Philipps Universität Marburg

Networked Wireless Sensor Systems for Ecological Monitoring

Dissertation

zur Erlangung des Doktorgrades der Naturwissenschaften
(Dr. rer. nat.)

dem Fachbereich Mathematik und Informatik
der Philipps-Universität Marburg
vorgelegt von

Master of Science (M.Sc.)
Patrick Lampe
geboren in Northeim

Marburg, im November 2022

Vom Fachbereich Mathematik und Informatik der Philipps-Universität Marburg (Hochschulkennziffer 1180) als Dissertation am 16. Dezember 2022 angenommen.

1. Gutachter: Prof. Dr.-Ing. Bernd Freisleben, Philipps-Universität Marburg
2. Gutachter: Prof. Dr.-Ing. Ralf Steinmetz, Technische Universität Darmstadt

Tag der Einreichung: 07. November 2022.
Tag der mündlichen Prüfung: 19. Dezember 2022.

Eidesstattliche Erklärung

Ich versichere, dass ich meine Dissertation selbstständig, ohne unerlaubte Hilfe angefertigt und mich dabei keiner anderen als der von mir ausdrücklich bezeichneten Quellen und Hilfen bedient habe. Die Dissertation wurde in der jetzigen oder einer ähnlichen Form noch bei keiner anderen Hochschule eingereicht und hat noch keinen sonstigen Prüfungszwecken gedient.

Datum

Unterschrift

Abstract

Due to the climate catastrophe, spatially and temporally comprehensive monitoring of the environment is becoming increasingly important to detect changes quickly. The problem is that human work, due to the limited number of experts, can only be scaled to a limited extent, and humans are not really suitable for performing permanent monitoring. In addition, there are possible cognitive limitations during data collection. If such data is collected almost simultaneously, it can no longer be perceived in its entirety by humans, resulting in data loss. Furthermore, data can be distorted by human assumptions (bias), or observations can be disturbed by the mere presence of humans. Considering these problems, a shift to automated monitoring methods seems inevitable.

This dissertation presents a new approach to create networked wireless sensor systems for ecological monitoring. Especially forested areas often have difficult conditions for humans and material, i.e., the proposed approach must also be able to deal with these real-world problems.

The main field of investigation chosen in this dissertation is the observation of bats. According to the "IUCN Red List of Threatened Species", one-third of them is either considered to be threatened or there is not enough data about them to make a statement. In addition, bats are good indicators of ecosystem health, i.e., monitoring bats can also provide information about the entire ecosystem's health. Nevertheless, the considerations and practical design ideas presented in this thesis can be applied to other fields of environmental monitoring.

The thesis introduces a novel approach to comprehensively monitor bats, divided into the components of monitoring the ecosystem and monitoring the bats themselves. In these areas, scientific work is presented and concluded with a view of a concrete overall system.

In the area of ecosystem monitoring, approaches on monitoring trees, insects, birds, and human disturbances are presented. In this context, new networked sensor nodes are developed and combined to form a complete system. A novel approach is proposed for equipping existing unchangeable systems and devices with new functions and thus integrating them into existing networked sensor systems. This novel approach is presented using the concrete integration of an existing system into the overall monitoring system.

In the area of bat monitoring, approaches are presented that enable the tracking of bats with different sensors and provide an automatic evaluation. New evaluation methods using machine learning techniques and the development of an operating system for particular sensor nodes are used. As a combination of newly developed hardware and software, a sensor node is presented that enables multi-sensory monitoring of bats triggered by bat activity. As a result of this, video, audio and VHF data can be collected, and ultimately one sensor's findings can be transferred to different other sensors.

Deutsche Kurzfassung

In Anbetracht der Klimakatastrophe wird ein flächendeckendes und zeitlich hoch aufgelöstes Monitoring der Umwelt immer wichtiger, um Veränderungen schnell zu detektieren. Hierbei ergibt sich das Problem, dass menschliche Arbeit durch die relativ geringe Anzahl an Expert*innen nur eingeschränkt skalierbar ist und Menschen auch nicht für ein permanentes Monitoring in der Natur geeignet sind. Hinzu kommen mögliche kognitive Einschränkungen in der Gesamtheit der Erfassung von Daten. Treten viele Daten nahezu zur selben Zeit auf, können diese von Menschen nicht mehr in ihrer Gänze wahrgenommen werden, und es kommt zu Datenverlusten. Des Weiteren können Daten durch menschliche Annahmen verfälscht (Bias) oder die Beobachtungen durch die reine Anwesenheit von Menschen gestört werden. Daher scheint die Nutzung automatisierter Verfahren zum Monitoring unausweichlich.

Diese Dissertation präsentiert neue Ansätze zur Erstellung eines vernetzten drahtlosen Sensorsystems zum ökologischen Monitoring. Gerade in bewaldeten Gebieten herrschen oftmals schwierige Bedingungen für Mensch und Material, so dass die vorgestellten Ansätze auch mit diesen Realbedingungen umgehen können müssen. Als Hauptuntersuchungsfeld wird vor allem das Monitoring von Fledermäusen betrachtet. Fledermäuse sind ökologisch besonders wichtig, da ein Drittel von ihnen nach der "IUCN Red List of Threatened Species" als bedroht gilt oder nicht genügend Daten vorhanden sind, um eine Aussage über ihr Bedrohungspotential treffen zu können. Hinzu kommt, dass Fledermäuse sehr gute Indikatoren für die Gesundheit eines Ökosystems sind und somit die Beobachtung von Fledermäusen auch Aufschluss über den Zustand des gesamten Ökosystems liefern kann. Allerdings lassen sich die in der Dissertation vorgestellten Ansätze auch auf weitere Felder des Umweltmonitorings anwenden.

Die Arbeit stellt insbesondere einen neuen Ansatz zum umfassenden Monitoring von Fledermäusen vor, der neben den Fledermäusen selbst auch das umgebende Ökosystem betrachtet. In diesen Bereichen werden wissenschaftliche Arbeiten vorgestellt und mit einem Blick auf ein konkretes Gesamtsystem abgeschlossen.

Im Bereich des Ökosystem-Monitorings werden Ansätze zum Monitoring von Bäumen, Insekten, Vögeln, sowie Störungen durch den Menschen vorgestellt. Einerseits werden hierbei neue vernetzte Sensorknoten entwickelt und zu einem Gesamtsystem zusammengefügt. Andererseits wird auch eine neuartige Methode vorgestellt, wie bestehende unveränderbare Systeme und Geräte mit neuen Funktionen ausgestattet werden können. Dies wird anhand der konkreten Integration eines bestehenden Systems in das gezeigte Gesamtsystem illustriert.

Im Bereich des Fledermaus-Monitorings werden Ansätze präsentiert, welche mit unterschiedlichen Sensoren das Tracking von Fledermäusen ermöglichen und eine automatische Auswertung liefern. Hierbei kommen sowohl neue Auswertungsmethoden auf der Grundlage maschineller Lernverfahren, als auch die Konzeption eines Betriebssystems für spezielle Sensorknoten zum Einsatz. Als Kombination von neu entwickelter Hard- und Software wird ein Sensorsystem vorgestellt, welches eine multi-sensorische Beobachtung, die durch Aktivitäten von Fledermäusen getriggert wird, ermöglicht. Hiermit können Video-, Audio- und VHF-Daten in einer neuartigen Kombination erhoben werden und neue Schlüsse aus den Daten eines Sensors auf die Daten eines anderen Sensors gezogen werden.

Acknowledgments

First of all, I would like to thank my motivator and mentor, Prof. Dr. Bernd Freisleben, for his advice and support during my research life. His challenging and motivating nature pushed my boundary in the case of research many times and carried me through many lows. I am grateful for the insights of many different projects that I carried out under his guidance.

I also want to thank Prof. Dr. Ralf Steinmetz from the TU Darmstadt for taking the time to review this work and for the opportunity to work with him and his research group in the DFG SFB 1053 MAKI.

Furthermore, I would like to thank my past and present colleagues and students at the Distributed Systems Group in Marburg for fruitful discussions, open communication, the productive and entertaining environment, and many helpful discussion while eating lunch or drinking coffee. Over the last seven years, these have been (in alphabetical order): Dr. Lars Baumgärtner Hicham Bellafkir, Christian Birk, Dr. Pablo Graubner, Mechthild Kessler, Nikolaus Korfhage, Matthias Leinweber, Dr. Markus Mühling, Alvar Penning, Falk Schellenberg, Nils Schmidt, Daniel Schneider, Stefan Schulz, Michael Schwarz, Dr. Roland Schwarzkopf, Sara Steinsiek, Markus Sommer, Christian Uhl, and Markus Vogelbacher.

During my Ph.D., two research projects provide financial support to me. I am very grateful for this financial support.

The DFG SFB 1053 MAKI provided an excellent environment for basic research in the area of the future Internet. I would especially like to thank the co-authors of my contributions in this area and the people who made this research environment possible (in alphabetical order): Dr. Michaela Bock, Dr. Patrick Felka, Prof. Dr. Bernd Freisleben, Dr. Pablo Graubner, Prof. Dr. Matthias Hollick, Prof. Dr. Boris Koldehofe, Dr. Katharina Keller, Dr. Manisha Luthra, Julia Müller, Prof. Dr. Mira Mezini, Dr. Tobias Meuser, Dr. Björn Richerzhagen, Prof. Dr. Ralf Steinmetz, and Dr. Denny Stohr.

The LOEWE Nature 4.0 project, funded by the Hessian Ministry of Science and Art (HMWK), introduced me to environmental monitoring and provided exciting tasks, challenges, and a motivating use case. It also brings together many researchers from different research fields, and I am very grateful to get to know each other. I want to thank my co-authors and other team members from Nature 4.0 for their unusual ideas, comprehensive explanations of the tasks in their respective domains, and their energy in the many different joint projects (in alphabetical order): Marcel Becker, Prof. Dr. Nina Farwig, Prof. Dr. Bernd Freisleben, Dr. Nicolas Friess, Lea Leister, Dr. Patrick Lieser, Tobias Geitz, Kim Lindner, Julia Maier, Prof. Dr. Thomas Nauss, Betty Neumann, Dr. Christoph Reudenbach, Tobias Richter, Dr. Sascha Rösner, Dr. Raphael Royaute, Dr. Dana G. Schabo, Prof. Dr. Bernhard Seeger, and Julian Zobel. In particular, I would like to thank Jannis Gottwald, who introduced me to several fields of environmental monitoring in old-fashioned ways and always had new ideas to kick out this old-fashioned stuff. I am thankful for spending many days and nights together while catching bats, building sensor nodes, or discussing ideas to get even better results for our research projects.

I would especially like to thank Dr. Lars Baumgärtner. He introduced me to practical research and supervised my bachelor's and master's thesis. Without close contact with the HMWK LOEWE NICER project he was working on, maybe my answer to staying in science could be another.

I also would like to thank Mechthild Kessler for taking care of almost everything. She has been the good soul of the working group.

I would also like to thank Dr. Jonas Höchst and Dr. Artur Sterz for our time together at the University of Marburg. Seven years of being in the same office and doing research together tightens our friendship after being in nearly all university events together since the first semester. Especially the discussions, motivation, and support while working together or hanging around in our free time are always empowering for me. Without both of you by my side, I certainly wouldn't have made it through the time of my PhD.

Finally, I want to thank my family for always being there. My parents, Susanne and Norbert Lampe, supported me in all my activities from early childhood and encouraged me to pursue all my goals in life. They made me the curious, open-minded person I am today, for which I am deeply grateful. And, of course, my brother Dennis Lampe for being the lovely and straightforward person you are and for all the good moments in the past and the future. Last but not least, I want to thank all my good friends, who always helped me through many difficult times, pushed me when I was unmotivated, provided a cozy place when needed, and got some other ideas while having a good time together.

My Contributions

As indicated in my acknowledgments, I was lucky to be part of an interdisciplinary team. Through this, some new perspectives and previously unknown connections became apparent. I had many insights into other disciplines that would otherwise have been difficult to gain. In addition, this collaboration enabled results at some points that would probably have been difficult or impossible without the cooperation [88, 222]. Therefore, it is challenging to attribute the specific portions of a publication to a particular person. Since I use parts of published papers in this thesis and they are also based on joint work as described, I will write down the contributions to the individual publications as best I can in the following.

Chapter 3 is based on my ideas and considerations of how a networked wireless sensor system for bat monitoring should be designed under adverse conditions.

The design of the software and the hardware of the publication [81] in Chapter 5.5 came significantly from me, as well as the initial implementation of the software. Especially in the technical conception, I benefited greatly from Jannis Gottwald's expertise in bats and their behavior, so we did this in close cooperation. Jannis Gottwald led the field study, and we shared the task of maintaining the stations in the field with Julia Maier, Lea Leister, Tobias Richter, and Betty Neumann. Later, Dr. Jonas Höchst and myself developed the software further and evaluated it together. The evaluation of the field study and the writing of the paper was mainly done by Jannis Gottwald with input from Dr. Jonas Höchst and myself. Both Jannis Gottwald and I are shared first authors. Prof. Dr. Bernd Freisleben, Dr. Nicolas Friess, and Prof. Dr. Thomas Nauss critically revised the different versions of the system and contributed to its optimization.

In the second publication [165], which I present in Chapter 4.2, the hardware design and the software were conceived and created by myself. Technical input from Jonas Mielke-Möglich and Dr. Lea Heidrich was essential to specify the requirements. Both designed and planned the field study together and also carried it out. Together, we deployed the stations, and they conducted the field study operationally for the rest of the season. The publication's writing was separated into technical and ecological parts and was mainly done by Jonas Mielke-Möglich and Dr. Lea Heidrich for the environmental domain and by myself for the technical part. The analysis of the results come equally from Jonas Mielke-Möglich and Dr. Lea Heidrich. Additionally, Mario Fickus was involved and took over parts of the evaluation. Jannis Gottwald and Dr. Nicolas Friess were involved with ideas and gave input, especially in the initial phase. Prof. Dr. Thomas Nauss, Prof. Dr. Roland Brandl, Prof. Dr. Martin Brändle, and Prof. Dr. Bernd Freisleben improved the written text by proofreading the manuscript.

The idea for publication [13] was developed between Dr. Lars Baumgärtner and myself and was implemented by Alvar Penning and Dr. Lars Baumgärtner. Alvar Penning and myself did the evaluation. Dr. Lars Baumgärtner mainly did the paper writing. Dr. Björn Richerzhagen, Prof. Dr. Ralf Steinmetz, and Prof. Dr. Bernd Freisleben reviewed the paper and improved the writing.

The idea behind the open-source software for reliable VHF wildlife tracking [104] presented in Section 5.3 emerged from Jannis Gottwald, Dr. Jonas Höchst, and myself. Jannis Gottwald contributed to the requirements of engineering based on his domain knowledge and experience in fieldwork. Dr. Jonas Höchst designed and implemented the software system; the hardware design is based on prior work and was improved by Jannis Gottwald and myself. Julian Zobel implemented the LoRa communication module and provided suggestions for improvements. Jannis Gottwald, Dr. Jonas Höchst, and myself did the evaluation jointly. Prof. Dr. Thomas Nauss, Prof. Dr. Ralf Steinmetz, and Prof. Dr. Bernd Freisleben reviewed the paper and provided improvements concerning the writing.

The idea and concept for *unobtrusive mechanism interception* used in Chapter 4.3 came mainly from myself, with input from Dr. Jonas Höchst, Dr. Artur Sterz, and Markus Sommer. This concept was published in paper [141] and includes a prototype, which Dr. Artur Sterz and Dr. Jonas Höchst implemented. I was the lead author for the publication, and all co-authors contributed to the text.

The idea for *ForestEdge* came from myself, and I also did the reverse engineering of the Tree Talker protocol. On this basis, Markus Sommer and myself developed the concept for this interceptor where Markus Sommer led the implementation with contributions from Christian Uhl and myself. The evaluation was done mainly by Markus Sommer and myself. Dr. Jonas Höchst, Dr. Artur Sterz, and Prof. Dr. Bernd Freisleben participated in many discussions, gave necessary input, and reviewed different versions of the publication.

Jannis Gottwald and myself proposed the idea for the classification of VHF signals. It was, as a prototype, implemented for different activity classes by Sara Steinsiek in her master thesis, which I closely supervised. Ideas and problem solutions for data preparation and improvements of data quality, which resulted in publication [104], were found by Sara Steinsiek and myself. Based on these findings and the prototype from Sara Steinsiek, Jannis Gottwald created a simplified method for classification into active and passive phases. Dr. Raphaël Royauté and Jannis Gottwald had done the case study and the statistical part of this publication. The field work, i.e., collecting ground truth (observing animals and noting behavior) and tagging animals with VHF transmitters, was done by Marcel Becker, Tobias Geitz, Lea Leister, Kim Lindner, Julia Maier, Dr. Sascha Rösner, and Dr. Dana G. Schabo. Prof. Dr. Bernd Freisleben, Prof. Dr. Roland Brandl, Prof. Dr. Thomas Müller, Prof. Dr. Nina Farwig, and Prof. Dr. Thomas Nauss were involved in discussions and the revision of the publication.

The idea to classify bat calls occurring in Germany came from Jannis Gottwald and myself, and we also obtained labeled training data. The initial conception of the work was done between Hicham Bellafkir, Markus Vogelbacher, Jannis Gottwald, and myself. The implementation, including finding the appropriate neural network architecture, training the neural network, and evaluating the results, was done by Hicham Bellafkir, Markus Vogelbacher, Dr. Markus Mühling, and Nikolaus Korfhage. Dr. Nicolas Frieß, Prof. Dr. Thomas Nauss, and Prof. Dr. Bernd Freisleben contributed significantly with their input to raise the quality of the publication.

Bird@Edge [103] presented in 4.5 was jointly designed by Hicham Bellafkir, Dr. Jonas Höchst, Markus Vogelbacher, Dr. Markus Mühling, Daniel Schneider, and myself. Dr. Jonas Höchst realized the system software and hardware design with contributions from Hicham Bellafkir and myself. The neural network architecture design, training, evaluation, and embedded implementation were led by Hicham Bellafkir and Markus Vogelbacher, with contributions by Dr. Markus Mühling and Daniel Schneider. Dr. Jonas Höchst did the experimental evaluation of the system. Kim Lindner, Dr. Sascha Rösner, Dr. Dana G. Schabo, and Prof. Dr. Nina Farwig provided domain knowledge and reviewed the paper with Prof. Dr. Bernd Freisleben and provided valuable feedback for further improving the manuscript.

SmartFace [139] emerged from the NICER project, with Dr. Lars Baumgärtner and myself creating the idea. I conceptualized and implemented *SmartFace* under the close supervision of Dr. Lars Baumgärtner. Dr. Lars Baumgärtner and myself wrote the publication, and Prof. Dr. Ralf Steinmetz and Prof. Dr. Bernd Freisleben contributed improvements.

Contents

1 Introduction

Environmental monitoring is a long-established, but in most cases highly manual field of work [243]. Many research findings have been achieved through monitoring and experiments designed, conducted, and evaluated by humans [243]. Several factors complicate manual research in this field. One of them is the time-consuming nature of fieldwork and the lack of possibilities to control the collected results during the field season. These problems lead to experiments being conducted over an entire season, even though it may be evident after a few weeks that the results will not be usable. For example, a researcher has to set up, supervise, and dismantle an experiment and also has to collect information by monitoring on-site, and the effort to collect a data point is quite considerable. Thus, considering all these tasks, in most cases no time is left to check the data for validity. In addition, these tasks must primarily be performed by people with prior knowledge in identifying species, capturing moths, recognizing bird calls, or climbing trees to harvest leaves. For all these activities, well-trained people are needed to perform these tasks conscientiously. Furthermore, even if well-trained people are doing the job, failures can happen, and the data sets can be biased by our human nature or even the presence of an observer [116, 243]. In addition, for a spatially and temporally highly resolved collection of data, one would need many well-trained people who can perform the tasks in parallel at different locations. In practice, the spatial or temporal distribution is reduced to carry out the tasks with the given number of people. Additionally during the season, evaluations are often skipped, because any evaluation would mean further losses in local or temporal resolution. Thus, manual fieldwork has two main problems: (a) it is very time-consuming, so experiments cannot be evaluated during the season, and (b) it is not scalable.

Martin Luther King said the following words with respect to the Vietnam War: *"We are now faced with the fact that tomorrow is today. We are confronted with the fierce urgency of now. In this unfolding conundrum of life and history, there is such a thing as being too late. Procrastination is still the thief of time. Life often leaves us standing bare, naked, and dejected with a lost opportunity. The 'tide in the affairs of men' does not remain at the flood; it ebbs. We may cry out desperately for time to pause in her passage, but time is deaf to every plea and rushes on. Over the bleached bones and jumbled residue of numerous civilizations are written the pathetic words: Too late."* Undoubtedly, we could also follow Mojib Latif [143] in mapping this to the climate catastrophe and our ability to fight it.

Given the imminent climate catastrophe, area-wide and temporally high-resolution environmental data monitoring is becoming more and more critical [222, 243]. On the one hand, this involves logging changes in the environment and generating conclusions and expectations from there. On the other hand, they are also predicting how the system will change. Especially the responses of species to climate change are among the big questions in biology [26]. This reaction plays a role in looking at biodiversity and the state of the environment and estimating the frequency of, e.g., extreme weather events or predicting them. Both resolution dimensions are essential; a broad collection of good monitoring data and high-resolution data collection are needed to meet the requirements. For example, the International Union for Conservation of Nature (IUCN) Red List of Threatened Species lists more than 120,000 species; up to 17,000 are listed with a 'Data deficient' status [243].

Especially for bats, the situation is critical. About one-third of all bat species are classified as threatened or listed with a 'Data deficient' status by the IUCN. About half of all bat species show an unknown or declining population trend [68]. This trend reflects the need to observe bats as notable species since bats face similar threats during climate change [68]. Nevertheless, the available data set for bats is much smaller, as the IUCN Red List also reflects. Furthermore, bats are good indicators of harmful environmental conditions, also described in the article by Aodha et al. [153]: *"Monitoring of bat species and their population dynamics can act as an important indicator of ecosystem health as they are particularly sensitive to habitat conversion and climate change."* In the context of the climate catastrophe and the following biodiversity crisis, it must be possible to collect findings in this area without long delays and on a broad scale because only in this way findings can lead to changes and countermeasures on time [16]. If research in environmental monitoring wants to achieve more than just describing effects retrospectively, it is essential for findings to be made available within a shorter time frame. Making findings available would enable a feedback loop for chosen actions, which is essential in evaluating their effects and efficiency.

Due to the technical development in the last years and decades, microprocessors and single-board computers can record data from various sensors and perform complex processing steps on this computers [116, 222, 243]. Furthermore, the falling prices for technical devices, microprocessors, and sensors are responsible, because sensor nodes with commercial off-the-shelf (COTS) components can lead to a large-scale ecological understanding [243]. The broadening of the market for sensors and the increase in unit numbers play a significant role here. Currently, the situation is aggravated by the Russian war on Ukraine and the Corona pandemic, and inexpensive components are hard to obtain. Nevertheless, it is desirable to digitize ecological monitoring processes, because this is the only way an area-wide collection of ecological data can be obtained in a temporally high resolution [243]. To achieve these effects in the future, a system for ecological monitoring must be as flexible and modular as possible. Individual components must therefore be interchangeable in terms of hardware and software components.

1.1 Motivation

The state of the art in various disciplines that perform ecological monitoring is to employ human experts [243]. Here, the process can only scale with more humans, which are complex and time-consuming to train, not cost-effective with fair payment, error-prone, and do not provide results with the same quality among each other. Automatization of these tasks can achieve both scalability and comparability. Data recording, forwarding, and processing must be automated and made intelligent without the need for much human intervention. Available and cost-effective components must be selected for sensor systems, since this is the only way to ensure scalability [88]. During and before transmission, domain knowledge must be used to enable preprocessing and compressed data. Preprocessing and compressing data allows data to be made available in near real-time for automatic knowledge generation.

1.2 Problem Statement

Observing the behavior of bats to estimate the environment's health is a time-consuming and, in most cases, manual task. If devices are used, the data is processed with much time between recording and processing.

The main question tackled in this thesis is: How can we perform ecological monitoring in an automatic and time-consuming manner? In particular, the thesis addresses the problem of performing bat monitoring automatically in an effective manner.

This problem can be split into two parts. The first part is devoted to the automatic monitoring of the ecosystem surrounding bats to measure external factors that are relevant for bats. The second part is to automatically monitor the behavior of bats: What are bats doing? Where are bats flying, sleeping, or hunting? Which species or which individual is flying, sleeping, or hunting? In general, the following questions should be answered: Who does an action? Where does the action take place? Which action takes place?

The goal of this thesis is to develop a sensor system that can answer these questions. Especially they should be answered for all bats, some species, and individuals.

1.3 Contributions

This thesis presents a novel approach for developing networked wireless sensor systems under ecological conditions. This approach is primarily dedicated to bat research, but also includes general design criteria for remotely located networked wireless sensing systems. Challenges of remote scenarios are described, and solutions for different problems are presented. A novel approach for automatic bat monitoring is presented and detailed in the subsequent chapters. Finally, a concrete implementation of such a system is presented. The monitoring system is divided into two parts. First, the part of ecosystem monitoring has to be adapted to the particular use case, but most parts could also be used for different situations. Second, the part of the use of case-specific monitoring, namely bat monitoring, must be realized.

Ecosystem Monitoring

For ecosystem monitoring, the thesis presents contributions to monitoring the environment in which bats live. With *EcoSense*, the thesis shows how COTS components can be used to monitor environmental factors. With the *automatic non-lethal moth trap (AMT)*, a robust and field-tested method to automatically estimate the occurrence of moths is presented. With *Unobtrusive Mechanism Interception*, the conceptual basis to integrate legacy devices into an existing network and to provide them with new functionalities is presented. *Unobtrusive Mechanism Interception* is applied to the *Tree Talker* in *ForestEdge*, thus integrating an existing sensor node design and adding new functionalities. With *Bird@Edge*, we present an approach for edge processing of audio data to determine bird calls in soundscapes. With *SmartFace*, the possibility is offered to recognize faces on photos in an energy-efficient manner to discard person-related data when taking pictures and disturbing bats by humans.

Bat Monitoring

For bat monitoring, we first present a method for recognizing bat species in continuous audio. Using the *Bird@Edge* approach and a developed neural network for bat species recognition, the *Bat@Edge* system can be built for automatic bat recognition at the edge of the Internet. With *tRackIT OS*, we refine an existing sensor node design into a robust, error-avoiding, and live data-exchanging solution. A further contribution with respect to bat monitoring is the automatic classification of VHF data into active and passive behavior of birds and bats. With *BatRack*, a multi-sensor approach is presented, which offers the possibility to fuse data via the interaction of different sensors and thus enrich the data of a sensor by the findings of another sensor.

Software Design for a Bat Monitoring System

We present six design principles that can help build a networked sensor system from scratch. Furthermore, we present the software design of a bat monitoring system with the integration of the approaches described above.

1.4 Publications

During the work on this thesis, the following papers were published:

1. **Patrick Lampe**, Markus Sommer, Artur Sterz, Jonas Höchst, Christian Uhl, and Bernd Freisleben. “ForestEdge: Unobtrusive Mechanism Interception in Environmental Monitoring.” In: *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. IEEE. 2022, pp. 264–266
2. **Patrick Lampe**, Markus Sommer, Artur Sterz, Jonas Höchst, Christian Uhl, and Bernd Freisleben. “Unobtrusive Mechanism Interception.” In: *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. IEEE. 2022, pp. 303–306
3. Jonas Höchst, Hicham Bellafkir, **Patrick Lampe**, Markus Vogelbacher, Markus Mühling, Daniel Schneider, Kim Lindner, Sascha Rösner, Dana G. Schabo, Nina Farwig, and Bernd Freisleben. “Bird@Edge: Bird Species Recognition at the Edge.” In: *International Conference on Networked Systems (NETYS)*. Springer. May 2022, pp. 69–86
4. Hicham Bellafkir, Markus Vogelbacher, Jannis Gottwald, Markus Mühling, Nikolaus Korfhage, **Patrick Lampe**, Nicolas Frieß, Thomas Nauss, and Bernd Freisleben. “Bat Echolocation Call Detection and Species Recognition by Transformers with Self-attention.” In:

International Conference on Intelligent Systems and Pattern Recognition. Springer. 2022, pp. 189–203

5. Jonas Mielke Moeglich[1], **Patrick Lampe[1]**, Mario Fickus, Jannis Gottwald, Thomas Nauss, Roland Brandl, Martin Braendle, Nicolas Friess, Bernd Freisleben, and Lea Heidrich. "Automated Non-lethal Moth Traps can be used for Robust Estimates of Moth Abundance." In: *submitted; under review* ()

6. Jannis Gottwald, Raphael Royaute, Marcel Becker, Tobias Geitz, Jonas Hoechst, **Patrick Lampe**, Lea Leister, Kim Lindner, Julia Maier, Sascha Roesner, et al. "Classifying the Activity States of Small Vertebrates using Automated VHF telemetry." In: *submitted; under review* ()

7. Jannis Gottwald[2], **Patrick Lampe[2]**, Jonas Höchst, Nicolas Friess, Julia Maier, Lea Leister, Betty Neumann, Tobias Richter, Bernd Freisleben, and Thomas Nauss. "BatRack: An Open-Source Multi-Sensor Device for Wildlife Research." In: *Methods in Ecology and Evolution* (July 2021), pp. 1867–1874

8. Jonas Höchst, Jannis Gottwald, **Patrick Lampe**, Julian Zobel, Thomas Nauss, Ralf Steinmetz, and Bernd Freisleben. "tRackIT OS: Open-Source Software for Reliable VHF Wildlife Tracking." In: *51. Jahrestagung der Gesellschaft für Informatik, Digitale Kulturen, INFORMATIK 2021, Berlin, Germany*. LNI. GI, Sept. 2021, pp. 425–442

9. Julian Zobel, Paul Frommelt, Patrick Lieser, Jonas Höchst, **Patrick Lampe**, Bernd Freisleben, and Ralf Steinmetz. "Energy-efficient Mobile Sensor Data Offloading via WiFi using LoRa-based Connectivity Estimations." In: *51. Jahrestagung der Gesellschaft für Informatik, Digitale Kulturen, INFORMATIK 2021, Berlin, Germany*. LNI. GI, Sept. 2021, pp. 461–479

10. Johnny Nguyen, Karl Kesper, Gunter Kräling, Christian Birk, Peter Mross, Nico Hofeditz, Jonas Höchst, **Patrick Lampe**, Alvar Penning, Bastian Leutenecker-Twelsiek, Carsten Schindler, Helwig Buchenauer, David Geisel, Caroline Sommer, Ronald Henning, Pascal Wallot, Thomas Wiesmann, Björn Beutel, Gunter Schneider, Enrique Castro-Camus, and Martin Koch. "Repurposing CPAP Machines as Stripped-Down Ventilators." In: *Scientific Reports* 11.1 (June 2021), pp. 1–9

11. Jonas Höchst, Alvar Penning, **Patrick Lampe**, and Bernd Freisleben. "PIMOD: A Tool for Configuring Single-Board Computer Operating System Images." In: *IEEE Global Humanitarian Technology Conference (GHTC)*. Seattle, USA, Oct. 2020, pp. 1–8

12. Manisha Luthra, Boris Koldehofe, Jonas Höchst, **Patrick Lampe**, Ali Haider Rizvi, and Bernd Freisleben. "INetCEP: In-Network Complex Event Processing for Information-Centric Networking." In: *15th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. Cambridge, UK, Sept. 2019, pp. 1–13

13. Artur Sterz, Lars Baumgärtner, Jonas Höchst, **Patrick Lampe**, and Bernd Freisleben. "OPPLOAD: Offloading Computational Workflows in Opportunistic Networks." In: *IEEE*

[1] shared first authorship

[2] shared first authorship

44th Conference on Local Computer Networks (LCN). Osnabrück, Germany, Oct. 2019, pp. 381–388

14. Lars Baumgärtner, **Patrick Lampe**, Jonas Höchst, Ragnar Mogk, Artur Sterz, Pascal Weisenburger, Mira Mezini, and Bernd Freisleben. "Smart Street Lights and Mobile Citizen Apps for Resilient Communication in a Digital City." In: *IEEE Global Humanitarian Technology Conference (GHTC).* Seattle, USA, Oct. 2019, pp. 1–8

15. Pablo Graubner, **Patrick Lampe**, Jonas Höchst, Lars Baumgärtner, Mira Mezini, and Bernd Freisleben. "Opportunistic Named Functions in Disruption-tolerant Emergency Networks." In: *ACM International Conference on Computing Frontiers 2018 (ACM CF).* Ischia, Italy: ACM, May 2018, pp. 129–137

16. Lars Baumgärtner, Alvar Penning, **Patrick Lampe**, Björn Richerzhagen, Ralf Steinmetz, and Bernd Freisleben. "Environmental Monitoring using Low-Cost Hardware and Infrastructureless Wireless Communication." In: *IEEE Global Humanitarian Technology Conference (GHTC).* IEEE. 2018, pp. 1–8

17. Nils Schmidt, Lars Baumgärtner, **Patrick Lampe**, Kurt Geihs, and Bernd Freisleben. "Miniworld: Resource-aware Distributed Network Emulation via Full Virtualization." In: *IEEE Symposium on Computers and Communications (ISCC).* IEEE. 2017, pp. 818–825

18. **Patrick Lampe**, Lars Baumgärtner, Ralf Steinmetz, and Bernd Freisleben. "Smartface: Efficient Face Detection on Smartphones for Wireless on-demand Emergency Networks." In: *24th International Conference on Telecommunications (ICT).* IEEE. 2017, pp. 1–7

19. Lars Baumgärtner, Paul Gardner-Stephen, Pablo Graubner, Jeremy Lakeman, Jonas Höchst, **Patrick Lampe**, Nils Schmidt, Stefan Schulz, Artur Sterz, and Bernd Freisleben. "An Experimental Evaluation of Delay-tolerant Networking with Serval." In: *IEEE Global Humanitarian Technology Conference (GHTC).* Seattle, USA, Oct. 2016, pp. 70–79

1.5 Organization

This thesis is organized as follows:

Chapter 2 introduces the background for the research done in this dissertation. It contains the biological scenarios of ecological monitoring used in this thesis and briefly introduces sensor systems, machine learning, and wireless communication.

Chapter 3 presents the general approach of this thesis with the main problem statements in this area of research. They are followed by specific challenges in forested areas, a critical use case for bat monitoring. After that, solutions for problems in sensor systems are presented. The final section presents a novel design for automated bat monitoring to answer the questions stated in Section 1.2.

Chapter 4 presents research to monitor the ecosystem surrounding bats. This chapter includes six sections. The first section introduces *EcoSense*, a solution that presents many COTS components and describes the structure of building a networked sensor system. The second section is about a design and field study of an automated moth trap (AMT), which is a novel non-lethal method to measure the abundance of moths. The third section is about the approach of *Unobtrusive Mechanism Interception*, which is used in *ForestEdge* to add new functionality to the proprietary *TreeTalker* system. The fifth section on *Bird@Edge* shows how to process the audio of bird songs on the edge of the Internet. The last section is about *SmartFace*, a solution to detect human faces in images.

In Chapter 5, research is presented to automatically monitor bats. First, an approach for bat call detection and species recognition is presented. With the *Bat@Edge* infrastructure and the developed neural networks for bat call detection, bat detection on the edge is possible, as described in the second section. Then, *tRackIT OS* is presented as a user-friendly and reliable solution to record VHF signals. The next section concerns the classification of VHF signals in active and passive parts for bats (and other animals) equipped with a VHF transmitter. In the last section, *BatRack*, a multi-sensor device for the coupled monitoring of different sensors is presented.

In Chapter 6, the software design of a fully automated bat monitoring system is presented. First of all, design principles are introduced, and after that, the networked wireless sensor system's substantial components are presented.

Chapter 7 concludes this thesis and discusses possible areas of future work.

2

Fundamentals

This chapter explains concepts, terms, and technologies required throughout this thesis. In Section 2.1, the main aspects of the classical biological scenarios are introduced. Section 2.2 introduces sensor nodes and different sensor types, especially sensor types that are used in this thesis. Section 2.3 gives an overview of wireless communication. Section 2.4 describes basic concepts of machine learning and highlights the differences between unsupervised and supervised learning.

2.1 Ecological Scenarios

Observing and understanding ecosystems is a central component of biology. New technology can improve the monitoring of changes in species or entire ecosystems. However, according to the work of Hahn et al. [88], this often requires more than the mere adaptation of off-the-shelf devices that neither scale, nor are they reasonably adapted to the exact demands. Furthermore, there is a lack of reliable methods and use case adapted devices that can deal with the prevailing conditions, such as energy constraints and robustness. Wiens [26] stated: *"Dozens of studies have shown local extinctions and declines that appear to be associated with recent, human-related climate change. For most of these cases, the primary cause of the declines has not been identified, highlighting our worryingly limited knowledge of this crucial issue. However, where causes have been identified, changing species interactions is essential in most cases. Because many of these impacts have already happened and climate has changed only a little relative to the predicted changes in the next 100 years, our results suggest that these shifting interactions may make even small climatic changes dangerous for the survival of populations and species."*

2.1.1 Ecological Monitoring

As stated by Spellerberg [223], monitoring is the systematic measurement of variables and processes over time. Spellerberg also divided the field into three different fields of monitoring. First, census is part of monitoring programs with the purpose of getting population counts. Second, surveillance measures variables over time to get a timeseries of data. Finally, monitoring does the same as surveillance, but it is assumed that the measurement has a specific reason. With this definition, the term ecological monitoring as defined by Spellerberg has the meaning of a systematic collection of ecological data in a standardized manner and with regular intervals

[223]. Also, Nichols et al. [177] differentiate between surveillance and monitoring in the same manner.

2.1.2 Bat Behavior Monitoring

To study the behavior of bats, bat calls are often recorded [153, 184]. By automatically analyzing these audio recordings, common species, in particular, can be easily identified. To get recordings of rarer species, individuals of these species are held close to a microphone in a flying cage or tent, and thus all calls can be assigned to this species [216]. Thus, it is questionable whether the calls in these situations correspond to those in routine flights. Another possibility is to get the recordings in the field and equip a bat with a transmitter beforehand. As a result, in the case of territorial species, an assignment of calls to species can occur when a tagged individual approaches the microphone. The individual procedures for recording bat behavior are described below. The monitoring of bats is critical because a decreasing bat population could be used as an indicator of harmful environmental conditions [188].

Bat Call Detection

The beginning of the processing chain is often the detection of bat calls. Since a bat primarily uses calls as a tool for localization, these are realistically broadband and quite the same in certain situations. Roughly, a distinction is made between social and non-social calls. Social calls, depending on the species, are sometimes audible to humans; the location calls are mostly not. Calls of bats can differ significantly between species, as shown in Figure 2.1.

However, bat calls often appear in the recordings along with other sounds, such as the cracking of branches or grasshoppers, which also provide broadband signals. Thus, manually and automatically, bat calls must be separated from other broadband sounds. Other narrowband sounds or sounds that are audible to humans are usually easy to filter with, for example, high-pass filters. Preliminary work, such as the BatDetect project [153], exists in this area.

Bat Call Classification

As the second step in processing, both manual and automatic, the detected calls must be assigned to species of bats. For this purpose, processing methods range from measuring spectrograms to neural networks. The goal of this processing step is to generate a species list. Distinguishing individuals based on calls is not yet possible at the time of this work.

VHF Signal Detection

To get an insight into where bats are located, bats are equipped with VHF tags, and the signals are received either manually by cross-tracking or automatically, as described by Gottwald et al. [80]. Cross-tracking allows cross-bearings with several receivers, i.e., two beams, which are recorded by the two measurement points around space against the north and intersect in the area. As a result, with the classical method, large distances often arise between both measuring

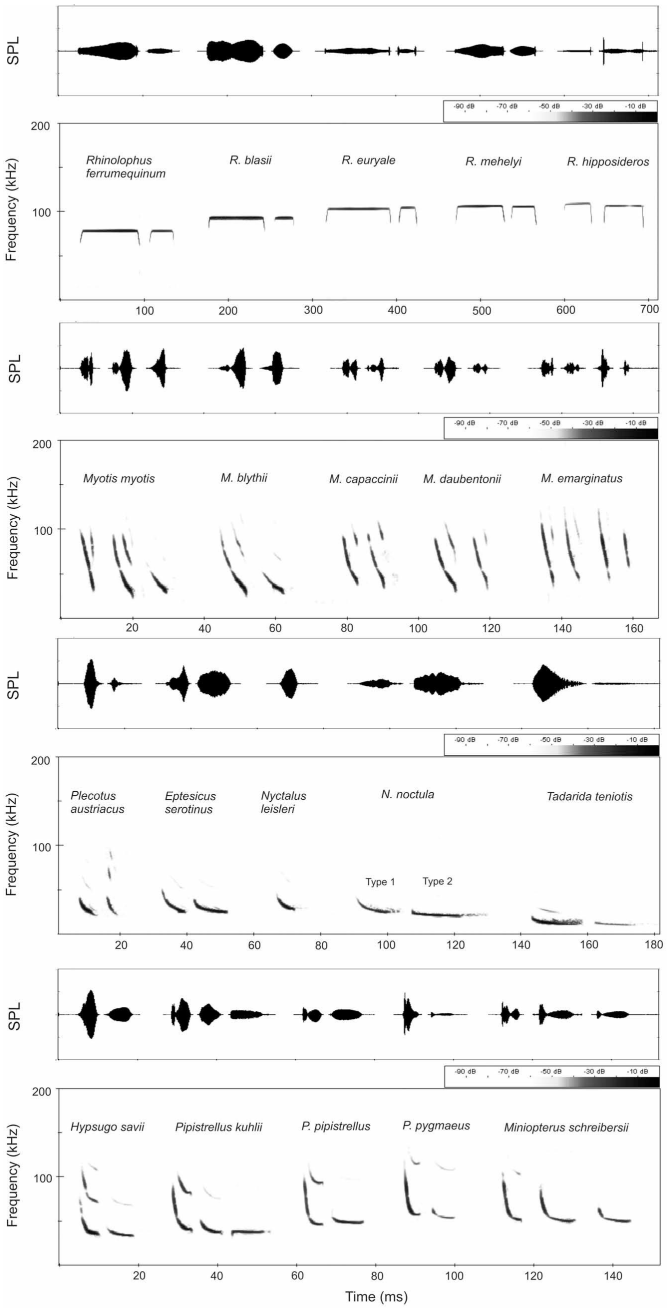


Figure 2.1: Different bat calls of bats in Greece (Papadatou et al. [180])

persons, since the method needs either many measuring persons or large spaces to cover the detection area. In this case, considerable error values arise even with a minimum angular error of 5%, as shown in Figure 2.2.

The VHF signals are continuously recorded in the automatic case, and the bearing is taken based on significantly closer stations. Here, the system benefits from scalability due to the automatic processing.

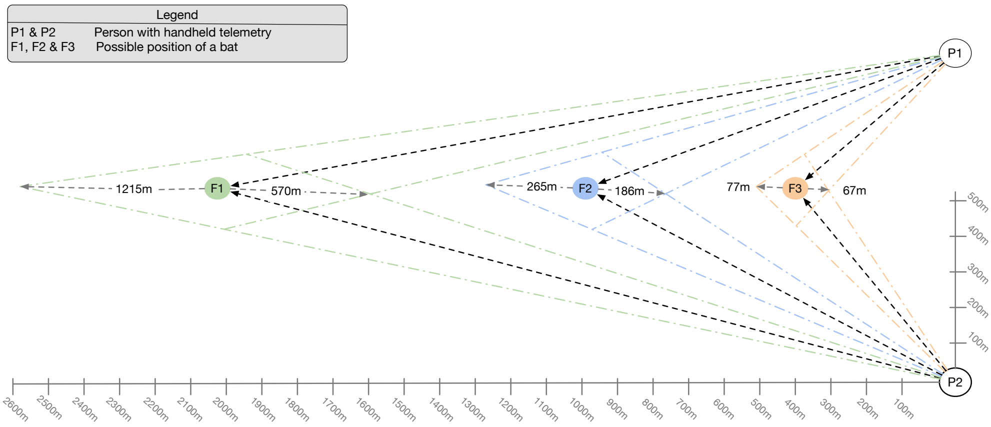


Figure 2.2: Schematic description of angle error to position error when the angle error is constant by 5 degrees and the target varies in field position and distance

.

2.1.3 Insect Abundance

In the last years, the number of insects has become more and more the focus of research, because here, also in a short time, significant changes have been observed [26, 166]. One possible approach to measure insect abundance is manually deployed lethal light traps where the moths attracted by the light fall in baskets by using, e.g., chloroform. This method allows the number, mass, and composition of the killed individuals to be determined by manual identification following the experiment [166]. This procedure is accepted in the community but poses risks to the population by killing individuals, as shown, e.g., by Gezon et al. [77]. Therefore, researchers should not use the method permanently and continuously for monitoring. However, high temporal resolution and permanent monitoring of species in the forest are essential to monitor, quantify, and evaluate the effects of climate change and possible measures to increase the insect population.

2.1.4 Bird Song Detection and Classification

Another aspect of biodiversity assessment is the study of the occurrence of birds. These can be detected and mapped through observations and through the classification of bird songs. Here, it is essential to recognize a single song, as well as to determine the species in a mix of many birds. From such a soundscape, a species list is created on a grid map when experts walk around the area. With this list, researchers can determine the occurrences of certain species in different areas.

2.2 Sensor Nodes

Sensor nodes consist of components that automatically record data from the installed sensors. A sensor node can contain only one sensor or several sensors. A central computing unit processes all these sensors and can consist of a microprocessor or a single-board computer. More powerful architectures are also conceivable, but the scenario limits the energy consumption of sensor nodes installed in remote areas, therefore they do not have a fixed power supply.

Sensors in sensor nodes can be pretty straightforward, such as a temperature sensor or a humidity sensor. Alternatively, a light sensor detects the presence of light. Nevertheless, even for this type of sensor, there are some differences. For example, a light sensor can measure the concrete brightness or the brightness of different bands of light. Measuring the brightness on different bands offers the advantage that changes in the spectrum can be detected. Thus, for example, the sensor node can estimate the condition of the foliage [245]. These sensors provide relatively small amounts of data, in the range of a few bytes per measurement, depending on the sensors' accuracy.

2.2.1 Microphones

Microphones can be used to record audio. Some audio sensors work with a digital interface and offer the audio data via, e.g., USB or I2C. Other microphones have to be read out in a fully analog manner. Thus, the choice of a microphone depends on the application. This choice is especially noticeable in the characteristics of microphones. Here, the range extends from cardioid or super-cardioid microphones, which have relatively directional characteristics, to omnidirectional microphones, which are almost equally sensitive in all directions. In addition to the different characteristics, the microphones' frequency ranges are also different, i.e. there are microphones that mainly cover the human voice to microphones that are relatively sensitive in the range audible to humans to ultrasonic microphones, which can have a sampling rate of up to 384 kHz. For these ultrasonic microphones, the sampling rate of 384 kHz results, using the sampling theorem of Nyquist and Shannon, in a maximal recordable frequency of 192 kHz. Another distinguishing feature of microphones is the sensitivity curve over the frequency range. Here, the sensitivity can vary depending on the frequency; the characteristics can also differ depending on the frequency. Thus, there are many distinguishing features of microphones. Also, the amount of data generated by audio recordings varies enormously with the sampling rate, the sampling depth, and the compression used.

2.2.2 Cameras

With cameras, the quality of the taken images also depends on many factors. First, the sensor's resolution varies. However, the sensor size and the associated light sensitivity are also decisive for the quality of the images. Especially in low-light situations, the lens' sensor size and light intensity can distinguish between a dark image and one in which the action is still clearly visible. When choosing a suitable sensor in this area, a researcher must also consider the scenario.

Additional sensors without infrared (IR) filters and an IR illuminator can be used for recording in the dark at short distances. Most animals do not perceive light in the infrared spectrum, so they are not disturbed by this setup. However, the camera's sensor can take much brighter pictures or videos due to the IR light, even without relatively large sensors and fast lens shutters. At short distances, this has the advantage of making the system potentially cheaper. The sensor node can store photos and videos raw or compressed directly. Using only grayscale in images can also lead to clear savings in memory requirements and, depending on the scenario, hardly any loss of quality. Thus, the amount of data can only be estimated based on the specific scenario and hardware.

2.2.3 VHF Receivers

A signal processor and an antenna are the main components for receiving VHF signals. The antenna must amplify signals in the correct band to effectively record the appropriate signals. The right band means that antennas are particularly receptive to the incoming signals in a specific range of the frequency range due to their design and that there is a relatively high attenuation of the signals in other ranges. This difference means that the choice of antenna can also limit the signals that can be received. Furthermore, antennas differ in their sensitivity; thus, weak signals can still be received with an antenna with higher sensitivity and no longer with a less sensitive one. In this area, the antenna choice primarily changes the system's range and susceptibility to unwanted extraneous signals, which can be recorded. Thus, the task of filtering the raw signals also depends on the hardware specifications of the antenna.

Another part of signal detection is the signal processor, and here there are different possibilities to process the signals and to distinguish signals from background noise. Here, signal detection benefits from not receiving too many unwanted signals through the antenna so that the range of the system stands in conflict to the precise detection of wanted signals. Here, the signal transmitters could play a significant role and encode their transmitted signals in order to be able to detect the wanted signals.

Another factor that can be mentioned is the characteristic of the antenna. There are two categories of antennas, omnidirectional and directional antennas. Here, the omnidirectional antennas are equally receptive in all directions, while the directional antennas have a direction in which they are significantly more sensitive than in other directions.

2.3 Wireless Communication

In environmental monitoring, network infrastructure could be present, or in remote areas, it is possible for no communication infrastructure being present. In both cases and under changing conditions (harsh weather, partial power outages), sensor nodes have to be able to send data and status information. Therefore, installing a wired network in remote areas is impossible, so the network has to be based on wireless technologies.

2.3.1 OSI Reference Model

The OSI reference model was developed in the 1970s and is based on seven layers. Interfaces exist between the individual layers so that they can be exchanged upwards and downwards. Thus, a protocol on one layer can be exchanged by another protocol on the same layer, since the interfaces are the same. This interchangeability means that applications can use the

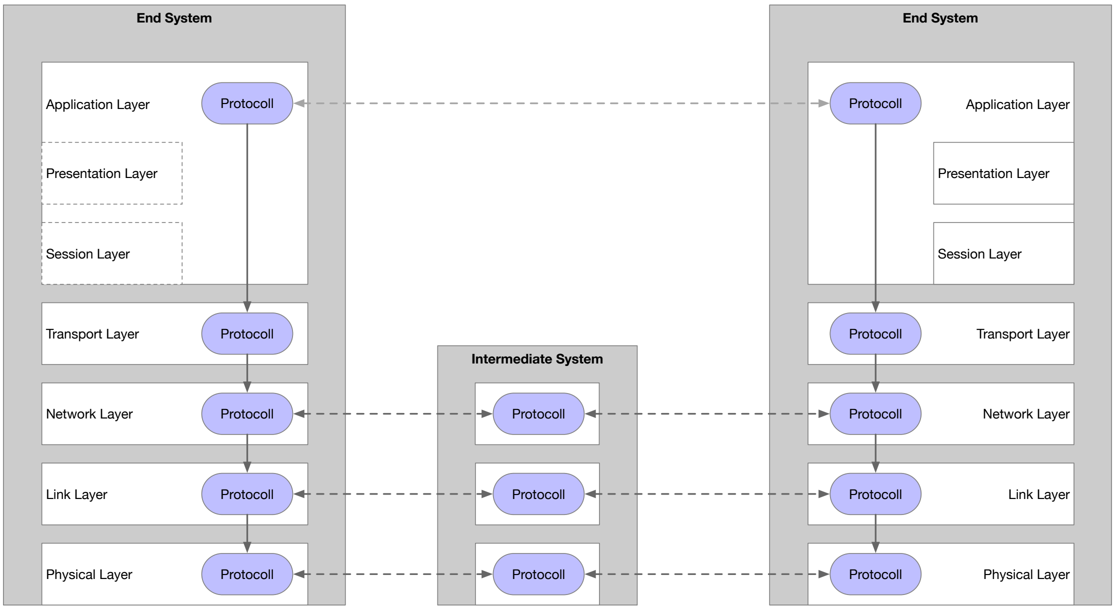


Figure 2.3: OSI reference model according to [267]

transmission technologies of the layers below, regardless of which specific technology is used.

Figure 2.3 shows the different layers of the OSI model. Here, the first layer is responsible for the physical transmission of data, and the subsequent layers build on this foundation. The data link layer acts as a logical link between different instances in the network. Packets are formed on the network layer and sent between networks. The nodes between these two networks forward or route packets. From the transport layer (layer four) onward, senders and receivers exist. A logical end-to-end transmission is established and provided to the application layer, which builds on this end-to-end communication.

2.3.2 WLAN

WLAN represents the possibility of local communication. In most cases, a method of the 802.11 standards is used. Examples are 802.11n or 802.11ac, which are often used as WLAN standards in end devices today, but other standards are also used. WLAN is part of the lower two layers of the OSI reference model, and thus WLAN can also be replaced by other methods on the first and second layers [56]. The currently widespread standards of WLAN use frequency bands around 2.4 GHz or 5 GHz, but bands around 6 GHz and 60GHz are also used, and there are isolated other bands [130]. In addition to the infrastructure mode, mainly used between terminals and routers, there is also the ad-hoc mode, which works without a fixed station between two WLAN devices [56].

2.3.3 Cellular Networks

Cellular networks are designed to deliver a high throughput link on long distances and in remote areas [66]. Current networks are often based on LTE or 5G in western countries. With this network structure as a basis, bandwidth-heavy data can be transmitted. For the use of transmitting photos or even small videos, this technology can be used, but for the traffic in most countries, significant costs arise.

2.3.4 Low-Power Wide-Area Networks

Low-power wide-area (LPWA) networks are designed to achieve the goal of enabling the transmission of small data packets over long distances [66]. Two significant representatives of LPWA are Sigfox and LoRa, but LoRa gets the broadest research interest at the moment [66]. LoRa uses an unlicensed 900 MHz Band in North America and 868 MHz in Europe [66]. For Internet-of-Things (IoT) applications, LoRa is an appealing candidate, because it delivers messages with low-energy demands over long ranges [66].

2.3.5 Satellite Links

In contrast to terrestrial stations, satellite links are based on infrastructure deployed in the orbit, so they are not equally vulnerable to disasters [66]. The Low Earth Orbit (LEO) and Geostationary Earth Orbit (GEO) links exist in satellite communication. Both can be used to achieve communication and add a new or additional link to existing networks. In most satellite networks, the data rates are low, and they are mainly used for short messages, status messages, or alert sharing. With *Starlink*, a commercial satellite network with much more throughput capability was introduced in 2020. *Starlink* can also handle streams of video, photo, or audio data in remote areas, but has significant power consumption and a significant subscription fee.

2.3.6 Disruption-tolerant Networks

To enable communication with satellites and alien robots, NASA[1] and other space agencies started the development of delay- or disruption tolerant networks (DTNs). These remote devices have few communication possibilities, primarily due to the conditions in space, and also unstable links due to interference. Real-time routing has been replaced with DTNs to allow communication by transferring data from one node to another. This scheme is called store-carry-forward and is not only used in space communication today. NASA illustrates the differences in a network with many disruptions in Figure 2.4. Here, it is shown that DTNs can deliver more data when the network is disrupted.

[1] `https://www.nasa.gov/directorates/heo/scan/engineering/technology/delay_disruption_tolerant_networking`

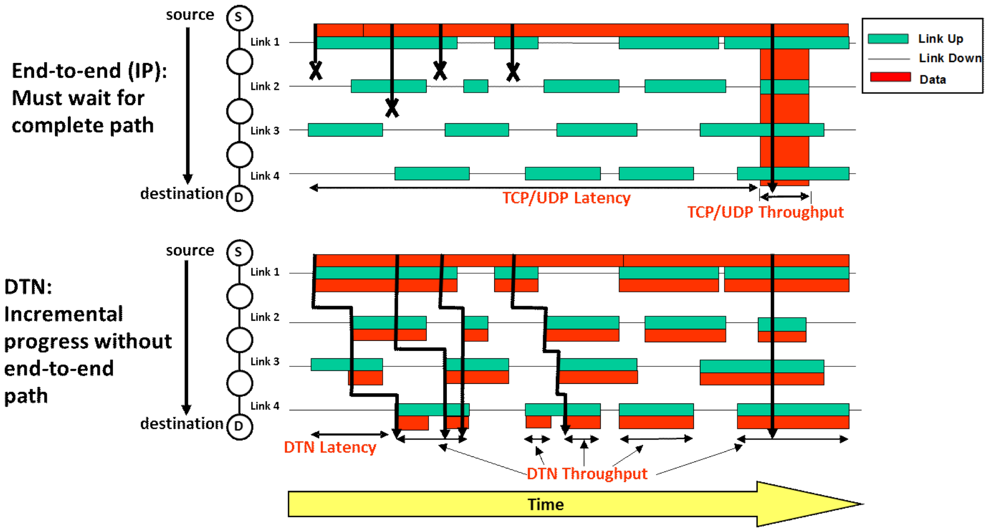


Figure 2.4: IP end-to-end dataflow vs. DTN store-and-forward dataflow as in `https://www.nasa.gov/sites/default/files/dtn_0.png`

2.4 Machine Learning

In machine learning (ML), there are several methods and algorithms to solve the problem of getting a machine to learn from data.

2.4.1 Unsupervised and Supervised Machine Learning

These algorithms can be broadly divided into unsupervised and supervised learning methods.

The main difference between unsupervised and supervised learning is the dataset [87]. In the case of unsupervised learning, the algorithm tries to find a predefined number of clusters or classes in the data [87]. This cluster finding could be done by, e.g., a neural autoencoder that gets a strict number of input neurons, maps this input to several middle neurons, and tries to produce the same result on the output neurons as on the input neurons. With this method, a researcher could make a classification or clustering. In Figure 2.5 a neural autoencoder is shown.

In contrast to this unsupervised method, the supervised approach is to get a dataset with a label for each data point [87]. With this labeled data, a model is trained to learn labels for data and, after that, can tag unseen data with these learned labels [87]. In this way, data can be classified automatically into known classes.

2.4.2 Neural Networks

Neural networks offer a way to define a nonlinear function to fit the given learning data to the provided labels [175]. A neuron is a unit of this neural network that gets input $x_1, \ldots, x_n$ and

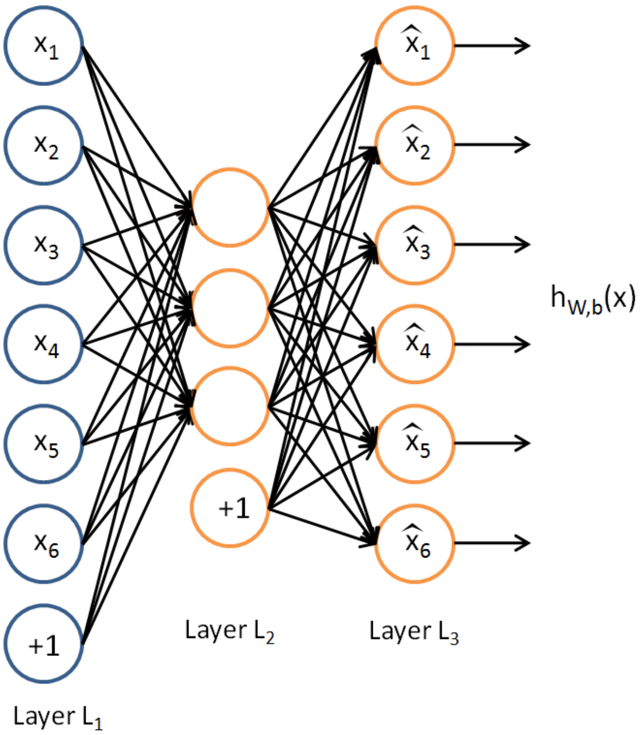


Figure 2.5: An abstract illustration of a neural autoencoder by Andrew Ng [175]

generates an output to the activation function in the neuron [175]. The resulting structure can be seen in Figure 2.6.

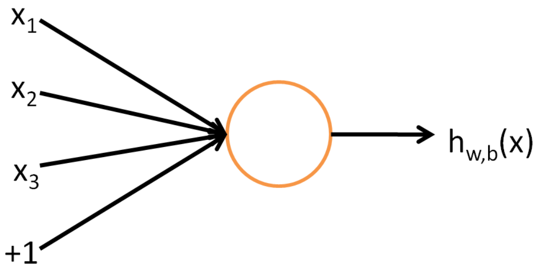


Figure 2.6: An abstract illustration of a neuron by Andrew Ng [175]

A neural network is a composition of many neurons so that the output of each neuron can potentially be the input of another neuron [175]. A small neural network is shown in Figure 2.7. The first layer of a neural network is the input layer, and the last is the output layer [175].

Backpropagation is combined with a cost function to learn the nonlinear function based on training data. Backpropagation is done in the training phase, and in case a training dataset results in wrong labels, backpropagation takes place and changes the internal state of a neuron [175]. An additional part of building a neural network is the network structure. Here, there are many possible structures; an abstract illustration is presented in Figure 2.8.

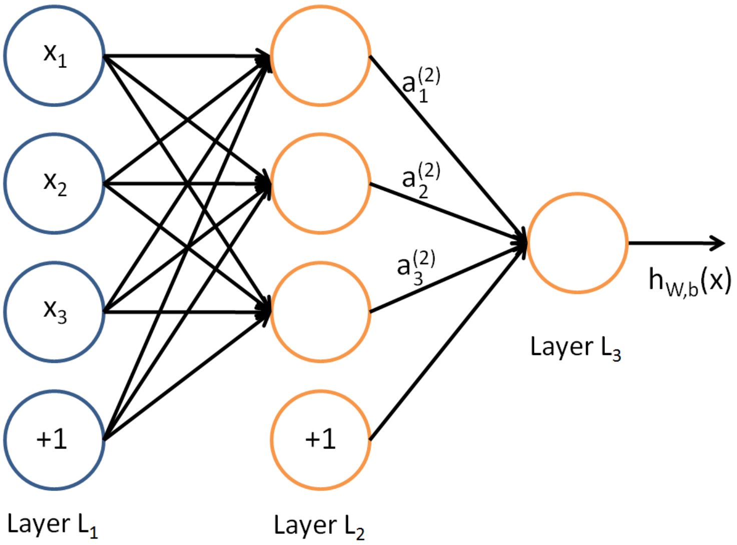


Figure 2.7: An abstract illustration of a neural network by Andrew Ng [175]

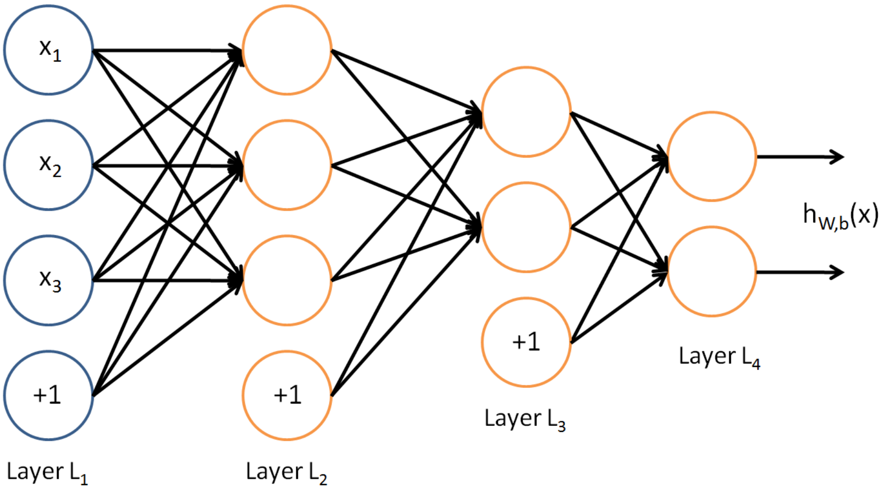


Figure 2.8: An abstract illustration of a neural network structure with more than one layer by Andrew Ng [175]

2.5 Summary

This chapter introduced terms, concepts and technologies used in this thesis. Section 2.1 explained ecological scenarios and typical methods used in this field. Also, a definition of monitoring was given. Section 2.2 discussed technologies for sensor nodes. In Section 2.3 fundamental concepts for wireless networks were described. Basic concepts and terms for machine learning were presented in Section 2.4.

3 Sensor-based Ecological Monitoring

This chapter presents challenges in the field of ecological monitoring, especially for bat monitoring, and a case study of using low-cost commercial off-the-shelf hardware components and wireless technologies to build a sensor network. As stated in Chapter 2, bats are good indicators of harmful environmental conditions. Thus, observing and understanding bat behavior plays a significant role in estimating ecosystem health and changes. As a result, spatial and timely high-resolution monitoring of bats and their actions could be a good indicator of the ecosystem's well-being. Nevertheless, humans are not suitable to perform high-resolution monitoring, as described in Section 1.1. The main reasons are that humans are time-consuming to train, are not cost-effective with fair payment, error-prone, and do not provide comparable results.

3.1 Bat Monitoring

Today, bat monitoring is mostly performed manually, especially regarding the bats' positions. Here, transmitters are stuck on the bats, and positions are taken with manual telemetry and 3-point direction finding. This process is very time-consuming, but it is currently, in most cases, the only way to assign a location to an individual for a specific time. Only some projects try to tackle the problem of handheld telemetry and published computerized VHF telemetry systems. Examples are *Atlas* [154], the *Motus Wildlife Tracking System* [231], and the predecessor of our *tRackIT* stations [80].

To decide which areas are important for bats, researchers have to check the presence of bats in different regions. Checking the presence of bats can be done with audio recordings and telemetry. Telemetry is a time-consuming method with the disadvantages that bats must be caught and equipped with a transmitter and that it harms the bats. In the case of audio recording, in the classical method, data is collected manually by people walking through a forest at night with a microphone. Alternatively, researchers deploy sensor nodes beforehand in the forest. A researcher, while walking around, documents the presence of bats in a particular area in case of audio recordings by hand. In the case of a fixed sensor box, the area is easy to determine, because the sensor node has a fixed location. Through these methods, the presence of bats in a specific area can be well documented.

However, following the data recording, the species must also be determined. The bat species can be determined manually by humans and measuring the recorded spectrograms or by analyzing the recorded audio sequences by software. Executing this process immediately after the recording would save time and detect possible recording errors on time.

However, suppose a specific individual is the target of the investigation. In this case, it is impossible with today's methods to obtain this information through audio recordings since humans cannot distinguish between different individuals in audio. The only option is to use transmitters to distinguish between the different individuals. In this case, the determination of the species is straightforward, since the animal has to be caught and thus examined in detail. Moreover, because the transmitter is attached to a specific animal, the distinction between different individuals is a technical issue in differentiating between different transmitters. This issue is well understood and can be handled on a technical level.

To study the behavior of bats and especially to show differences or changes in their behavior and thus to be able to react to this, it is necessary to include external factors. Bats depend strongly on their environment and their food supply. These external factors must be recorded to attribute a change in behavior to specific external factors. For this purpose, the question of hunting grounds, the food supply, and the condition of the entire ecosystem arises. Wind, precipitation, and concrete brightness must be mentioned here. These factors can be decisive for the behavior of bats and should, therefore, be recorded.

In moth monitoring, a lot of work has been done in the past, but mostly work using lethal moth traps. Non-lethal moth traps were only conceptually available in the last years and are not yet well tested. One exception is the publication of Bjerge et al. [17], which was the basis for our non-lethal moth trap, where we added features and robustness for the deployment in the forest.

In general, the behavior of bats in different situations should be detectable, and also by various sensors, because thus the observable behavior can be transferred from one sensor to another. For example, the simultaneous acquisition of audio and VHF data could make it possible to assign audio sequences to a specific individual. This tagged audio could be the starting point for automatically recognizing individuals in recorded audio. This automatic recognition would have the advantage that a transmitter and thus the intervention in animal welfare would no longer have to be carried out. Such a multi-sensor device needed for this type of data has not been published previously, to the best of our knowledge. To also label VHF and audio data with bat behavior, the data fusion of audio, VHF, and camera data must be achieved. This data fusion allows automatically detecting bat behavior in audio or VHF sequences. Furthermore, it could also be achieved by our project *BatRack*, which is presented in Section 5.5.

3.2 Problem Statement

To be able to answer questions in bat research, different sub-questions have to be answered, as shown in Figure 3.1. Answering these sub-questions is necessary to get a good assessment of the whole field.

Researchers must collect ecological data to find correlations with weather, microclimate, food availability, or roosting sites. By surveying moths and especially recording the number of moths over a broad local area and over time, correlations can be found with the occurrence

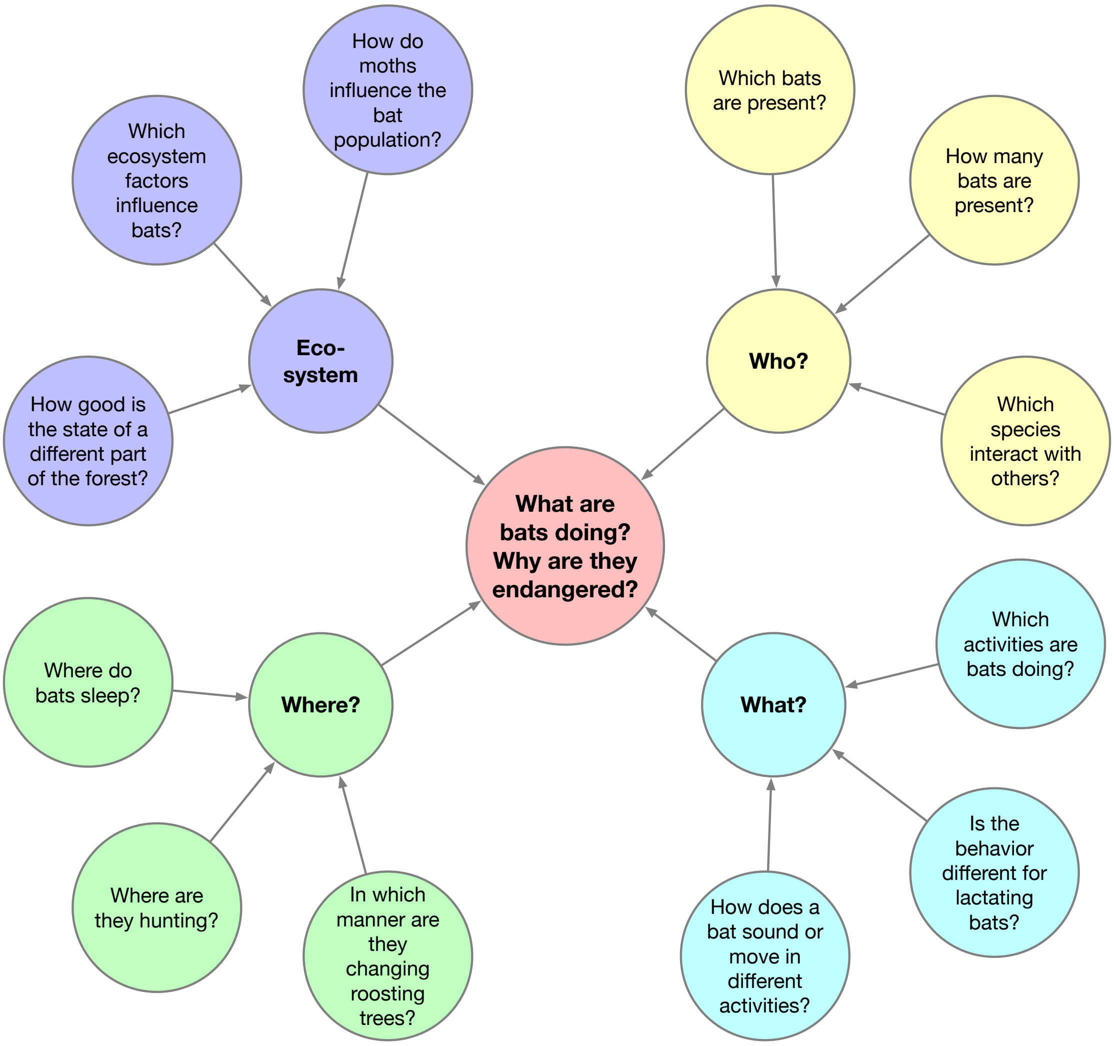

How do moths influence the bat population?
Which ecosystem factors influence bats?
How good is the state of a different part of the forest?
Eco-system
Which bats are present?
How many bats are present?
Who?
Which species interact with others?
What are bats doing? Why are they endangered?
Where do bats sleep?
Where?
Where are they hunting?
In which manner are they changing roosting trees?
Which activities are bats doing?
What?
Is the behavior different for lactating bats?
How does a bat sound or move in different activities?

Figure 3.1: Problem statements in bat monitoring

and activities of bats. By the large-scale examination of different trees, statements can be made about the condition of the forest in various parcels of the forest and, thus, about factors influencing bats in these parcels. It is also essential to know which factors influence the behavior and occurrence of bats. From the answers to these questions, it is easier to find answers to the factors and circumstances that must be present to protect bats and entire ecosystems. Bats are one of the first species to be driven out by circumstances and indicate that this ecosystem is no longer intact. Thus, all the findings can be used to keep ecosystems intact and improve their conditions.

Second, where bats are lactating or, more in general, the positions of bats over time are a field of research to get an impression of what bats are doing and where bats live. Furthermore, by investigating the positions of bats, statements can be made about their roosts and habitat. Here, the questions arise, how many different roosts does a bat colony need per year, and how often do they change them? The answers to these questions allow conclusions about the effects on the bat population. For example, the estimation of clearing forest areas is significantly influenced by the measure of the harm to the animal population. Thus, a significantly more accurate estimate of clearing effects or the impact of environmental damage due to natural events can be derived. From the nature of the different tree cavities, i.e., which type of tree and which season, conclusions can be drawn about the necessary components of forests for the continued existence of forest bats. Thus, through these answers, an estimation can be made for the effect of monocultures and environmental influences on the development of the bat population and, with that, the evolution of an ecosystem. The third question that can be answered from the positional data of bats is: How often and where are bats active? Here, different species can be compared, and the preference for other prey or hunting techniques can be derived from differences between species.

The third area for questions is: Who is present? Here, statements can be made about the occurrence of different bat species in other forest areas. This method illuminates this question more precisely and for all existing individuals. One or several individuals must be provided with a transmitter, but all current individuals can be seized. Thus, this kind of data acquisition is again less complex and more broadly possible. It is also less prone to human errors, since the quality of the data does not depend on the catch success for the transmission. However, some methodical questions still have to be answered to get a method that can distinguish individuals and thus only allows a census of bats.

It is necessary to study the actions itself to get a complete overview of the activity of bats. Likewise, an energy requirement can be derived from the activity, which must be absorbed through food. The activity of bats can determine correlations with weather and microclimate, and thus conclusions can be made about the behavior and adaptability of bats to external conditions. A method must be created to observe a bat and determine its activity. If this is successful, researchers can answer several questions. These include the question of how the movement patterns differ in different activities, e.g., whether nursing bats behave or hunt differently or in other areas than non-nursing related animals.

With the intersection of audio data and thus calls, the question of whether bat calls differ in different activities can be answered. If so, one could also predict activity by recording the calls and thus determine species, numbers, and activity by deploying a wide-area audio sensor net. Therefore, it is necessary to have a method that is non-invasive to an animal to assess its occurrence and action in specific areas. This non-invasive method could then be used to develop a simple and scalable system that would provide a consistent and detailed record of bats and, thus, a way to detect bat population declines quickly.

3.3 Challenges of Ecological Monitoring in Forests

Especial in harsh conditions, ecological monitoring is a challenging task that has to fulfill several requirements. In particular, bats live in a forest ecosystem, and a feasible automatic wireless sensor system for ecological monitoring in a forest ecosystem faces the following challenges.

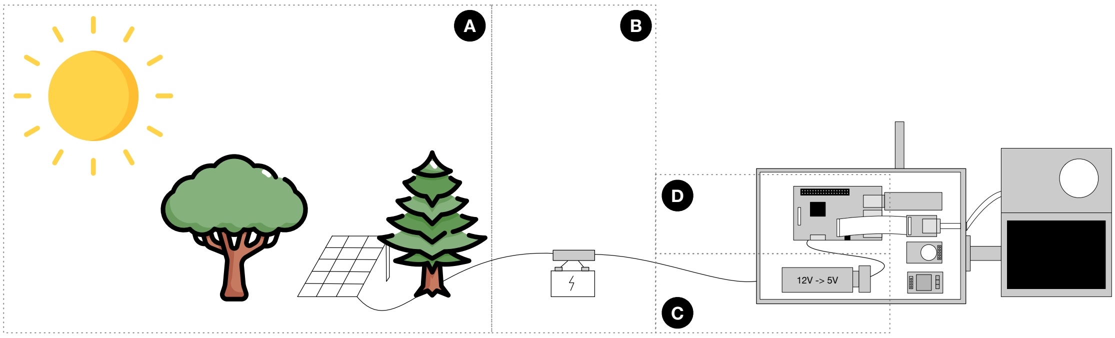


Figure 3.2: Different stages of power generation, voltage transformation, and computation, and the stages to optimize the energy system: (A) energy generation with photovoltaic modules in the forest, (B) battery and solar controller, (C) voltage transformer, (D) computation unit of the sensor node

3.3.1 Energy Constraints

In forested areas, most places do not have sufficient power supply. One reason is that these places are often remote and thus not connected to the local power grids. Another reason is that photovoltaic systems can only generate energy in forested areas with considerable effort. A researcher must find clearings in the forest, the photovoltaic panels must be correctly positioned on the clearings, and a researcher must clear them of vegetation. Furthermore, the problem arises that most commercially available partially shaded panels hardly provide electricity. In the open areas, this problem only arises in a weakened form, and the choice of the solar panel and other components moves into the background. However, other power generation components in the forest also play an essential role, so these should also be selected sensibly. Explicitly, the choice of solar controllers and voltage transformers are potential sources of energy dissipation. Only by choosing sensible components, the operating time of a sensor node can be maximized, even in a cloudy or rainy period, and the manual maintenance effort is minimized.

In contrast, to write energy-efficient software, the hardware part is, in most cases, more expensive than less energy-efficient parts. Nevertheless, continuous operation of the sensor nodes is nearly impossible without an energy-efficient hardware setup. In Figure 3.2, the different areas of energy optimization are shown.

3.3.2 Harsh Environments

A sensor node used for a long period of time is exposed to different influences compared to experiments and sensors set up manually for short periods. These sensor nodes and their accessories must be much more robust and durable than established methods with manual care. One example is the manual supervision of lethal insect traps. They are deployed on promising (non-rainy) days, attract insects during the night, and researchers collect and evaluate the results the following day. Multi-day operation is also conceivable here, but with the restriction that these traps can be emptied every morning and thus also retrieved in lousy weather. In the case of automatic sensor nodes, manual maintenance of the sensor nodes is only envisaged in the event of a fault and is therefore minimized. Otherwise, scalability could not be achieved because manual work is too time-consuming. Moreover, collecting the sensor nodes in bad weather is not foreseen.

3.3.3 Changing Conditions

The changes over time, in which the sensor node automatically takes over the ecological monitoring, are not trivially solvable. For example, the daily course of the sun changes over a year. The vegetation changes constantly, and sensor nodes and their accessories must adapt to these circumstances, or the conditions have to be manually improved for the sensor nodes. A robust solution for these problems is desirable because it reduces manual effort and allows scaling.

3.3.4 Unreliable Connectivity

Especially in forested areas, due to the changing vegetation, unreliable connections to cell towers or other nodes are not uncommon. Here, the network operator can interrupt long running connections or a different cell tower utilization could be possible. However in a self deployed network, nodes could fail in scenarios where the stored and newly generated energy is not sufficient to guarantee a stable energy supply.

3.3.5 Resource Limitations

Based on the challenges and limitations described above, the hardware must only rely on minimal resources. Here, microcontrollers, and single-board computers in particular, are a suitable way to save energy and resources but still offer plenty of options for reading out sensors, processing, and forwarding the data. These computers and controllers must be able to read out the sensors used. The reading of sensors requires GPIO pins for simple sensors and various interfaces for more complex or data-intensive sensors. For this reason, the data to be recorded often determines which sensors must be used and, thus, which computing resources must be employed. Thus, the choice of resources depends mainly on the use case and should be selected so that the data is read, processed, and forwarded with as few resources as possible.

3.4 Sensor Systems

Sensor systems can be divided into hardware components, data collection, transmission, and energy supply. In all these subareas, preliminary work or hardware can be used to solve the problems described. In the following, a categorization in the described subareas and a classification is presented to build a system for automatic environmental monitoring.

3.4.1 Components of a Sensor System

The simplest way to divide hardware components into two categories is based on their primary task: The data recording is done by the sensors and the data processing is carried out by the computing units. Due to different possibilities for data transmission, however, not all components in these two categories are compatible. For example, computing units can only read some sensors out in analog form, others only provide specific digital protocols, and still others can only be integrated via individual connectors. As a result, a researcher must select the processing unit according to the sensors needed to perform the acquisition.

As a further aspect for the processing unit, the ability to forward and store the data is required. Here, the categories of microcontrollers and single-board computers differ significantly. The single-board computers usually offer significantly more interfaces for storage, and the factory often provides some standards for transmission. In contrast, others can be upgraded, for example, via USB. However, in contrast to microcontrollers, single-board computers are usually much less energy-efficient.

Also, more or less energy-saving sensors exist. With these, the energy consumption is partly systematically conditioned, e.g., with sensors for specific cases, in addition, to the concrete conversion of the sensor and the pertinent printed circuit board. Thus, a researcher can already optimize some systems in the design phase with the choice of sensors.

3.4.2 Data Collection

To define the requirements for a scalable, automatic, user-friendly ecological monitoring system, a researcher must clarify which data a sensor node should collect. Since the data needed clearly depends on the problem, it is abstracted here and divided into data classes. The data rate is the product of temporal resolution and size per data point. For the first factor, the data classes can be divided into continuously recorded data collection and data collection according to a schedule. As a third variant, some data collections are triggered by an external event. Looking at the three different variants of data collection, a reasonably fixed data rate can be determined for the first, while with the second variant, this depends mainly on the schedule settings, and with triggered recordings, upper estimates can be made, but no concrete data rates can be determined.

In the second factor, the size of each data point can range from a simple float, int, or str, to a collection of these data types per measurement, to photos or videos. Here, the memory and transmission requirements differ significantly. Figure 3.3 shows the resulting data rates for different sensor nodes presented in this work.

3.4.3 Communication

To equip each sensor node with its uplink, we can choose disruption-tolerant networks (DTN), mesh networks, or no concrete network formation at all. An uplink in a mesh network or DTN can be provided with one or more points and different technologies. The technology used results in different transmission rates, robustness, and energy consumption of the system.

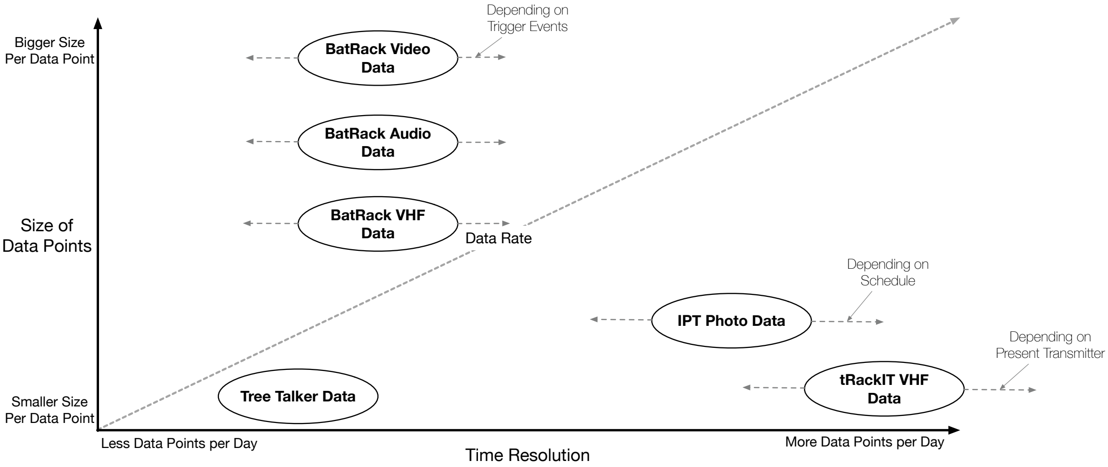


Figure 3.3: Data rates of the different sensor node projects of this thesis - categorized by the size of each data point and the time resolution of the recording

Communication is essential to enable real-time data processing and get status information from the system. Real-time processing is essential to gain knowledge of the transmitted data, evaluate the setup and results about their quality, and fix possible errors quickly. LTE, WLAN bridges or satellite links can also be used to establish communication. These solutions depend on infrastructure and can only be used in areas where this infrastructure is present.

3.4.4 Power Supply and Consumption

Regarding power supply, there are significant challenges due to varying vegetation. The whole power system offers significant opportunities to optimize the power supply. As demonstrated in Section 3.3, photovoltaic power is a suitable power source, but has some challenges, which can partially be addressed with commercially available technology. One example are hotspot-free solar panels, which provide significantly higher power output than conventional solar panels in partially shaded conditions. Maximum Power Point Tracking (MPPT) solar chargers can increase the battery's charging efficiency compared to Pulse Width Modulation (PWM) chargers. The researcher should also adjust the battery size of the sensor system so that the system can survive periods of lousy weather without failure. Furthermore, the researcher should check the efficiency of the components used in the sensor node, and they should use more efficient parts if possible.

3.5 Networked Sensor Systems for Bat Monitoring

Section 3.2 raises questions for the research of bats and thus for predicting the state of the whole ecosystem. These can partly be answered with new methods presented in this thesis. For this purpose, an assignment of solutions to questions will be done in the following section. The assignment of the projects to the individual problems is shown in Figure 3.4.

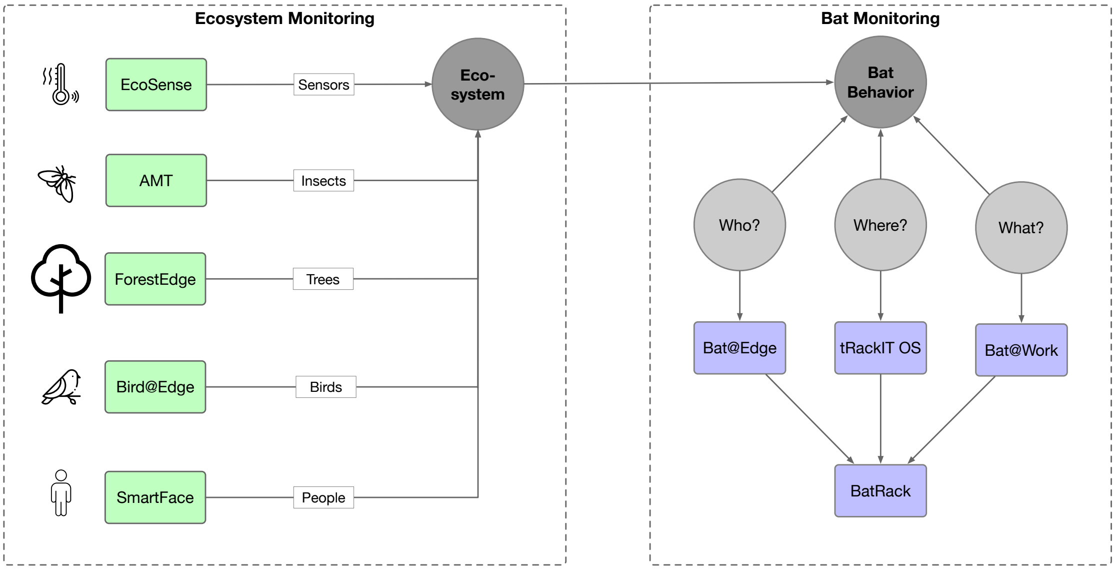


Figure 3.4: Problems with matching publications

The work can be divided into two parts. The first part addresses issues in monitoring the ecosystem in which bats live. This task can be split into sub-questions about people, trees, birds, insects, and sensors. For these sub-questions, Chapter 4 highlights publications in this area. The second part is bat monitoring, which can be split into the sub-questions of "who?", "where?" and "what?". These sub-questions are answered by the publications in Chapter 5.

3.5.1 Ecosystem Monitoring

The publications presented in Section 4 answer sub-questions of Chapter 3.2. The *EcoSense* project provides the basis and an overview of various sensors and their power consumption and presents a structure for disseminating data. In this project, the questions about environmental factors and how to measure them are described in a scalable way. In this context, scalability stands for the possibility of autonomous operation and the lowest possible costs in both acquisition and operation, which is why the energy consumption of the presented sensor nodes is also determined.

Another project is the *Automated Moth Trap (AMT)*, which automates the detection of moths and is thus a food source for bats. As with *EcoSense*, the *AMT* also focuses on low-cost components and operations. For this purpose, domain knowledge enables the most data and energy-saving operation possible by employing intelligent scheduling adapted to the use case.

As a third project, the concept of *Unobtrusive Mechanism Interception* is presented to equip existing systems with new functions. The method of *Unobtrusive Mechanism Interception* is then used in the *ForestEdge* project. Thus a possibility is created to provide the proprietary system of the *TreeTalker* with variable measuring intervals. These can depend now on, e.g., the battery condition and prevent a data gap. Furthermore, the incoming data is checked for plausibility, and the sensor node is instructed to repeat the measurement if an outlier is detected.

Fourth, the project *Bird@Edge* is presented. It provides an edge computing infrastructure to couple several network-capable microphones and an edge processing device. This infrastructure enables the microphones themselves to be manufactured relatively inexpensively and also to be relatively energy efficient. These microphones send the received data to the edge device, which processes the data in real time and relays the results. Thus, bird species lists can be made available to researchers immediately, and they can work with the results.

Finally, humans are part of the ecosystem, and the possibility of their presence has to be addressed. The camera traps for bats from *BatRack* are in public places. Therefore, human faces must be excluded from the camera images since no personal data may be collected. For this problem, the results of the *SmartFace* project are used since it enables energy-efficient recognition of faces on end devices. Thus we can immediately delete these unwanted recordings. Afterward, we can log the occurrence of these events to analyze the impact of human presence on bat behavior.

To summarize, these projects can answer the questions raised in Section 3.2 for the ecosystem part and thus provide answers for this subarea, as shown in Figure 3.4.

3.5.2 Bat Monitoring

The publications presented in Chapter 5 answer questions about bat monitoring.

The first project contributes to how audio can be used to recognize bat calls. It introduces a neural network to detect and classify them into different species and is tightly coupled to the *Bat@Edge* project.

Bat@Edge provides an edge computing infrastructure to couple several network-capable microphones and an edge processing device. In combination with the neural network mentioned above, the questions from the "Who?" area shown in Figure 3.4 can be answered. Furthermore, the question of how many bats are in this area can be estimated if bat individuals can be distinguished in the audio data.

tRackIT OS is a solution to the scalable acquisition of a bat position and the low-maintenance provision of this service. Through the combination of error-aware software and real-time data processing, this system helps answer the question in the subarea "Where?" in Figure 3.4.

The project *Bat@Work* answers questions about "What?". In this project, the signals of bats with attached transmitters are divided into the classes of active and passive. The different behaviors of two other bat species are presented through a study. Through these findings, conclusions can be drawn about the hunting behavior of the different species, and the project also provides answers to make bat monitoring more efficient in the future. If a researcher wants to study a specific species in the future, the researcher can use this data to adapt the technology to the monitoring results. This result means that the technical devices no longer have to be active the entire night but can be used in a data and power-saving manner.

With all the projects in mind, *BatRack*, which links several sensors, is presented. The sensors used are an ultrasonic microphone, an infrared spotlight with an infrared camera, and a VHF receiver. Thus, the bat calls can be recorded, while the camera can determine their activity. Additionally, the VHF receiver can assign these data to those of a bat with a transmitter. With this new device, tightly coupled types of data can be recorded. For example, we can observe a specific activity with the camera and tag the audio and VHF data with this activity for further use.

3.6 Summary

This chapter introduced the topic of bat monitoring and problems arising in this field of work. The research field can be divided into the following sub-questions:

- What does the ecosystem do?
- Who is acting?
- What action is performed?
- Where does the action take place?

The "who/what/where" questions were answered explicitly for bats. Networked sensor systems can answer these sub-questions. However, these sensor systems are still confronted with use-case-specific challenges and environmental disturbances, which are indeed accelerated by climate change itself. The most challenging part, monitoring in forested areas, was selected. To overcome these challenges, solutions for components for networked sensor systems, data collection, communication, and power supply were presented.

Furthermore, an overview of the contribution of networked sensor systems for bat monitoring was given.

4 Ecosystem Monitoring

This chapter contains research results on the topic of ecosystem monitoring. Monitoring the ecosystem is relevant for bats to understand their behavior.

First, an approach is presented to build sensor nodes with COTS components that can measure environmental parameters in a cost-efficient and energy-efficient manner. This approach appeared in publication [13]. Here, the main focus is on comparing sensors with similar functionality and investigating automatic data processing for data reduction. The results show that low-cost and intelligent sensor nodes can be built, new data sources can be used, or already acquirable data can be recorded in a new way.

The second publication [165] presents a non-lethal moth photo trap. Preliminary work exists in lethal moth traps that carry the disadvantage of killing the insects and potentially decimating the population. Furthermore, some publications exist in non-lethal photo traps that were not used in the context of a field study but existed as prototypical developments only for testing. The presented automated non-lethal moth trap (AMT) implements a schedule for data reduction and operates in the field with a power supply for a permanent timed recording in possible harsh environments.

With *Unobtrusive Mechanism Interception*, an approach is presented to equip existing sensor nodes with new functionality without needing to change or replace proprietary hardware. As a result, publication [141] contributes to the scalability of the approach since, in environmental monitoring projects, existing hardware can be further used. Thus, the costs for the conversion to a networked wireless sensor network are reduced. In a further publication [140], this approach is applied to the used sensor nodes. The result is called *ForestEdge*, indicating that *unobtrusive mechanism interception* can be applied to particular sensor systems. In addition, the approach can add some functions to the proprietary hardware, and the monitoring results can be improved by outlier detection. For this purpose, *ForestEdge* was investigated in a grey box manner. The communication via LoRa was reverse-engineered and mimicked to answer and give commands to the proprietary hardware correctly.

As a further publication [103], *Bird@Edge* is presented. Here, a neural network was ported to an edge platform and used for live analysis of bird calls in a forest. It is shown how processing at the edge leads to a significant data rate reduction. This processing increases the transmission chance and decreases the costs for the transmission because especially in remote areas, the costs for higher bandwidths increase. This approach also shows the ability to track the abundance of birds in the monitored area automatically. Furthermore, bird abundance can be correlated with bat abundance or bat activity.

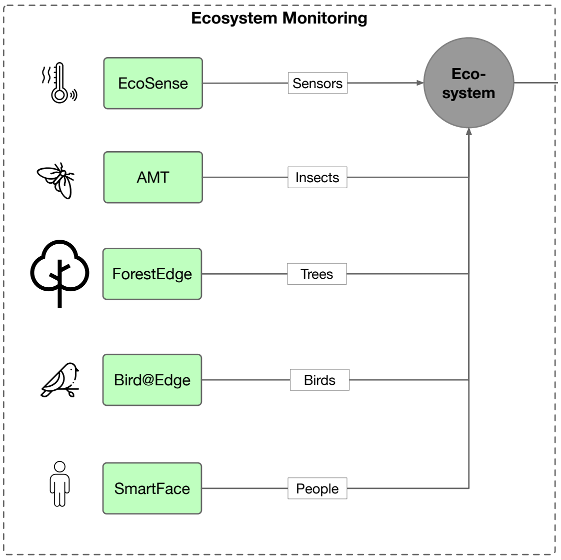


Figure 4.1: Publications on ecosystem monitoring

The last publication [139] in this chapter presents an energy-saving approach for recognizing faces in images on low-power mobile devices. This approach can reduce data by not transmitting images of faces from camera traps. In addition, it also has the advantage of recording sensor data in a way that complies with data protection regulations, since recording personal data without consent is not readily permitted in the EU or is at least controversial. Furthermore, the approach allows us to get an impression of when humans are present in the research area and to store these events to check for signs of disturbance of humans for bats or other animals.

In Figure 4.1, the publications in this chapter that deal with monitoring the ecosystem are displayed. They address the challenges discussed in Section 3.3. In particular, robust and cost-effective systems are presented, which allows scaling these approaches to an area-wide network. Furthermore, contributions to the topics discussed in Section 3.4 are made, sensors are compared, communication paths are shown, and the power supply in the forest is optimized. In addition, the contributions are designed to work in harsh environments. Last but not least, the research contributes to an open networked sensor system to monitor our environment and especially to monitor bats and their activities. It also can be used in different use cases, not only for bat research.

The non-lethal moth trap[1], ForestEdge[2], and Bird@Edge[3] are available via GitHub under open-source licenses. For EcoSense, the components developed for our setups, such as the rf95 modem firmware, the BLE modem, Serval LBARD support, and the software customized for our Raspbian distribution, are all released as open-source software on GitHub.

Parts of this chapter have been published previously [13, 103, 139–141, 165]

[1] https://github.com/Nature40/InsectPhotoTrapOS
[2] https://github.com/umr-ds/TTtalker
[3] https://github.com/umr-ds/BirdEdge

4.1 EcoSense: Environmental Monitoring using Low-Cost Hardware and Infrastructureless Wireless Communication

4.1.1 Introduction

Environmental monitoring without being able to rely on existing electricity and communication infrastructures is required in several use cases, such as biological and ecological studies (e.g., animal tracking, air quality measurement). In these situations, several constraints need to be considered when a technical solution for environmental monitoring is designed: lack of power supply, lack of communication infrastructures, harsh weather, low maintenance possibilities, weight restrictions for animal-attached sensors, availability of scenario-specific sensors, and cost factors.

There are several computing platforms readily available that can be extended to provide stationary services as well as mobile, low power tracking devices. Furthermore, there are many affordable, off-the-shelf sensors that are potentially useful in such scenarios, but proper information on how to integrate them and their specific requirements, such as real world energy consumption, are often not easily accessible. Moreover, integrating the computing power, flexibility and mobility of smartphones and tablets is an interesting option. Since satellite communication is expensive and limited, and cellular services are often not available in remote places, low-cost long-range (up to 16 km) radio technologies, such as LoRa, are quite useful. Although they are mostly associated with IoT applications in an infrastructure-based LoRaWAN mode, they can also be used for device-to-device communication and have the benefit of license-free frequency bands in any country of the world. Due to the low bandwidth of these radio transceivers, only small pieces of data can be transmitted, increasing the need to preprocess data prior to long-range transmissions.

In this section, we present a flexible and affordable sensor, computation, and communication platform for environmental monitoring that relies on low-cost hardware and infrastructureless communication. It uses delay-/disruption-tolerant networking (DTN) for non-time critical tasks over different wireless links, provides on-device data processing capabilities based on machine learning methods, and integrates mobile sensor nodes, static devices, and smartphones. In particular, we make the following contributions:

- We present a novel sensor, computation, and communication platform for static and mobile environmental monitoring setups, and for users with mobile devices.
- We present a novel energy-efficient approach for on-device image classification.
- We present a novel approach to provide long range communication capabilities for Bluetooth-enabled devices.
- We present experimental evaluations of (a) commonly available and affordable platforms, (b) the power consumption of several sensors, and (c) the real world communication ranges and power demands of various radio link technologies for environmental monitoring.

Parts of this section have been published in Lars Baumgärtner, Alvar Penning, Patrick Lampe, Björn Richerzhagen, Ralf Steinmetz, and Bernd Freisleben. "Environmental Monitoring using Low-Cost Hardware and Infrastructureless Wireless Communication." In: *IEEE Global Humanitarian Technology Conference (GHTC)*. IEEE. 2018, pp. 1–8.

4.1.2 Related Work

Several wireless sensor platforms for environmental monitoring were presented in the literature, but they are often either based on specific technologies or tailored to dedicated use cases. For example, the OpenSense project aims to bring community-driven environmental monitoring to life with an open platform and accessible results [1]. However, it is specifically designed for air pollution monitoring, relies on existing communication infrastructures and hardware specifically built for this use case. On the other hand, it integrates static sensor nodes as well as mobile nodes, and most recently also wearables [152]. Similarly, Citi-Sense-MOB [28] and AirSenseEUR[4] [76] work for air quality measurements in urban environments, but also rely on technologies such as GRPS for data transmission, constant power supply (e.g., from the bus or car the sensor is mounted on), and custom/closed-source printed circuit boards (PCBs).

Some custom-built platforms can be adopted to various scenarios. Problems associated with custom-built platforms are their cost and availability; sometimes they are specifically tailored to a particular geographic region [144, 217].

Llamas et al. [149] use Arduino and Raspberry Pi computers for building a reusable open sensor platform. Their application is human gait identification. Long range communication or infrastructureless operation are not considered.

The senseBox project[5] provides a sensor platform for citizen science projects [259]. There is a strong focus on education and publicly available sensor data, but senseBox lacks flexibility when used for more than just raw sensor data aggregation and also relies on existing communication infrastructures. Luftdaten[6] [70] also uses cheap microcontroller units (MCUs) configured for air quality measurement in a citizen science project, but also relies on existing communication infrastructures.

The EU project WAZIUP tries to bring IoT technologies to developing countries [189]. WAZIUP provides a low-cost communication infrastructure (LoRaWAN), while we try to work without any infrastructure on a purely peer-to-peer basis. We incorporate DTN technologies and feature heterogeneous radio-link setups. Furthermore, conserving energy plays a more vital role in our use cases, and we focus on providing a complete system for remote sensing and communication.

Some proposals use mobile devices or wearables as sensors, and some approaches provide low-cost, long-range radios usable from smartphones, but they are often very impractical (a direct USB connection to a smartphone is required) or require operating system modifications [156].

[4] https://airsenseur.org/website/

[5] https://www.sensebox.de

[6] https://www.luftdaten.info

4.1.3 Sensor Platforms for Environmental Monitoring

Static Sensor Platforms (SSPs) are the backbone of our environmental monitoring platform. They have less restrictions on size or weight than Mobile Sensor Platforms (MSPs), and energy problems can be handled easier by adding solar panels, wind turbines, or larger batteries. SSPs can be used for relaying packets, collecting sensor data on roof-tops, on trees (e.g., wildlife cameras), or on smart street lamps. MSPs, on the other hand, are used to tag animals and, therefore, must be kept as light and small as possible. Energy is a more valuable resource. Since humans with smartphones and tablets can also act as sensors, these must be integrated as well. Adding long-range infrastructureless communication to mobile devices also has the benefit that it can be used in case of an emergency for sending messages to other participants. Finally, it is necessary to preprocess the gathered sensor data to reduce the needed bandwidth for transmission. This is challenging when using power-efficient devices as a basis for SSPs. The different sensor platforms are discussed in more detail below.

Static Sensor Platforms

SSPs can easily utilize different energy sources, such as solar panels, car batteries, or electrical wall outlets. Therefore, they can be built using regular single board computers (SBCs), such as Raspberry Pi 3 or Zero, since these platforms offer a compromise between computational power and low energy consumption. To also function as a network hub for mobile users and MSPs, different radio link technologies can be incorporated - Bluetooth, WiFi mesh, and for longer range, license-free, low-bandwidth communication via LoRa. Since energy consumption of SSPs is not as critical as in MSPs, SSPs can constantly listen on the different wireless interfaces and act as hubs or relays for smaller mobile nodes passing by. For data dissemination, DTN technologies such as Serval[7] can be used, since they perform well in infrastructureless environments [9]. A variety of sensors can be added, e.g.:

- camera, night-vision camera, thermal camera
- microphone
- GPS/GLONASS
- temperature, humidity, barometric pressure
- air quality
- rain- and soil-moisture sensors

Usually, these are connected directly through GPIO pins or various bus systems (e.g., serial, SPI, i2c). Most SBCs directly provide these interfaces, only analog-digital pins are often lacking, but can easily be added externally.

[7] https://github.com/servalproject/serval-dna

Mobile Sensor Platforms

MSPs have more constraints than SSPs, since they must be as light and small as possible to be attached to animals or additionally mounted on an Unmanned Ground or Aerial Vehicles (UGV/UAV) [11]. Apart from the weight of the total system, power consumption is the main bottleneck. Due to these limitations, the choice of radio link technologies and the types of sensors are quite restricted. MSPs are based on small MCUs, since they require much less power and often provide deep sleep capabilities as well as various I/O pins for digital and analog sensor inputs.

Long-range Radio Links for Mobile Devices

In remote areas, cellular coverage or WiFi access points for communication using standard smartphones are often not available. Building custom radio links into phones is quite expensive and economically not interesting for large carriers. Therefore, we need a different approach with the following requirements:

1. it should work with any operating system,
2. it should be compatible with any mobile phone,
3. it should be license-free "worldwide",
4. it should provide infrastructureless long-range (>1 km) communication,
5. it should be energy-efficient,
6. it should be affordable.

The best solution for items 1 and 2 is to rely on platform-agnostic standards such as Bluetooth Low Energy (BLE) that can be used from Apple iOS and Google Android, or almost any other operating system. To cover items 3 and 4, we use the LoRa standard that in theory provides up to 16 km of range and is available in different frequency bands (433/868/915 MHz) for worldwide usage. LoRa also offers the LoRaWAN standard as a higher layer with built-in encryption, but it also requires some infrastructure, making it unsuitable for our task. Energy efficiency (item 5) is partly achieved by using BLE, but also by using small MCUs to handle communication. By finding a suitable MCU with built-in Bluetooth and LoRa chipsets, the price for such a smartphone add-on is also kept low. Alternatively, an SBCs equipped with a LoRa modem can be paired using Bluetooth with a smartphone. Even though power consumption will be higher, this system has the benefit that software such as Serval can run directly on the SBC and expose only UI relevant functions via BLE, preserving smartphone energy due to lesser notifications and wake-ups.

On-Device Data Processing

Since long-range links over LoRa only provide a few kilobits per second of bandwidth, transmitting large amounts of data in its raw form is not feasible. Therefore, processing the sensor data at least partially on the SSP to filter out unwanted content or already produce analysis

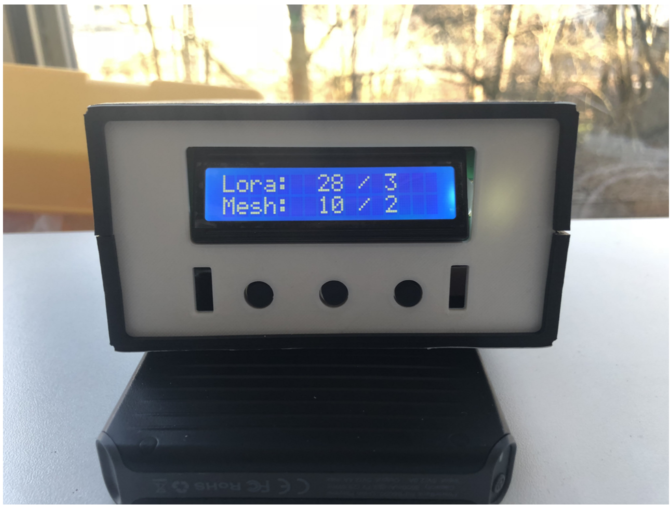


Figure 4.2: Base station for monitoring, relaying and processing.

results is favorable. In our use cases, we heavily rely on visual object/concept detection in images to filter out relevant information. Our previous work has shown how this approach can significantly reduce the amount of data necessary to be transmitted [139]. To cope with the limited CPU power of most SBCs, we integrate external hardware to accelerate the visual classification process. The challenge is to limit the power consumed by the accelerator device when it is currently not in use.

4.1.4 Implementation

We now discuss implementation details of our platforms.

Static Sensor Platform

The SSP can act as a relay for MSPs and mobile devices carried by users. Furthermore, it can be used without any sensors as a base station and uplink to the Internet (see Fig. 4.2). To provide these features, we focused on ARM-based SBCs, more specifically the Raspberry Pi family. These devices are readily available worldwide, are proven technology made specifically for tinkerers and well documented. There are several cameras that can be directly attached, plus many (p)HATs with additional hardware and many GPIO pins for direct sensor attachment. Depending on particular requirements (size, energy etc.), there are different models available, such as the Pi Zero W and the Pi 3. Many wildlife cameras and weather stations have already been built on this proven hardware platform, and Raspbian provides a solid and familiar Linux foundation. We modified a Raspbian OS Image to provide features specific to our SSP. It can easily setup a mesh network, provide Bluetooth debugging capabilities, open an access point, and start services, such as GPS logging and serval-dna. Also, the energy consumption can be optimized by deactivating unused features, such as the HDMI port of the Raspberry. All this can be configured through a simple text file on the small FAT32 partition on the SD card[8]. This has the benefit that the default image can be written to a new SD card and then can be configured from any computer with a simple text editor prior to actually booting the system. There is no need for Linux knowledge or extra drivers to read the filesystem, all is kept in one file, in one place for ease of use. The single biggest feature missing in the Raspberry Pi family is proper power management and deep sleep, but there are several after-market solutions[9][10] offering this functionality. Another benefit of the newer Pi's (Zero W(H) and 3) is that Wi-Fi and Bluetooth are already present. The only major interface left to be added for our system is a LoRa transceiver for long-range communication.

LoRa Radio Modem

There are several MCUs on the market using Cortex M0, ATmega32u4, or ESP32 chips with onboard rfm95 transceiver modules. Using the RadioHead library[11], these can easily be used in the Arduino programming environment to send packet data from device to device. We developed a firmware to expose this functionality via an *AT* command set over the built-in USB-Serial, similar to the one found in classic dial-up modems. Therefore, any device with an USB port can use such a modem, and no special drivers are needed. The source of the modem firmware can be found on github[12]. Furthermore, Serval, our DTN middleware, was made aware of this communication channel by changing LBARD and writing a driver for our firmware[13].

Mobile Sensor Platform

Since the MSP should be as light and small as possible, we focused on MCUs with integrated Wi-Fi and/or LoRa transceivers. For convenience, performance and portability, all of our programming was done using the Arduino programming environment in contrast to ESP's IDF or MicroPython. Hence, only minimal changes were necessary to switch from one MCU to another. To preserve battery power, the mobile sensors are mostly kept in deep sleep until a timeout or external trigger wakes them up. Also, they are programmed as send-only devices, since relaying data would require constant listening and would prevent deep sleep, thus draining the battery faster.

Long-range Radio Links for Mobile Devices

Since SBCs, such as the Raspberry Pi 3 and Pi Zero W, provide Bluetooth capabilities and can control devices with our radio modem firmware, such a setup can easily be used as a bridge for smartphones. For a simple prototype, we used the bleno framework[14] to expose system functionality via BLE. This is also useful for debugging head-less systems. Therefore, we provide

[8] https://github.com/buschfunkproject/heckenschere
[9] https://spellfoundry.com/product/sleepy-pi-2/
[10] http://www.uugear.com/witty-pi-realtime-clock-power-management-for-raspberry-pi/
[11] http://www.airspayce.com/mikem/arduino/RadioHead/
[12] https://github.com/gh0st42/rf95modem
[13] https://github.com/gh0st42/lbard-ng
[14] https://github.com/noble/bleno

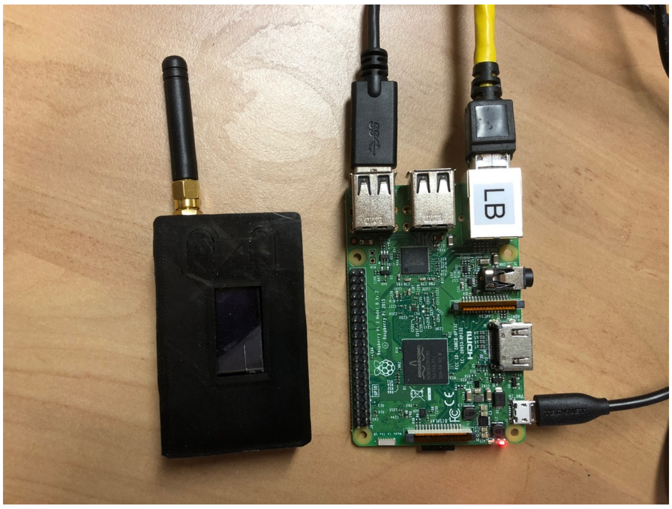


Figure 4.3: BLE LoRa modem with plain Raspberry Pi 3 for size comparison.

a simple echo service to broadcast any information via BLE[15] from our customized Raspbian distribution. Despite the flexibility this setup provides, it consumes quite a lot of power and is rather large in size (Pi Zero + LoRa modem + battery pack + antenna).

For a more lightweight solution, we enhanced the radio modem firmware to directly use BLE as a serial UART replacement. The packet size restrictions of LoRa and the BLE implementation in most RF95-based chipsets are quite similar. There are ESP32-based MCUs, such as the ones from Heltec and TTGO, that provide WiFi, Bluetooth and LoRa on a single board. The resulting system can be paired with any BLE capable (mobile) device, requires much less power, and for the 868/915 MHz bands, a rather short antenna. A modified version of the firmware can also be found online[16]. This system is rather small and comparable to commercial products, such as the GoTenna[17], but more open and flexible. The BLE LoRa modem based on a TTGO ESP32 chip, including a small 750 mAh LiPo, a custom 3D printed case, and a 868 MHz antenna is shown in Fig. 4.3 right next to a Raspberry Pi 3 for a size reference.

On-Device Data Processing

For data processing on SBCs, we implemented a solution with two execution options. The first one uses the device CPU itself, and the second one utilizes an Intel® Movidius™ Neural Compute Stick (NCS). Visual object/concept detection is used for image processing, and for this purpose we use two neural network models. The first one is InceptionNet v3 [227] with 1001 different detectable classes, and the second one is also an InceptionNet v3 model trained on the Open Images Dataset[18]. This second neural network can detect 6012 different classes and is suitable for a more detailed analysis of the given images.

[15] https://github.com/buschfunkproject/bledebug
[16] https://github.com/gh0st42/rf95modem/tree/ble_modem
[17] https://www.gotenna.com
[18] https://storage.googleapis.com/openimages/web/index.html

These neural network models are executed either with Tensorflow 1.1.0 built for ARM CPUs or on the Movidius compute stick. Additionally, the neural networks have to be converted with the Movidius compiler *mvNCCompile* to run the pretrained versions on the compute stick.

Table 4.1: Overview of SBC and MCU Platforms

	Processor	**RAM**	**Network**	**Idle** - Power in mW		**Load** - Power in mW	
				Mean	Max	Mean	Max
Pi 1 Model B+	1x 700 MHz	512 MB	Eth.	931	1288	1043	1418
Pi 3 Model B	4x 1200 MHz	1024 MB	Eth., WiFi, ᛒ 4.1	1107	1845	2224	3285
Pi Zero W	1x 1000 MHz	512 MB	WiFi, ᛒ 4.1	415	702	662	1057
ESP32, TTGO	1x 240 MHz	520 KB	WiFi, ᛒ 4.2	366	427		
ESP8266, Feather Huzzah	1x 80 MHz	80 KB	WiFi	426	1321		
Feather M0	1x 48 MHz	32 KB	Optionally WiFi, LoRa	25	29		
Feather 32u4	1x 8 MHz	2 KB	Optionally LoRa	24	27		
Waspmote	1x 14.74 MHz	8 KB		145	150		

During our evaluation we found that the Intel Movidius NCS consumes an average power of 1955 mW when it is idle. On a Raspberry Pi, we can disable the entire USB subsystem, but in this case the Ethernet and all other USB ports are also disabled. Since various sensors and potential extra radio link interfaces might be connected over USB, this behavior is not desirable. Disabling only the specific port of the NCS to save energy when no processing has to be done is more favorable. T o achieve this, hub-ctrl[19] is used to turn off the power of the compute stick in case it is currently not needed. This preserves a lot of energy, since the compute stick is expected to have more idle times than compute times. The possibility to have the compute stick as a backup without an energy penalty and only activating it when really needed makes it even more useful.

The visual object/concept detection software itself was written in Python 2.7 and outputs the five main visual concepts for each given image for further processing or discarding of irrelevant images. Thus, the energy for transmitting irrelevant images can be saved. Especially in a multi-hop scenario, a large amount of energy can be saved by applying preprocessing functions to recorded image data [83].

4.1.5 Experimental Evaluation

We performed several experimental evaluations regarding power consumption, transmission ranges, and execution times. We used the Monsoon Power Monitor for power measurements. All energy tests were repeated 2-3 times and ran for about one minute each, resulting in plenty of time to get a realistic power reading. Only the stress and compute stick evaluations were not terminated by time but by the given task.

[19] https://github.com/codazoda/hub-ctrl.c

Table 4.2: Overview of different Sensors

	Function	**Power** in mW	
		Mean	Max
MAX30105	Particle	19	145
CCS811	VOC, TVOC, eCO2	52	186
PIR	PIR	51	55
SIM28 GPS	GPS	250	302
BME280	Temp., Baro., Humidity	56	65
MICS-2714	NO2	42	45
MICS-5524	CO	41	60
AMG8833	IR thermal	179	2248
Envirophat	Temp., Baro., Color, Accel., Magnetometer	180	1868
Pi Camera NV	Night Vision Camera	1173	2290
Pi Camera v2.1	Camera	393	2231

Platform Comparisons

We evaluated Raspberry Pi SBCs as well as MCUs from different vendors for their idle power consumption and under full load. Other chipset-specific features such as RAM, onboard networking capabilities etc. are also listed in Table 4.1. When looking at the different Raspberry Pi models, it is evident that when idling, the Pi Zero W (415 mW) consumes less than half of what a Pi 1 (931 mW) needs and just about a third of what the base Pi 3 Model B (1107 mW) needs. The Pi 3 Model B+ was not tested, but due to its increased CPU power it is expected to consume even more power. Both Pi 1 and Pi 3 offer more USB ports onboard and Ethernet. T he Pi 3 offers more processing power and cores than the relatively power-hungry Pi 1 that provides even less processing power than the Pi Zero W. Under load, the Pi Zero W still consumes less power than a Pi 1 idle. Due to its 4 cores and high clock speed, the Pi 3 uses 2224 mW under load and peaks to 3285 mW at certain times. Thus, if saving power is the priority, then the Pi Zero W is the best choice. If multiple CPU cores, onboard Ethernet, and more USB ports are essential, then the Pi 3 Model B is the best candidate.

For our MSP we evaluated several MCU candidates, as shown in the lower part of the table. Some of them like the ESP32 board from TTGO are much more powerful with 240 MHz and 520 KB RAM compared to systems like the Waspmote with 14.74 MHz and 8 KB RAM. Not all systems provide deep sleep functionality by default. Even though the Waspmote or Feather M0 system consume less power than the ESP32, they also lack the radio link interfaces. Due to the flexibility through the included radio interfaces, cost of the board and provided CPU/RAM as well as deep sleep capabilities and I/O pins, we mainly built our MSPs on ESP32 boards. Should RAM not be an issue and a single radio-link interface be sufficient, the Feather M0 or even the low-power 32u4 are good alternatives with minimal power requirements.

Evaluation of Sensor Requirements

We measured the typical power consumption of different sensor types, as shown in Table 4.2. The sensors range from simple temperature and particle sensors over GPS receivers to complex optical systems such as (night vision) cameras.

Most of these sensors consume an average power of about 50 mW or less. A notable exception is GPS with about 250 mW and the cameras with 393 mW / 1173 mW. It is also noteworthy that the night vision camera draws 3 times as much power as the regular one. A simple 8x8 thermal imaging sensor, such as the AMG8833, on the other hand, only consumes an average of 179 mW. When adding these sensors to an MSP or SSP, we must also consider the maximum peaks. These can go up to 2290 mW for the optical sensors.

Radio Link Comparisons

We evaluated the power consumed by different MCUs with onboard radio transceivers, ranging from Wi-Fi to LoRa, as shown in Table 4.3. This table shows the power consumption from the MCUs with included radio transceivers. It is obvious that Wi-Fi communication is at least twice as expensive as LoRa (TTGO ESP32), and in case of the Feather M0, about 10 times as expensive. There is also a bigger difference regarding LoRa power consumption when comparing both Feather devices (ca 46 mW) to the TTGO ESP32 (235 mW). Furthermore, sending via LoRa is much more expensive at roughly 528 to 816 mW. If more RAM and CPU power is needed or if more than one radio transceiver with small physical dimensions is required, than the TTGO is the best choice. The Feather sticks are available only either with Wi-Fi or LoRa onboard.

Table 4.3: Power consumption of MCUs including radio transceivers

	Power in mW	
	Mean	Max
ESP32, TTGO, WiFi	591	1353
ESP8266, Feather Huzzah, WiFi	405	2118
Feather M0, WiFi	475	1577
ESP32, TTGO, LoRa	235	816
Feather M0, LoRa	45	528
Feather 32u4, LoRa	47	568

To evaluate the LoRa transceiver modules' communication range, we deployed our sensor box (Fig. 4.4) and measured the transmission range using one of our GPS modules. We used ESP32 TTGO modules in various configurations. One chip was tuned to the 433 MHz band and equipped with a small wire antenna from the supplier. The next one was set to work in the 868 MHz band, also with the short vendor-supplied antenna. Finally, we set up another TTGO in the 868 MHz band but on a different channel with larger 3.5 dBi magnetic sucker antennas on both ends. The sending interval chosen for the fixed station was set to 7 seconds for each device.

The city of Marburg is surrounded by many hills and forests, therefore, the sending station was deployed on a viewpoint near the university with line-of-sight to most parts of the city. The

Table 4.4: Comparison of Concept Detection on Pi vs. Neural Compute Stick

	Idle - Power in mW			**Starting** - Power in mW			**Analysis** - Power in mW		
	Mean	Max	Time	Mean	Max	Time	Mean	Max	Time per Image
Pi 3b, 1K, CPU	1107	1845	59.29 sec.	2626	2895	19.66 sec.	2723	3537	13.27 sec.
Pi 3b, 1K, Movidius	1955	2095	59.76 sec.	2372	2884	3.92 sec.	2760	3388	0.61 sec.
Pi 3b, 6K, CPU	1107	1845	59.29 sec.	2659	3203	75.03 sec.	2871	3537	5.46 sec.
Pi 3b, 6K, Movidius	1955	2095	59.76 sec.	2507	2879	12.64 sec.	3313	3667	0.82 sec.

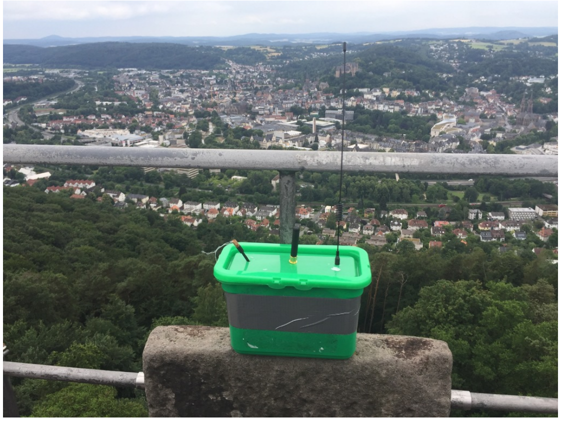


Figure 4.4: Waterproof static sensor box deployed.

receiving nodes were mounted on the dashboard of a car, and we drove around the campus, through the city, up to the castle on the opposite side of the city, and then along the valley, until we lost the signal. The route we took, including received beacons, is shown in Fig. 4.5.

The longest distance covered was almost 6.5 km with the 3.5 dBi sucker antenna. The 433 MHz setup and the other 868 MHz node reached 1-2 km less, but with a much higher packet loss. Also, there was no more line-of-sight from either maximum distance points, it was blocked by hills, houses and trees. The 1.5-2 km distance to the downtown area was covered by all three devices despite trees and houses in the way.

The RSSI over distance is shown in Figure 4.6, where up to 2-3 km all devices still received signals quite often. As expected, the 3.5 dBi antenna has the best reception throughout the test. This is especially clear for distances over 3 km where the duck antennas both fall short. The total number of received packets in percent can be seen in Fig. 4.7. The clear winner here, again, is the 3.5 dBi sucker antenna. Both stock antennas from TTGO perform very similar with reception rates between 10% and 15% during our test. Due to our constant movement and the diverse topographic areas, these numbers should be seen in relation to the sucker antenna, not as absolute numbers. Traffic might be responsible for remaining longer in areas where no reception was possible, resulting in a higher overall packet loss rate.

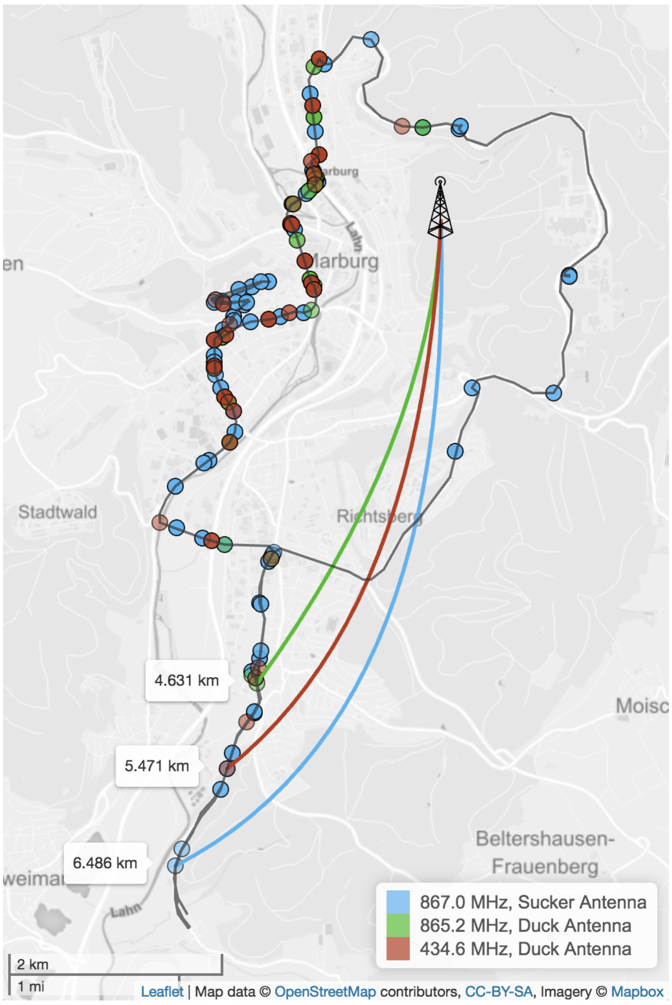

Stadtwald
Marburg
Richtsberg
Moisc
Beltershausen-Frauenberg
weimar
Lahn
4.631 km
5.471 km
6.486 km
🟦 867.0 MHz, Sucker Antenna
🟩 865.2 MHz, Duck Antenna
🟥 434.6 MHz, Duck Antenna
2 km
1 mi
Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA, Imagery © Mapbox

Figure 4.5: Communication range of different LoRa setups.

On-Device Data Processing

For on-device data processing, we used 1481 images for CPU and neural compute stick execution. Each of these images is 3000 x 2000 pixels in size, since 6 MP are a good compromise between storage space and visual details. This dataset is described in our previous work [139]. For evaluation purposes, we built a Python tool to batch process these images. First, the program loads the necessary libraries and the pretrained neural network, then the images are processed. The power consumption and time for the tasks is shown in Table 4.4.

First, we compared the idle power consumption of the Pi 3 to one with an attached neural compute stick. While the max peak is almost identical, the average power consumed is nearly double when a NCS is added. In the starting phase, the neural network is loaded either into RAM for the plain Pi 3 or onto the NCS. Both setups draw around 2500 mW with a slight edge for the NCS setup. Taking runtime into account, the compute stick is clearly more efficient by loading the trained model 5-6 times faster. When performing the actual visual concept detection, the mean power is between 2723 and 3313 mW and peaks at 3667 mW, with a slightly

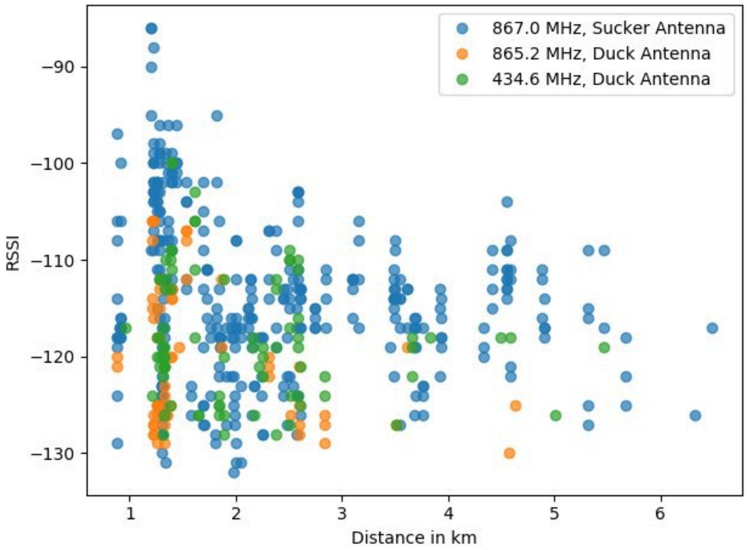


Figure 4.6: RSSI vs. Distance

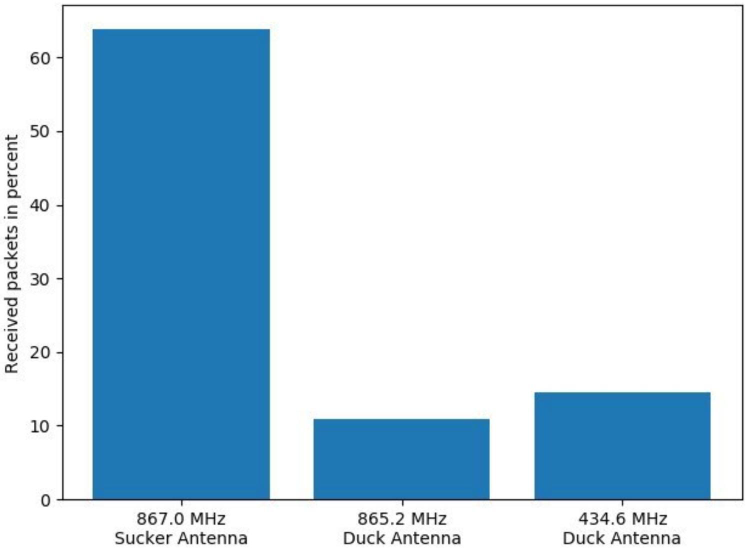


Figure 4.7: Number of received packets in relation to packets sent.

lower power demand by the CPU setups. Given the much longer runtimes of the CPU version, 6x and 21x slower per image, the NCS is still much more efficient, in terms of speed and energy. The main drawback of the tested compute stick is its high power consumption when idle. This problem was solved by deactivating the specific USB port when the stick is currently not needed, reducing the power consumption overhead to zero.

Setup Cost

Since we want to provide affordable solutions that can easily be customized to specific needs, we also provide a short overview of the cost of our systems. All prices are in (rounded) Euro, taken in June 2018 from Amazon, Aliexpress etc. All our cases are custom printed on a 3D printer. We also did successful deployments with cheap waterproof junction boxes from the local hardware store (<10 €).

The cheapest system is our BLE LoRa Modem with a total system cost of about 14 € (Table 4.5). A 750 mAh battery lasts roughly between half a day and a full day. This component can easily be swapped for a bigger battery or powerbank.

Table 4.5: Cost of BLE LoRa Modem

Part	Estimate Price	Comment
TTGO LoRa SX1276 ESP32	9 €	incl. antenna
LiPo Battery 750mAh	3 €	
3D printed case	2 €	
Total Price	14 €	

The MSP example shown in Table 4.6 is usable for GPS tracking of larger animals plus some added sensors for collecting weather data. In real world deployments with a 750 mAh battery and gathering samples every 15 seconds, the power lasted for about 6 days. With a reasonably lightweight and cheap 1500 mAh battery and a lower sampling rate, we can achieve runtimes over two weeks. The total cost for this setup is around 46 € per device, mainly dominated by the price of the SD card for data logging and the GPS module.

In Table 4.7, we show a SSP example used for relaying data as well as actively gathering sensor data and processing it on device. Depending on whether a compute stick is used or not, the cost varies between 135 € and 201 €. The cost for a blank relay-only system would be around 65 €. The rest is the cost of the added sensors and the NCS. What this setup is lacking, is a

Table 4.6: Example configuration for a MSP

Part	Estimate Price	Comment
TTGO LoRa SX1276 ESP32	9 €	incl. antenna
LiPo Battery 1500mAh	3 €	
3D printed case	4 €	
Ublox NEO GPS	12 €	incl. antenna
SD-Card Breakout	2 €	
32 GB SD Card	13 €	
Real Time Clock	1 €	
Temperature & Humidity Sensor	1 €	
Gyro Sensor	1 €	
Total Price	46 €	

Table 4.7: Example configuration for a SSP (w/o power supply)

Part	**Estimate Price**	**Comment**
Raspberry Pi 3	32 €	
TTGO LoRa SX1276 ESP32	9 €	incl. antenna
3D printed case	10 €	
Ublox NEO GPS	12 €	incl. antenna
32 GB SD Card	13 €	
Real Time Clock	1 €	
Temperature & Humidity Sensor	1 €	
Pi Camera	20 €	night vision opt.
Movidius NCS	*66 €*	*optionally*
Thermal Imaging	32 €	
Rain Sensor	1 €	
Soil Moisture	3 €	
PIR Motion Detector	1 €	
Total Price	135 € / *201 €*	

power supply. Depending on the location of deployment, a battery system (20 € - 200 €) and/or a solar panel (50 € - 200 €) could be added if a direct power supply is not available.

4.2 Automated Non-lethal Moth Traps Can be Used for Robust Estimates of Moth Abundance

4.2.1 Introduction

Recent reports on the decline in insect abundance and biomass [90, 91, 212] have raised considerable public concern together with calls for the immediate implementation of conservation strategies [95]. However, shortcomings in data availability have led to questions regarding the generalizability of this decline [52, 218]. The resulting controversy can best be resolved by regular and comprehensive assessments of insect communities [46, 96, 167, 249]. Nonetheless, such efforts have been hindered by a scarcity of technical as well as taxonomic expense and the logistical challenges associated with traditional survey methods. The development of automated monitoring systems that can be readily implemented and allow broad spatial and taxonomic coverage is therefore needed [166].

Visual surveys such as transect walks, net sweeping or hand catches are labor intensive. Conventional traps can reduce the demands of field work, but they are limited by their high maintenance and post-sampling efforts [166]. Furthermore, caught specimens must be stored at least until their identification, but storage or freezer space may not be available or is limited. Lastly, conventional insect monitoring methods usually involve killing the insects for subsequent determination and biomass assessment, which has raised ethical concerns: Some traps can result in the slow death of the insects, which must be weighed against the knowledge and benefits for conservation gained from their use. The debate over consciousness of insects and their ability to feel pain should be recognized [62]. Even in the absence of detrimental effects imposed by the long-term monitoring of insect populations [77], less destructive alternatives should be considered whenever possible [50].

Given the above considerations, most monitoring campaigns focus on specific taxa or locations, with the latter often being chosen based on accessibility and a high density of the target taxon [46, 63]. However, this could result in a bias due to non-random site selection and, in turn, to biased estimates of abundance or biomass and of the associated trends (missing zero effect, [46])). Another practice is to focus on sampling periods that are assumed to adequately cover insect diversity [179, 206]). However, such temporal snapshots, whether based on daily or on annual data collection, imply a loss of information [179], because phenological shifts and species occurring outside the general phenology peak will remain undetected (snapshot & groundhog effect, [46]). Monitoring networks that collect data on a daily basis, such as the Rothamsted insect survey [64], are a notable and rare exception. Yet even this dataset, made up of > 10 million records, and thus probably more accurate than any other, cannot provide complete species inventories at the spatial resolution relevant for conservation [203].

A limitation of manual species identification is the steadily decline in the number of taxonomic experts [55], such that the number of sampling points cannot be increased indefinitely. Machine-based automated species determination, especially artificial intelligence (AI)-based image recognition, reduces both the workload and the cost associated with monitoring, thus allowing the number of sampling points to be scaled to the required spatial and temporal extent and resolution [46, 166]. Furthermore, non-lethal monitoring systems can provide real-time information about the state of populations while eliminating the ethical concerns related to lethal sampling [46, 166].

The development of a functioning prototype and a working AI system based on comprehensive training data requires an adaptive process whose applicability and functionality in real world settings is constantly re-evaluated and accordingly adjusted [88]. However, as scalable monitoring is often subject to financial constraints, automated monitoring systems must be as cost efficient as possible [88]. While all of these considerations are crucial in themselves, their relevance will depend on whether the automated system is actually able to monitor the targeted taxonomic group in the field.

Generally, every change in the design of a trap affects its capture efficiency [191] and therefore the conclusions regarding species composition, richness, and abundance. For example, flight interception traps consisting of only one collecting jar at the bottom, rather than an additional collecting jar at the top, might underestimate taxa that tend to move upwards after collision [132]. Even in traps targeting single species, slight adjustments such as the width of the trap opening or the addition of a rain cover can affect the number of captured individuals [18, 22, 86]. In extreme cases, design-related differences in community and/or abundance data can bias ecological conclusions [204], especially when the effectiveness of the trap system depends on environmental conditions and species traits [23].

In most monitoring systems, including lethal ones, the probability of detecting an individual depends on its activity [46, 108, 261]. This dependency is likely to be reinforced in camera-based, non-lethal systems, in which individuals are recorded only when they enter the camera's field of view. Since, unlike in lethal systems, individuals are not permanently removed, they might be counted several times if they leave and re-enter the detection area. In such cases, variations in activity, and thus movement, might change not only the probability of detection but also the likelihood of duplicated counts. This could in turn lead to discrepancies in abundance trends determined by conventional vs. camera-based monitoring systems. The efficiency of newly designed traps should therefore be compared with that of conventional traps, as should the ability of the newly designed trap to accurately capture the response of the target group to changes in the environment. In this study, we investigated whether an automated moth trap (AMT), which attract moths to a screen via ultraviolet (UV) light and then photographs them [17, 107], leads to abundance trends consistent with those determined using conventional (lethal) light traps.

Moths are one of the most diverse insect groups and they are closely tied to their ecosystems [65]. Accordingly, they have a long history in monitoring studies [203] and are often the commonly targeted taxon in new identification algorithms[20] [29, 213, 262]. However, moths are highly sensitive to trap design [21, 57, 96]. For example, differences in relative abundance were observed between manual sampling, in which species are recorded directly when they land, and funnel traps, which are not necessarily entered by all moths [21]. An AMT, on the other hand, captures the moths directly, but only if they land on the screen photographed by the camera.

[20] https://nationalmothweek.org/2017/07/21/introducing-lepsnap-image-recognition-for-moths-butterflies-guest-post-by-andre-poremski/

In this study we deployed an AMT and a conventional bucket trap at the same sites throughout the late summer, shortly after the phenological peak of moths. We reasoned that, if the two trap types are equally efficient in capturing local moth abundance, they should show a similar temporal decline in moth individuals. In addition, by deploying the AMT under realistic in-field conditions in a relatively remote area, we were able to assess its usability while also identifying its weaknesses, which can then be addressed in subsequent iterations.

Parts of this section have been published in Jonas Mielke Moeglich[21], Patrick Lampe[1], Mario Fickus, Jannis Gottwald, Thomas Nauss, Roland Brandl, Martin Braendle, Nicolas Friess, Bernd Freisleben, and Lea Heidrich. "Automated Non-lethal Moth Traps can be used for Robust Estimates of Moth Abundance." In: *submitted; under review* ().

4.2.2 Material & Methods

Study Sites

Our study was part of the Nature 4.0 framework [22] and was conducted in the "Marburg Open Forest", a $1.5km^2$ university-owned forest area near Caldern, in Hesse, Germany. The forest is a typical beech-dominated managed forest embedded in an agricultural landscape. The 18 trap sites were selected from the existing Nature 4.0 sites and were evenly distributed over the research area, thus covering small-scale climatic, structural, and ecological differences. Each site included a beech tree (50 to 120 years-old) in its center and is separated from the other sites by least 50 m. An overview map can be found in 4.8.

Conventional Traps

The conventional light traps 4.9 consisted of a super-actinic UV light tube (Sylvania blacklight F15W/T8/BL 368, 12 V, 15W, wavelength peak: 368 nm) that was used to attract the moths, which were then directed via a funnel into the attached 10-l bucket. The bucket contained a closed chloroform jar from which a piece of cloth was drawn through a small hole in the lid to act as a wick, thus filling the bucket with the killing agent. Pieces of egg carton were added to allow the caught moths to rest, thus also reducing stress and wing damage. The UV light was powered by a 12 V lead battery. A light switch (Kemo Germany M197) attached to the battery automatically turned the light on at sunset and shut the light off at sunrise. In the morning, the traps were emptied and all moths were collected and stored in a freezer for later sorting.

Automated Moth Traps

A prerequisite of our AMT was that it should operate in a real-world environment and in a variety of settings. Therefore, the design requirements of the AMT were as follows:

- modular and flexible, allowing both extensibility and adaptability for further applications

[21] shared first authorship
[22] www.natur40.org

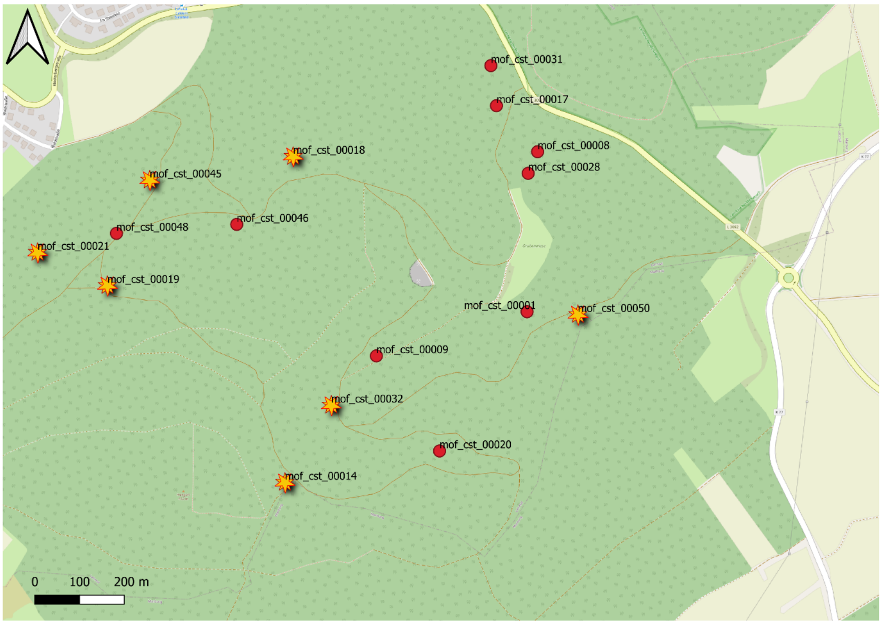


Figure 4.8: Map of the Marburg Open Forest (MOF) and our study sites. Symbols indicate an exemplary randomized setup of AMTs (star) and conventional traps (red) for one night. The following night, the trap types were exchanged such that sites assigned an AMT the first night received a conventional trap the following night and vice versa (two nights = one round).

- reproducible, such that the AMT could be easily assembled by other researchers
- cost efficiency, to allow simultaneous installments of multiple AMTs
- robust, to allow continuous operation of the trap in outdoor settings
- easy to use and configurable, to support long deployment periods while minimizing the work load

Following these requirements, we adopted the basic design of Bjerge and colleagues (2021) but used our own software and customized the hardware to meet our needs: The AMT consists of several power consumers (LED-UV light, LED screen, Raspberry Pi, ring light, and camera). To achieve a comparable runtime of the two trap types without access to a power grid and to reduce energy consumption as well as overall acquisition costs, the 15 W super-actinic UV light tube (176€) in the conventional trap was replaced in the AMT by a LED-UV tube [23]. Below the LED-UV tube, we installed a waterproof, white, cloth-lined LED screen illuminated by four rows of 30 cool-white LEDs (6000 K, 240 lumens/m). The LED screen acted as a resting site for the moths. The illuminated screen allowed for standardized photos [17], which were taken by an oppositely placed camera box with an attached ring light for illumination. Both the camera and the ring light were directly connected to the Raspberry Pi computer. The Raspberry Pi's software was programmed to trigger a customizable schedule, according to which photos were taken and the light (UV and LED screen) was controlled. A 5MP Raspberry Pi-camera,

[23] `www.entosphinx.cz` 37.12 LED/UV lamp, 9.6 W, wavelength from 395 to 405 nm, 33€

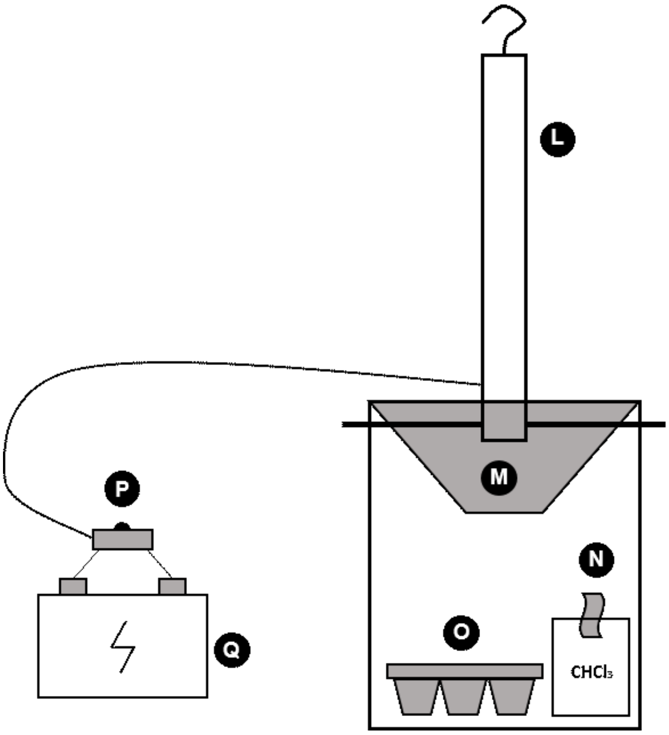


Figure 4.9: Setup of a conventional bucket light trap L) UV tube M) Funnel N) Chloroform O) Egg carton P) Light switch Q) 12 V battery

v1.3 was chosen to further reduce costs. The whole system was mounted on a tripod to avoid disturbance by animals, foliage, and puddling. To enable the longest possible operation time, only the UV light and LED screen were powered by a 12 V battery. The Raspberry Pi, which is susceptible to voltage fluctuations, as well as the ring light were powered by a 5 V power bank inside the camera box to ensure stable voltage even when the LEDs were being powered. This setup also avoided energy loss during voltage conversion, since the Raspberry Pi requires a 5 V power supply. To ensure the reproducibility of the design, the trap was constructed from off-the-shelf hardware components. All of the components, their prices, and the measured power consumption are listed in chapter 4.2.2 and table 4.8 (for AMT's architecture and a detailed description of the trap design, see figure 4.10).

Power Consumption

Power consumption was measured to provide an overview of how long the ATM can be operated using different energy sources:

- UV LED: 0.8A@12.5V   10W
- LED Board: 0.78A@12.5V   9.75W
- Ring light: limited by the output of the power bank
- Power bank: 2.1A@5.1V   11W

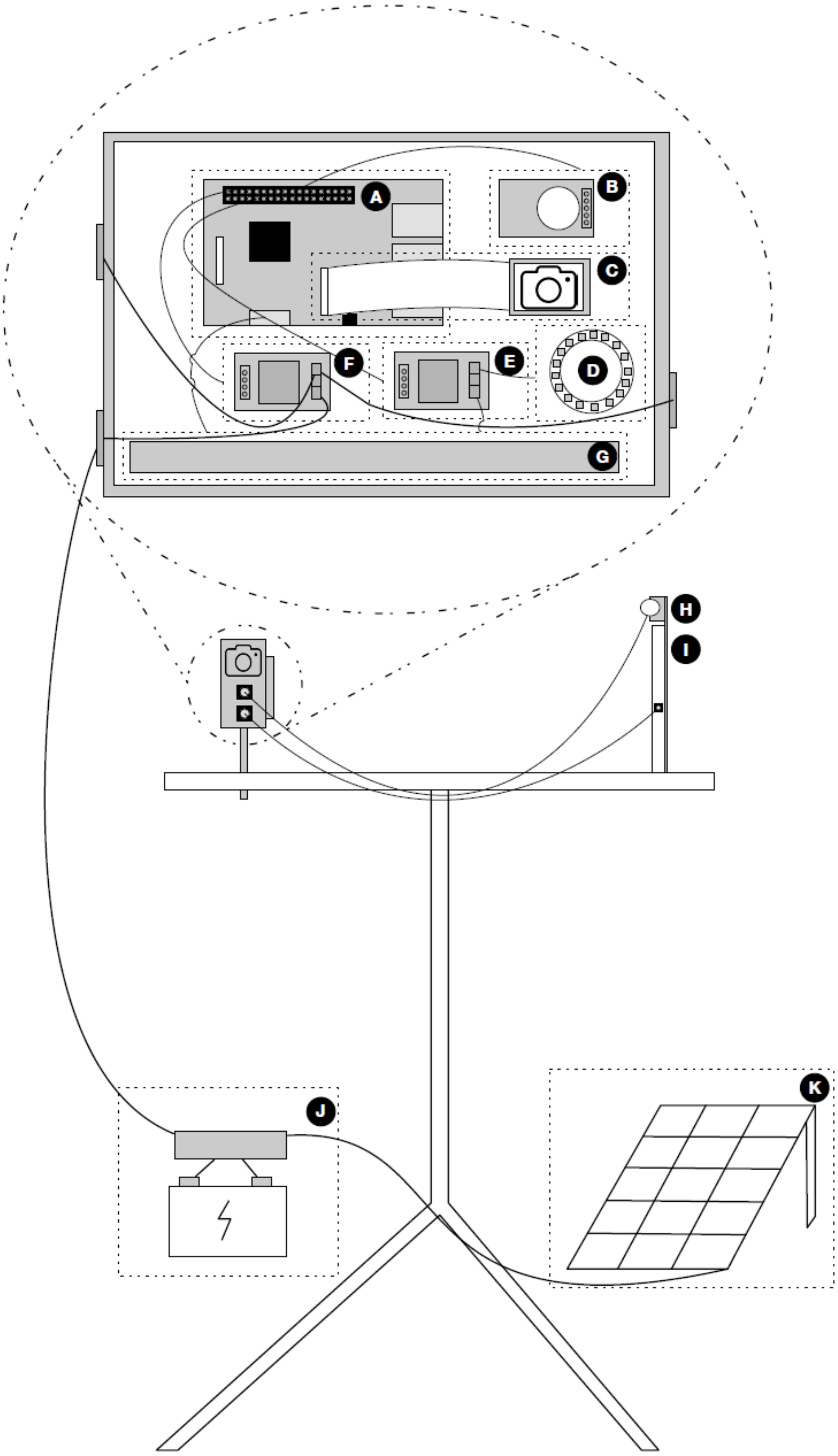


Figure 4.10: Setup of our AMT A) Raspberry Pi B) Real-time clock C) Camera D)Ring light E & F) Relays G) Power bank H) UV LED I) LED screen J) Battery box K) Solar panel

- Raspberry Pi: 0.35A@5.1V   1.8W

The used energy is the sum of the energy used in moth attraction multiplied by the percentage of attractions per hour and the energy consumption per photo multiplied by the number of photos taken in one hour and the power consumption of the Raspberry Pi. The energy per hour is calculated as follows:

$$Energy\_use\_per\_h = t * (uv + led) + (n * s * rl/3600) + rp \tag{4.1}$$

where t is the percentage of attraction, n is photos per hour, s is the duration of ring light per photo, uv is Watts of UV light, led is Watts of LED board, rl is watts of ring light, and rp is Watts used by the Raspberry Pi.

- 50 min attraction: 50/60 * (10 + 9.75) + (100 * 3 * 11 / 3600) + 1.8 = 19.175 Wh
- 40 min attraction: 40/60 * (10 + 9.75) + (80 * 3 * 11 / 3600) + 1.8 = 15.7 Wh
- 30 min attraction: 30/60 * (10 + 9.75) + (60 * 3 * 11 / 3600) + 1.8 = 12.225 Wh
- 20 min attraction: 20/60 * (10 + 9.75) + (40 * 3 * 11 / 3600) + 1.8 = 8.75 Wh
- 10 min attraction: 10/60 * (10 + 9.75) + (20 * 3 * 11 / 3600) + 1.8 = 5.275 Wh

The system is modular, with a sensor box at its core that switched the UV light and the LED screen on and off by means of relays. Both the UV light and the LED screen were connected to the sensor box via waterproof cables and connectors, so that all circuitry was contained in the sensor box. The specific UV lamp or LED screen could thus be replaced without having to change the sensor box itself. Only the ring light, which was also controlled by a relay, was permanently connected to the sensor box. To ensure robustness, all components, except the ring light, were waterproof. Ease of use was ensured by using standardized plugs and reverse-polarity-protected sockets for the individual components. In addition, the software was designed to be configurable and to start automatically.

Our software implementation was based on an extended version of the software of [81], which was built using the pimod tool [105] to configure operating system images in a user-friendly manner. Triggering the camera and the control pins of the Raspberry Pi by the program switched the external components (i.e., the ring light, LED screen, UV light) on and off via attached relays. AMT's software was configurable by an yml file. A downloadable image that can be customized with pimod to fit the needs of the project is available in our GitHub repository:[24]

An additional feature of the software was that the sensor box includes a Wi-Fi connection, which allows wireless observation and configuration. The box could be accessed via ssh (Secure Shell) or via the live view of the camera using http (Hypertext Transfer Protocol) and a browser on the mobile device.

[24] https://github.com/Nature40/InsectPhotoTrapOS/releases/tag/IPTv1.0.7

Table 4.8: Price list of AMT components

	Modul	Parts (exemplary products)	Quantity	Price [€] (29.11.21)	Price per trap [€]
A1	Camera box	Raspberry Pi 3 B	1	59.99	59.99
A2	Camera box	SD card, SanDisk microSDHC Ultra, 64Gb	1	15.99	15.99
A3	Camera box	Case	1	23.00	23.00
A4	Camera box	Mounting plate (3d print)	1	1.00	1.00
A5	Camera box	Jumper wires	0.1	5.35	0.54
A6	Camera box	Neutrik NAC3FPX power socket female	2	8.26	16.52
A6	Camera box	Neutrik NAC3MPX power socket male	1	3.66	3.66
A6	Camera box	Neutrik SCNAC-FPX	3	1.08	3.24
B	Camera box	DS3231 RTC Modul I2C real-time clock AT24C32 for Arduino; with LIR2032 battery	1	5.00	5.00
C	Camera box	Raspberry Pi camera 5MP, v1.3	1	26.04	6.55
D	Camera box	LED ring light	1	8.99	8.99
E,F	Camera box	Relay KY019RM	2	3.79	7.58
G	Power	EasyAcc Powerbank, 20000mAh	1	47.00	47.00
		Subtotal camera box			199.06
H	Light box	Entosphinx 37.12 LED/UV LAMP, 395-405 nm	1	33.00	33.00
I1	Light box	Box for backlight	1	23.30	23.30
I2	Light box	LEPRO LED strip, 5 m, 12 V, 6000 K, 240 lumens/m for backlight	0.4	16.44	6.58
I3	Light box	Aluminum foil in format of box for backlight	0.2	3.00	0.60
I4	Light box	Lee filter in format of box for backlight	0.03	65.00	1.95
I5	Light box	Bedsheet in format of box for backlight	0.1	10.00	1.00
I6	Light box	Spray adhesive	0.2	15.00	3.00
I7	Light box	Neutrik NAC3MPX power socket male	1	3.66	3.66
I7	Light box	Neutrik SCNAC-FPX	1	1.08	1.08
I8	Light box	Neutrik NAC3MX-W plug male	1	7.44	7.44
I9	Light box	Cable connections	1		1.00
		Subtotal light box			82.61
J1	Power	Battery, 12 V, 12 Ah	1	32.18	32.18
J2	Power	Battery box	0.5	52.80	26.40
J3	Power	Neutrik NAC3FPX power socket female	1	8.26	8.26
J3	Power	Neutrik SCNAC-FPX	1	1.08	1.08
	Power	Fuse	1	0.50	0.50
		Subtotal power box			68.42
K	Cable	Neutrik NAC3FX-W plug female	2	9.08	18.16
K	Cable	Neutrik NAC3MX-W plug male	2	7.44	14.88
K	Cable	Cable	1		2.00
		Subtotal cable			35.04
	Mounting	Backlight and UV light holder self-made	1	9.30	9.30
	Mounting	Tripod, Gravity GLSTBTV28	1	72.00	72.00
	Mounting	Threaded rods	1	1.00	1.00
	Mounting	Screws	4	1.00	4.00
		Subtotal mounting			86.30
		Other items			20.00
	Total				**491.43**

Sampling/Experimental Design

The two trap types were tested at separate times, to avoid their mutual interference during sampling. Thus, the eight AMTs and eight conventional trap systems were randomly assigned to the 16 sites for one night. The following night, the trap types were exchanged such that sites assigned an AMT the first night received a conventional trap the following night and vice versa (two nights = one round). This routine was performed five times from July 2021 to August 2021. The AMTs were scheduled to take photos every 30 s. Activation of the lights (UV and LED screen) was scheduled to start at 10 p.m. and end at 6:00 a.m., with an on time of 50 consecutive min/h followed by 10 min of lights out. This was done to test the functionality of the hourly schedule and to roughly mimic the light phase of the conventional trap.

Software Configuration

Exemplary configuration of our test unit and sample runs:

```
[TimedInsectTrapAnalysesUnit]
light_pin = 26
uv_light_pin = 21
hourly = True
uv_light_start_time = 00
uv_light_end_time = 50
adjust_time_in_seconds = 2
take_photo_every_seconds = 30

[run.1]
start = 21:10
stop = 23:59:59

[run.2]
start = 00:00
stop = 05:50
```

[TimedInsectTrapAnalysesUnit] defines the unit itself and the associated parameters, including the pins used for the respective relays as well as the start and end times of the hourly routine. These values can be adapted as needed, included depending on the unit and the research question. The time the camera needs to focus is specified by adjust_time_in_seconds: for a camera with a fixed focus, this option can be set to 0. The time between two photos is specified by take_photo_every_seconds, for photos taken periodically between uv_light_start_time and uv_light_end_time. To map as many use cases as possible, different runs can be defined and the parameters modified in each one. For example, take_photo_every_seconds can be set to 30 in run 1 and to 60 in run 2.

With this schedule, the theoretical power consumption was 19.175 Wh (see chapter 4.2.2).

Moth Counting

Moth abundance in the photos taken with the AMT was determined by manually counting the number of moths in each frame. For the total number of moths per night, only new arrivals on the LED screen were counted; in other words, when the total number of resting insects in a photo superseded the total number of resting insects in the prior photo, the moth count was raised. This method was used because movements of the insects on the LED screen made it difficult to identify individuals. In addition, the possibility that the same individual flew out of the photo, swirled around the light, and then again rested on the LED screen could not be ruled out. Accordingly, in the frame by frame counts of the insects, only the total number of currently sitting insects was considered. Non-target insects were not counted. Due to varying image quality, very small insects could not be identified with confidence. Thus, moths smaller than approximately 0.8 cm and without an easily identifiable characteristic moth profile could not be included in the count, since the distinction from, e.g., small Diptera species was no longer possible due to blurriness. Samples from conventional bucket traps were determined to the insect order and moths counted using a binocular.

Statistical Analyses

The analyses are aimed at testing whether the number of moths captured by the AMTs scale isometrically to the number of moths captured by the conventional traps, thus yielding similar seasonal patterns in the number of individuals.

Our tests for an isometric relationship between the two traps are based on the assumption that the two nights of each round were comparable, i.e., that the number of moths do not differ when one or the other trap system was installed. Unfortunately, due to several failures of the AMTs (described and discussed below), only some of the sampled locations had repeated measures see figure 4.11. Thus, we tested a priori for date-wise differences between the number of moths caught by the conventional traps, which had a larger sample size due to the AMT malfunctions. This pairwise comparison, using an ANOVA followed by a post-hoc test, revealed that rounds one, three, and five had comparable numbers of individuals; in rounds two and four, the number of moths during the respective test nights differed, e.g. due to weather changes such as rising temperatures (see figure 4.11, round 4). We therefore applied the following analyses to both the full data set and to a reduced data set in which rounds two and four were omitted First, a major axis (MA) method was used to test for isometric scaling between the abundance captured by both trap types per round and plot. In contrast to a simple linear (ordinary least squares, OLS) model, the MA method accounts for variations in the X variable (see Smith (2009) for details), defined here as the number of moths caught by conventional traps. Thus, it does not assume that the use of conventional traps provides a true baseline of available moths. Rather, the MA method reflects the fact that the two traps capture only a fraction of the true abundance and that both are dependent on a temporal and spatial baseline. The MA was calculated with 999 permutations, using the lmodel2-function of the lmodel2 package [25]. Note that some of the plots were repeatedly sampled, while others were sampled only once or twice due to trap malfunctions (see figure 4.12). This hindered the implementation of plot-ID as a random effect,

[25] https://CRAN.R-project.org/package=lmodel2

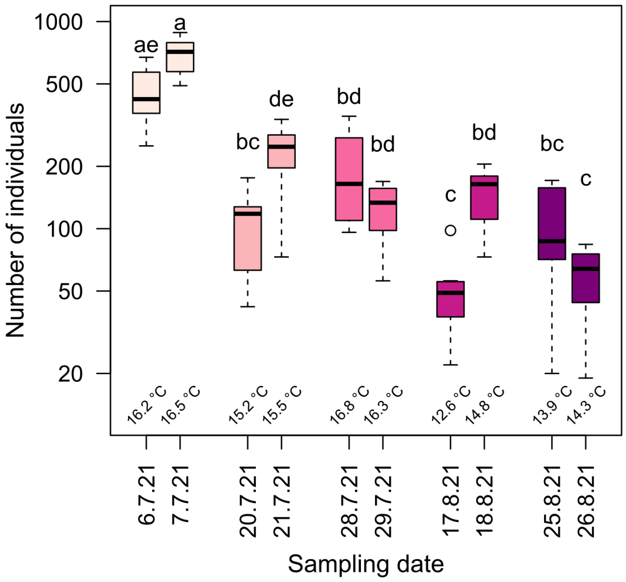


Figure 4.11: Results of the ANOVA and subsequent Tukey HSD to test for pairwise differences in the sampling dates. Daily mean temperature is also given. Boxes are color-coded according to the sampling round. Pairs that are not significantly different from each other share the same label. Round 2, sampled on 20th and 21st July, and round 4, sampled on 17th and 18th August, differ from each other; round 1, sampled on 6th and 7th July, round 3, sampled on 28th and 29th July, and round 5, sampled on 25th and 26th August, do not (p-values 0.001 and 0.025, respectively).

because the resulting singularities could have led to overfitting and to numerical problems and would have made standard inferential procedures inappropriate [8]. However, an ANOVA did not show any significant differences between the plots, suggesting that the plot-ID had a neglectable influence. To handle the discrete nature of the moth counts, the counts from AMTs and conventional traps were loge-transformed in all analyses. This allowed retention of the data of both axes within a comparable range. Whether the slopes resulting from all models equaled one was determined using the slope.test function of the smatr-package [251].

We then tested whether the two trap systems differed in depicting abundance trends, by taking advantage of the phenology of moths. In Europe, moths usually peak in abundance and diversity around July [197, 203]. The later the sampling time thereafter, the fewer moth individuals in the area. We thus used the Julian day, which well-reflected the time lapse between the rounds and thereby possible seasonal trends, as an independent variable to model the number of individuals using an OLS. We then centered the Julian day such that its intercept was in the middle of the sampling period, which thus reflects the differences in abundance. The trap type was included as an interaction term to test for differences in the systems in depicting the temporal trend in abundance:

$$Model : log_e(\#individuals)\ trap\_type * centred(julian\_day) \tag{4.2}$$

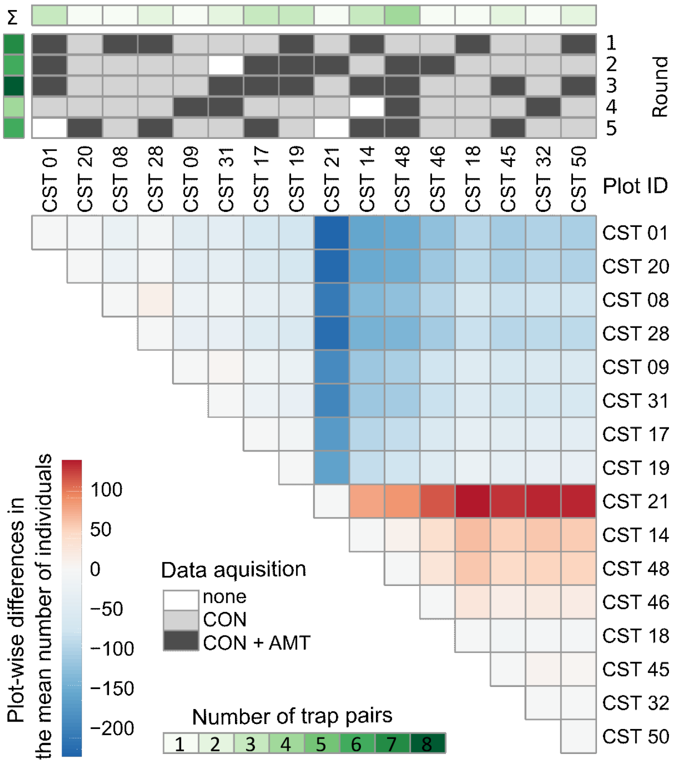


Figure 4.12: Plot-wise differences in the mean number of individuals, determined using the package "pheatmap". Differences are indicated along a color scale from red (positive) to blue (negative). The rows and column above the main plot indicate whether the conventional traps had been set up during the respective round. None of the differences were significant.

For consistency, the number of individuals was loge-transformed and only data points with records from both trap types included (see electronic repository for analyses on full data sets). We also performed all analyses with GLMs with an untransformed dependent variable, a loge-transformed independent variable, and a negative-binomial distribution. However, no noteworthy differences were detected. These models can be reviewed in our electronic repository. Linear hypothesis tests were conducted using the package "car" [64]. All figures were created using the package "ggplot2" [255]. All analyses were conducted using R [26] and RStudio [27].

4.2.3 Results

The conventional traps captured 17,164 individuals (mean = 226, SE = 25, min = 19, max = 885) in total and 6,755 individuals (mean = 218, SE = 36, min = 20, max = 791) considering only samples with accompanying AMTs. The latter captured a total of 4,065 individuals (mean = 131, SE = 24, min = 13, max = 616). On average, AMTs captured 87 fewer individuals. However, during 8 out of 31 comparable nights more individuals were captured by the AMTs than by the conventional

[26] https://www.r-project.org/
[27] https://www.rstudio.com/

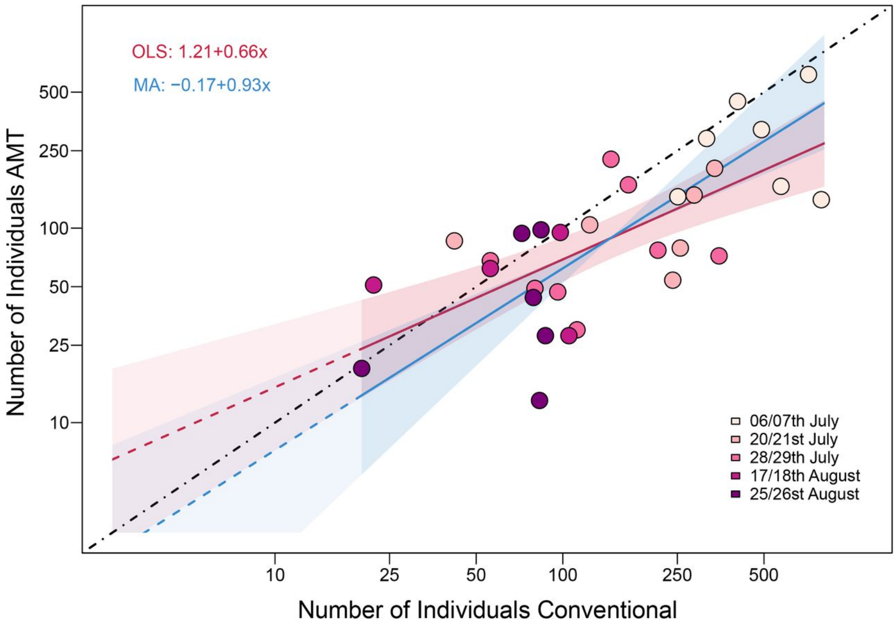


Figure 4.13: The number of individuals captured by automated moth traps (AMTs) and conventional light traps per plot and round. The points are color-coded according to the sampling dates. The black line represents an isometric relationship with an intercept of zero and a slope of one, the red line the ordinary least squares (OLS) model, and the blue line the major axis (MA) model, with shaded areas as confidence intervals. Note that the extrapolation on the left (less intensively shaded) only displays the uncertainty around the intercept. Both axes were log-transformed.

traps. The variance between the two trap systems did not differ (F-test on log-transformed data F1,30 = 0.91, p = 0.80). In the OLS model presented here, the relationship between the trap types in terms of moth abundance was isometric (slope = 0.81, linear hypothesis test: F1,19 = 1.35, p = 0.26). Note, however, that this was not the case when considering all rounds (slope = 0.66, linear hypothesis test: F1,30 = 7.11, p = 0.01). The slopes of the MA regression models were close to one, and the intercepts did not significantly differ from zero (Table 4.9, Figure 4.13). Sampling conducted later in the year resulted in fewer recorded individuals overall, and AMTs captured significantly less individuals than the conventional traps (Table 4.10). Nonetheless, the two trap types did not differ in depicting the seasonal decline of moths in the study area (Figure 4.14).

4.2.4 Discussion

Automation can facilitate the monitoring of insects at high temporal and spatial resolution, but automated monitoring must be able to track processes and changes in nature [38]. In addition, these systems must be user-friendly, easy to maintain, cost efficient, allow for long operation times, and sufficiently robust to withstand environmental conditions, such as rain, dew, and wide temperature fluctuations [191]. In this study we tested the ability of an AMT to depict ecological patterns based on satisfactory data quality, as well as its in-field applicability. While

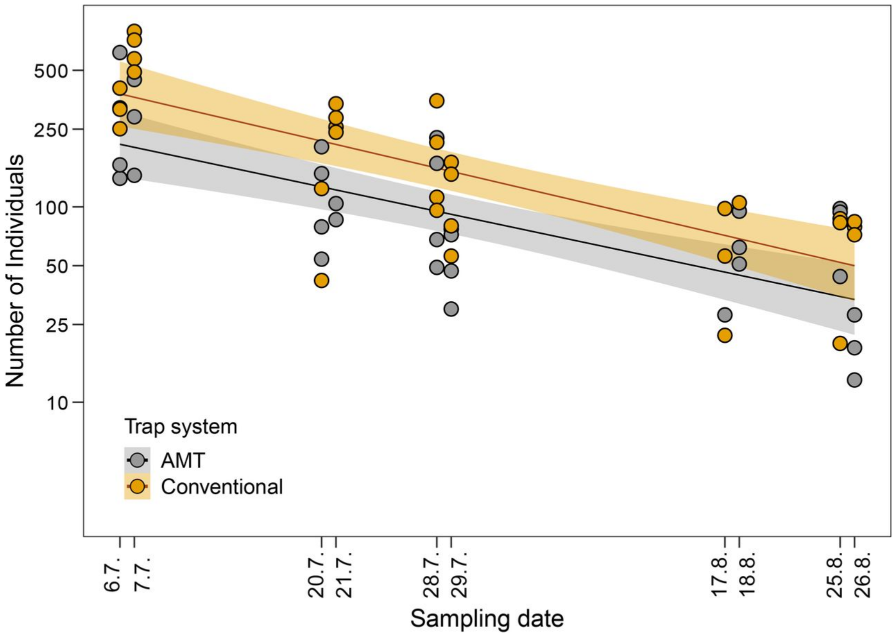


Figure 4.14: The number of individuals captured by all AMTs (grey) and by conventional light traps (orange) per sampling date. The positions of the data points indicate the Julian day of the sampling date. The shaded areas show the confidence intervals of the OLS regression. Only balanced data are shown. Note that the y-axis was log-transformed.

Table 4.9: Results of the ordinary least squares (OLS) and major axis (MA) regressions of the number of individuals captured by the automated moth trap (AMT) and by a conventional light trap based on a complete (all rounds) and a reduced (rounds 1,3,5) dataset. The results of a linear hypothesis test (LHT), testing whether slopes deviate from one, are also displayed. Abundance values were log-transformed prior to the analyses.

	Model	*Intercept*	*CI of intercept* (2.5% / 95%)	*Slope*	*CI of slope* (2.5% / 95%)	*p*	*F (LHT)*	*p (LHT)*
All	OLS	1.21	(-0.12 / 2.52)	0.66	(0.39 / 0.92)	**<0.001**	7.11	**0.01**
rounds	MA	-0.18	(-2.53 / 1.42)	0.94	(0.61 / 1.41)	**<0.001**	0.12	0.73
Rounds	OLS	0.39	(-1.36 / 2.15)	0.81	(0.47 / 1.15)	**<0.001**	1.35	0.26
1,3,5	MA	-1.08	(-4.19 / 0.86)	1.10	(0.72 / 1.71)	**<0.001**	0.23	0.63

Table 4.10: Results of the OLS regression testing potential differences in the trap systems concerning seasonal trends in moth abundance, using the centered Julian day to account for differences in time intervals and the interaction with the trap type. Results are displayed for a balanced data set when the data from conventional traps and their corresponding AMTs were considered. Abundance values were log-transformed prior to the analyses.

Predictors	*Estimates*	*Std. error*	*t*	*p*
Intercept	4.93	0.11	44.3	**<0.001**
Julian day (centered)	-0.04	0.01	-6.32	**<0.001**
AMT	-0.50	0.16	-3.12	0.003
AMT:Julian day	0.00	0.01	0.47	0.640
Observations		62		
R2 / R2 adjusted		0.59 / 0.57		

our in-field test demonstrated the comparability of AMTs and conventional traps in terms of capturing trends in moth abundance, the need for improvements in the AMT's design was also identified.

Abundance and Abundance Trends

Before AMTs can be used in monitoring schemes, their ability to accurately capture ecological patterns must be ensured. A quantitative assessment is the first step in analyzing variations in the ecological patterns of insects, such as those arising from changes in the environment and in the viability of populations [74, 96]. However, trap counts only provide an estimate of the total number of individuals [164]. Given that the counting mode of AMTs differs fundamentally from that of conventional traps and that AMTs might attract other species than conventionally collected, due to different insect settling behaviors [21, 261], the abundances determined by AMTs may be over- or underestimated. Indeed, in this study, compared to the conventional traps, the AMTs captured a smaller fraction of the total abundance of moths per round and plot. This difference was expected, given the differences in the attraction range of the LED used in the AMTs vs. that of the super-actinic light used in the conventional traps and that in some runs, the Raspberry Pi delayed the execution of the schedule at midnight. Moreover, an underestimation of the total number of moths can arise if the moths are counted without considering individual identities, such as if one moth replaces another between frames. Conversely, a duplicated count bias and thus an over-estimation is also possible, as some moths may have flown away during the 10 min without light and then returned to the screen after the light had been switched on again. These duplicated counts might have contributed to the few instances in which the AMT captured a higher fraction of the total population than the conventional traps. Yet, these instances also coincided with the operation of the AMTs during nights that were slightly warmer than those during paired conventional trap operation (see figure 4.15), indicating that the effect of weather conditions exceeds the effect of the trap type. Without marking and tracking individual moths, it is not possible to completely disentangle the effect of changes

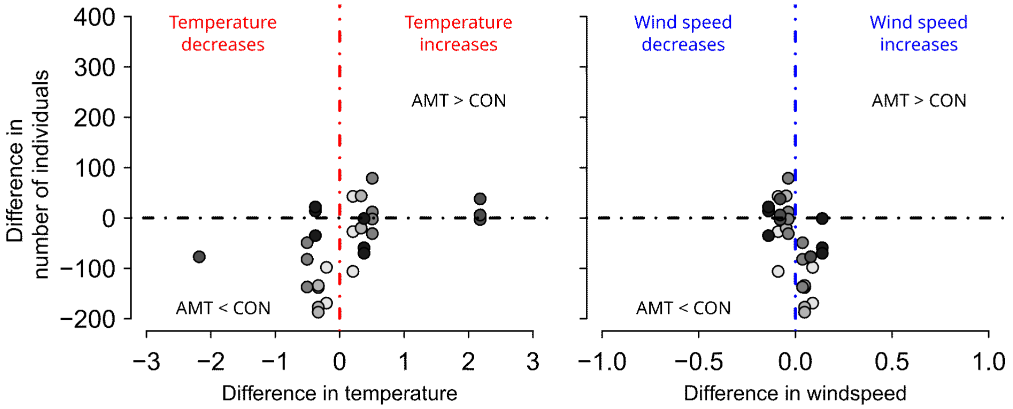


Figure 4.15: The difference in number of individuals captured by automated moth traps (AMTs) and conventional light traps and A) the difference in temperature and B) the difference in windspeed between following nights at each site. The points are color-coded according to the sampling rounds.

in ambient weather conditions between trapping events and potential inaccuracies in the counting method.

Despite these ambiguities, the changes in population size determined using the AMTs corresponded 1 to 1 with those determined using the conventional traps. This was also demonstrated by the seasonal trends in population size, which were similarly captured by the two trap types. Overall, the AMT is sufficiently robust to capture important trends in moth abundance, thus providing fundamental ecological information, especially since abundance is commonly used to evaluate insect communities [131, 258]. The utility of AMTs will be expanded even further by the development of more advanced functions, such as species detection and justifies further investments.

Field-work Applicability and Lessons Learned

In evaluating the in-field applicability and usability of our ATM, two issues came to our attention that need to be addressed. First, the DIY prototypes constructed from consumer-class hardware suffered problems in the field, despite prior laboratory testing. These problems resulted in data loss see figure 4.12. Out of a total of 90 sampling events (5 rounds × 18 plots), the AMTs worked completely fault-free and delivered data in 34%. In the remaining events, failures resulted in partial to total data loss. The problems responsible for the failures included the loosening of soldered connections during transport over rough terrain, corrupted SD cards, and modifications/undocumented changes by the manufacturer in the specifications of individual components that led to incompatibilities with the Raspberry Pi. As technical failures such as these can only be identified during field work and/or during long-term deployment, they demonstrate the importance of thoroughly testing trap prototypes in order to achieve a truly operational design. For example, although the LED screen was initially thought to be beneficial for the standardization of photos, during the course of our study it became apparent that, while the LED screen guaranteed an even background, backlighting led to underexposure of the moths themselves. The use of a flash strong enough to evenly illuminate the background without underexposing the moths would make the LED screen obsolete, thus also reducing acquisition and energy costs.

An Honest Evaluation of the Costs and Benefits

As the aim of this study was to determine whether our AMT could provide information on moth abundances and the changes therein, the generation of a fully autarkic, automated monitoring system was beyond its scope. The power supply of the AMT tested in this study consisted of portable energy storage, since the number of AMTs was limited and the traps had to be shared between plots. However, in a long-term deployment the additional investment in photovoltaic systems or, in shadier locations, shade-tolerant solar panels, would be worthwhile. A similarly constructed bat monitoring system, described in a previous study [81], was likewise operated autonomously. In addition, a shorter illumination time would save energy; for example, a reduction to 30 min would save 7 Wh, or even 12 Wh without the LED screen (see chapter 4.2.2). For data transfer, rather than frequent visits to the trap sites to exchange the SD card, the research site could be remotely accessed. Data could be directly extracted using LTE sticks [81], or, when restricted by mobile phone reception, LoRa [61], which would again require additional infrastructure. A further necessity is data-processing software (a trained neural network) that would allow the Raspberry Pi to analyze and process the images in-field, thus reducing the size of the output files.

These investments in solar power and automatic data transfer will add to the current total material costs of 491€ per trap, in addition to incurring further running costs. Nevertheless, the cost of an updated version of the DIY-system presented herein would still be at the lower end of the price range of systems utilizing more elaborate components, such as high-end cameras (e.g., AMMOD [248]), or custom-built cases (e.g., `www.diopsis.eu`). The cost still obviously exceeds that of conventional traps, including the model used in this study (240€) but also those in other studies (e.g., [254]. The acquisition costs of automated monitoring systems will amortize over the long run, through a reduction of the workload both in the field and during AI-based post-processing. Our study did not address the latter as our focus at this stage was to acquire data on the number of individuals, without the uncertainties of AI, which despite showing promising results, has to be fully trained and validated. However, a full validation can be expected in the near future [111] and will allow automated identification to become nearly instantaneous. Admittedly, automated counting will not necessarily lead to a crucial cost savings relative to the acquisition cost of AMTs, as the post-processing costs of conventional traps are relatively low if only abundance data are considered.

The fastest amortization and greatest entomological value would be gained by automated species identification. The manual identification of species is both time-consuming and costly (yet also rewarding), and remains the greatest challenge in large-scale monitoring using conventional traps. For example, for this study the cost of sending the moths caught during each sampling event (night/plot) using the conventional trap to a specialist for identification would have been 110€ (on average, 218 individuals, current common price 0.5€ per individual). A preliminary test showed that a slight upgrade in the camera (from RPI CAM 5 MP, v1.3, resolution 2592 * 1944 px (6.55€) to RASP CAM HQ 12 MP with wide-angle lens, resolution 4056 * 3040px

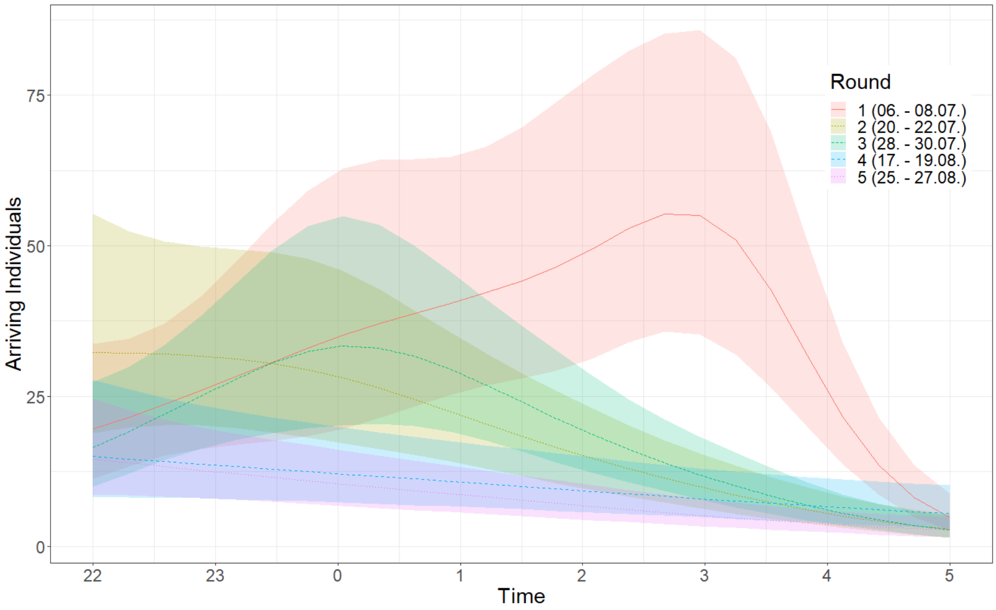


Figure 4.16: Generalized additive models (GAMs) of individuals arriving per time, listed by rounds. The round was inserted as a linear term; time was added using a cubic spline basis function, with $k = 5$. To account for the effect of the plot in the repeated measurements of arriving individuals, the trap location was included as a random effect. The colored lines represent the GAMs of the different rounds, and the shaded areas the confidence intervals. According to the model $R^2_{adj} = 0.722$, the largest number of individuals overall was captured during round one, i.e., the first and earliest of the rounds. Also, moth arrivals as recorded by the AMT in round one peaked at 3 a.m., whereas in the other rounds the peak occurred at 11 p.m. or 12 a.m.. Since data from sunset (around 9 p.m.) was missing, such that the earliest hour of moth activity was not included in the model, an even earlier peak may have been missed.

(88€)) and an adjustment of the illumination would produce images in which conspicuous species and genera could at least be identified [17, 134]. This upgrade would provide data not only on total moth abundance but also on species abundance and composition. Further studies need then to test for differences in the species composition obtained with the AMTs vs. conventional traps.

Potential and Outlook

The flexibility of the AMT, such as the possibility to customize the light regime, allows for a broad range of applications in both monitoring and research. First and foremost, the AMT provides additional information on the precise timing of the sightings [131]), and thus on the effect of environmental conditions, with high accuracy and detail (figure 4.11 and S7).

4.3 Unobtrusive Mechanism Interception

4.3.1 Introduction

Networked systems are often based on proprietary hardware/software components (e.g., applications, access points) and communication between them (e.g., network interfaces, system calls) that manufacturers might not be willing to adapt or update if new requirements arise. Possible reasons are: insufficient monetization of deprecated components and/or technical hurdles such as missing infrastructure. Furthermore, long and costly standardization and deployment processes may also hinder or slow down updates, which is not only true for proprietary components. Generally, this leaves users with no choice but to accept the non-optimal functionality of networked systems.

To add or modify functionality to/of existing networked systems or applications without touching proprietary components, developers use abstractions and workarounds that unobtrusively intercept and modify the communication between proprietary components and their environments. For example, developers introduced Network Address Translation (NAT) as a solution for insufficient IPv4 addresses, or Wine to run Windows applications on Unix-like operating systems.

We argue that such solutions are not exceptions, but essential for networked systems to provide highly desired improvements in a fast manner. However, there is no approach that can be used by developers to systematically identify possible starting points and implement such functionality.

Therefore, we present *mechanism interception*, a novel approach to implement functional additions to or modifications of existing networked systems without touching any proprietary components. We distinguish between *system*, i.e., components containing non-changeable parts, and *environment*, i.e., components containing modifiable parts under control. Behavioral changes are achieved by functionality-enhancing yet unobtrusive *interceptors*, i.e., hardware/-software components that are introduced between environment and system to add or update mechanisms representing some functionality. Examples of interceptors are an updated software library, a newly deployed edge device, or an enhanced cloud service. Interceptors must be unobtrusive to avoid disrupting or even destroying applications, but still provide added or modified mechanisms to the networked systems. We illustrate our approach by a case study, where we unobtrusively add a vertical handover mechanism between Wi-Fi and LTE to a mobile device without disconnecting existing TCP sessions. Our results indicate that mechanism interception is a compelling approach to achieve improved service quality and provide previously unavailable functionality in an unobtrusive manner.

Parts of this section have been published in Patrick Lampe, Markus Sommer, Artur Sterz, Jonas Höchst, Christian Uhl, and Bernd Freisleben. “Unobtrusive Mechanism Interception.” In: *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. IEEE. 2022, pp. 303–306.

4.3.2 Related Work

The basic idea of making network protocols extensible has been investigated in the context of active networks [236, 237]. ANTS [253] enables new protocols to be deployed on routers as well as terminals through platform-independent code. Pathak et al. [183] propose a system that provides applications with additional TCP interfaces and makes TCP adaptable in a fine-grained way. With software-defined networks [159] and the network programming language P4 [19], network programmability is provided. Tran et al. [242] present a method to add functionality to the TCP kernel implementation using eBPF. De Coninck et al. [43, 44] present a method to dynamically tune QUIC for each connection via extensions.

Alpine [53] and MultiStack [109] are userspace implementations of network stacks that allow changes to be tested quickly. An implementation of TCP in userspace is presented by Jeong et al. [115]. With NUSE, the network stack of the Linux kernel can be used as a userspace library [232] and existing programs can use modified protocols without modifications to the program itself. Heuschkel et al. [100] present VirtualStack, which allows different userspace network stacks to be used in one system. ClickNF [72] is an extension of a software-defined router in which the lower four layers of the network stack can be exchanged in a modular way.

Balinsky et al. [7] describe a method to prevent data leaks with system call interception and DAVINCI [51] uses system call hooking to build a fully transparent dynamic analysis tool for Android apps. Somy et al. [220] propose a sandbox architecture for serverless functions based on system call monitoring and whitelisting.

4.3.3 Unobtrusive Mechanism Interception

We motivate the novel concept of unobtrusive mechanism interception by the following examples:

Network Address Translation

The Internet Protocol version 4 (IPv4) was designed in the late 1970s, allowing only roughly 4.3 billion hosts in the Internet. At that time, the number of participating hosts in the Internet was rather low with less than 1000 and the designers of IPv4 did not anticipate, that the number of hosts in the Internet would increase dramatically as it did especially since the 2000s. Thus, they chose a 32 Bit field for addressing, allowing only roughly 4.3 billion hosts in the Internet. With over 5.5 billion smartphones globally [235], there are already more devices than the 4.3 billion hosts supported by via IPV4 not to mention servers, wearables, embedded systems and the entire IoT area. This development raised the need for solutions. Although there is a newer version, IPv6, for quite some time, allowing 2^{128} hosts in the Internet, its roll-out has been very slow. The protocol itself was already standardized 1998, but as of 2021, only 32% to 37% of the Internet's traffic is IPv6 based. To tackle the looming threat of address scarcity, Network Address Translation (NAT) has been introduced. Using NAT, a network gateway intercepts IP packets before routing and translates the IP addresses and TCP/UDP ports between an inbound, private IP address and an outbound, public IP address. When a reply arrives at the gateway, the IP addresses and TCP/UDP ports are translated back using the same mapping. Using NAT, it is not necessary to update or somehow alter the hosts of the private network, since NAT unobtrusively translates addresses without private hosts even noticing it. The observations that can be made for this example are the following. Thus, there is a system (i.e., the private host) that uses a mechanism (i.e., IP routing) provided by the environment (i.e., the gateway) to communicate with hosts in other networks. The NAT component adds a new functionality to the environment that intercepts IP packets and unobtrusively masquerades the IP addresses and TCP/UDP ports.

Wine

As of 2022, Windows is the dominant operating system for desktop computers, with a share of about 75%. Therefore, the Windows ecosystem contains the largest number of applications, many of which are not executable on Linux or other POSIX-compatible operating systems. To allow Windows applications to be executed on Linux or other POSIX-compatible operating systems, the Wine project[28] was initiated. It re-implements Windows system calls and libraries, such that system calls of Windows applications are translated to their corresponding POSIX equivalents. This example follows the same pattern as above. One could now argue that the Linux kernel could also support Windows system calls and system libraries. However, this would be time consuming and only solve the problem for the Linux kernel but not for, e.g., macOS, where Apple would be in charge re-implementing Windows system calls and system libraries.

These are just two examples. A further example is 6in4 or 4in6 tunneling, where the system can only use IPv4 or IPv6 and cannot be updated. The environment (i.e., a gateway) unobtrusively intercepts IPv4/IPv6 packets and encapsulates them in IPv6/IPv4 packets. Similarly, Talaminos-Barroso et al. [229] present a middleware that translates different Internet-of-Things (IoT) protocols to allow IoT devices that use incompatible protocols to communicate with each other. We argue that this approach is necessary in modern networked systems to allow adding new or update existing functionality on systems that are not controllable.

Mechanism Interception

In all examples, there is a system (i.e., a private host or a Windows application) that cannot be upgraded or modified. It uses mechanisms (i.e., IP routing or system calls) to achieve a specific task (i.e., communications between networks or executing a Windows application). Developers can control the environment (i.e., the NAT router or the operating system) and use a new mechanism that intercepts information from an old mechanism and translates it to the new mechanism (i.e., translating between private and public IP addresses or translating between Windows and POSIX system calls).

Our novel approach of unobtrusive mechanism interception is shown in Fig. 4.17. It consists of the following components:

[28] https://www.winehq.org

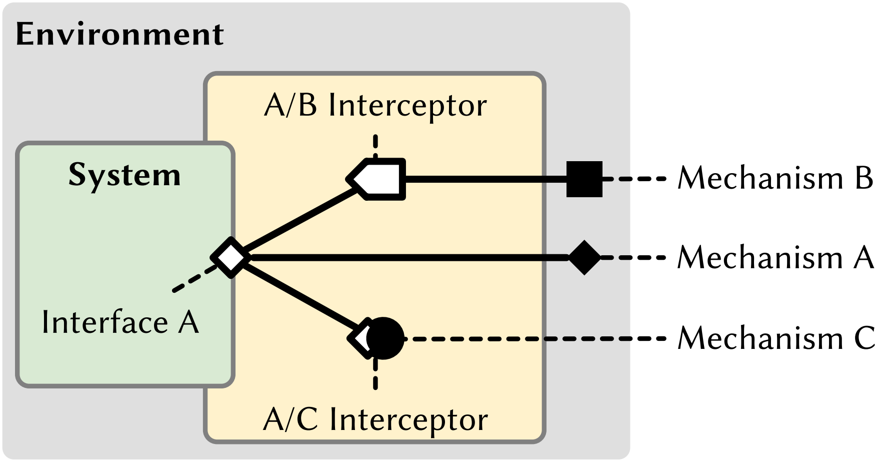


Figure 4.17: System, Environment, and Interceptor

System and Environment

We consider a system (green in Fig. 4.17) that is intended to achieve a particular task. The system or parts of it cannot be changed because it contains proprietary components or depends on other external factors, such as high coordination costs for changes due to long standardization or technical hurdles. In the NAT example, the system is the host that wants to communicate with hosts in other networks. The system is surrounded by an environment (gray in Fig. 4.17) that provides functionality to be used by the system. The system depends on the functionality provided by the environment to achieve the intended task. In the NAT example, the host needs the routing functionality of the gateway to enable communication with hosts in other networks.

Mechanism

A mechanism is the implementation of a functional unit of the environment used by the system via its corresponding interface to achieve its task [5]. The mechansim/interface mapping is not unique. For example, for the TCP mechanism, the interface could be the Sockets API or the Wi-Fi connection between two devices. In Fig. 4.17, the mechanisms are represented in black geometric figures and their interfaces in corresponding white geometric figures. In the NAT example, the mechanism provided by the environment that the system uses is IP routing and the corresponding network interface between host and NAT router.

Unobtrusive Interceptor

In the area of software engineering, the "interceptor architectural pattern allows services to be added transparently to a framework and triggered automatically when certain events occur" [208]. In this programming pattern, a framework provides interfaces so that programs using the framework can transparently intercept the flow of data at specific events. While the goal of our interceptor is similar, i.e., to transparently intercept the data flow to enable new functionality, the perspective is reversed. An interceptor in our approach is part of the environment and provides mechanisms to the system via interfaces, which are represented as combinations of geometric figures in the yellow area in Fig. 4.17. In contrast to the interceptor pattern, our interceptor is used by the system but configured by the environment. To allow a system to use the mechanisms, the interceptor is inserted between the system and the mechanism and changes the data flow. The used mechanism can also be introduced into the environment as part of the interceptor and does not have to exist before. In the NAT example, the newly added mechanism is *IP masquerading*.

To not influence or even disturb the actual task of the system, the existing mechanism must be unobtrusively replaced by the provided mechanism of the interceptor. The system should not notice the change of the mechanism; the used interfaces pretend that the original mechanism is still in place. Furthermore, the behavior of an interceptor must be unobtrusive so that the system can continue to provide the functionality and not provide error cases for its termination.

Guide to Action

To implement unobtrusive mechanism interceptors, we propose a 3-step procedure for developers.

Identification of Problem and Solution

The first step is to identify the problem that needs to be solved. Usually, problem identification is the result of prolonged use of a system where limitations occur or are noticed. Once the problem is identified, a solution must be found. Typically, there are several possibilities from which feasible solutions can be distilled. In the NAT example, the problem is IPv4 address scarcity, and the solution might be to translate addresses between multiple networks or to adjust routing on hosts so that unique IPv4 addresses are not necessary.

Identification of System and Environment

In the second step, developers need to identify the system in terms the problem to be solved, i.e., the component that cannot be changed or controlled, and the environment that can be controlled or changed by an interceptor. Some solutions of the first step must be discarded, since they would require a change of the system. In the NAT example, the system is the host that communicates with hosts in other networks, and the environment is the local network or NAT router. The alternative suggestion of adapting hosts so that IPv4 addresses do not have to be unique is omitted because the host is not changeable.

Identification of Mechanisms and Interceptors

In the third step, mechanisms must be found to solve the problem by considering the feasibility and unobtrusiveness of an interceptor that translates between the old and the new mechanism. This makes it essential that the mechanisms are functionally equivalent, for example, that they are on the same layer in the case of network protocols. In the NAT example, the functionally equivalent mechanism is an adaptation of IP routing. This interceptor implements an alternative IP routing mechanism, i.e., functionally equivalent to the old mechanism. Finally, the interceptor has to be implemented and deployed.

4.3.4 TunnelHandovers

We present the use case of an unobtrusive mechanism interceptor that uses common Linux mechanisms to enable handovers without TCP session disconnects, and without the need to roll out new, complicated, or niche protocols.

Problem and Solution

A vertical handover (henceforth simply called ”handover”) is the process of switching a networked device from one network access technology to another [106]. Probably the best-known example is the switch from a mobile phone’s Wi-Fi connection to a cellular connection, e.g., when leaving home on the way to the office. The core problem with handovers are protocols such as TCP, which were designed before wireless communication became popular. Thus, disconnections on the transport layer were not really considered during protocol design. If the connection on the transport layer is interrupted, the connections on the application layer must also be re-established, which can lead to different implications for different applications, such as interruptions of audio streams when listening to music or making phone calls. Several solutions to this problem have been proposed. There are standards that explicitly support mobility of nodes and implement mechanisms to support handovers, most notably Multipath-TCP, SCTP, and QUIC. However, all of these approaches have the core problem that they have to be deployed and supported by a major number of participants in the network. Furthermore, all of these solutions require application developers to update and adjust their applications to support these protocols.

System and Environment

The system in this scenario is an TCP-based application that we cannot change, e.g., by adding UDP functionality, since it is a pre-compiled application from some app store. The environment is the operating system, which we can change by implementing and deploying an interceptor.

Mechanism and Interceptor

We use two building blocks: WireGuard and `LD_PRELOAD`. WireGuard is a simple, fast, and secure VPN software that transmits data encapsulated in UDP datagrams. The use of UDP eliminates the problem of TCP connection losses during handovers, since UDP is not connection-oriented. For a WireGuard tunnel to work, a tunnel endpoint somewhere in the network is required, where the WireGuard tunnel is terminated and the encapsulated TCP connection is further relayed to the original destination.

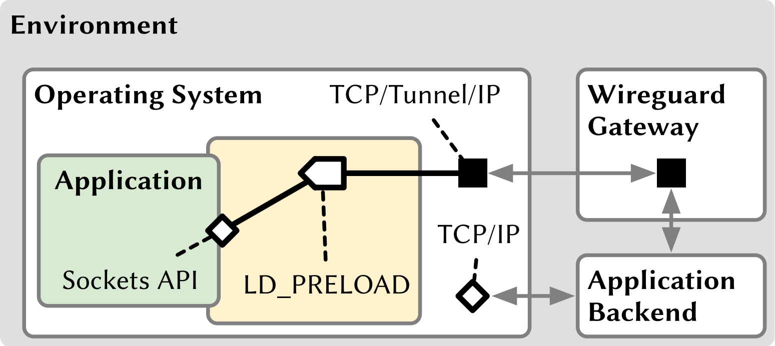


Figure 4.18: The system architecture for the TunnelHandover approach

The use of WireGuard alone does not automatically enable existing TCP-based applications to support handovers. Rather, they have to be instructed to transfer their data over the WireGuard connection. One way to do this is to modify the routing table of the system in a way that all traffic from the host is sent over the tunnel. However, this would mean that non-TCP based applications would also be routed through the tunnel, including UDP or QUIC based applications that do not have the problem of a missing handover capability, making the routing table an inappropriate interface for the TCP mechanism. To mitigate this potential drawback, `LD_PRELOAD` is used. The `LD_PRELOAD` mechanism allows intercepting functions of dynamically linked libraries, enabling us to intercept the Socket API such that only connections of the application started with our `LD_PRELOAD` modifications use the WireGuard tunnel. Thus, the mechanism *TCP/IP* is replaced by the mechanism *TCP/Tunnel/IP*. For this purpose, an interceptor is used that implements the Socket API used by the system, i.e., the application.

Fig. 4.18 presents the components of our TunnelHandover approach. A WireGuard daemon is executed on the operating system, indicated by the black square. When a program is executed with our interceptor, the TCP packets of this particular application are intercepted and redirected to the local WireGuard daemon using `LD_PRELOAD`. The UDP-based WireGuard packets are then sent to the WireGuard endpoint located in the network, which unpacks the encapsulated TCP connection and acts as a relay to forward the application data to the desired destination using conventional TCP/IP mechanisms. The responses of the server with which the application communicates are sent back to the host via the same route, i.e., via the WireGuard tunnel.

Experimental Evaluation

We evaluated the proposed interceptor for seamless handovers by emulation using the Common Open Research Emulator (CORE) in a leaving home scenario. Here, the handover is performed from Wi-Fi to LTE. The following nodes were emulated: client (with the TCP-based application to be intercepted), Wi-Fi access point (AP) and LTE access network (AN) to handover between, backbone router where both AP and AN are connected to for the network uplink, WireGuard gateway to terminate the WireGuard connection, and application server running the server of the client's application. To simulate a TCP-based application, we used iPerf3[29], where the

[29] `https://iperf.fr`

client initiates the connection to the server via Wi-Fi at experiment start. We also conducted experiments without the *TCP/Tunnel/IP* mechanism for comparison, using the regular *TCP/IP* mechanisms of the Linux kernel.

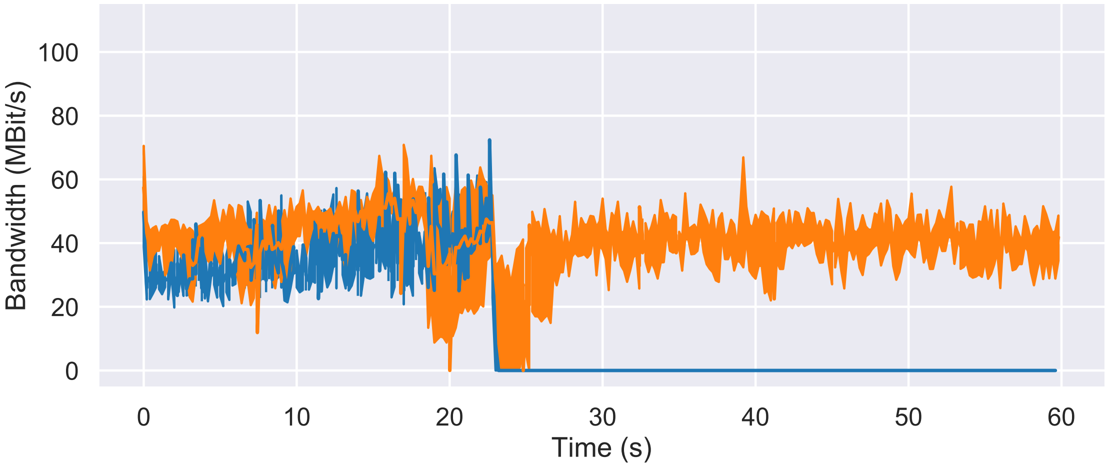


Figure 4.19: Network throughput with and without vertical handover

Fig. 4.19 shows the results for the conducted experiment. The x-axis denotes the experimental time, while the y-axis denotes the achieved bandwidth. The orange graph shows experiments where the proposed handover mechanism is enabled, the blue graph shows the results without employing the *TCP/Tunnel/IP* mechanism. Without the*TCP/Tunnel/IP* mechanism, the throughput drops to 0 because the TCP connection is not re-established after the connection is changed. Using the proposed `LD_PRELOAD` interceptor, a short throughput degradation is visible at around 25 seconds. This is due to the Wi-Fi connection being lost immediately, resulting in a short connection drop until the kernel recognizes the lost connection and propagates the changes to the routing tables.

4.4 ForestEdge: Unobtrusive Mechanism Interception in Environmental Monitoring

4.4.1 Introduction

Networked systems for environmental monitoring are often based on highly specialized integrated hardware and software components with a limited feature set. Many of these systems are created for exactly one use case, with a particular focus on cost. If such systems are operated for a long time, several components might not be state of the art anymore and might lack functionality that has become relevant over time. Furthermore, lengthy and costly standardization and deployment processes may also hinder or slow down updates, which is not only valid for proprietary components.

Sensor networks for environmental monitoring were previously criticized for being too costly to operate [150]. Advances in other fields, i.e., technologies borrowed from the Internet of Things (IoT) and machine learning, including denoising methods and suitable standards for communication, allow environmental monitoring applications to become smarter [244]. Using new technology, environmental monitoring can be realized in a more cost-efficient manner. An example is the *TreeTalker* platform. It is a low-cost networked system for monitoring the health of trees in a natural ecosystem [245], which is currently in use in many areas around the world[30]. The platform consists of sensor devices called *TreeTalkers (TT)* and a base station called *TTCloud*, which can be used to relay data received via LoRa and upload it to a cloud storage provider. Our university operates 100 TreeTalkers as part of the *Nature 4.0* research project[69]. The TreeTalker platform is proprietary and does not give buyers access to the software's source code. There is also no intention from the vendor that these devices can be upgraded after purchase. The original implementation is limited in functionality and restricted to pure data acquisition. Hence, modern advances and technologies, such as machine learning methods, anomaly detection algorithms, or dynamical measurements, are not utilizable in the TreeTalker platform.

In this section, we replace TTCloud with a versatile gateway based on open source software and components-off-the-shelf (COTS) hardware to allow dynamic reconfiguration of all TTs and to have a common basis for the automated analysis of collected data based on the novel concept of *unobtrusive mechanism interception* [141].

Parts of this section have been published in Patrick Lampe, Markus Sommer, Artur Sterz, Jonas Höchst, Christian Uhl, and Bernd Freisleben. "ForestEdge: Unobtrusive Mechanism Interception in Environmental Monitoring." In: *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. IEEE. 2022, pp. 264–266.

4.4.2 Background

To replace *TTCloud* with our gateway, we performed a black-box analysis of a *TT's* behavior and network communication to understand the communication protocol. TTCloud's command message contains four significant values influencing a TT's behavior. *Sleep* is the time the

[30] `https://www.nature4shop.com/our-vision/`

device will sit idle between measurements (default is one hour). *Heating time* is the time the TT will run its internal heater before taking the second heat-probe measurement. Finally, *time slot* and *slot length* govern the time the device will wait between measurements and sending measurement data; these values are multiplied to get the time to wait before transmitting. The waiting time between measurements is user-configurable during deployment. The time slot length will also be the same for all nodes, while the gateway will assign each TT a unique time slot. The only way for a user to access the collected data is by downloading a CSV file containing the measurement data from the vendor's web server or via a serial interface on the device. Neither a TT nor the cloud backend implements an interactive interface or an evaluation/error check of the data.

TTCloud is controlled via SMS. Collected data is sent to the cloud backend via a mobile data connection using a LoRa-based call-response protocol initiated by a TT. LoRa-certified hardware provides a dedicated network layer protocol, *LoRaWAN*, which handles collisions, addressing, and other issues resulting from the shared-medium characteristics of LoRa. However, the TT vendor decided to forgo LoRaWAN in favor of using the LoRa PHY layer directly. Thus, communication with each TT must be scheduled manually to avoid collisions. Especially in congested frequency bands, operating a TT network without collision avoidance is problematic, often resulting in poor reliability.

To add or modify the functionality of an existing networked system, we rely on the novel approach of *unobtrusive mechanism interception* [141]. Therefore, we distinguish between *system*, i.e., components containing proprietary or other non-changeable parts, and *environment*, i.e., components containing modifiable parts under our control. Behavioral changes are achieved by functionality-enhancing yet unobtrusive *interceptors*, i.e., hardware/software components introduced between the environment and the system to add or update mechanisms to represent some functionality. To showcase the use of mechanism interception, we illustrate our approach on a specific proprietary hardware system, namely the *TT*. We call the resulting interceptor *ForestEdge*.

4.4.3 ForestEdge

With the insights gathered from our black-box analysis, we can fully replace TTCloud with our open-source alternative, which allows us to bring mechanism interception to the TT network. By intercepting and redirecting TT messages, we can increase both the functionality and the reliability of the TT network: (a) we can make the system interactive, (b) we can perform statistical analyses on measurement data to find and rectify anomalies by triggering repeated measurements, and (c) we can maximize the battery lifetime by adjusting the measurement frequency to the remaining battery power.

According to the approach of unobtrusive mechanism interception, the following components are present: (a) The TT is the *system*; its goal is the collection of accurate environmental data and to make it available to users. (b) The *environment* is a forest-wide LoRa network that serves as the gateway to connect the set of TTs to the Internet. It receives the TTs' data via LoRa and can process and relay it to other systems in the Internet. ForestEdge is modular and allows the use of arbitrary communication technologies, such as WiFi, LTE, 5G, or a satellite uplink for data transmission. (c) The *mechanism* that allows the set of TTs to talk to

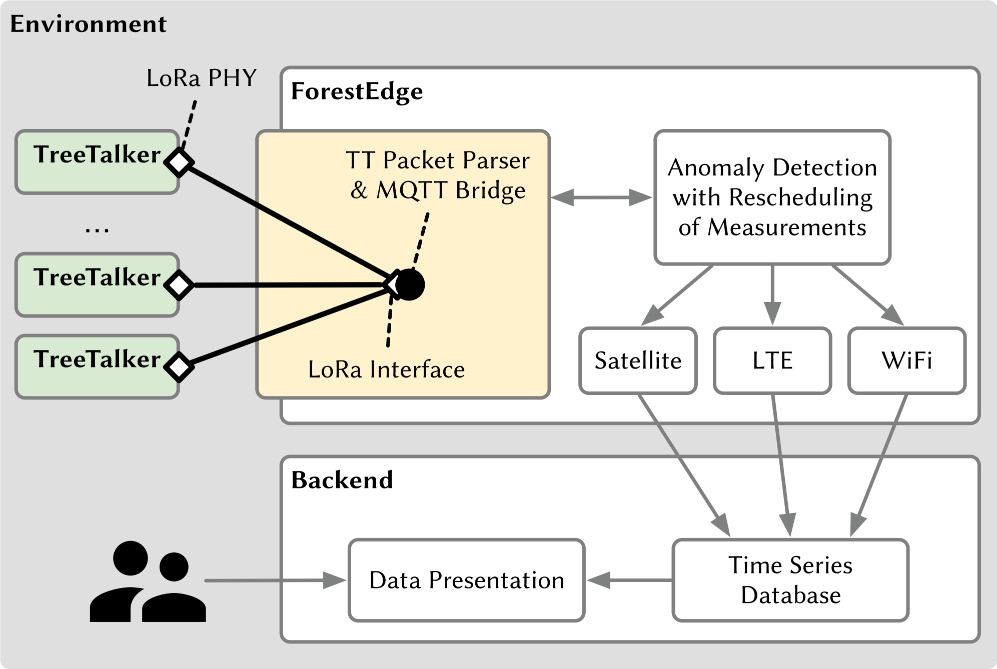


Figure 4.20: TT scenario with system, environment, and interceptor

the system is LoRa. (d) *ForestEdge* is the *interceptor*. It replaces TTCloud, communicates with the set of TTs via LoRa, and ultimately allows users to access the data in a more convenient format. Furthermore, it enhances the system's functionality by introducing anomaly detection algorithms, improving the data quality. This approach is unobtrusive, since the TTs cannot distinguish between TTCloud and ForestEdge.

Fig. 4.20 shows the architecture of ForestEdge. It is based on two separate decision-making entities. The local decision engine is located on the physical device, in this case, a Raspberry Pi, in the forest and communicates directly with a set of TTs. It only has data from its directly connected sensor nodes. This component is labeled *ForestEdge* in the figure; it consists of the actual interceptor, shown in the yellow area, which includes the LoRa transceiver and a translation layer that re-marshals the packets and sends them on via MQTT. The anomaly detection algorithm checks if a measurement seems reasonable, and if successful, the packet is transmitted onward via different communication technologies. ForestEdge decodes LoRa packets from a TT and submits the measurement data via MQTT. When the local decision engine receives data via MQTT, it stores it locally and forwards it to a global aggregator, which periodically aggregates data from all the nodes and computes relevant benchmark values. When generating a reply packet, the local decision engine performs a statistical evaluation of the corresponding TT's previous measurement data and how it relates to the aggregator-supplied benchmark values. For simplicity, we decided to use a statistical analysis where a value is labeled as an "anomaly" if it deviates from the expected value by more than two standard deviations. The expected value is computed from the ForestEdge node's local data and aggregated values supplied by the backend. However, employing a more sophisticated analysis method (e.g., based on machine learning) is possible with little to no modification of

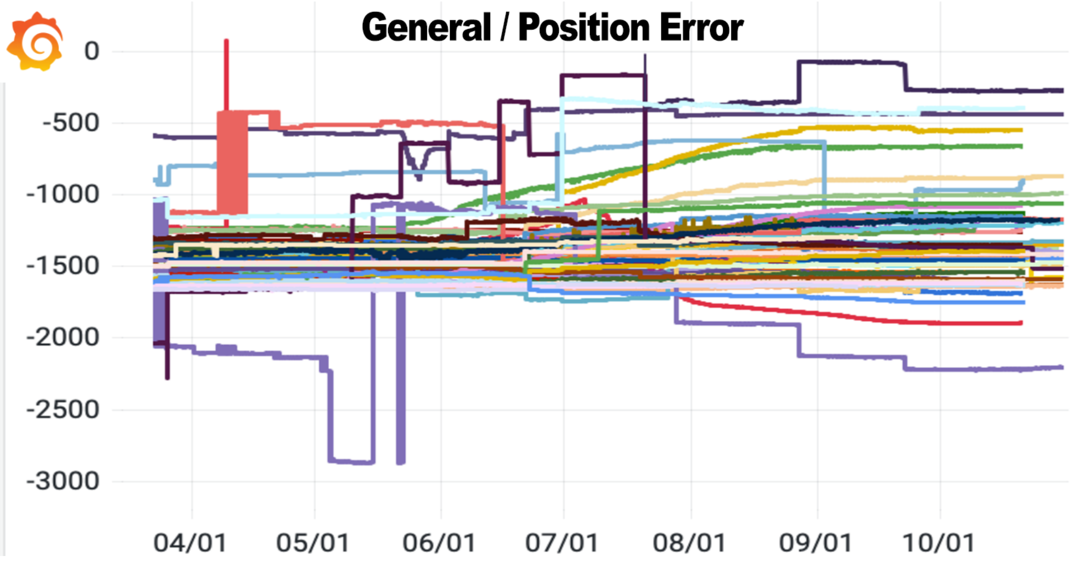


Figure 4.21: Erratic data from the deployed TT in the Marburg Open Forest

the architecture. If ForestEdge detects an anomaly, a TT is required to rerun its measurements by sending a command packet with an updated shorter sleep time.

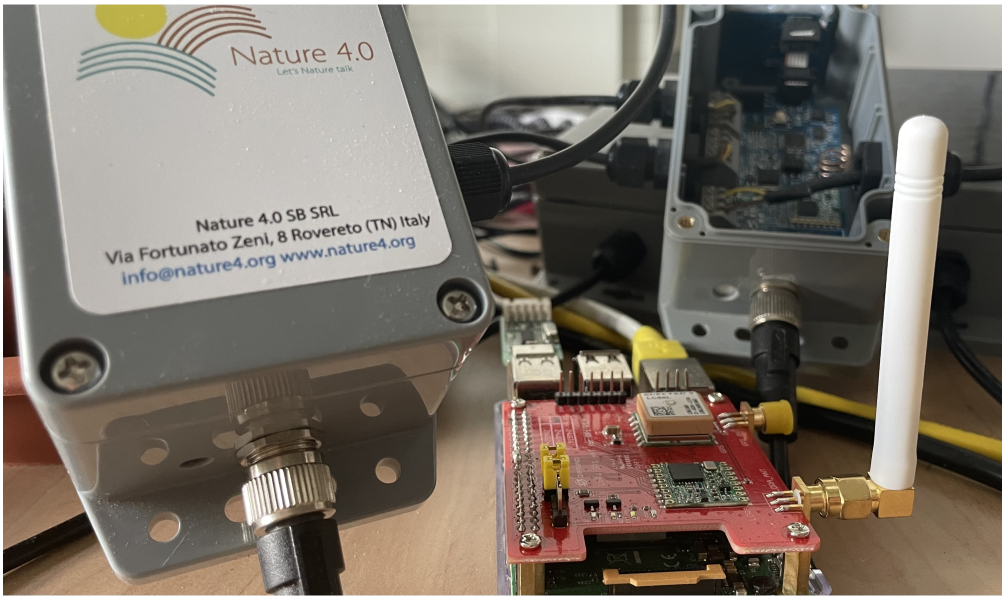
The second decision-making entity, labeled *Backend* in Fig. 4.20, is a global decision engine, i.e., a centralized aggregator in the backend. It has global knowledge of all nodes in the network. The backend receives data from all ForestEdge instances and saves them in a time-series database to aggregate and redistribute them to aid stations in making statistical evaluations. It also contains a presentation layer through which users can view the data.

4.4.4 Anomaly Detection at the ForestEdge

A disadvantage of the TT devices is the fluctuating quality of on-device measurements. To compensate for this shortcoming, we implemented local anomaly detection, which can trigger a new measurement. To make meaningful decisions, we compare a long-running and a short-running time window of the data. If we assume the data is normally distributed, the calculation of the twofold standard deviation for short-running windows is already sufficient to make decisions, e.g., whether a new measurement is necessary or these new values can be appointed as the new standard. In general, the preferred decision is to reduce the upcoming measurement interval to verify and increase the accuracy of the collected data. If no anomaly is found, the sleep time until the subsequent measurement is calculated based on the TT's reported remaining battery capacity. This maximizes the device's lifetime running with a battery by reducing the measurement frequency when the solar panel produces little energy and increases the frequency when there is a power surplus. We used real-world data gathered from 100 TTs over the past two years to evaluate whether our anomaly detection works as intended. The evaluation was performed by injecting data into the global decision engine in chronological order, thus doing a replay.

Table 4.11: Anomalous and critical events found in historical data

Type	Position	Movement	Stem Temp.	Air Temp.
Anomalous	2182	108	443	0
Critical	6313	105	183	1



Nature 4.0
Let's Nature talk
Nature 4.0 SB SRL
Via Fortunato Zeni, 8 Rovereto (TN) Italy
info@nature4.org www.nature4.org

Figure 4.22: Original TreeTalker (left) and the ForestEdge hardware (right)

Table 4.11 indicates that most events are produced by the accelerometer and, more specifically, by the values denoting the sensor's position. Further investigation of the data revealed that this seems to be due to a systematical failure in either the installed sensor itself or the method in which the onboard controller queries it. The data reported by the TT for this sensor seems to be completely erratic and not suitable as a basis for further decision-making, as shown in Figure 4.21. This failure had previously not been noticed by the operators, but with the aid of ForestEdge, we were able to discover it before it could impact other decision-making processes.

4.5 *Bird@Edge*: Bird Species Recognition at the Edge

4.5.1 Introduction

The continuous loss of biodiversity is particularly evident from the sharp decline of bird populations in recent decades. Birds are important for many ecosystems, since they interconnect habitats, resources, and biological processes, and thus serve as important early warning bioindicators of an ecosystem's health. Thus, changes in bird species in time and space should be detected as early as possible.

Traditionally, this is achieved by human experts who walk around a natural habitat to look at birds and listen to bird sounds, identify bird species present in the sounds, and take notes of their occurrence. In recent years, this is often supported by placing microphones in the natural habitats of birds and recording their sounds. The audio data recorded in this way is then evaluated either manually by human experts or by means of automatic analysis methods to recognize bird species in soundscapes. The disadvantages of this approach are: (a) there is a potentially large amount of recorded audio data that can usually only be evaluated after the end of the recording time, and (b) there is an inherent time delay between recording the audio data and delivering the recognition results.

In this section, we combine Edge Computing and Artificial Intelligence (AI) to present Bird@Edge, an *Edge AI* system for recognizing bird species in audio recordings to support real-time biodiversity monitoring. Bird@Edge is based on embedded edge devices operating in a distributed system to enable efficient, continuous evaluation of soundscapes recorded in forests. Multiple microphones based on ESP32 microcontroller units (called Bird@Edge Mics) stream audio to a local Bird@Edge Station, on which bird species recognition is performed. The recognition results of different Bird@Edge Stations are transmitted to a backend cloud for further analysis, e.g., by biodiversity researchers.

To recognize bird species in soundscapes, a deep neural network based on the EfficientNet-B3 architecture is trained and optimized for execution on embedded edge devices and deployed on a NVIDIA Jetson Nano board using the DeepStream SDK. Our experimental results show that our deep neural network model outperforms the state-of-the-art BirdNET neural network on several data sets and achieves a recognition quality of up to 95.2% mean average precision on soundscape recordings in the Marburg Open Forest, a research and teaching forest of the University of Marburg, Germany. Measurements of the power consumption of a Bird@Edge Station and the Bird@Edge Mics highlight the real-world applicability of the approach. All software and firmware components of Bird@Edge are available under open source licenses[31]. Our contributions are:

- We present a novel Edge AI approach for recognizing bird species in audio recordings; it supports efficient live data transmission and provides high-quality recognition results.
- We propose a deep neural network based on the EfficientNet-B3 architecture optimized for execution on embedded edge devices to identify bird species in soundscapes.

[31] `https://github.com/umr-ds/BirdEdge`

- We evaluate our Edge AI approach in terms of recognition quality, runtime performance, and power consumption.

Parts of this section have been published in Jonas Höchst, Hicham Bellafkir, Patrick Lampe, Markus Vogelbacher, Markus Mühling, Daniel Schneider, Kim Lindner, Sascha Rösner, Dana G. Schabo, Nina Farwig, and Bernd Freisleben. "Bird@Edge: Bird Species Recognition at the Edge." In: *International Conference on Networked Systems (NETYS)*. Springer. May 2022, pp. 69–86.

4.5.2 Related Work

In this section, we discuss related work with respect to current machine learning approaches for bird species recognition in audio recordings and edge AI approaches for biodiversity monitoring.

Bird Species Recognition

For many years, bird populations were monitored manually by ornithologists who identified birds visually and acoustically on site. The introduction of autonomous recording units (ARU) opened new possibilities. Although such passively recorded data does not provide any visual information, the resulting bird surveys conducted by humans from sound recordings are comparable to traditional monitoring approaches in the field [42].

Furthermore, machine learning methods, such as Convolutional Neural Networks (CNN), are increasingly being used for automatically recognizing bird species in soundscapes. For example, BirdNET is a task-specific CNN architecture trained on a large audio data set using extensive data pre-processing, augmentation, and mixup that achieves state-of-the-art performance [121]. The audio spectrograms are generated using a Fast Fourier Transform (FFT) with a high temporal resolution. BirdNet is based on a ResNet [97] architecture and is capable of identifying 984 North American and European bird species.

More recently, BirdNET-Lite[32] has been released. This neural network is optimized for mobile and edge devices and can recognize more than 6,000 bird species. It takes raw audio as its input and generates spectrograms on-the-fly. Mühling et al. [169] proposed a task-specific neural network created by neural architecture search [269]. It won the BirdCLEF 2020 challenge [119]. It also operates on raw audio data and contains multiple auxiliary heads and recurrent layers.

Recently, Vision Transformers (ViT) achieved great improvements in computer vision tasks [49] and audio event classification[78]. Puget [192] adopted a ViT architecture for bird song recognition and achieved results comparable to CNNs. However, the annual birdcall identification challenge (BirdCLEF [120]) is currently dominated by approaches based on CNNs. The top approaches typically use ensembles of CNNs and heavy parameter tuning. The winning approach at BirdCLEF 2021, for example, uses Mel spectrograms, network architectures based on ResNet-50 [97], and gradient boosting to refine the results using metadata. The runners-up Henkel et al. [99] presented an ensemble of nine CNNs. During training, they used 30 second

[32] `https://github.com/kahst/BirdNET-Lite`

Mel spectrograms to mitigate the effect of the weakly labeled training data and applied a novel mixup scheme within and across training samples for extensive data augmentation. Furthermore, a binary bird call/no bird call classifier contributed to the final result. However, combining several machine learning models leads to a considerably increased computational effort.

Edge AI for Biodiversity Monitoring

Executing machine learning algorithms on edge devices leads to a quantitative increase of data through continuous observation, where previously only individual data points could be collected with manual effort, often including a bias of individual experiences depending on, e.g., habitat or bird species. Merenda et al. [161] survey several approaches based on the execution of machine learning methods on hardware with limited resources. Gallacher et al. [71] deployed 15 sensors in a large urban park to process recorded audio data of bats locally, which allowed monitoring their activities for several months. Given that the system has only been operated in an urban environment, the limitations of this approach are that network connectivity must be available via WiFi, and that a fixed power supply must be present. Novel deep learning approaches presented by Disabato et al. [48] further improved bird song recognition at the edge. These approaches provide high accuracy while reducing computational and memory requirements, with limited battery lifetimes of up to 12.4 days on an STM32H743ZI microcontroller. Likewise, Zualkernan et al. [271] compare different edge computing platforms based on neural networks using bat species classification as an example. While the NVIDIA Jetson Nano is the only device capable of executing a TensorRT model on its GPU, both the Raspberry Pi 3B+ and the Google Coral showed good results when executing a reduced TensorFlow-Lite model.

4.5.3 Bird@Edge

Bird@Edge is designed as an *Edge AI* system based on distributed embedded edge devices to enable efficient, continuous evaluation of soundscapes recorded in forests. Multiple Bird@Edge Mics stream audio wirelessly to a local Bird@Edge Station, on which bird species recognition is performed. The recognition results of different Bird@Edge Stations are transmitted to a backend for further analysis. The results are stored in a time series database and can be visualized, as shown in Fig. 4.23. Using hidden microphones also supports recognizing very elusive species that are hard to detect while ecologists are present in field to conduct a census.

A Bird@Edge Station consumes significantly more power than a microphone node, but can run a neural network for bird species recognition for more than one audio stream. We can feed 1 to 10 audio streams into the neural network and thus operate a variable number of Bird@Edge Mics at one Bird@Edge Station. Different numbers of Bird@Edge Mics may be present when a new microphone node appears (e.g., by switching it on) or leaves (e.g., due to battery shortage).

To generate a list of bird species at a Bird@Edge Station, chunks of an incoming audio stream are passed to the neural network, which may return multiple results, since we process mixtures of recorded bird songs, i.e., soundscapes. These potentially multiple results per audio segment are then collected and aggregated into larger intervals in the time series database in the

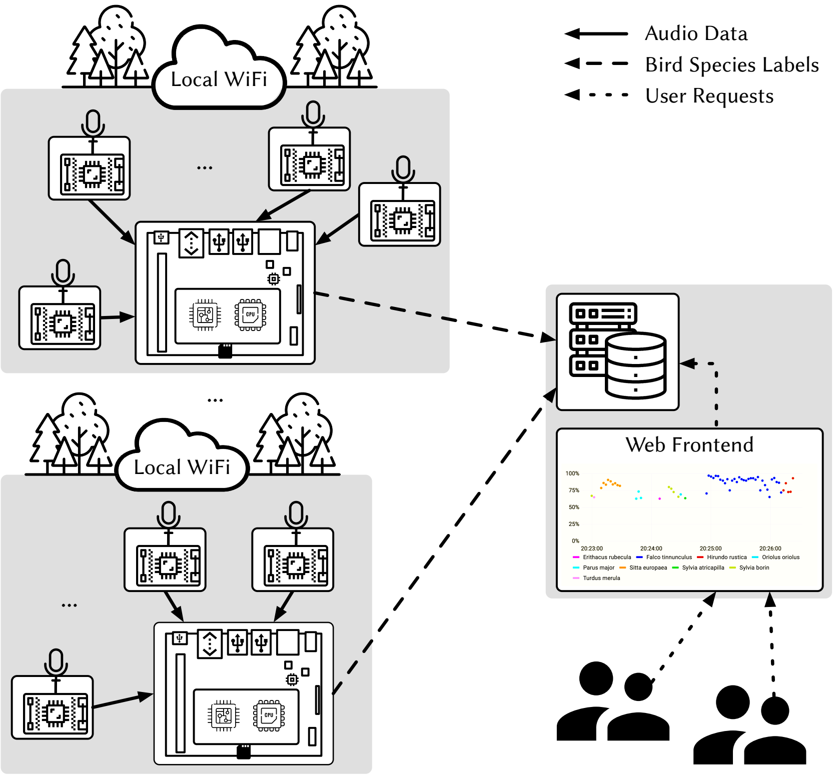


Figure 4.23: Overview over the Bird@Edge system

backend cloud. The size of the interval can be dynamically changed and visualized in near real-time. In addition, the status of the microphone nodes and potential problems can be detected much faster than collecting data only every few days.

Bird@Edge Hardware

The hardware used for Bird@Edge consists of (a) Bird@Edge Mics, which are in charge of recording and transmitting audio at the deployed location; (b) Bird@Edge Stations, which receive audio streams from multiple Bird@Edge Mics and execute the Bird@Edge processing pipeline. Figure 4.24 provides an overview of the hardware components used in Bird@Edge.

A Bird@Edge Mic consists of an Espressif ESP32 microcontroller that has a dual-core CPU running at 80 MHz, Bluetooth and WiFi connectivity, as well as multiple input and output options, including an I2S (Inter-IC Sound) bus. Connected to it is a Knowles SPH0645LM4H microphone capable of recording audio in the range between 50 Hz and 15 kHz[33]. A Bird@Edge Mic can be powered either using single 18650 Li-ion cells or using one of the widely available USB power banks. The price of a Bird@Edge mic of 22€ to 50€ is composed of the ESP32, depending on the offer and model 5€ to 15€, the I2S microphone 7 - 12€ and a battery for 10€ -

[33] https://www.knowles.com/docs/default-source/default-document-library/sph0645lm4h-1-datasheet.pdf

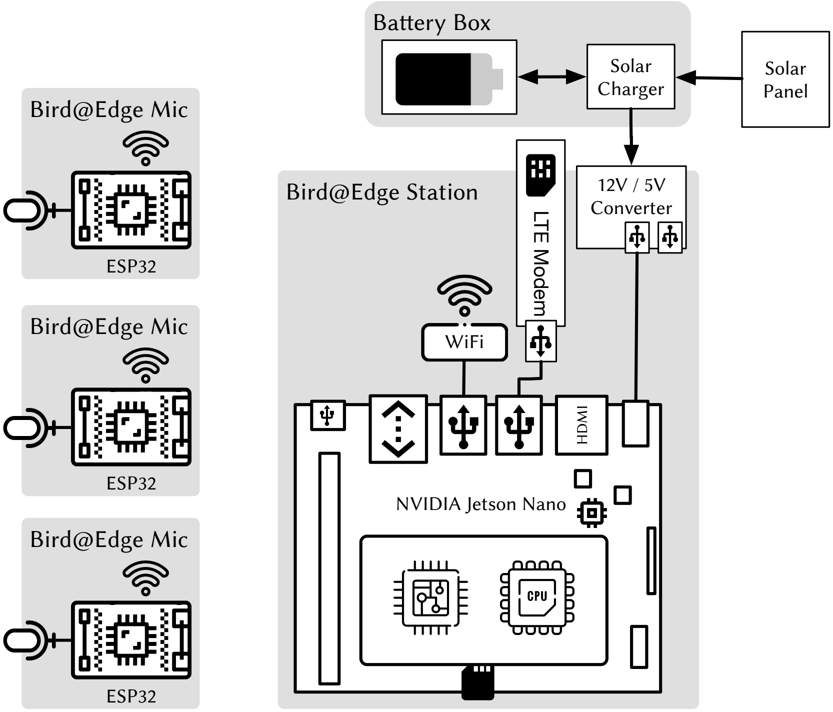


Figure 4.24: Bird@Edge hardware components

20€. All components can be placed in a small case of 10 x 10 x 5 centimeters, which does not exceed the weight of 500 grams.

At the heart of a Bird@Edge Station is a NVIDIA Jetson Nano. It allows the efficient execution of machine learning models in a low power environment. A Realtek RTL8812BU-based USB WiFi is used to enable wireless networking with the board and allow connection to the Bird@Edge Mics. In addition, a Huawei E3372H LTE modem is installed to connect to the Internet in rural areas. The station is powered by 12V solar battery system connected to the Jetson Board via a 12V/5V step down converter. The hardware of a Bird@Edge Station costs about 110€, with 50€ for the Jetson Nano, 20€ for the USB WiFi adapter, and 40€ for the LTE modem. The components of an Bird@Edge Station, including a solar charge controller, can be fitted into an industrial enclosure measuring 25 x 18 x 12 centimeters, weighing less than 1.5 kilograms in total.

Bird@Edge Software

Bird@Edge consists of a variety of software components that enable its smooth configuration and operation. Figure 4.25 shows these software components, as well as the data flows and interaction possibilities of the users with the system.

The software for the Bird@Edge Mics is built using components of the Espressif Development Framework (ESP-IDF), i.e., HTTP Server, Multicast DNS Implementation, and I2S drivers. When booting up, the Bird@Edge Mic connects to the WiFi network (SSID: BirdEdge) with the best

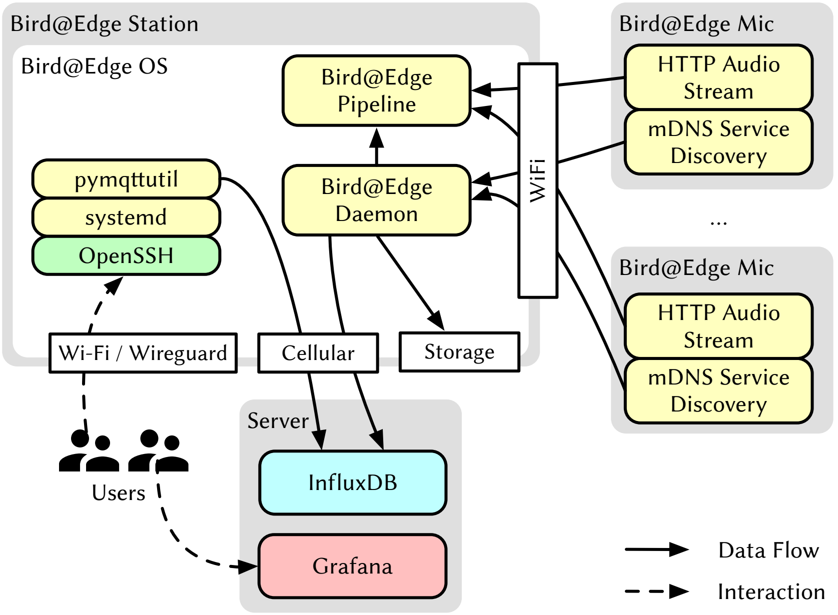


Figure 4.25: Bird@Edge software components

signal strength and reports its own accessibility via the service identifier mDNS. Then, the HTTP server is started, which provides the audio stream of the microphone for the Bird@Edge station. To detect connection interruptions, the WiFi connection is also checked at intervals of one second with the aid of ICMP and, if necessary, the WiFi connection is re-established. The Bird@Edge Mic software is available online[34].

The software running on a Bird@Edge Station is based on the NVIDIA Jetson Nano Development Kit Operating System, which in turn is based on Ubuntu Linux. The central component responsible for detecting the Bird@Edge Mics, executing the processing pipeline and transmitting the results is called birdedged (Bird@Edge Daemon). It continuously searches for newly connected Bird@Edge devices and restarts the processing pipeline accordingly when devices are found or dropped. Bird species recognition results from the processing pipeline are captured and transmitted to the InfluxDB server system. The server system that collects data from potentially multiple Bird@Edge implementations runs Grafana, a dashboard visualization WebUI designed specifically for stream data[35].

The operating system running on Bird@Edge Stations is built using pimod [105] and is available online[36]. NVIDIA's licenses do not allow to redistribute complete operating system images, however pimod allows to reduce the necessary steps and easily create the images.

[34] https://github.com/umr-ds/BirdEdge/tree/main/BirdEdge-Client
[35] https://grafana.com
[36] https://github.com/umr-ds/BirdAtEdge-OS

4.5.4 Recognizing Bird Species in Soundscapes

In this section, we describe our deep learning approach to bird species recognition in audio recordings including the preprocessing steps, the neural network as well as its optimization and deployment on the NVIDIA Jetson Nano edge device. The deep neural network model is designed to recognize 82 bird species indigenous in Germany and background noise that is typical for German forests.

Audio Preprocessing

We selected 44.1 kHz as the sampling rate and analyzed frequencies up to 22.05 kHz to cover the frequency ranges of the bird song patterns. The task is considered as a classification problem, aiming to recognize bird species in 5-second audio snippets. To avoid overfitting and enrich our data set, we randomly select these 5-second snippets and add randomly selected noise from up to four background samples. This encourages our model to focus on the patterns that are important for species recognition. The recognition is based on visual representations of the frequency spectrum as it changes over time, called spectrograms. In our case, we use Mel spectrograms that are generated using 128 Mel bands and an FFT window size of 1,024.

Neural Network Architecture

Our approach to recognize bird species relies on an EfficientNet-B3 [230] architecture pre-trained on ImageNet [201]. The model is fine-tuned in two phases to target domain using the Adam [127] optimizer. In the first phase, we only train the last, randomly initialized layer for 40 epochs with an initial learning rate of 0.004, while the remaining layers with pre-trained weights are frozen. In the second phase, we train all layers of the model until convergence, while reducing the initial learning rate by a factor of 10. Furthermore, a binary cross-entropy loss combined with modulation factors motivated by the success of focal loss [148] in the field of object detection are used to emphasize difficult samples during the training process. Since the underlying data set is only weakly labeled, we use positive training samples for one species as negative samples for the others. Furthermore, samples labeled negative from expert feedback are defined as hard negatives in the following. Our loss function is defined as follows:

$$L = \sum_{k=1}^{K} l(y_k, p_k),$$

$$l(y,p) = \begin{cases} -\alpha_{pos}(1-p)^{\gamma}\log(p) & \text{if } y \text{ is positive} \\ -\alpha_n p^{\gamma}\log(1-p) & \text{if } y \text{ is negative} \\ -\alpha_{hn} p^{\gamma}\log(1-p) & \text{if } y \text{ is hard negative} \end{cases}$$

where K is the number of bird classes, p_k is the predicted probability for the k-th class, y_k is the k-th ground truth label, α_{pos} is the weighting factor for positive labels, α_n for negative or undefined labels, α_{hn} for hard negative labels and γ is the focusing parameter.

We implemented our approach using the TensorFlow deep learning framework [157]. For audio processing and especially spectrogram generation, we use the librosa library [158].

Optimizing the Neural Network for Edge Devices

To speed up inference, we optimized our model using the TensorRT[37] library. This library includes an inference optimizer for CUDA-capable target devices that applies various operations, such as quantization and memory optimization, to reduce the inference time. In particular, the floating point precision is reduced by quantizing to FP16 or INT8, while maintaining high accuracy. We optimized our model by using FP16 quantization in addition to the original FP32 weights, since the NVIDIA Jetson Nano does not support INT8 computations natively. Furthermore, we applied target-specific auto-tuning to select the best algorithms and quantization for each layer.

Inference

We use the DeepStream SDK[38] to deploy our optimized model on the NVIDIA Jetson Nano board with high throughput rates. DeepStream is based on the GStreamer framework and provides a pipeline that takes an input stream and performs hardware accelerated inference on it. An overview of our pipeline composed with DeepStream is presented in Figure 4.26. First, the N HTTP streams are read and parsed from the WiFi signal. Since the microphone we used (see Section 4.5.3 for details) induces noise in the lowest frequency bands, we apply a highpass filter that attenuates all frequencies below 120 Hz to each stream. These frequencies are irrelevant for bird species recognition and can therefore be neglected. We prefer the Chebyshev highpass filter over the windowed sinc filter, because it is much faster. Next, we use DeepStream's stream muxer to bundle our streams into one batch and forward the data to the NvInferAudio plugin. This plugin provides inference for audio streams and automatically generates the respective

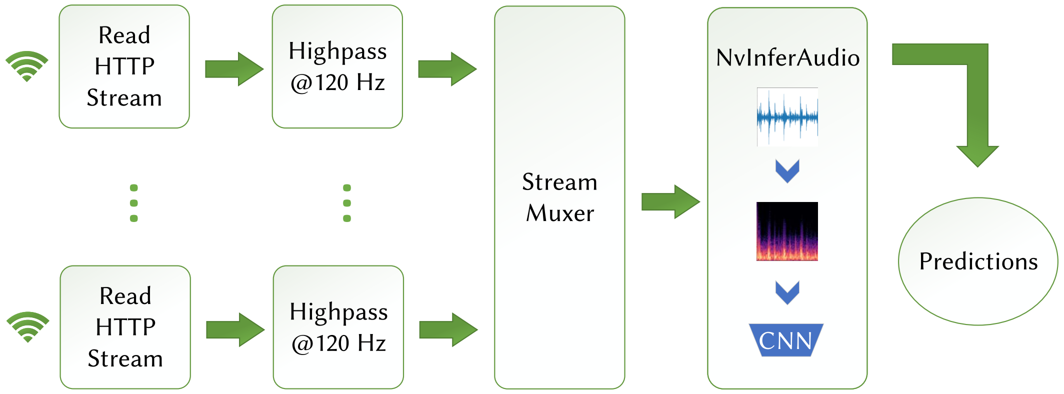


Figure 4.26: Overview of the Bird@Edge processing pipeline

[37] https://developer.nvidia.com/tensorrt

[38] https://developer.nvidia.com/deepstream-sdk

Mel spectrograms. Finally, the spectrograms are passed to our model with a batch size of N, and the obtained predictions are retrieved. To be able to process the streams in real-time with a high temporal resolution, we take 5 second snippets with a stride of one second.

4.5.5 Experimental Evaluation

In this section, we present experimental results in terms of (a) bird species recognition quality and execution speed, (b) visualization of bird recognition results, as well as (c) power consumption measurements of a Bird@Edge Station and a Bird@Edge Mic.

Bird Species Recognition Quality and Execution Speed

Data Sets.

Our neural network models were evaluated and compared to BirdNET [121] and BirdNET-Lite[32] on data sets collected from three sources. We recorded a first data set with AudioMoth devices [101] in the Marburg Open Forest (MOF). The recordings were labeled on a 5 second basis by human experts. In total, 33 species occur in the MOF data set. Since the amount of labeled data in the MOF data set is not sufficient to train a deep learning model, we acquired further data sets by crawling data from the online bird song collections Xeno-Canto [264] and iNaturalist [114]. The assets included in these data sets have often higher quality and contain less background noise. In our evaluation, we took up to 10% of the files of each class. To make sure that we do not feed snippets without bird calls, we first applied the heuristic used by Kahl et al. [121] and selected up to three 5 second snippets containing a bird call for each test file. Table 4.12 shows an overview of the training and test data.

Table 4.12: Overview of the training and test data.

Data Set	MOF	Xeno-Canto	iNaturalist
Training	4,294	104,989	30,631
Test	913	2,144	1,365

Quality Metrics.

To evaluate the performance of our bird species recognition approach, we use average precision (AP) as our quality metric. The AP score is the most commonly used quality measure for retrieval results and approximates the area under the recall-precision curve. The task of bird call recognition can be considered as a retrieval problem for each species where the annotated audio samples represent the relevant documents. Then, the AP score is calculated from the list of ranked documents as follows:

$$AP(\rho) = \frac{1}{|R \cap \rho^N|} \sum_{k=1}^{N} \frac{|R \cap \rho^k|}{k} \psi(i_k),$$

$$\text{with} \quad \psi(i_k) = \begin{cases} 1 & \text{if } i_k \in R \\ 0 & \text{otherwise} \end{cases}$$

where N is the length of the ranked document list (total number of analyzed audio snippets), $\rho^k = \{i_1, i_2, \ldots, i_k\}$ is the ranked document list up to rank k, R is the set of relevant documents (audio snippets containing a bird call), $\left|R \cap \rho^k\right|$ is the number of relevant documents in the top-k of ρ and $\psi(i_k)$ is the relevance function. Generally speaking, AP is the average of the precision values at each relevant document. To evaluate the overall performance, the mean AP score is calculated by taking the mean value of the AP scores from each species.

Table 4.13: Results (mAP).

Method	MOF	XC	iNat
BirdNET[121]	0.833	0.725	0.725
BirdNET-Lite[32]	0.859	0.737	0.714
EfficientNet-B3	0.952	0.820	0.811
Bird@Edge	0.952	0.816	0.819

Table 4.14: Model inference runtimes.

Model	Device	Inference time (ms)
BirdNET-Lite[32]	Raspberry Pi-4B	279
Bird@Edge (FP32)	Jetson Nano	64
Bird@Edge	Jetson Nano	54

Results.

First, we evaluated the recognition quality of our models, namely the original trained model (EfficientNet-B3) as well as the optimized model (Bird@Edge) and compare the results to BirdNET and BirdNET-Lite. While EfficientNet-B3 is evaluated with TensorFlow on a workstation, the Bird@Edge model is run on the NVIDIA Jetson Nano for inference.

BirdNET and BirdNET-Lite take the recording location as additional metadata along with the corresponding audio input. As longitude and latitude, we take the coordinates of the Marburg

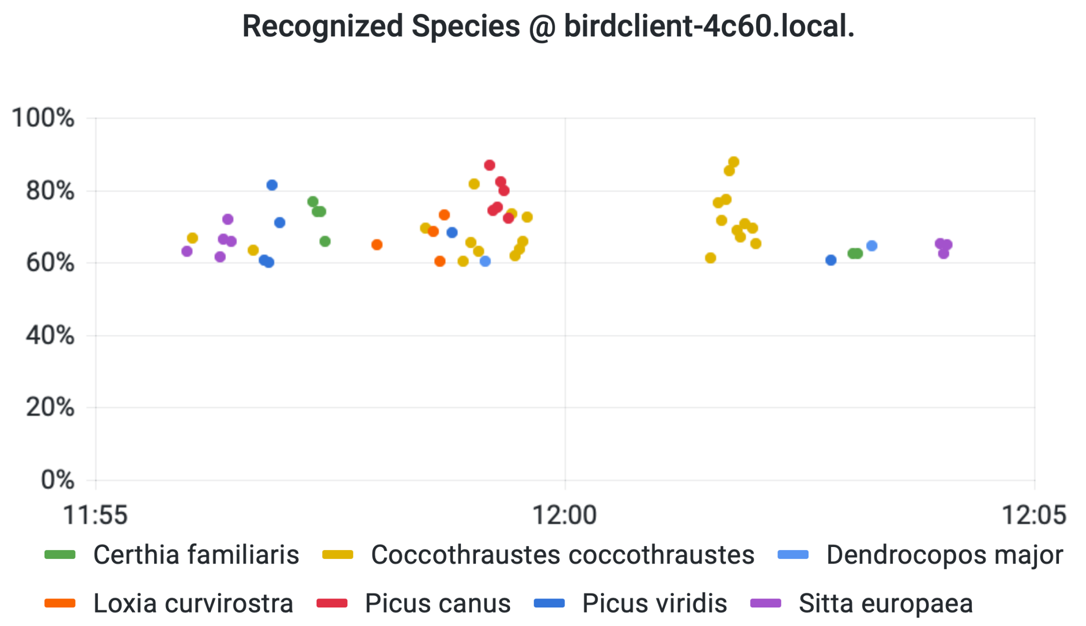


Figure 4.27: Grafana panel (x-axis: clock time; y-axis: recognition confidence) showing recognized bird species of a certain Bird@Edge Mic, based on Xeno-Canto file XC706150, recorded by user *brickegickel*

Open Forest for all data sets, since we only use bird species resident in this specific forest for evaluation. Since the length of the audio input of the BirdNET models differs from our approach, the 5 second samples are split into two 3 second snippets with an overlap of 1 second and the results are averaged for the final prediction.

Table 4.13 summarizes the experimental bird species recognition results. Our original model (EfficientNet-B3) outperforms BirdNET-Lite as well as BirdNET by roughly 10% in terms of mAP on all data sets considered. While keeping the recognition quality, the optimized Bird@Edge model achieves an inference runtime of 64 ms per spectrogram, as shown in Table 4.14. Adding model quantization with 16-bit floating point precision where appropriate effectively reduces the inference runtime on the NVIDIA Jetson Nano board by 10 ms. We also compared the runtimes of our models to BirdNET-Lite. Similar to BirdNET-Pi[39], we ran BirdNET-Lite on a Raspberry Pi-4B with 4 CPU threads in parallel. Table 4.14 reveals that our setting is more than four times faster.

Visualization of Bird Species Recognition Results

Figure 4.27 shows a Grafana screenshot of an automatically generated graph of the recognized bird species of a Bird@Edge station. To generate the figure, the publicly available soundscape audio file XC706150 from Xeno-Canto[40] of the target area was played back and captured by

[39] https://github.com/mcguirepr89/BirdNET-Pi

[40] https://xeno-canto.org/706150

the Bird@Edge Mic. The clock time is shown on the x-axis. The confidence of the recognition is plotted on the y-axis. The data is grouped according to the recognized bird species labels, distinguished by color. For every Bird@Edge Mic, a separate figure is generated, and its parameters, e.g., plotted time frame or selection of species, can be configured.

Some observations can be derived from this simple visualization. First, there are several recognized occurrences of *Coccothraustes coccothraustes* (hawfinch) in two clusters. *Picus canus* (grey-headed woodpecker) is detected multiple times over the duration of 12 seconds, and *Sitta europaea* (Eurasian nuthatch) is detected in two clusters each at the beginning and end of the observation period. All three observations indicate that individuals were heard on the recordings and were in the area at these times. For *Loxia curvirostra* (red crossbill) and *Dendrocopos major* (great spotted woodpecker), only 4 and 2 observations were made, respectively; these were probably heard only in the background. More sophisticated analyses can be performed based on researchers' requirements, such as heat maps of the occurrence of species based on their geo-positions, or time-based plots. This can include both short-term considerations, such as the time of day at which certain species are active, or long-term aspects, such as during which period a particular species is particularly active.

Power Consumption

An important aspect for the applicability of Bird@Edge in real applications is its power consumption. Therefore, we measured the power consumption of a Bird@Edge Station and a Bird@Edge Mic.

To measure the power consumption of a Bird@Edge Station, we used the internal power monitors of the NVIDIA Jetson Nano, since these enable the differentiation between CPU, GPU, and total power consumption. The power measurements were performed in different profiles: a) the 10 Watt maximum performance profile (default), b) the 5 Watt low power profile from NVIDIA, and c) a custom low power profile created for Bird@Edge. In this custom power profile, only 2 of the 4 CPU cores were used, running at a maximum frequency of 614 MHz, and the GPU was limited to a maximum of 230 MHz. As a baseline, the power consumption is measured with 5 connected Bird@Edge mics, and only the measured values while the pipeline is running are averaged. In this setup, the maximum performance mode requires 4.86 W, the low power profile 4.19 W and the custom low power mode only requires 3.16 W, i.e., roughly 35% compared to the maximum performance mode with no performance degradation observable. Our observations during the execution of the experiments suggest that the GPU's dynamic frequency scaling algorithm tends to be too conservative to permanently lower the clock and thus prevents the possible lower power consumption.

Figure 4.28 shows the power consumption of a Bird@Edge station in a short scenario with a changing number of connected Bird@Edge Mics. At the beginning, the system is switched on, but the neural network for bird species is not running; the system needs 2.1 W in this state. At t=0, the neural network is started with a Bird@Edge Mic already connected to the station. The neural network model is loaded into memory from t=6, for which the CPU requires up to 0.59 W. From time t=36, i.e., 30 seconds after the start of the pipeline, the neural network model runs and forwards results to the backend. In this phase, the Bird@Edge station requires an average of 3.18 W. At t=120 and t=180, 4 and 5 additional Bird@Edge Mics are switched on, which first

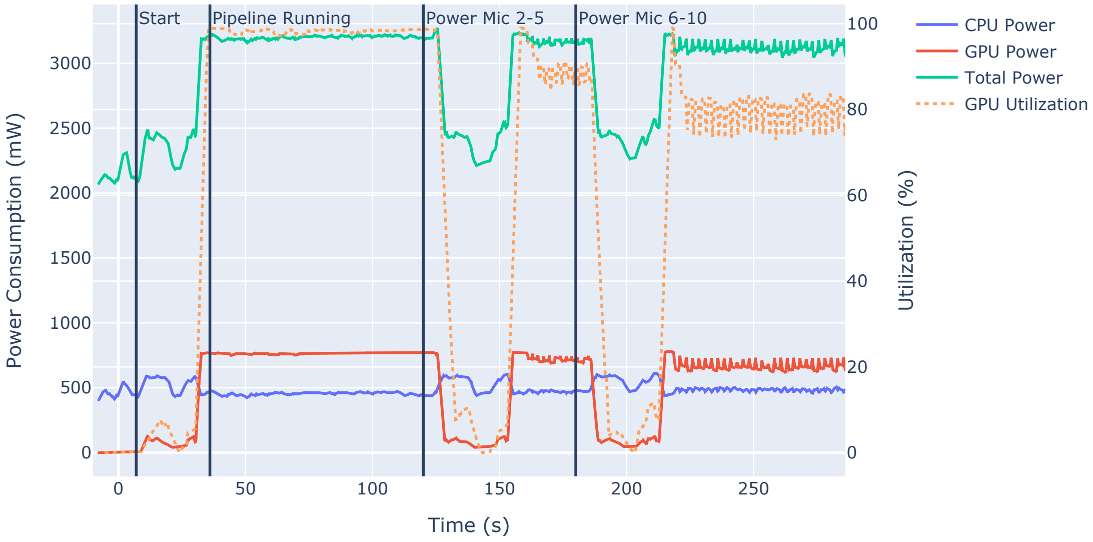


Figure 4.28: Power consumption of a Bird@Edge Station in a dynamic scenario.

connect to the Bird@Edge Station via WiFi, then are discovered via mDNS, which results in the reconfiguration of the processing pipeline and its restart. In both cases, the reboot took about 35 seconds, with 5 seconds for WiFi connection and discovery, and 30 seconds for pipeline reboot. With 5 and 10 Bird@Edge Mics connected, the Bird@Edge Station requires 3.16 W and 3.13 W, respectively. Particularly noteworthy is the station's lower power consumption when a larger number of Bird@Edge Mics are connected. Figure 4.28 indicates that GPU utilization is lower with many Bird@Edge Mics connected, compared to smaller numbers of Bird@Edge Mics (98% with 1 client, 88% with 5 clients, 80% with 10 clients). This is probably due to internals of the DeepStream SDK and may be influenced by the implementation, e.g., with respect to the handling of unused streams.

The magnitude of the Bird@Edge Station's power consumption necessitates the use of an LFP, VRLA or AGM battery, which at 12 Volts typically have a capacity of 50 to 200 Ah. The available 60 to 2400 Wh allow the operation of a Bird@Edge Station from 7 up to 31 days. In combination with a solar panel between 50 and 100 watt, a continuous operation is also possible during periods of weak sunlight.

To measure the power consumption of a Bird@Edge Mic, we used a Monsoon High Voltage Power Monitor[41] and connected the ESP32 using the 3.3 volt input. The measurements of a Bird@Edge Mic show a power usage of 665 mW whenever the stream is activated and data is sent to the station. The power measurements were performed with three different off-the-shelf ESP32 boards, since the additional electronics present on the boards can have an additional impact on power consumption. The three boards differed only slightly in terms of power consumption: 0.452 W was needed by the *Adafruit HUZZAH32*, 0.452 W by the *Joy-IT NodeMCU ESP32*, and 0.421 W by the *SparkFun ESP32 Thing Plus*. The latter[42] is the board

[41] https://www.msoon.com/high-voltage-power-monitor

[42] https://www.sparkfun.com/products/15663

of choice for our application, due to the lowest power consumption, a direct LiPo battery connector, and an external WiFi antenna.

To get a realistic estimation of the battery life of a Bird@Edge Mic, further measurements were performed with 3.7 Volt via the corresponding connectors for LiPo batteries. The SparkFun board required 0.468 W or 126.6 mA in operation, whereas the Adafruit board required 132.9 mA, or 0.492 W. LiPo batteries are available in a wide range of capacities, from 100 mAh to over 30000 mAh. Typical capacities, as they are found in smartphones and can be purchased cheaply, are around 3500 mAh, which allow a runtime of 27.6 hours. In combination with a small solar panel of around 10 Watts, continuous operation is thus easily feasible.

4.6 *SmartFace*: Efficient Face Detection on Mobile Devices for Wireless On-demand Networks

4.6.1 Introduction

During ecological monitoring, a common task is to take photos of scenes in the nature to detect leaf sprouts, observe bats, or spot for wild animals. This images are taken to get ecological insights. In most cases, taking photos of humans are not the primary goal, but can be used as an indicator for disturbances by humans. For this purpose, humans have to be detected on photos and especially their faces. Faces belong to personal data in the EU, and storing images with personal data may be a legal problem.

For desktop and server systems, several face detection libraries with different properties in terms of recall and precision are available, but their resource requirements do not match with the resources of today's mobile devices, since the latter still only provide a fraction of the computing power offered by a contemporary workstation.

In this section, we present a novel approach, called *SmartFace*, to perform face detection *in situ* on smartphones or tablets in an efficient manner. The approach is based on a novel two-stage combination of state-of-the-art face detection algorithms, further enhanced by region of interest selection, color space/depth reduction, resolution scaling, face size definition, image scaling, image cropping, and bounding box scaling. *SmartFace* improves the face detection rates within the same runtimes or obtains the same face detection rates within faster runtimes compared to the individual face detection algorithms used alone, and also reduces the amount of data that needs to be stored on disk and sent over the network. As a consequence, the battery life of mobile devices is extended, too. The main contributions are:

- We present a novel two-stage processing pipeline for resource-efficient face detection on mobile devices, with improved overall face detection rates and runtimes.
- We present an experimental study of image preprocessing parameters to obtain algorithm agnostic speed gains.

Parts of this section have been published in Patrick Lampe, Lars Baumgärtner, Ralf Steinmetz, and Bernd Freisleben. "Smartface: Efficient Face Detection on Smartphones for Wireless on-demand Emergency Networks." In: *24th International Conference on Telecommunications (ICT)*. IEEE. 2017, pp. 1–7.

4.6.2 Related Work

Ecological monitoring networks [190] typically rely on radio technologies, such as Bluetooth, LoRaWAN, WiFi, TETRA digital radio or satellite links. They either use telecommunication infrastructures, are distributed (peer-to-peer, mobile ad-hoc networks), or form hybrid architectures of both.

Several approaches utilize commodity mobile devices to realize hop-to-hop DTN [32, 73, 238]. Due to their peer-to-peer nature, they are well suited for infrastructure-less scenarios. Projects such as FireChat[43], Briar[44], Serval[45] and Forban[46] transfer messages and files between heterogeneous devices and share them locally through store-and-forward technology.

Furthermore, several face detection algorithms have been proposed. Viola and Jones [247] offer an approach in *OpenCV* that accelerates face detection. The tradeoff is that with an increased detection rate, the false positive rate also increases.

Felzenswalb et al. [58] present an object detection system for *dlib*. Since a sliding window over an entire image in different resolutions is used, the runtime increases with increasing image size. Therefore, small interesting regions within an image should be selected before the algorithm is applied.

Cheney et al. [31] perform a comparison of face detection algorithms. The authors state that often used test sets, such as LFW[112] and YouTube Faces [260], do not pose any challenges for current algorithms and should not be used any more for benchmarking. Instead, they use IJB-A [129] that consists of over 5000 images and 20,000 video frames for benchmarking, The main open source algorithms highlighted in their paper are from *Dlib* and *OpenCV*, the first one with good detection rates and the second one being one of the fastest.

The typical approach to perform face detection on mobile devices is to offload images to a server and run a face detection algorithm on the server [113, 221]. Using powerful servers makes it much easier to achieve good face detection rates, but in an ecological monitoring scenario such servers are often not available.

Feng et al. [59] present a cascaded classifier approach that has also been tested on a Samsung S6 smartphone, resulting in a runtime of 34 ms for detecting faces in a 640 x 480 pixel photo. Compared to current 8 or 16 megapixel cameras in today's smartphones, this size is very small, and no public code is available to verify the results.

4.6.3 *SmartFace*

The goals of *SmartFace* are as follows:

- Since the photos with recognized faces are deleted the false positive rate of the used face detection approach should be as low as possible.
- Since a data transfer can only happen after face detection has finished, the runtime of the used face detection approach should be minimized. This also means that the false negative rate and the false positive rate of the used face detection approach cannot be minimized at the same time, but the false negative rate should be as low as possible in accordance with the available computational resources on mobile devices.

To achieve these goals, our basic idea is to develop a two-stage processing pipeline in which the first stage is responsible for quickly selecting interesting regions in a given image, and the second stage performs in-depth face detection and validation in the interesting region selected

[43] http://opengarden.com/firechat

[44] https://briarproject.org/

[45] http://www.servalproject.org

[46] http://www.foo.be/forban/

in the first stage (see Fig. 4.29). Thus, the first stage is optimized for speed, favoring high recall over high precision. The second stage is optimized for quality, favoring high precision over high recall. By using adequate face detection algorithms that support the competing objectives of each stage, overall speed and quality improvements can be achieved.

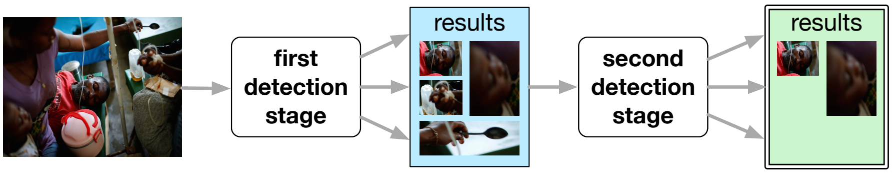


Figure 4.29: Basic two-stage face detector

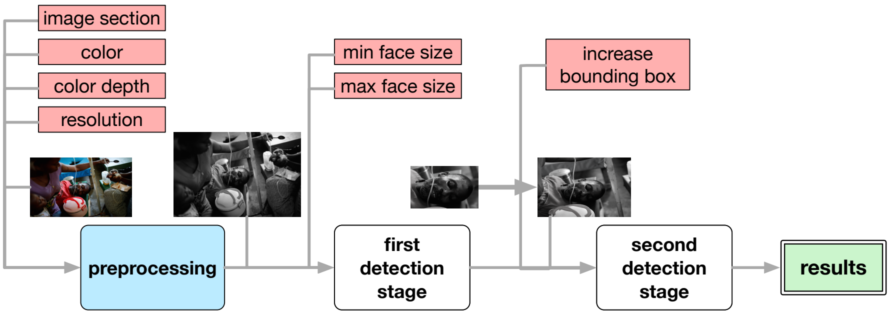


Figure 4.30: Two-stage face detector, preprocessing, and parameters

This two-stage face detection approach can be flexibly configured to the preferences of a particular usage scenario, e.g., (a) by reducing the runtimes to save battery power, (b) by improving the quality of the face detection results by running the algorithms for a longer time, or (c) by trying to get the best results in a fixed amount of time.

There are several parameters that can change the runtime and the quality of the results of a face detection algorithm without changing the algorithm itself. To support the goals of *SmartFace*, we consider the following set of parameters and their values:

Color space: color / greyscale / black-and-white
Color depth: 24-bit / 8-bit / 1-bit
Resolution: $100 \leq n \leq 5000$ in steps of 100 pixels
Image scaling: area, cubic, lanczos4, linear, nearest interpolation
Cropping: remove from each border 0%, 10%, 20%, 30% of the full image
Face size: min: 80 x 80 pixels; max: 1000 x 1000 pixels
Bounding box scaling $50 \leq n \leq 500$ in steps of 50

Some of these parameters are relevant for operations in a preprocessing step, while other parameters are applied to the first or second detection stages, respectively, as shown in Fig. 4.30. For example, having a larger bounding box in the second stage might increase the number of results, but might overlap with nearby faces in the same region. Here, duplicates have to be detected and eliminated. This is achieved by allowing bounding boxes to overlap to a certain degree after the second detection stage.

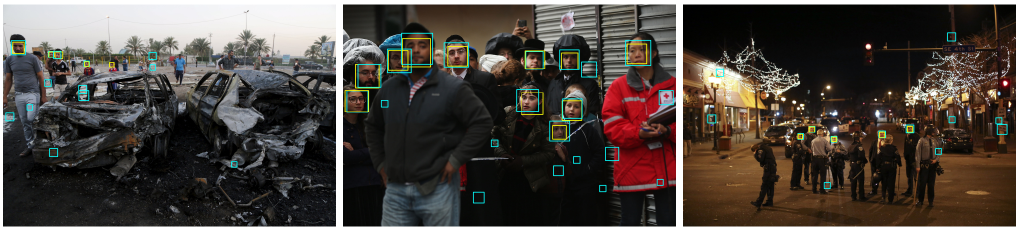


Figure 4.31: *SmartFace* in action

4.6.4 Implementation Issues

The implementation of *SmartFace* is based on three main components. The first component handles the parameters and the preprocessing steps. The second component is a fast face detection algorithm for the first stage of *SmartFace* to select regions of interest. The third component is a high-quality face detection algorithm for the second stage of *SmartFace* to operate on the selected regions of interest.

OpenCV's Viola/Jones algorithm [247] is used in the first stage, since it is a common out-of-the-box solution for face detection, in contrast to *OpenCV's* SURF that is also suitable for general object detection. It is comparatively fast [31], but has a relatively high false positive rate (which is acceptable as a pre-filter), as shown in Fig. 4.31 by the blue bounding boxes in each image. *Dlib* is used in the second stage. It is slower than the Viola/Jones algorithm, but has a lower false positive rate and a higher precision [31]. The faces detected by *dlib* are shown by the yellow bounding boxes in Fig. 4.31.

All parameters except the face sizes are independent of the used face detection algorithms. Therefore, the algorithms can be replaced by alternatives, still benefiting from the rest of our optimizations. The entire program flow is shown in Figure 4.32.

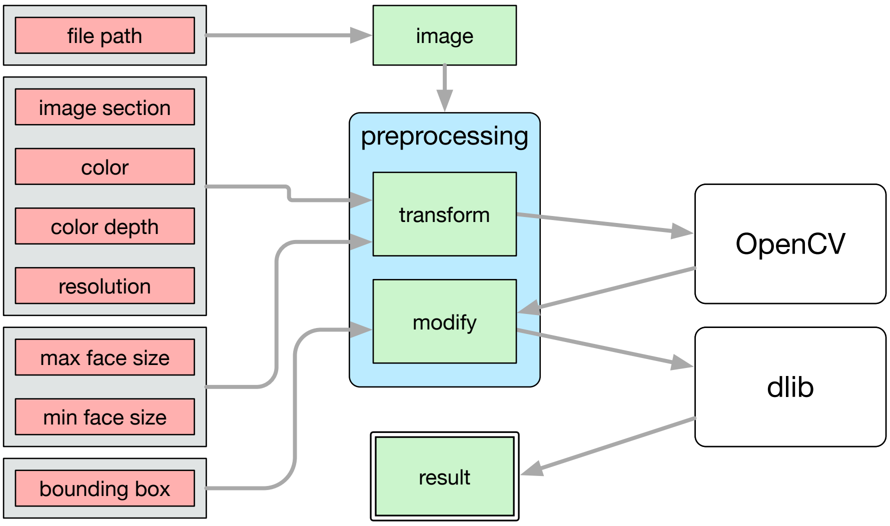


Figure 4.32: *SmartFace* implementation

To determine suitable parameters for our scenario, we performed a parameter scan by creating a test environment in *Bash* and using the created API to automatically iterate over the parameter sets using our image test set. The parameter scan is first performed with *dlib* to match the results to the behavior of *OpenCV*. The results are presented in Section 4.6.5.

4.6.5 Experimental Evaluation

Test Environment

We use three different devices in our experimental evaluation, as shown in Table 4.15: (a) a workstation as a reference platform, (b) a smartphone (OnePlus 3T), and (c) an older tablet (Nexus 7). The workstation operates under Ubuntu 16.04 LTS, and both Android devices use Android release 6.0.1.

Table 4.15: Device Specifications

Device	CPU	RAM	Storage
Workstation	Quad-core (i7-2600 @3,4GHz)	8GB	256 GB SSD
OnePlus 3T	Quad-core (2x2,35 GHz & 2x1,6 GHz)	6GB	128 GB Flash
Nexus 7	Quad-core (4x1,5GHz)	2GB	32 GB Flash

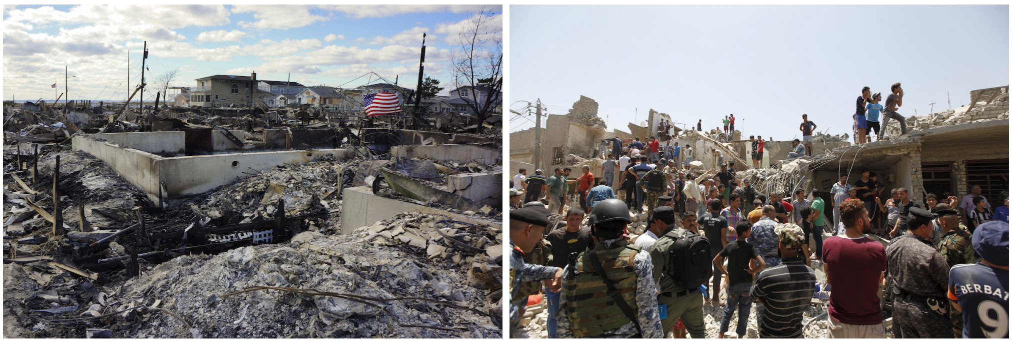
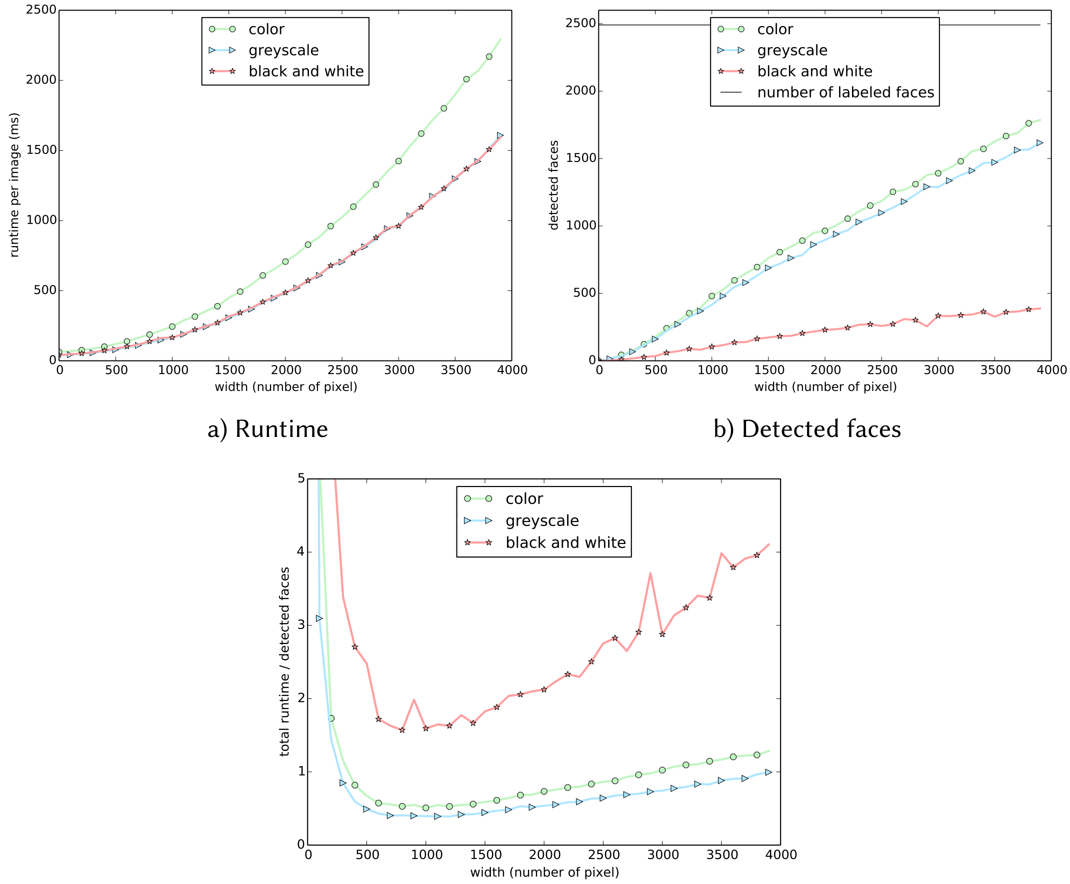


Figure 4.33: Examples from image test set

Image Test Set

To evaluate *SmartFace* on realistic images taken under challenging conditions, we created our own test set by randomly downloading images from the Internet (two examples are shown in Figure 4.33) using one of the search terms from the following list: *haiti earthquake, haiti earthquake people, haiti earthquake faces, earthquake faces, earthquake people, disaster people, disaster faces, disaster management, flooding people, flooding faces, natural hazard people, natural hazard faces, tsunami people, tsunami faces, night bushfire, bushfire people, firefighter disaster, explosion people, accident people, syrian civil war, crowd, crowd disasters, crowd control, crevices earthquake.*



a) Runtime

b) Detected faces

c) Runtime per #detected faces

Figure 4.34: Comparison of different color spaces and depths.

We developed a crawler for *Google Images* and downloaded the top 100 search results, finally obtaining a total of 2,400 images. The crawler downloads images with a resolution of 3000 x 2000 pixels. Thus, these images correspond to photos taken by a 6 megapixel camera. Smartphones can take photos with a higher resolution, but in a typical scenario with energy constrains it is not realistic to assume that top-notch hardware is commonly available. Thus, 6 megapixel photos are quite realistic and much more sophisticated than commonly used face detection test sets, such as LFW where images have a resolution of 250 x 250 pixels, and a large image part is covered by a face.

After removing duplicates, our test set consists of 1,481 images. This leads to 2.8 GB image data that needs to be processed in our benchmarks. In addition, we have manually labeled the faces for each image in our test set. It contains 2,419 faces to be detected. A list with 2-D coordinates is stored for each image in the test set. These coordinates represent a central point in the face, in many cases this is the tip of the nose. Furthermore, we follow the rule suggested by Cheney et al. [31] to only mark faces where both eyes can be seen.

Parameter Scan

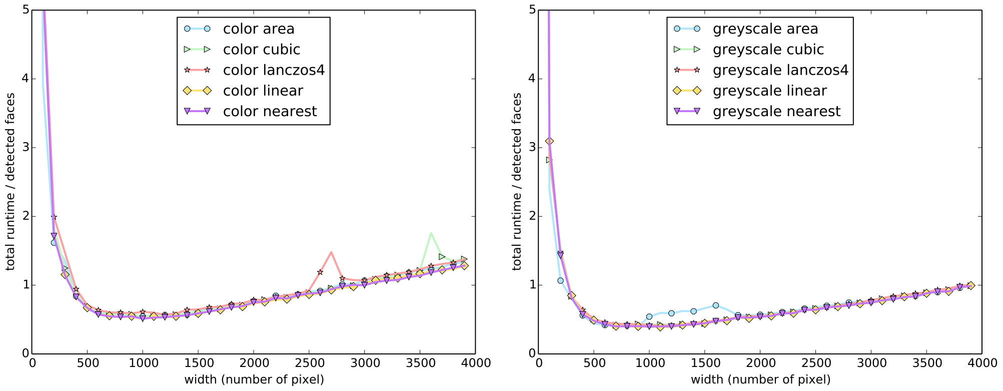
To evaluate the selected parameters, we present results performed with *dlib* below; each preprocessing experiment is repeated 10 times. The results using *OpenCV* are similar.

Color Space and Depth

In this experiment, the 24-bit input image is reduced to an 8-bit greyscale image and to a 1-bit black-and-white image. As indicated by Fig. 4.34a, the runtimes for greyscale and black-and-white images are nearly identical, and both are faster than for the fully colored image. This effect increases with increasing image size. Changing color space and depth also affects the detection rates, as shown in Fig. 4.34b. Using a greyscale image produces almost as good results as using the original image, but black-and-white yields significantly worse results. Fig. 4.34c shows the combination of the first two graphs; lower numbers are better. Again, greyscale and full color images are similar in their score, with a slight edge for greyscale due to its runtime being faster than its decrease in detection rate.

Image Scaling

These experiments are performed on the original and greyscale images; black-and-white is not considered due to the results from the color space tests. Although the most common interpolation mechanisms for scaling images such as *linear, area, cubic, nearest and lanczos4* produce slightly different images, they are all comparable in terms of runtime and number of detected faces (see Fig. 4.35).



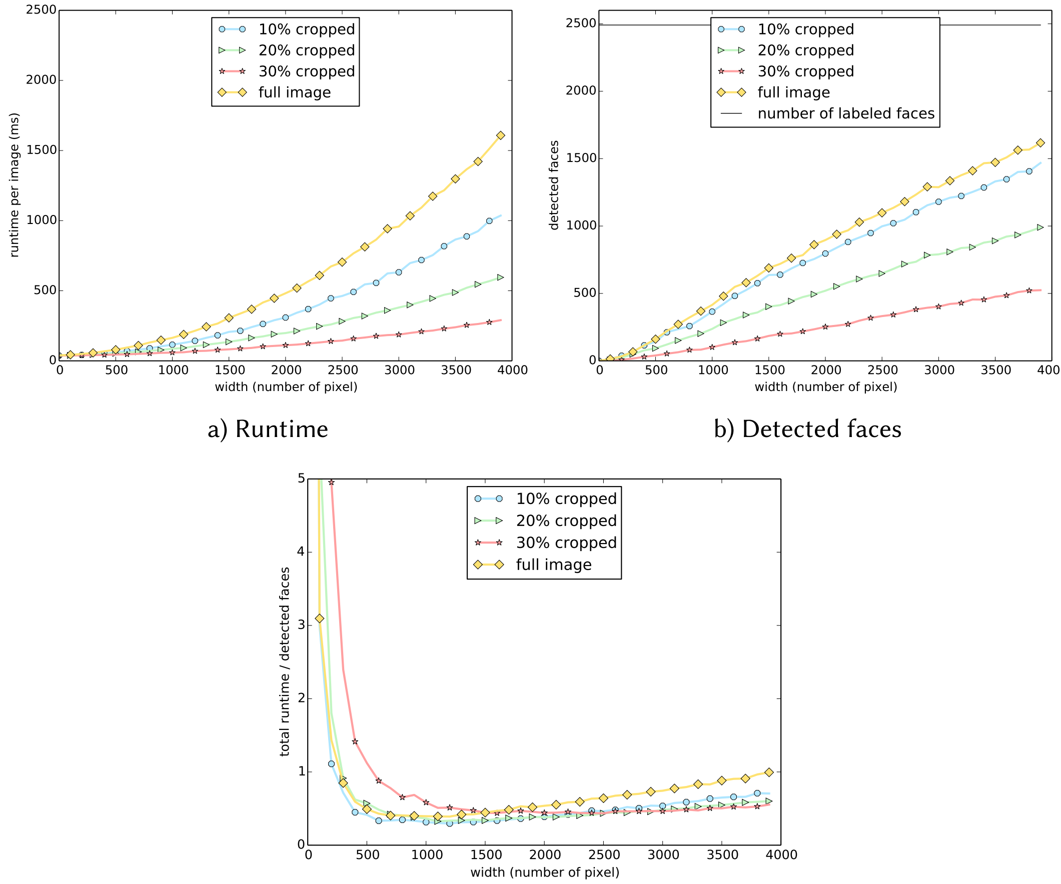
a) Color images

b) Grey scale images

Figure 4.35: #detected faces per detection time for different resolutions.

Image Cropping

The next parameter we evaluate is cropping the greyscale image. For each image, 0%, 10%, 20% and 30% are cut off from each border, reducing the image size, but also reducing areas where possible faces could be detected. As expected, image cropping reduces runtimes, even more when an original image is upscaled, as shown in Fig. 4.36a. Cropping the image by 30% from each border significantly reduces the number of detected faces, while cropping 10% decreases the detection rate only slightly (see Fig. 4.36b). Combining both scores shows an interesting result in Figure 4.36c where no single strategy wins. The behavior changes between 1,500 and 2,000 pixels. Cropping around 10% gives the most reliable results independent of image size. However, since we do not expect many portrait photos in our scenario, we cannot assume that humans are always centered in the images, therefore cutting off the edges might miss important information.



a) Runtime

b) Detected faces

c) Runtime per #detected faces

Figure 4.36: Comparison of cropped areas from grey scale images.

Face Detection Comparison

For a fast, offline face detection solution not only execution time is relevant but also its precision and recall.

Next, we compare the face detection quality of the *dlib* and *OpenCV* algorithms. Tables 4.16 and 4.17 show that *dlib* is superior with an overall F1 score of 0.63 compared to the 0.37 of *OpenCV*.

However, the recall value of 0.49 for *dlib* is much lower than the recall value for *OpenCV* with 0.79, whereas the precision of *dlib* with 0.87 clearly outperforms *OpenCV* with a precision of 0.24. This confirms that *OpenCV* is well suited for the first stage of *SmartFace*, while *dlib* is an ideal candidate for the second stage.

Table 4.16: Contingency table for *dlib*

	faces detected	**undetected**	
labeled faces	1,194	1,226	**Recall (R):** 0.49
unlabeled areas	178	—	
	Precision (P): 0.87		**F1-Score:** 0.63

Table 4.17: Contingency table for *OpenCV*

	faces detected	**undetected**	
labeled faces	1,967	502	**Recall (R):** 0.79
unlabeled areas	6,106	—	
	Precision (P): 0.24		**F1-Score:** 0.37

Table 4.18 shows the results for combining both algorithms within *SmartFace* where we tried to run *SmartFace* as fast as possible to obtain roughly the same number of detected faces as *dlib*. In contrast, in Table 4.19 we gave *SmartFace* the same amount of time as *dlib*, leading to a higher precision of 0.93 and also a higher recall of 0.70 compared *dlib*. This also results in a higher F1 score of 0.80 compared to 0.63 of *dlib*.

Table 4.18: Contingency table for *SmartFace* (faster runtime)

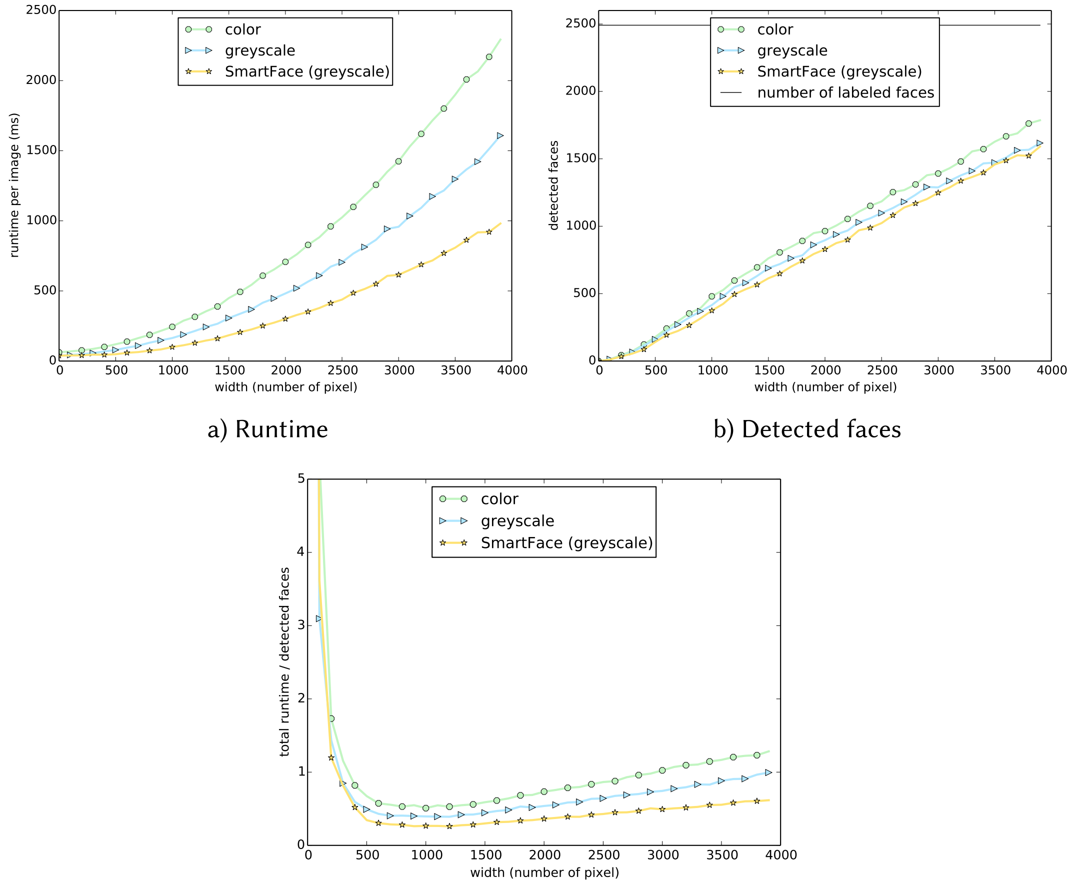
	faces detected	**undetected**	
labeled faces	1,172	1,263	**Recall (R):** 0.49
unlabeled areas	238	—	
	Precision (P): 0.83		**F1-Score:** 0.61

Table 4.19: Contingency table for *SmartFace* (higher quality)

	face detected	**undetected**	
labeled face	1,741	732	**Recall (R):** 0.70
unlabeled area	129	—	
	Precision (P): 0.93		**F1-Score:** 0.80

Compared to using color or greyscale images in *dlib*, *SmartFace* is up to two times faster depending on the image size, as shown in Figure 4.37a. The overall number of detected faces is only slightly smaller, particularly when compared to standard greyscale (Fig. 4.37b) images resulting in the best overall score, as indicated by Figure 4.37c.

Figure 4.38a shows a comparison of the best parameter combinations. The two horizontal lines are the baselines of *dlib* for runtime and number of detected faces without preprocessing, respectively. With an increased number of detected faces (80-100%), the runtime also increases for *dlib* with preprocessing. When reaching 100%, the runtime is still slightly better than *dlib* without preprocessing. Giving *dlib* with preprocessing the same amount of time as *dlib* without preprocessing results in a few more detected faces. On the other hand, *SmartFace* (i.e., the red/blue bars denoted by *optimized time* and *optimized detection*) is almost twice as fast as *dlib* with preprocessing, when limited to the same number of detected faces. If *SmartFace* gets the same amount of time as *dlib* with preprocessing, the number of detected faces increases significantly.



a) Runtime

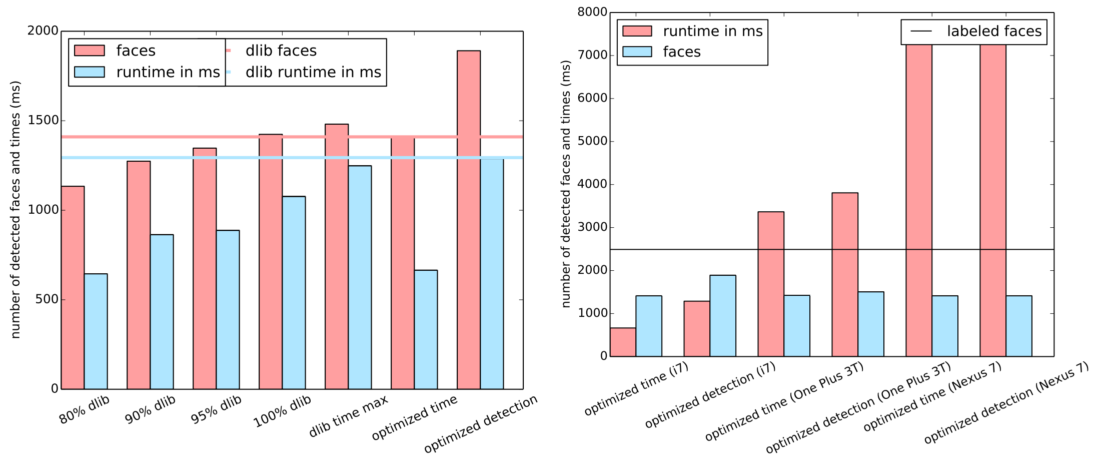
b) Detected faces

c) Runtime per number of detected faces

Figure 4.37: Overall benchmark

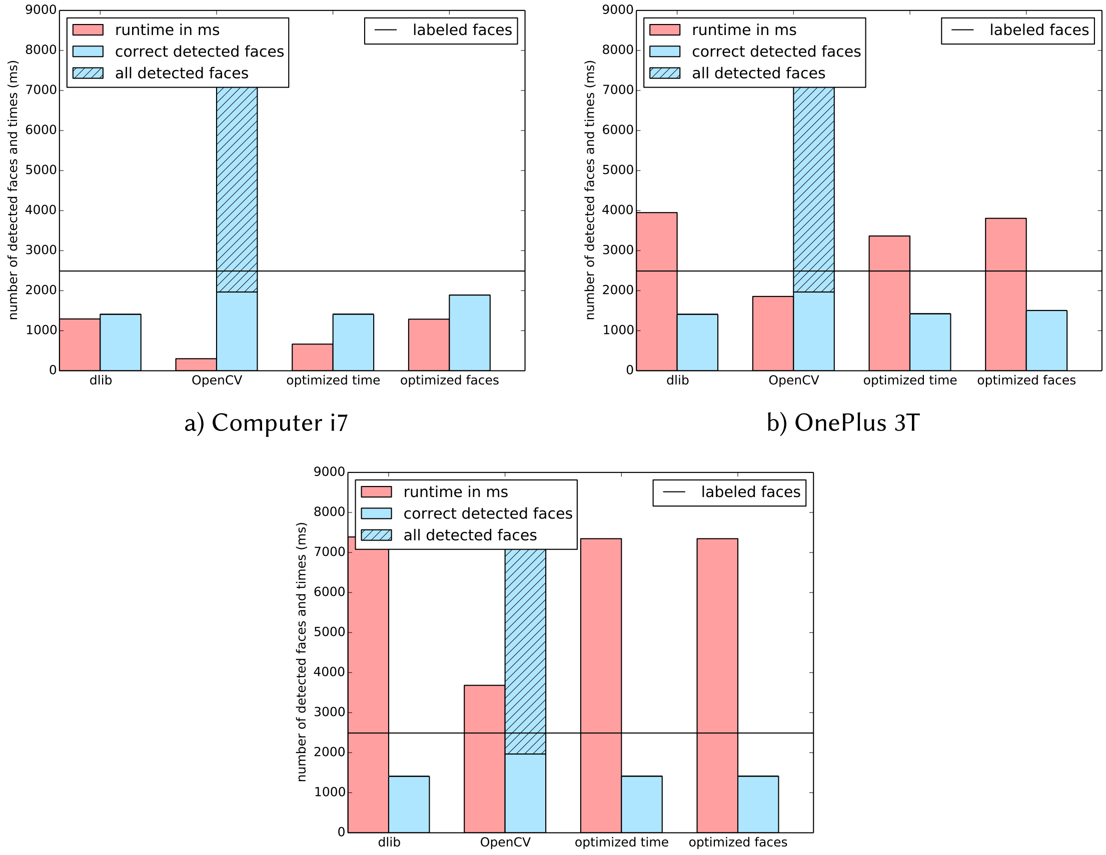
Device Performance

A comparison (see Fig. 4.38b) of *SmartFace* running on all three devices shows that the high-end *OnePlus 3T* is about 2-4 times slower than the *i7 workstation*. The mid-range tablet *Nexus 7* is about twice as slow as the *OnePlus 3T*. All three devices yield about the same high face



a) Best of all categories - dlib vs *SmartFace*

b) Direct comparison of devices



a) Computer i7

b) OnePlus 3T

c) Nexus 7

Figure 4.39: Individual device performance

detection rate, but the runtimes range between 3 and 7 seconds for the mobile devices. For our scenario with a DTN sharing photos, these runtimes are sufficently fast.

The individual performance of the three devices is shown in Fig. 4.39. Across all devices, *OpenCV* is clearly the fastest algorithm, but with an unacceptably high false positive rate. Figure 4.39a shows that depending on whether *SmartFace* is optimized for time or detection rate, it clearly outperforms *dlib*. Even on the resource-constraint mobile devices, the number of detected faces is increased.

4.7 Summary

This chapter presented work that contributes to ecosystem monitoring in forested areas. It was shown that, on the one hand, new sensor nodes can generate new knowledge, and, on the other hand, in some cases, legacy sensor boxes can be integrated and new functionalities can be added. The presented approaches can be used to monitor the ecosystem automatically with relatively low-cost hardware. This low-cost hardware leads to the ability to scale the system to an area-wide networked sensor system. In particular, the following contributions were presented:

EcoSense

We presented EcoSense, a flexible and affordable sensor, computation, and communication platform for environmental monitoring. We provided solutions for static and mobile setups, and integrated mobile devices, such as smartphones, into long-range infrastructure-less radio networks. Apart from evaluating typical power requirements of common platforms and sensors, we also included LoRa radio transceivers in our study and evaluated their realistic communication range. Furthermore, to optimize these long-range, low-bandwidth links, we provided a solution to on-device preprocessing image data using neural compute sticks in an energy-efficient and fast way. Finally, we presented the expected costs of the subsystems used in our evaluation.

AMT

We demonstrated the comparability of a low-cost automated moth trap (AMT) with a conventional trap in capturing trends in moth abundance. AMTs and automated species identification cannot replace the expertise of entomologists, since some species cannot be identified from photographs alone. However, an automated system can perform "taxonomical trivia" [131] and thus can serve as a complementary tool in species detection and identification. The data obtained can reveal ecological patterns, such as insect decline, and ultimately allow for scalable fine-meshed monitoring. Perhaps most importantly, AMTs can be deployed in the field for long periods without the ethical issues associated with conventional moth sampling.

Unobtrusive Mechanism Interception

We presented mechanism interception, a novel approach to implementing unobtrusive functional additions to or modifications of an existing networked system without touching any proprietary components. Mechanism-enhancing yet unobtrusive interceptors achieve behavioral changes. The results of the case studies indicated that a mechanism interception is a compelling approach to achieving improved service quality and unobtrusively providing previously unavailable functionality.

ForestEdge

By applying unobtrusive mechanism interception to a TreeTalker (TT) network, we were able to improve the overall functionality of a networked system that is not intended for upgrades and is locked by the vendor. Our prototype adds functionality without needing to replace the existing TT infrastructure, thus saving costs. We could also leverage behavioral modification to enable interactivity in an otherwise static system, which allowed us to recognize faulty sensors and identify corrupted data. Finally, we gained significant progress for the TT system in terms of a user-friendly interface and improved measurement accuracy.

Bird@Edge

In audio recordings, we presented Bird@Edge, an *Edge AI* system for recognizing bird species to support real-time biodiversity monitoring. Bird@Edge is composed of embedded edge devices, such as ESP32-based microphones and NVIDIA Jetson Nano boards, operating in a distributed system to enable efficient, continuous evaluation of soundscapes recorded in forests. We presented a deep neural network based on the EfficientNet-B3 architecture and optimized for execution on an NVIDIA Jetson Nano board to recognize bird species in soundscapes. It outperforms the state-of-the-art BirdNET neural network on several data sets. It achieves a recognition quality of up to 95.2% mean average precision on soundscape recordings in the Marburg Open Forest, a research and teaching forest of the University of Marburg, Germany. Measurements of the power consumption of Bird@Edge Station and Bird@Edge Mics show that the system has a sufficient demand of 3.18 W plus 0.492 W for each Bird@Edge Mic, which can be covered by reasonably sized batteries and solar panels, highlighting the real-world applicability of the approach.

SmartFace

We presented a novel approach to perform face detection on mobile devices efficiently to support the search for missing persons in wireless on-demand emergency networks. The approach relies on a two-stage combination of existing face detection algorithms, enhanced by region of interest selection, color space/depth reduction, resolution scaling, face size definition, image scaling, image cropping, and bounding box scaling. Experimental results showed that the proposed approach improves the overall face detection rate and the overall runtime compared to each of the individual face detection algorithms used alone and also reduces the amount of data that needs to be stored on disk and sent over the network.

5 Bat Monitoring

This chapter presents work on bat monitoring. It answers three questions about bats' behavior: 1) Where do bats act? 2) Which bat or species acts? 3) What do they do?

In Section 5.1, an approach is presented to allow the automatic processing of audio data by a neural network to detect bat calls. Through this approach, the collected data can be automatically categorized by species, and a species-specific distribution of bats can be detected using the meta-data of the sensor nodes. The approach appeared in publication [15].

A machine learning model for bat call detection was ported to an edge device, using the presented *Bird@Edge* approach as the basis. This led to the project of *Bat@Edge*, which allows detecting where and which bats are present, as presented in Section 5.2. Higher sampling rates lead to hardware changes and make it necessary to filter the data stream of ultrasonic audio on the microphone side.

Section 5.3 presents *tRackIT OS* [104] as a robust, error-aware, and self-healing approach to collect VHF signals on low-cost hardware. This work is the basis for long-running VHF monitoring tasks of bats equipped with a VHF transmitter. The results show that not only the data gets better, but also all processes that depend on this data get better results. Furthermore, the results can deliver insights into where bats are located.

Section 5.4 presents the automatic classification of VHF signals into active and passive [79]. The basis of this work was the collected data in the *BatRack* system, and thus developments intertwine here as described by design. The automatic evaluation of active and passive phases in bats based on the VHF data can give cost-effective information about where bats are active, i.e., to answer questions like: What is the habitat? What are the hunting grounds of bats? Long periods of inactivity for bats during the day indicate that they are in a roost; long active periods in the night with small spatial dimensions could indicate a hunting scenario.

Finally, *BatRack* is presented [81] that allows recording bats at night via video, VHF, and audio. This publication works with schedules and triggers in the scheduled time windows. This data saving was made possible because the installed sensors can interact with each other. Thus, audio sequences, which could correspond to a bat call, were taken as triggers for recording videos with an IR spotlight. Also, the recording of VHF signals and the classification of these signals into present and absent and present active or passive were used as triggers so that only recordings were made when at least one of the transmitted bats was in the area and active. This fusion of sensors allowed data to be recorded simultaneously on the video, audio, and VHF spectra without removing the animals from their natural habitat or exposing them to stressful situations. Based on this data and the temporal parallelism, audio and VHF data can now be

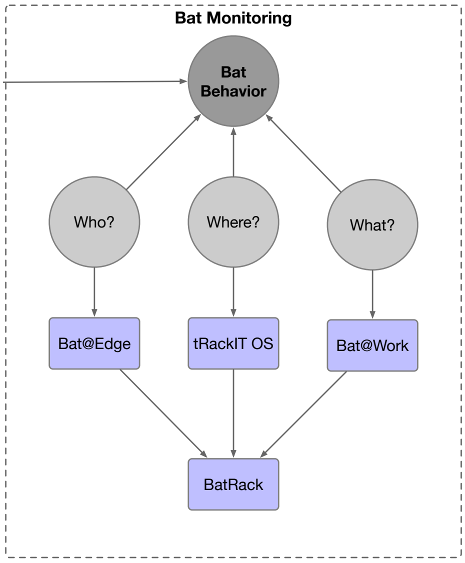


Figure 5.1: Publications on bat monitoring

tagged with behaviors that are detectable on the videos. To our knowledge, this tagging of data was not possible until the time of publication and has not yet been done.

This chapter contributes to the aim of bat monitoring as shown in Figure 5.1. Here, three questions are answered with specific publications, and on top of this, *BatRack* is presented as a solution that can answer all three questions. *BatRack* can also generate coupled data sets over the different sensors. Robust and cost-effective software and systems are presented, contributing to the scalability of sensor systems to observe bats in forested environments. All publications in this chapter also deal with the challenges discussed in Section 3.3 and give solutions to this challenges.

From this chapter, tRackIT OS[1], the classification of activity states [2], and two parts of BatRack[3] [4] are publicly available under open-source licences on GitHub. Our neural network models [5] for bat echolocation call detection and bat species recognition are publicly available on GitHub.

[1] https://github.com/Nature40/tRackIT-OS

[2] https://github.com/Nature40/tRackIT

[3] https://github.com/Nature40/BatRackOS

[4] https://github.com/Nature40/BatRack

[5] https://github.com/umr-ds/transformer4bats

5.1 Bat Echolocation Call Detection and Species Recognition by Transformers with Self-Attention

5.1.1 Introduction

Bats (Chiroptera) belong to the most widespread species group among terrestrial mammals. Except for the Arctic, Antarctic, and a few isolated islands, all regions of the earth are inhabited by bats [136]. With almost 1,400 recognized taxa, they represent almost one fifth to the mammalian diversity [68]. From pest control to seed dispersal, bats contribute to all four ecosystem services defined in the Millennium Ecosystem Assessment [137]. Thus, they are equally important for the ecosystems they inhabit and for mankind whose well-being depends on the integrity of these ecosystems. Furthermore, bats are important bioindicators for the health of the ecosystems they live in. For example, due to the high trophic level of insectivorous species, fluctuations in bat populations can be indicative of environmental changes affecting their prey of mostly small invertebrates, which are difficult to monitor themselves [117]. Unfortunately, about one third of all bat species are classified as threatened or data deficient by the International Union for Conservation of Nature (IUCN), and about half of all bat species show a declining or unknown population trend [68].

To monitor populations of bat species and thus biodiversity at scale, automatic bat echolocation call detection and bat species recognition approaches are required. With the success of Convolutional Neural Networks (CNNs) in various tasks, audio classification research has evolved over the last decade from models based on hand-crafted features like Mel Frequency Cerpstral Coefficients (MFCCs) to CNNs that directly match audio spectrograms to feature representations. Several state-of-the-art bat echolocation call detection and bat species recognition approaches are based on CNN architectures applied to audio spectrograms.

Currently, transformer neural network architectures based on self-attention, such as Vision Transformers (ViT) [49], are successfully applied in computer vision tasks. While CNNs have an inductive bias, such as spatial locality and translation equivariance, vision transformers have a lower bias and can capture global context even in the first layers of a neural network.

In this section, we present a novel approach for detecting bat echolocation calls and recognizing bat species in audio spectrograms. It is based on a transformer neural network architecture and relies on self-attention. To the best of our knowledge, this is the first work that utilizes fully attentional architectures to solve the problems of bat echolocation call detection and bat species recognition. In particular, our contributions are as follows:

- We present a novel self-attention approach for bat echolocation call detection and bat species recognition in audio spectrograms. It is based on pre-trained data-efficient image transformer models used as components in a workflow that we developed to process audio spectrograms of recorded bat echolocation calls.
- We show that the presented approach outperforms other state-of-the-art approaches for bat echolocation call detection and bat species recognition on several publicly available data sets. Our approach for bat echolocation call detection achieves a performance of up to 90.2% in terms of average precision, and our bat species recognition model obtains up to 88.7% accuracy for 14 bat classes occurring in Germany, some of which are difficult to

distinguish even for human experts. Furthermore, we demonstrate that the distribution of the training and test set is an important factor for determining the recognition accuracy.

- We make our transformer models for bat echolocation call detection and bat species recognition publicly available at `https://github.com/umr-ds/transformer4bats`. In this way, other researchers can use our models to detect and recognize bat calls contained in their audio recordings.

Parts of this section have been published in Hicham Bellafkir, Markus Vogelbacher, Jannis Gottwald, Markus Mühling, Nikolaus Korfhage, Patrick Lampe, Nicolas Frieß, Thomas Nauss, and Bernd Freisleben. "Bat Echolocation Call Detection and Species Recognition by Transformers with Self-attention." In: *International Conference on Intelligent Systems and Pattern Recognition*. Springer. 2022, pp. 189–203.

5.1.2 Related Work

Since manually detecting and recognizing bat echolocation calls is a tedious and time-consuming task, several automated methods have been proposed in the literature. For many years, handcrafted features extracted from the length of the call, the frequencies, and amplitudes were used to train machine learning algorithms. In recent years, several approaches based on CNNs outperformed these methods by learning the features in an end-to-end manner. Usually, the audio recordings are transformed into spectrograms that are then processed by a CNN.

Bat populations are found across all continents except the Arctic, the Antarctic, and a few isolated islands. Nevertheless, each region has an individual set of bat species. Roemer et al. [196] attempted to build a universal model to classify bat calls into species around the world. To do so, they first classified the calls into sonotypes and refined the results by determining the exact species. They used random forests to realize their approach.

Current approaches often use CNNs and mostly consider a specific geographic region. Consequently, models are available for regions in Europe [184, 211], North America [228], the Middle East [270], and Asia [30, 133].

While most approaches are based on spectrograms generated from standard waveform audio recordings, Tabak et al. [228] trained their CNN on sparse zero-cross data. Using a small ResNet-18 [97] neural network architecture, they achieved good results on out-of-distribution data. The way spectrograms are generated plays an important role for the given task. Zualkernan et al. [270] compared three different kinds of spectrograms, namely Short-Time Fourier Transform (STFT), Mel-Scaled Filterbanks (MSFB), and Mel-Frequency Cerpstral Coefficients (MFCC). For their use case, they found that Mel-Scaled Filterbanks work best. Paumen et al. [184] decided to work with MFCCs for data efficiency reasons. Most approaches make use of the Short-Time Fourier Transform [30, 133, 211]. Zualkernan et al. [270] and Paumen et al. [184] considered 3 and 1 second snippets, respectively, extracted from spectrograms, whereas the majority of approaches extract single calls [30, 133, 211] and can thus make use of a higher input resolution per call. Chen et al. [30] showed how manual labeling of bat call events can be improved by a simple peak detection algorithm for detecting potential regions, in order to significantly speed up the labeling process by human experts [133, 211].

Nevertheless, the human labeling process is a sophisticated task, infeasible for larger data sets, and the results can differ significantly between different annotators, resulting in a low inter-coder reliability [153].

Furthermore, an extensive publicly available bat call data collection is missing. In particular, certain species are inadequately represented in the existing data sets. To deal with small data sets and the class imbalance problem, Kobayashi et al. [133] used data augmentation. The authors applied Cutout, Random Erasing, and Salt-and-Pepper noise to the spectrograms.

In terms of the considered bat species, Schwab et al. [211] are closest to our work. The authors first applied a band-pass filter to the audio signal and computed the corresponding spectrogram. Next, they detected peaks in that spectrogram and cut out 10 ms windows around these points. A first CNN was trained to distinguish bat calls from noise and background. Based on the results of this model, a second CNN determined the exact bat species. Optionally, both models can be integrated into a single neural network to make the process more lightweight. The best results were achieved using a modified ResNet-50 [97] architecture. Paumen et al. [184] considered German bat species too, but merged similar bat species into higher classes. While most papers work with species that partly stem from the same kind, Zualkernan et al. [270] took solely groups of bat species into account, which is a much simpler task. In our approach, bat species are only merged if they are not distinguishable by human experts. A list of the bat species we considered is presented in Table 5.1.

To tackle the problem of hard to classify weak signals, Chen et al. [30] introduced a weak label along with a rechecking strategy that verifies whether the neighboring regions of the weak call belong to the same class. This is a helpful strategy for noisy, passively recorded data. However, this strategy requires additional manual labeling effort. Due to the different species being considered and the lack of publicly available data sets, it is difficult to compare the existing approaches on bat call recognition using deep learning techniques. Furthermore, classes with few samples can often be found in the data sets, which has the risk of overfitting.

To classify bat calls, these calls need to be detected in audio recordings in the first place. This corresponds to localizing audio events in time. Similar to recognition, detection has been performed using handcrafted methods such as amplitude threshold filtering or detection of changes in frequency. However, most available algorithms are closed commercial projects and thus lack transparency. Contrary to that trend, Mac Aodha et al. [153] developed an open-source software based on CNNs. Their software called *BatDetective* is based on a sliding window approach over time-expanded and band-pass filtered FFT spectrograms. Since the used data sets contain rather noisy audio recordings, the mean amplitude of each frequency band is removed [2] as a denoising operation. The resulting image is then fed into a CNN using a sliding window approach, where the window size is 23 ms. The final probabilities of the binary classification problem are obtained by applying non-maximum suppression. BatDetective achieves the best quantifiable results among all available tools by a large margin.

In recent years, CNNs have been the best choice for detecting and recognizing bat calls. Meanwhile, in many computer vision tasks, CNNs have been outperformed by transformer architectures. Transformers are based on self-attention mechanisms and have been applied, e.g., to image classification [49, 241], audio event tagging [78], and image captioning tasks [40, 266].

5.1.3 Methods

Self-attention architectures, in particular transformers, have recently become the model of choice in natural language processing (NLP), where they are often pre-trained on a large corpus of text and fine-tuned for different downstream tasks. Furthermore, transformer architectures, such as Vision Transformers (ViT) [49], have also become state-of-the-art approaches in several computer vision tasks. In an attempt to explain this success, Cordonnier et al. [39] showed that self-attention can express a CNN layer and that convolutional filters are learned in practice by self-attention. Furthermore, Dosovitskiy et al. [49] concluded that transformers do not generalize well when trained on insufficient amounts of data. Contrary to this claim, Touvron et al. [241] showed that it is possible to learn a model that generalizes well using only the ImageNet-1k data set at training time. The authors presented a training process and a distillation procedure based on a distillation token, which plays the same role as the class token used by Dosovitskiy et al. [49], with the exception that it aims to reproduce the distribution of the label vector estimated by the teacher. The authors showed that the trained fully attentional model (i.e., DeiT) can achieve competitive state-of-the-art results on ImageNet. This approach was also successfully used in the Audio Spectrogram Transformer (AST) [78] for audio event tagging using audio set[6] [75].

In the following, we present a novel self-attention approach based on pre-trained DeiT models for the automated analysis of bat calls in audio recordings. In Section 5.1.3, we describe our detection approach for bat echolocation calls, whereas our recognition method is explained in Section 5.1.3.

Bat Echolocation Call Detection

Our detection approach is based on self-attention and hence free of any convolutional layers. In the following, we describe the steps of our detection workflow, as shown in Figure 5.2. First, we generate a spectrogram from the audio recordings to enable the use of pre-trained image transformer architectures. Instead of using traditional MFCCs or PLP (perceptual linear prediction) coefficients, we use log Mel filterbank features that are state-of-the-art for bat call detection and recognition [270]. The input audio waveform of t seconds, time-expanded (TE) by factor 10, is converted into a sequence of 128-dimensional log Mel filterbank features computed by applying a 23 ms TE Hanning window with a large overlap of 84% to better capture short calls.

The resulting spectrogram is then split into multiple views with a sliding window approach, where each view is a 128×64 spectrogram that is equivalent to 230 ms TE. We selected the window size to be sufficiently long to capture the longest search calls in our training data. Each such view is then split into 16×16 patches with a stride of 10 and fed into a linear projection operation before using them as the input sequence to the transformer model to verify the presence of a bat call. The model takes $\lceil (H-16)/10 \rceil \times \lceil (W-16)/10 \rceil$ patches, which is 60 for an input of 128×64. We use overlapping patches that were shown to be beneficial by Gong et al. [78]. Since we split the original spectrogram into overlapping views,

[6] `https://research.google.com/audioset`

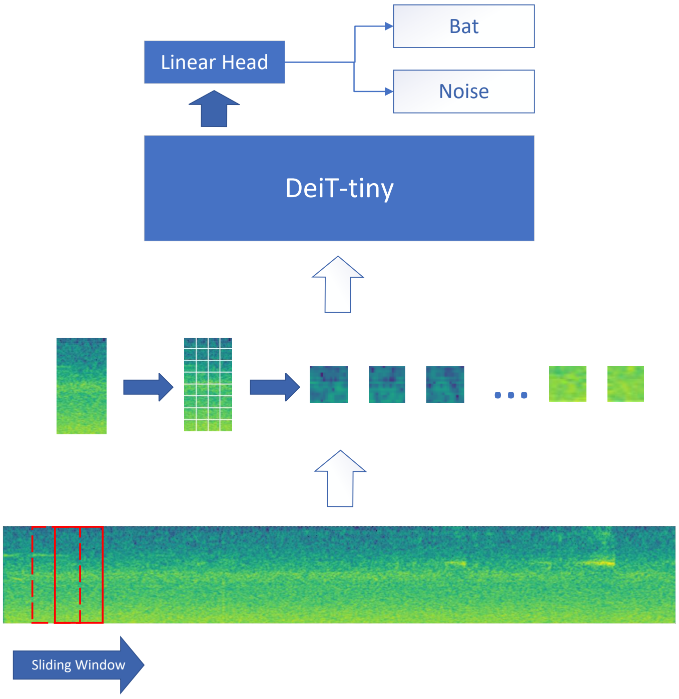


Figure 5.2: Workflow for bat echolocation call detection.

a one-dimensional non-maximum suppression operation is required to determine the final predicted time positions.

Our transformer architecture is a pre-trained tiny vision transformer (i.e., DeiT-tiny) [241] with approximately 5 million parameters. This architecture consists of 12 layers, each layer having 3 self-attention heads. The transfer learning approach with respect to processing audio data is applicable since the vision transformer can handle arbitrary sequence lengths. However, the pre-trained position embeddings need to be modified, since DeiT-tiny is pre-trained with a resolution of 224, which results in 14×14 position embeddings with patch sizes of 16×16. Since using randomly initialized position embeddings leads to poor overall performance in our experiments, we adopted the approach used by Gong et al. [78] and cut the first and second dimension of the 14×14 positional embedding to 12×5 and use it as the positional embedding in our training phase. Furthermore, we replaced the classification layer of the DeiT-tiny model to adapt the architecture to our audio detection task and initialized it randomly.

Bat Species Recognition

To build our recognition model, we consider only German bat species of the Tierstimmenarchiv[7] data set. Some of the bat species have very similar echolocation calls, e.g., the species of the Myotis kind. Nevertheless, we do not merge any of the classes, but try to distinguish them as annotated except for the Whiskered bat and the Brandt's bat which are not distinguishable by

[7] https://www.tierstimmenarchiv.de/

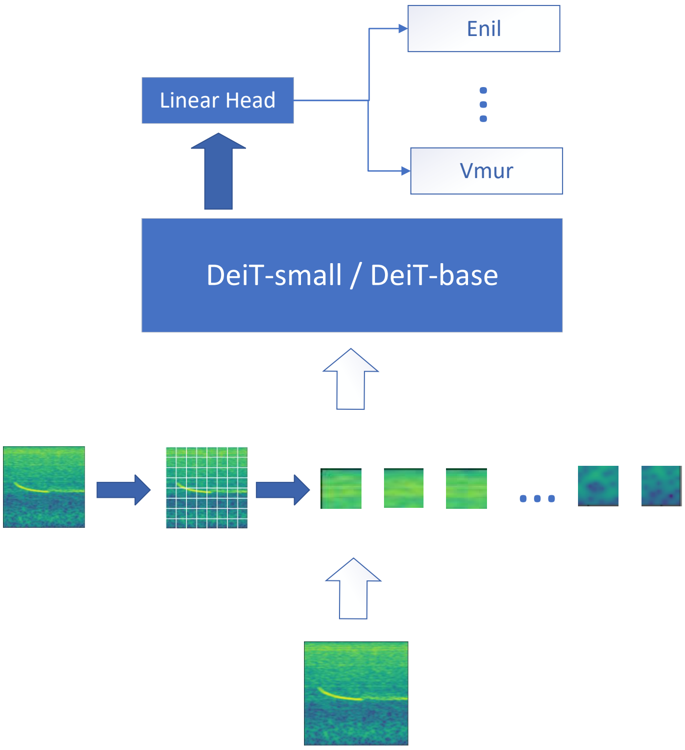


Figure 5.3: Workflow for bat species recognition.

their echolocation calls. All species considered for training our model are presented in Table 5.1.

Our recognition workflow is shown in Figure 5.3. To train our model, we first extracted echolocation calls from the Tierstimmenarchiv data set using our detection approach. For this purpose, the detection model was fine-tuned to the audio recordings of the Tierstimmenarchiv. We selected 90% as the threshold at which a detection is accepted. During training, we generate 128-dimensional log Mel filter bank features computed by applying a 23 ms TE Hanning window with a frame shift of 2.5 ms TE. We place each call in the midst of a 330 ms TE window, which results in an input resolution of 128 × 128. For the recognition task, we selected a deeper architecture to match the complexity of the task. We used two pre-trained data-efficient image base transformers (DeiT-small and DeiT-base) as alternatives. Both architectures consist of 12 layers as in DeiT-tiny, but DeiT-small has 6 self-attention heads per layer and DeiT-base has 12 self-attention heads per layer, whereas DeiT-tiny has only 3 self-attention heads per layer. Each image is then split (as in the detection workflow) into 16 × 16 patches with a stride of 10 in both dimensions, which results in 144 patches in this configuration.

5.1.4 Experiments

In this section, our methods are evaluated on different data sets. First, we describe the applied quality metrics as well as the used data sets in Section 5.1.4 and Section 5.1.4, respectively.

Afterwards, we present our bat call detection results and the conducted experiments for bat species recognition in Section 5.1.4.

In all experiments, a workstation equipped with an AMD EPYC™ 7702P 64-Core CPU, 256 GB RAM, and four NVIDIA® A100-PCIe-40GB GPUs were used. We implemented our approach using the PyTorch deep learning framework [182], utilizing the Torchaudio library [265] for audio and signal processing. The pre-trained DeiT models are available in the PyTorch Image Models (timm) library [256].

Quality Metrics

To evaluate the performance of our bat call detection approach, average precision (AP) and recall at 95% precision are used as our quality metrics. The AP score is the most commonly used quality measure for retrieval results and approximates the area under the recall-precision curve. The task of bat call detection can be considered as a retrieval problem where the annotated bat calls represent the relevant documents. Then, the AP score is calculated from the list of ranked documents as follows:

$$AP(\rho) = \frac{1}{|R \cap \rho^N|} \sum_{k=1}^{N} \frac{|R \cap \rho^k|}{k} \psi(i_k), \tag{5.1}$$

$$\text{with} \quad \psi(i_k) = \begin{cases} 1 & \text{if } i_k \in R \\ 0 & \text{otherwise} \end{cases}$$

where N is the length of the ranked document list (total number of analyzed audio snippets), $\rho^k = \{i_1, i_2, \ldots, i_k\}$ is the ranked document list up to rank k, R is the set of relevant documents (audio snippets containing a bat call), $\left|R \cap \rho^k\right|$ is the number of relevant documents in the top-k of ρ and $\psi(i_k)$ is the relevance function. Generally speaking, AP is the average of the precision values at each relevant document.

To determine whether the model underestimates the number of bat calls in the test set, we further calculate recall at 95% precision, i.e., only 5% of false positives are allowed for this metric.

To evaluate the performance of our bat call recognition approach, we use the accuracy metric as well as the F1-score. These metrics are widely used for evaluating classification models.

Data

Altogether, three different data sets are used as our test material for the detection task: two data sets provided by the Indicator Bats Program (iBats) [118] and a third data set provided by the Norfolk Bat Survey [174]. They consist of time-expanded ultrasonic acoustic data recorded between 2005 and 2011. The number of recordings and bat calls per test set are summarized in Table 5.2. We used 2,812 recordings containing 4,782 calls for training; these are a subset of the iBats R&B data collection. The described training and test split matches the one used by Mac Aodha et al. [153].

Table 5.1: Overview of the data set used for bat species recognition, showing the distribution of bat species including the number of recordings, the duration and the number of detected calls per species.

Species	Code	Recordings	Duration (s)(TE)	Detected calls
Eptesicus nilssonii	Enil	26	778	557
Eptesicus serotinus	Eser	75	2,018	1,926
Myotis brandtii/mystacinus	Mbart	24	1,770	2,033
Myotis dasycneme	Mdas	69	1,557	1,613
Myotis daubentonii	Mdau	128	3,370	4,764
Myotis emarginatus	Mema	28	450	597
Myotis myotis	Mmyo	52	1,202	1,598
Myotis nattereri	Mnat	77	1,969	2,714
Nyctalus leisleri	Nlei	78	2,338	1,770
Nyctalus noctula	Nnoc	95	3,236	2,574
Pipistrellus kuhlii	Pkuh	138	3,655	3,493
Pipistrellus nathusii	Pnat	140	3,760	3,279
Pipistrellus pipistrellus	Ppip	267	6,696	6,484
Vespertilio murinus	Vmur	69	1,788	1,654
Σ		1,266	34,587	35,056

Table 5.2: Number of recordings and bat calls per test set used for bat call detection.

Test set	Number of recordings	Number of calls
iBats R&B	500	1,604
iBats UK	434	842
Norfolk	500	1,345

Following Schwab et al. [211], we use a data set provided by the Tierstimmenarchiv for the recognition task. The data was originally recorded by Skiba [219] and later digitized. The mono audio files were provided in the WAV format with a sampling rate of 96 kHz and a bit depth of 24. Table 5.1 shows the amount of data per class along with the number of detected calls. As mentioned in Section 5.1.3, we first extracted these calls using our detection model. Although the annotations are given only per file, it is viable to assume that all detected calls in a certain recording belong to the corresponding manually annotated class of the file. This procedure is applicable because the data recorded by Skiba consists of high quality active recordings, i.e., the bats were captured for the recording and were close to the microphone.

Results

Bat Call Detection.

As mentioned above, our bat call detection model is trained using 2,812 recordings containing 4,782 calls. To augment the data, three different time crops were sampled per call, which contain the call or part of the call. For each call, six negative crops were randomly sampled, so that the overall training set contains 14,346 positive samples (bat call) and 28,692 negative samples. Furthermore, data augmentation techniques for spectrograms introduced by Park et al. [181]

Table 5.3: Summary of the training settings for the detection and recognition models.

Hyperparameter	Detection	Recognition
learning rate	$5e^{-5}$	$1e^{-4}$
warm up	✓	✓
optimizer	ADAM [127]	ADAM
batch size	64	64
input resolution	128 × 64	128 × 128
time/frequency masking	✓	✓
patch stride	10	10
random noise	✓	✓

for speech recognition were used, such as time and frequency masking. The training settings are summarized in Table 5.3.

We adopted the evaluation protocol used by Mac Aodha et al. [153]. A detection is considered as a true positive if its distance to the ground truth is smaller than a given threshold. We used a threshold of 100 ms TE, similar to Mac Aodha et al. [153]. Table 5.4 summarizes the results of our approach and compares them to state-of-the-art approaches. The results of the random forest approach, CNN_{FAST} and CNN_{FULL} are reported by Mac Aodha et al. [153]. Our self-attention approach outperforms the other approaches on all three data sets.

Table 5.4: Bat call detection results on three different datasets and comparison to state-of-the-art approaches.

Average precision				
Testset	Method			
	Random Forest [153]	CNN_{FAST} [153]	CNN_{FULL} [153]	DeiT-tiny
iBats R&B	0.674	0.863	0.895	**0.900**
iBats UK	0.648	0.781	0.866	**0.902**
Norfolk	0.630	0.861	0.882	**0.898**
Recall at 95% precision				
iBats R&B	0.568	0.777	0.818	**0.821**
iBats UK	0.324	0.570	0.670	**0.681**
Norfolk	0.049	0.781	0.754	**0.801**

The model was trained for a maximum of five epochs. While the training took about 146 seconds per epoch, the average inference runtime is 0.093 seconds for an audio signal with a length of one second using the hardware/software system described at the beginning of Section 5.1.4.

Bat Call Recognition.

In this section, we evaluate our bat call recognition model. We compare our self-attention approach to the ResNet-50 architecture that was used by Schwab et al. [211]. Additionally, we evaluate the use of Log-Spectrograms vs. Log-Mel-Spectrograms. Furthermore, we show that the way we split the data set into training and test data has a great impact on the result.

First, we follow the split described by Schwab et al. [211], where 20% of all echolocation calls are used for evaluation, meaning that different calls emitted by the same individual in the same call sequence can be found in both the test and the training set. Since these calls are very similar, this approach is prone to overfitting and leads to high performance values. To better evaluate the generalization capabilities of the model, we split the data set on a recording level. For each class, we use 20% of the files for testing and all remaining files for training to provide a more appropriate evaluation strategy. Table 5.5 summarizes the results using both splits.

Table 5.5: Comparison of bat call classification results of the DeiT-base model using call-based and recording-based splits of the data set.

Species	F1-score	
	Call-based split	Recording-based split
Enil	0.99	0.75
Eser	0.99	0.82
Mbart	0.99	0.95
Mdas	1.00	0.80
Mdau	0.99	0.92
Mema	1.00	0.98
Mmyo	1.00	0.98
Mnat	1.00	0.93
Nlei	0.99	0.76
Nnoc	1.00	0.91
Pkuh	0.99	0.86
Pnat	0.98	0.78
Ppip	0.99	0.94
Vmur	1.00	0.79
Avg	0.99	0.87

Table 5.6: Recognition results.

Method	Accuracy
DeiT-base + Log-Mel-Spectrogram	**0.887**
DeiT-small + Log-Mel-Spectrogram	0.882
ResNet-50 + Log-Mel-Spectrogram	0.884
DeiT-base + Log-Spectrogram	0.881
ResNet-50 + Log-Spectrogram	0.879

Indeed, the evaluation using a call-based split reaches very high F1-score values (≥ 0.98) for all species, which is consistent with the findings of Schwab et al. [211]. In contrast, these excellent scores cannot be confirmed using a file based split of the data set. However, our model still reaches very good results on this test split, since no class drops below 0.75, as Table 5.5 confirms. Therefore, we will use the latter, more meaningful data set split in the remainder of this paper.

In the following, we compare two transformer architectures, namely DeiT-base and DeiT-small, with the ResNet-50 CNN architecture. With 6 self-attention heads per layer, DeiT-small is a compromise between DeiT-tiny (3 heads) and DeiT-base (12 heads). As our input, we use both Log-Spectrograms as well as Log-Mel-Spectrograms. Table 5.6 shows that we achieve better

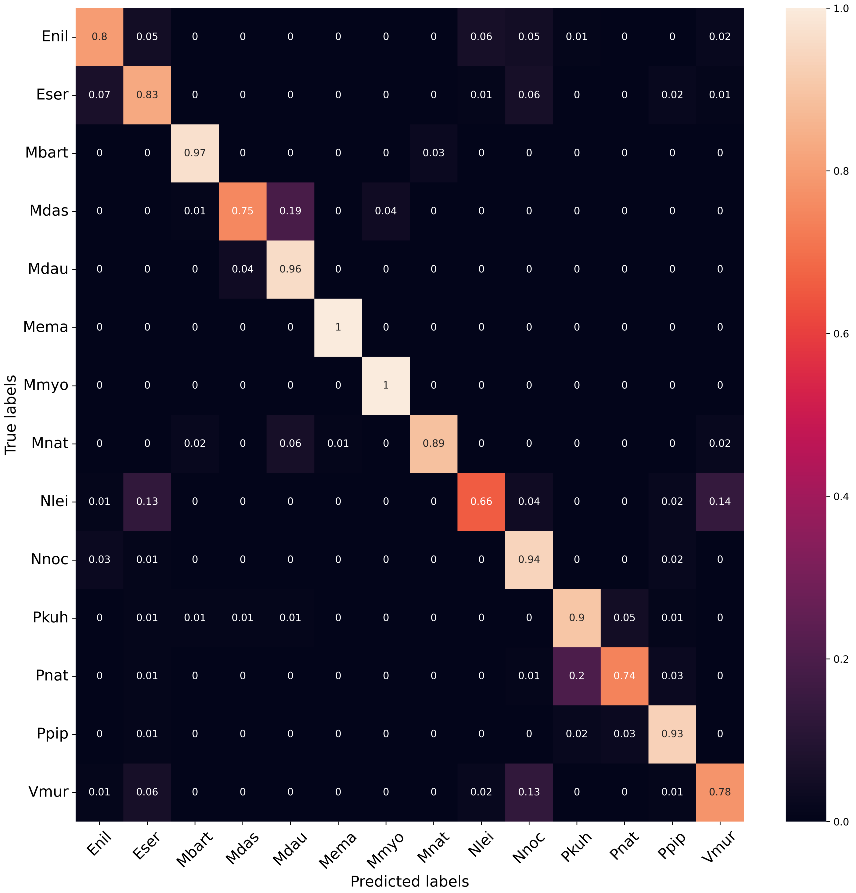


Figure 5.4: Normalized confusion matrix of the bat species classification results.

results than the ResNet-50 on both kinds of spectrograms. Furthermore, our best model, i.e., DeiT-base, achieves a 0.6% higher accuracy using Log-Mel-Spectrograms than using simple Log-Spectrograms.

Next, we consider the results of the best model (DeiT-base with Log-Mel-Spectrograms) for individual bat species. For this purpose, the confusion matrix of the classification model is visualized in Figure 5.4. The matrix shows a distinct diagonal line, which confirms the good classification results. Furthermore, it reveals that test samples are often mistaken for another species of the same kind. For instance, the model tends to confuse Enil and Eser, Mdas and Mdau or Pkuh and Pnat. Many confusions that span across two different kinds are quite difficult to distinguish as well, e.g., the bats of the Nyctaloid group (Nlei, Nnoc, Eser, Enil, Vmur) [184]. Most of the mistakes made by the model concern species that are hard to distinguish even for human experts, and the model predicted the higher order class correctly.

The recognition model was trained for a maximum of 50 epochs. While the training of the DeiT-base model took about 413 seconds per epoch, the average inference runtime is 0.02 seconds per bat call using the hardware/software system described at the beginning of Section 5.1.4.

5.2 Bat@Edge: Bat Species Recognition at the Edge

5.2.1 Introduction

As described in Section 5.1, automatic bat call detection and species recognition are necessary to monitor the population. The task of automatically processing audio data can be performed using the approach presented in Section 4.5, but the precondition is that the data is transferred in files to the edge device operating in the field. The transfer is challenging because the audio data is in raw audio format, and the data rate is relatively high with ultrasonic microphones. Today's most used solution is to record the audio on a SD card and collect the SD card every few days by humans. If something went wrong in between, researchers only notice it the next time while collecting the SD cards.

5.2.2 Related Work

To get an overview over the execution of machine learning methods, a survey was written by Merenda et al. [161]. A similar approach was presented by Gallacher et al. [71] who deployed 15 sensors in a large urban park to process recorded audio data of bats locally. This deployment allows monitoring of their activities for several months. But this system is only operated in urban areas and is dependent on network connectivity and fixed power supply.

Different edge computing platforms for bat species classification are compared by Zualkernan et al. [271]. Here, the Raspberry Pi 3B+ and the Google Coral showed good results for TensorFlow-Lite models, but for the execution of the TensorRT model on GPU only the NVIDIA Jetson Nano is capable.

5.2.3 Design

In Section 5.1, approaches for call detection and species recognition were presented, as well *Bird@Edge* in Section 4.5. *Bird@Edge* is based on an edge computing infrastructure divided into three components. These components are: (a) the microphones for field deployment, (b) the central processing unit in the field, and (c) recognition assurance and visualization in the backend. To design a system for automatically detecting bat calls in the field, all these three components and their cooperation must be enabled. In the following, the different components and the challenges are presented.

Bat@Edge Mics

Unlike bird calls, the frequency spectrum of bat calls is much broader and thus more challenging. The technology used in Bird@Edge is based on *ESP32* devices with a microphone tethered via I2S. In this case, *ESP32* devices are responsible for sampling the incoming audio signals. The maximum possible sampling rate depends on the available computing power and the transfer rate of the interface, in this case, the I2S. The *ESP32* with 2x240 Mhz offers a reasonable basis for sampling 40 kHz to investigate the spectrum from 0 Hz to 20 kHz audio. For more extensive frequency ranges, however, more computing power is needed, so an *ESP32* is no longer sufficient.

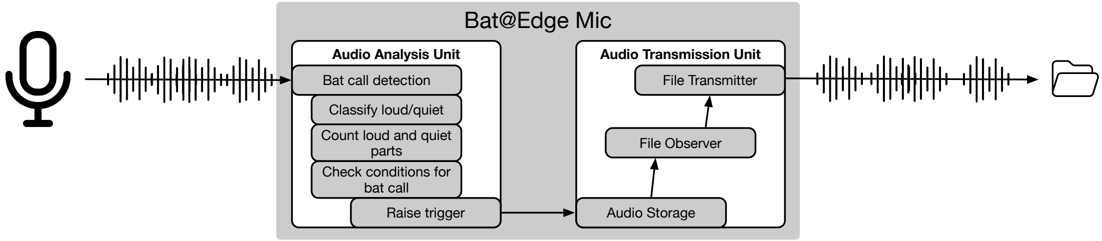


Figure 5.5: The processing chain of a *Bat@Edge Mic* from recording audio to transmitting the file

For this purpose, specially developed hardware solutions such as the Teensy Board can be used as an alternative to sampling analog microphones with a high sampling rate. Another option is a USB ultrasonic microphone with a Raspberry Pi. This setup is similar to the audio part of *BatRack* shown in Section 5.5. In *Bat@Edge*, an ultrasonic microphone is also operated on a Raspberry Pi, and the continuous data stream is examined for bat calls, as shown in Figure 5.5. If bat calls are detected, the sequence is saved as a file and is ready for further processing or transmission. As an estimate of the amount of data, each frame can be sampled at 16 bits, and a sampling rate of 382k samples is possible. This setting results in a maximum data rate of 764 kB/s in phases where bat calls are detected. These detected bat calls must be transmitted after recording. WiFi is a suitable transmission technology for transmitting the recorded ultrasonic audio data at these data rates using files.

Bat@Edge Station

Bats are only active at night, and the execution of call detection can also be performed during the day. This use case does not necessarily require real-time processing, so a significantly longer execution time is acceptable. Furthermore, a filtering step on the microphone can minimize the sequences to be processed. Thus, a neural network is only executed to process audio in which a bat call is present with a high probability. The execution chain is as follows: incoming data triggers the execution, and if all data is processed, the processing pipeline is shut down to save energy. For this purpose, a neural network is adapted to the computing unit, and the results of the execution of the neural network are saved.

The first step for the *Bat@Edge Station* is receiving the audio in files. This task can be done by any data transfer protocol on a higher layer protocol like MQTT, FTP, Rsync, or DTN (like Serval or DTN7). As the next step, an audio observer monitors the upload folder, which handles new audio files, as shown in Figure 5.7. The trigger for new audio files is only thrown if the file is completely uploaded. If necessary, the first processing step is resampling the audio for the sampling rate of the used machine learning model. This step could be skipped if the microphone and the CNN share the same sampling rate. The second step is to detect species in the transmitted and resampled audio with possible bat calls. Therefore, the stored audio file is given to the neural network, and after the processing ends, the results are stored for further transmission. These stored results can be transmitted to the backend after processing.

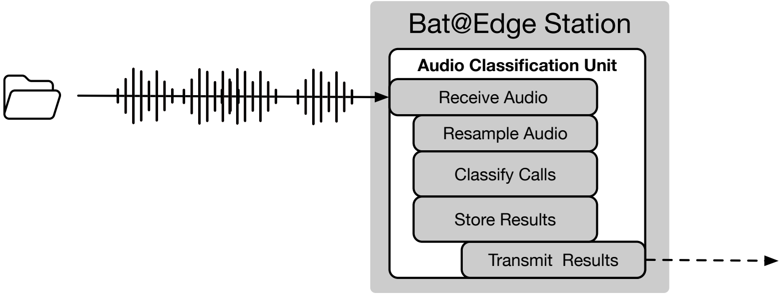


Figure 5.6: The arriving audio file and the following processing chain until the results are submitted

Bat@Edge System

For transmitting the results, a lightweight transport protocol is used. Since the data consists only of labels of species recognition, this data is much smaller than the raw audio data. Due to less data, the transmission can be done using different technologies, e.g., LTE, Mesh Networks, or WiFi bridges. The used technology is highly dependent on the use case and should be selected with this use case in mind.

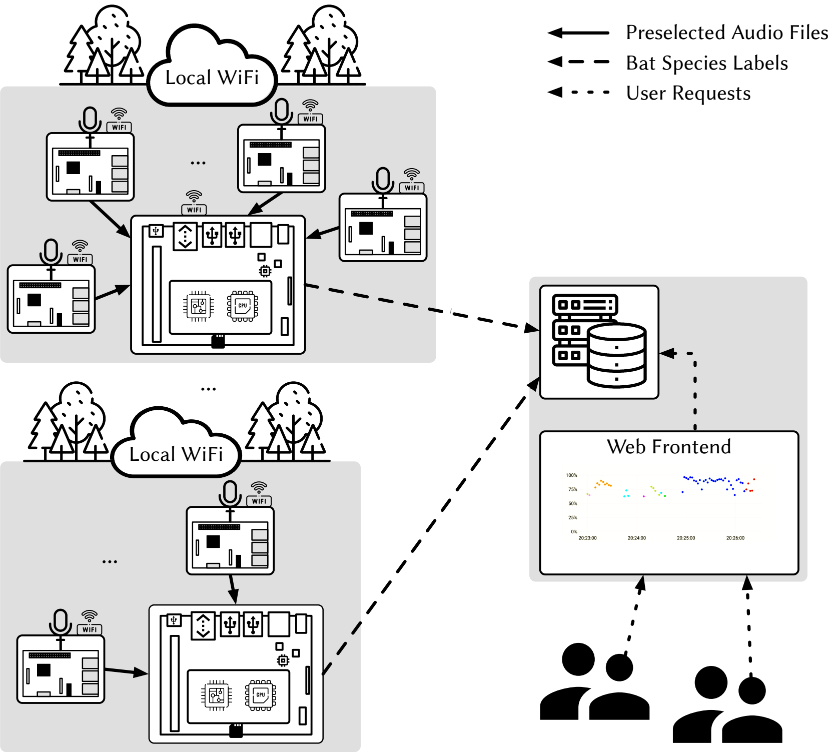


Figure 5.7: The Bat@Edge System with two local WiFi clouds and a *Bat@Edge Station* each and a *Bat@Edge backend* with result visualization

The backend saves the labels to a time series database and visualizes them, as shown in Figure 5.7. This process allows errors to be detected as quickly as possible and automatic evaluations to be performed with this data. The system is designed to enable as fast as possible processing in the field, which makes it possible to transmit the results.

5.2.4 Experimental Evaluation

In this section, we present experimental results for the *Bat@Edge Station* in terms of (a) the execution time of the processing chain, (b) the utilization of the computing resources, and (c) the power consumption while idling, resampling or classifying bat species.

Execution Time

The evaluation is based on data collected with *BatRack* in the Marburg Open Forest in 2021 to come close to a real-world scenario. For the evaluation, a Jetson Nano is used with a bat detection CNN and the stored real-world audio data. The steps on the *Bat@Edge Station* for the evaluation are the same as described in Section 5.2.3. The arriving audio is resampled if necessary, and the CNN is executed.

The process is done with pre-sampled data and without getting an impression of what time the resampling takes. The transmission is done by *scp* in a batch, and the processing is triggered automatically by the *Bat@Edge Station*. In this scenario, the concrete execution time is measured, but the neural network is loaded every time from scratch with the lack of the processing chain. This reloading is ineffective in this scenario, but in a real-world scenario, it would be better to avoid holding the CNN all the time on load and wasting energy. The performance could be better in a real-world scenario with no batch transmission.

This evaluation is done by a short time window of data from the night of 02.08.2021 from 21:00 till 23:59:59, i.e, for three hours. The question for the experiment is: is the execution time less than the observation time? The execution time over all files without sampling is 3,505 seconds or 58.40 minutes. Additionally, the load on the system is about 0.95 to 1.3, which indicates some leftover computational resources.

With the resampling task as the first step of the execution chain, the execution time over all files is 12,109 seconds or 201.80 minutes, and the average execution time per file, including CNN loading and sampling, is nearly 78 seconds. In contrast, the average execution time for the pre-sampled files is nearly 23 seconds. In this experiment, it is clear that the execution time of the tasks with resampling takes longer than the three hours of monitoring time. In this case, only some files get processed live, and the idle time is zero. For this scenario, a continuous running CNN would be the better option because it does not load the model for every file.

The execution time is nearly 3.4 times higher in the resampling scenario, which can be avoided by sampling the microphone at the same sampling rate. So, the *Bat@Edge Mics* have to record the audio in the same sampling rate as the neural network to avoid this overhead. If the sampling rate is not selectable on the microphone, the time overhead could be minimized by executing the resampling task on multiple cores. At the moment, only one core is doing this task, using multiple cores could speed up the process. The amount of data the *Bat@Edge Station* gets is different every night and for every position of the different *Bat@Edge Mics*, so a comparison for different nights should be made next.

Bat Activity in Different Nights

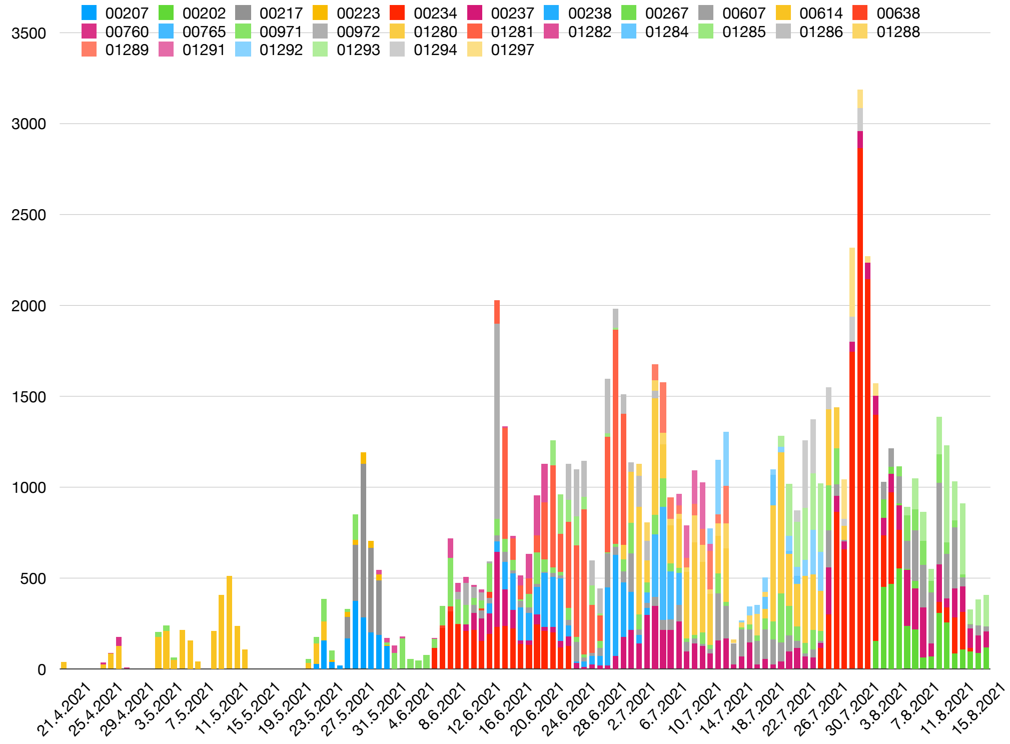


Figure 5.8: Summed sequences by day over all *Bat@Edge Stations*

The activity levels of bats vary significantly from night to night. These differences can be due, for example, to the weather; thus, the supply of nocturnal insects and with these nocturnal insects the food sources is different. The activity level per station is highly variable here, because the bats' primary location can also differ on a daily base. Bats only sometimes roost in the same tree, so a location that is highly frequented one night may be less of a focus the next night.

This behavior over the season and with several stations can be seen in Figure 5.8. Here, the sequences recorded by the *BatRack Station* are plotted per day and station. In our test scenario, we needed more hardware to continuously equip all habitat trees with a *BatRack* station. Nine stations were distributed on a total of 30 habitat trees, and by manually recording the night roosts of the bats, the stations were repositioned during the day. This repositioning results in better detection of the unique animals, but with this repositioning, the minimum number of calls per night still needs to be determined. Thus, the setup of the stations tries to record as many calls and sequences as possible with the given number of stations. This number of sequences cannot be expected for an area-wide and fixed installation with this number of stations. If the number of stations gets higher, the load for a *Bat@Edge Station* could be too high for a single *Bat@Edge Station*, so more than one *Bat@Edge Station* should be deployed.

As Figure 5.8 shows, a peak from over 3,000 sequences is possible but a rare event. Therefore, the *Bat@Edge Station* also stores the data to be processed in a queue and evaluates it one after the other. Thus, the evaluation can also be done over the entire day. However, some nights may produce more sequences than can be processed live.

5.2.5 Utilization of Computing Resources

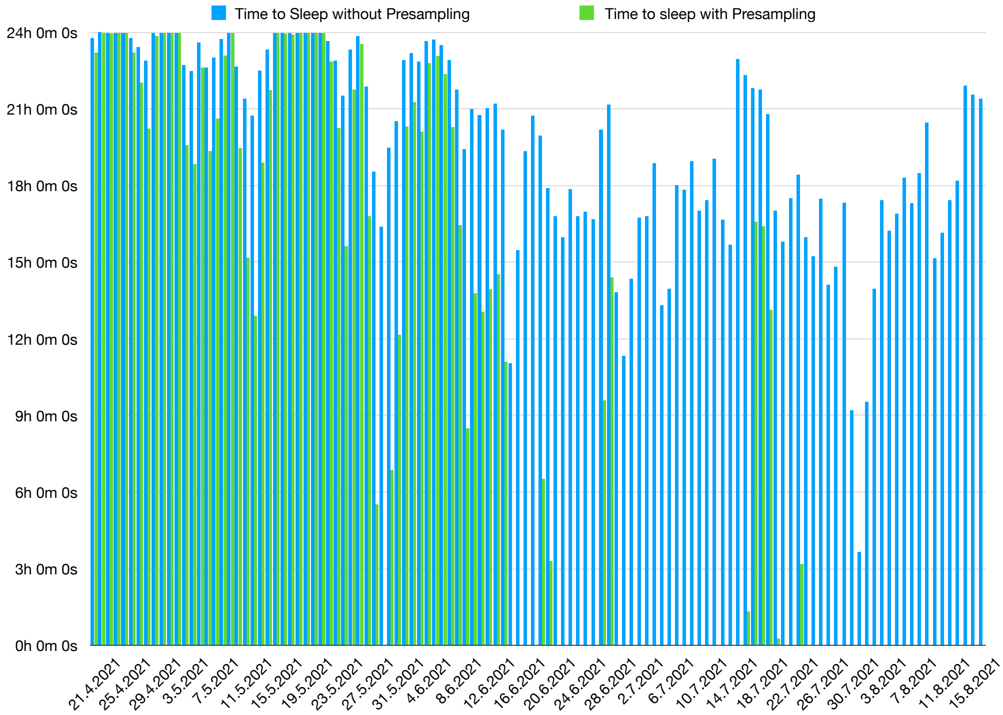


Figure 5.9: Modeling the sleeping time of the Nvidia Jetso Nano when processing the data of a day

To investigate the utilization of the *Bat@Edge* sensor network, all sequences are summed up for a whole day and get multiplied by 23 seconds for "without presampling" and 78 seconds for "with presampling" to get an impression of how utilized the network would be in a real-world scenario. The time not used for processing is shown as the time to sleep, as shown by Figure 5.9. If a node needs more than 24 hours a day to compute the received data, the time is subtracted from the next day. Alternatively, the day after the next day if more is needed, and so on and so forth. Thus, a "traffic jam" of tasks for the "with presampling" scenario is visible in Figure 5.9. Here, the Jetson Nano is processing the data over a few days all day long, and at the end of data collection, about 61 hours of processing time are left over. In this case, a second Jetson Nano would get more timely results and can handle the load without a "traffic jam".

Energy Consumption

For the power consumption measurement, the power measurement tool *tegrastats* of the Jetson Nano is used. Here, the power consumption of the CPU, GPU, and the whole system can be monitored. To get an impression what power which task consumes, a large file for processing is stored on the Jetson Nano. Next, the pipeline gets started and idles for nearly two minutes. The total power consumption in this state is about 1.1 W with nearly zero watts for the GPU and 0.12 W for the CPU, as shown in Figure 5.10. While idling, the "system" consumes the most power (about 1 W), but the total power consumption stays low.

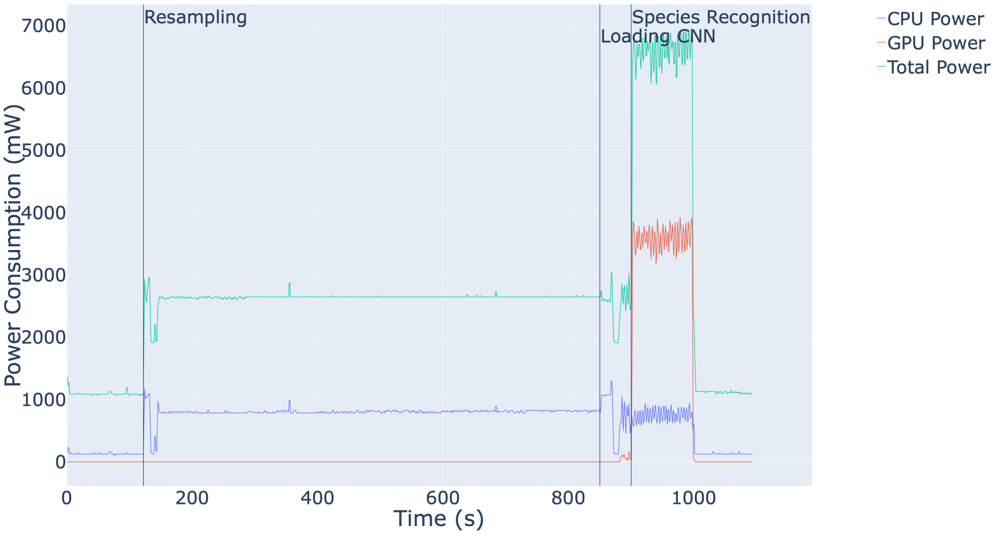


Figure 5.10: Power consumption of the Nvidia Jetson Nano while (a) idling (b) resampling (c) recognizing species

After two minutes, a file is inserted, and the pipeline detects and processes the file. Now, the average total power consumption is about 2.6 W, and the CPU consumes 0.8 W, so the rest of the system consumes about 1.8 W. The GPU consumption stays by approximately zero watts.

The next step *Bat@Edge* is starting the CNN. Here, the power consumption is more variable, but on the average, there is no significant difference in the power consumption of the resampling task. Here, only the start of the power consumption of the GPU can be seen.

The last step of the processing chain is bat species recognition. In this state, the main power consumption driver is the GPU power consumption which is about 3.6 W, and the total power consumption is about 6.6 W.

5.3 *tRackIT OS*: Open-source Software for Reliable VHF Wildlife Tracking

5.3.1 Introduction

In an increasingly densely populated and anthropogenically dominated environment, a scientific analysis of the consequences of human-wildlife interaction is essential for developing evidence-based guidelines for conservation [122]. Understanding the impact of altered habitats on the spatial distribution of species [205], the effects of human infrastructures such as roads [6, 110], and reasons for increased mortality of endangered species [145] is crucial for preserving biodiversity in a crowded world. Movement data of animals generated by recent technological advances support more detailed forms of analysis and insights into the behavior and ecology of threatened species than ever before [25, 250, 263].

Wildlife observations can be realized with a variety of technologies. For example, GPS technology can be used to equip animals and record their movements independently of other communication infrastructures. However, size, weight, and battery life constraints prevent the use of GPS for most European songbirds and bats.

Manual radio telemetry is another option for observing small animals. However, it is extremely labor-intensive, limited to a small number of individuals that can be tracked simultaneously [37], and results in spatial and temporal data with poor resolution, which might not be sufficient for meaningful scientific analyses [168].

Automated radio telemetry systems can minimize many of these disadvantages [124, 252]. In previous work, some of us presented a system based on commodity-off-the-shelf (COTS) hardware for automatic radio tracking of small animals based on Very High Frequency (VHF) tags [80] as part of the open source project *radio-tracking.eu*[8]. However, three seasons of long-term stationary operating time of the system in the *Marburg Open Forest* (i.e., the teaching and research forest of the University of Marburg, Germany) revealed several deficits, such as the lack of failure handling, inadequate interfaces for data transmission and health-state monitoring, and problems with time synchronization of received signals between receivers of the same station and among different stations.

In this section, *tRackIT OS*, an open-source operating system distribution for reliable VHF radio tracking of small animals, is presented. *tRackIT OS* runs on a *tRackIT station*; its basic hardware design is due to Gottwald et al. [80]. We developed *tRackIT OS* to provide new software functionality according to our experiences in studying the movement ecology of both diurnal and nocturnal wildlife with a network of 15 *tRackIT stations* in densely forested terrain. In particular, we present:

- a novel approach for automated signal detection of VHF radio tracking tags,
- means to provide reliable operation of *tRackIT stations* under harsh conditions,
- efficient live data transmission for monitoring data and detected signals,
- a novel web-based user interface for intuitive configuration of *tRackIT stations*,

[8] `https://radio-tracking.eu`

- a comparative evaluation of *tRackIT OS* compared to the state-of-the-art.

Parts of this section have been published in Jonas Höchst, Jannis Gottwald, Patrick Lampe, Julian Zobel, Thomas Nauss, Ralf Steinmetz, and Bernd Freisleben. "tRackIT OS: Open-Source Software for Reliable VHF Wildlife Tracking." In: *51. Jahrestagung der Gesellschaft für Informatik, Digitale Kulturen, INFORMATIK 2021, Berlin, Germany*. LNI. GI, Sept. 2021, pp. 425–442.

5.3.2 Related Work

Ripperger et al. present a comprehensive overview of existing systems for localizing small animals using different technologies [195]. The most recent projects on automated VHF transmitter tracking are ARTS [124], Atlas [252], and Motus (also called SensorGnome) [231].

ARTS consisted of towers with a height of 40 meters and top-mounted antenna arrays [124], but the system was taken down in 2010 and replaced by camera traps and GPS transmitters. ARTS was able to determine the position of a tagged individual by triangulation with an spatial accuracy of 50 meters, but rotating through channels with different frequencies reduces the time span in which each individual can be observed. *tRackIT* supports more detailed observations of movements using a higher number of stations at lower cost and less effort in construction.

The Atlas project achieves great spatial accuracy by using the *time difference of arrival* (TOA) method for direction estimates as seen from the receiver, while costs for the developed tags are low [252]. However, implementation of the receiving stations is quite expensive, a fact that probably explains why the system is only deployed in three areas in the Netherlands, England, and Northern Israel. *tRackIT* achieves comparable results with stations built from commodity off the shelf hardware at a lower price point.

Motus[9] is a globally operating network of VHF receiver stations hosted by different collaborators and supporting researchers [231]. Despite its open source character, an implementation of Motus at US$ 3000 for a single SensorGnome[10] receiver with three 9-element Yagi antennas, and US$ 7500 for a Lotek SRX800 receiver station with four 9-element Yagi antennas is costly [146], leading to a trade-off between spatial resolution and coverage. By default, the implemented radio receiver listens at a single center frequency and can detect pulses from tags in a narrow band of ± 24 kHz around its center frequency. This limits the number of distinguishable frequencies, i.e., the number of detectable individuals, substantially. Motus has delivered great insights into the ecology of different species in more than 120 research projects [231], but investigating fine-grained spatial movements by triangulation is not supported by the system. The wide frequency band that can be used by *tRackIT* supports both fine-grained temporal resolutions and observations of many individuals.

[9] Motus Wildlife Tracking System: `https://motus.org`

[10] SensorGnome Project: `https://sensorgnome.org`

5.3.3 *tRackIT OS*

A *tRackIT* system consists of (a) VHF radio tags mounted on animals, (b) *tRackIT stations* for receiving signals emitted by VHF tags, (c) *tRackIT OS* running on *tRackIT stations* for detecting and matching signals received on multiple antennas, (d) *tRackIT servers* for collecting and presenting data transmitted from *tRackIT stations*, and (e) *tRackIT analytics modules* for deriving ecological knowledge from the collected data.

In this section, we present design and implementation issues of *tRackIT OS*, the operating system distribution for *tRackIT stations*.

Requirements

Our experiences from three seasons of field work have indicated that automatic telemetry can only be a useful substitute of its manual counterpart if certain requirements are met:

1. *Low entry barrier*. To make automatic radio telemetry accessible to the widest possible user community, both hardware and software as well as data processing and analysis must be conveniently accessible, easy to use, and inexpensive.
2. *Reliability*. The used equipment must reliably record signals originating from VHF transmitters and minimize the amount of interference. Any component failures caused by adverse conditions, such as unstable power supplies, fluctuating temperatures, and hardware-based failures should be detected and handled automatically.
3. *Data availability*. In many application areas, like mortality studies [98], fast data availability is highly important. Thus, direct data transmission from the field with the shortest possible delay between recording and transmission is desirable.

tRackIT Station

To deploy an operational installation in the field, a *tRackIT station* is equipped with directional antennas in the four cardinal directions, a solar panel, and a battery box. The basic hardware design is due to Gottwald et al. [80]. We have slightly adapted the hardware by including an active USB hub, a better LTE modem, and a LoRa (Long Range Wireless Radio Frequency Technology[11]) expansion board (LoRa HAT), as shown in Figure 5.11.

The 'brain' of a *tRackIT station* is a Raspberry Pi 3 Model B that consists of a quad-core 1.2 GHz ARM-Cortex-A53 and 1 GB of RAM. It offers various input/output options, including Wi-Fi and 4 USB ports. The system is powered through a 5V USB port and is capable of powering connected USB devices. The four directional antennas are connected to four software-defined radios (SDR) (Nooelec NESDR SMArt v4) for signal analysis. Since these SDRs require more power than provided by the Raspberry Pi, an active 4-port USB hub (Anker 4-port Ultra Slim USB 3.0 Data Hub, A7518) is used to connect the devices. An LTE modem (Huawei E3372H) and a local prepaid data plan is used to establish a mobile Internet connection. The battery

[11] Semtech: `https://www.semtech.com/lora/`

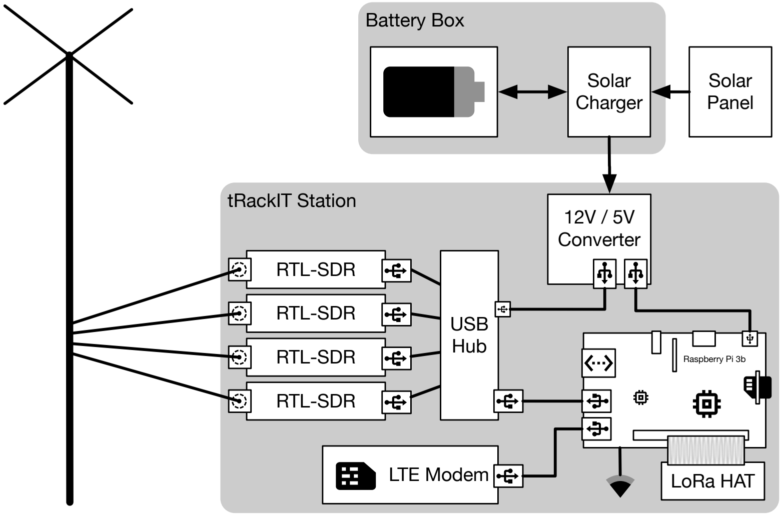


Figure 5.11: The hardware components of a *tRackIT* station.

box provides a 12 V source that is converted using a step down converter rated for 2 × 2.4 A at 5 V. For *tRackIT stations* relying on LoRa for data publishing, a Dragino SX127X GPS HAT[12] is used. For receiving and forwarding *tRackIT stations*, the Dragino PG1301 LoRa Concentrator is used[13]. The basic hardware of a *tRackIT station* costs a total of about 200 €, consisting of 35 € for the Raspberry Pi 3B+, 4 × 35 € for the Nooelec SDRs, 15 € for the active USB hub, and 10 € for the power supply unit. The optional communication modules cost 50 € in the case of the Huawei LTE modem and/or 35 € (LoRa HAT) / 110 € (LoRa Concentrator) for the LoRa publish / receive upgrade.

tRackIT OS Components

The operating system (OS) plays a crucial role in the reliable autonomous operation of the presented hardware. We developed a custom distribution of the Raspberry Pi OS, called *tRackIT OS*. The primary task of *tRackIT OS* is to execute a signal detection module, called *pyradio-tracking*, in a reliable manner. The secondary task is to interface with users (a) interactively while setting up the station, and (b) continuously during autonomous operation for extended monitoring. *tRackIT OS* is built using PIMOD [105], which allows configuration of single-board computer system images in a reproducible manner. The resources required to build *tRackIT OS* as well as the OS image itself are released under a GPL 3.0 license[14].

In Figure 5.12, the main software components of *tRackIT OS* are presented. Station-initiated communication is handled using the *Message Queuing Telemetry Transport* (MQTT) protocol, with *mosquitto* as an MQTT client and server implementation [147] for message distribution. It is configured such that incoming messages are forwarded to remote MQTT brokers for

[12] Dragino SX127X: `https://www.dragino.com/products/lora/item/106-lora-gps-hat.html`
[13] Dragino PG1301: `https://www.dragino.com/products/lora/item/149-lora-gps-hat.html`
[14] *tRackIT OS*, available online `https://github.com/Nature40/tRackIT-OS`

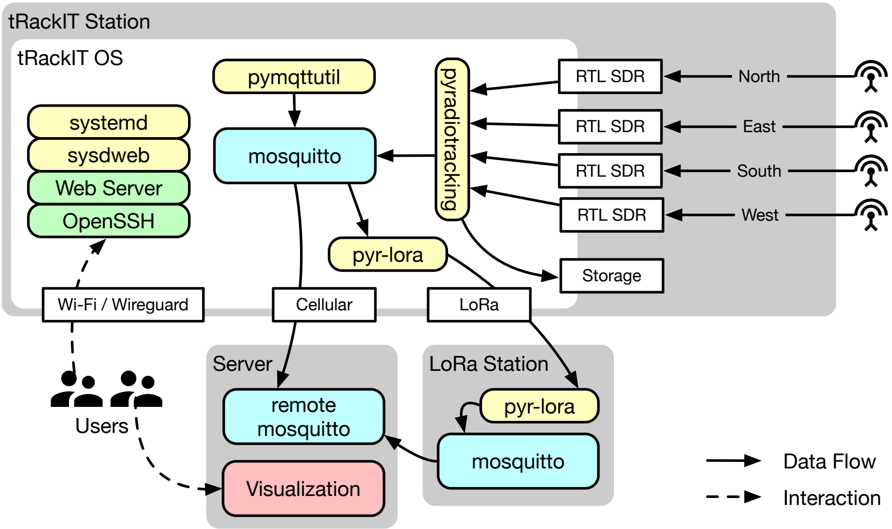


Figure 5.12: Overview of the main software components of a *tRackIT OS* distribution.

further processing. These brokers are also responsible for detecting and resolving connection failures.

The core software component for signal detection is called *pyradiotracking*. The component reads samples from all four SDRs, as well as detects, filters, and matches signals of VHF tags. Detected signals are saved to local storage, displayed via a custom web user interface, and published to a local message bus that is responsible for data distribution. Section 5.3.3 discusses the implementation details of *pyradiotracking*.

For system monitoring, we implemented a custom tool called *pymqttutil* in the Python programming language. It is released under a GPL 3.0 license[15]. The tool executes configurable Python statements in a fixed schedule and publishes the corresponding results via MQTT. It is configured such that relevant system metrics are published in a 5 minute interval, i.e., temperature, system uptime, system load, memory usage, CPU frequency, network addresses, storage utilization, and cellular data usage.

All services are managed by *systemd*. The WebUI *sysdweb*[16] for *systemd* is configured to allow easy log access and service control for (mobile) users. The *Caddy* web server is used to provide convenient access to the local storage, *pyradiotracking* and *sysdweb*. Finally, OpenSSH provides direct system access for local and remote users. To allow secure remote access, *wireguard* is used as a virtual private network (VPN).

[15] *pymqttutil*, available online: `https://github.com/Nature40/pymqttutil`
[16] *sysdweb*, available online: `https://github.com/Nature40/sysdweb`

Signal Detection

The signal detection algorithm is implemented in the *pyradiotracking* Python package, which is released under a GPL 3.0 license[17]. In Figure 5.13, the stages of signal processing are presented in a block diagram. First, spectrograms of the incoming IQ samples are created, which are used to detect signals. The detected signals are then filtered for shadow signals of lower power in neighboring frequencies and sent to a central signal queue. The detected signals of multiple antennas are matched and written to a local file, published to the MQTT message bus, and visualized in the local dashboard.

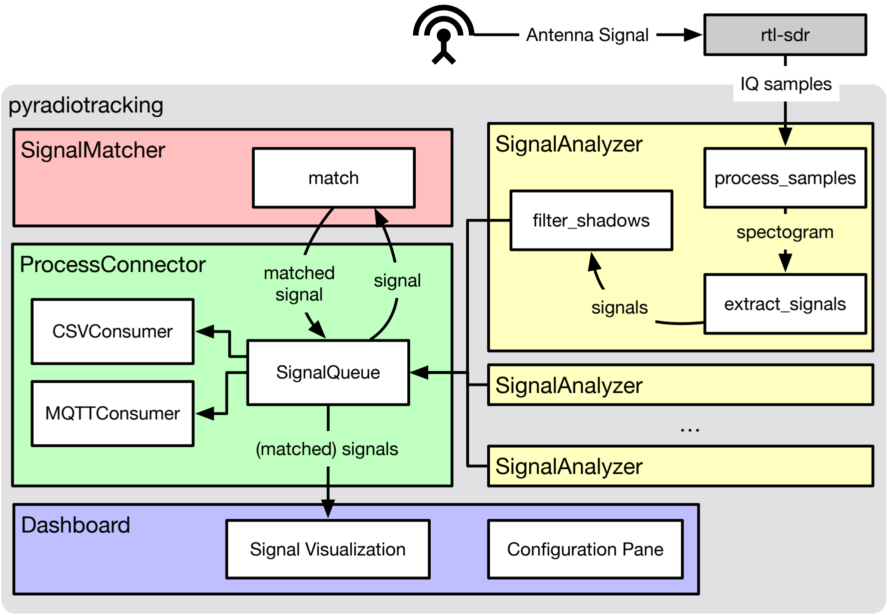


Figure 5.13: Signal analysis stages implemented in *pyradiotracking*.

To illustrate how the different stages work, data of the length of one second is used as an example. An SDR is configured such that a center frequency of 150.150 MHz, a sample rate of 300 kHz, and a fixed gain of 49.6 dB are used. A test tag with the frequency of 150.172 MHz and a signal duration of 40 ms was placed near to the receiving antenna. In Figures 5.14, 5.15, and 5.16, three stages of signal processing are visualized.

Figure 5.14 shows the raw IQ samples received by the SDR. Following the example configuration described above, there are 300,000 samples, hence 600 kilobyte of data collected in one second. In the time interval of $t_0 = 0.45\,s$ to $t_1 = 0.49\,s$, the IQ samples contain high values, which appear as a rectangle in the visualization. This rather sharp rectangle indicates that the gain value is set too high and the signal is clipping. When setting up stations for regular operation, the gain value must be chosen such that a good compromise of gain and clipping is achieved.

To detect single signals from the received data, a spectrogram is computed and processed further. This is achieved by applying consecutive Short-time Fourier Transforms (STFT) [4] to

[17] *pyradiotracking*, available online: `https://github.com/Nature40/pyradiotracking`

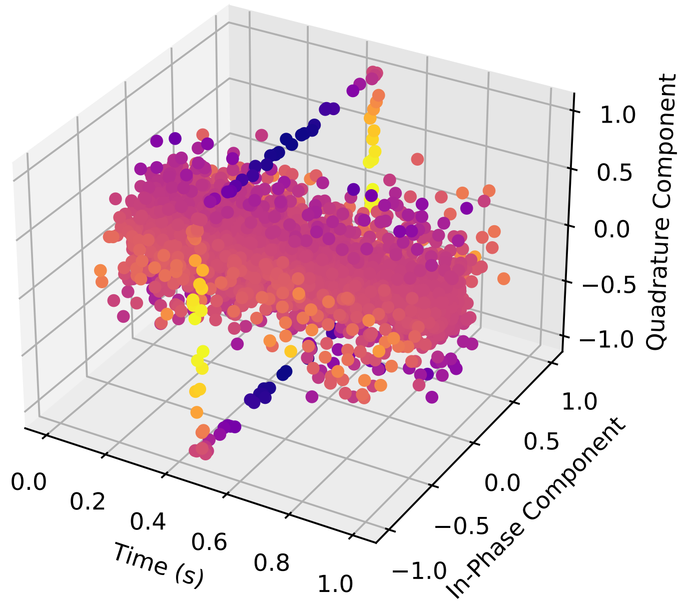


Figure 5.14: IQ samples of one second, as received by RTL-SDR.

the data. Figure 5.15 shows the spectrogram computed from the previously presented example data. The STFTs are computed with 256 samples per Fast Fourier Transform (FFT) and no overlapping samples. The Hamming window function is applied to smoothen discontinuities at the start and the end of the processed FFT. In this configuration, the bandwidth of 300 kHz is divided into 256 bands and a frequency resolution of 1,171 kHz, to achieve a time resolution of $1.0\,s/1,171 = 0.853\,ms$.

In Figure 5.16, signal detection on individual frequencies is visualized. The signal power (dBW) in the logarithmic scale is plotted for four example frequencies near the test sender's frequency, and the signal emitted by the test sender can be observed in three of those. The gray dashed horizontal line indicates the configured signal power threshold of -60 dBW. The gray dotted vertical lines show scan points used for initial signal detection. The blue arrow marks the total detected signal length. Signal detection is achieved by (a) iterating through all frequencies and (b) iterating through time using scan points placed according to the minimal detectable signal duration of 8 ms in our example. The signal-to-noise ratio (SNR) is calculated using the ratio of the current power and the average signal power of this frequency. If signal power and SNR at the scan points are above the configured thresholds, a potential signal is detected. The scan is then continued by evaluating the thresholds for all neighboring values until the thresholds are undershot, indicated by the blue arrows. If the duration of the detected signal is within the set limits, further complementary features are computed and added to a list for further processing.

After all signals of a spectrogram are extracted, shadow filtering is performed. We define a shadow signal as a signal that matches another signal in duration and time, but has a lower detected power. In the example of Figure 5.16, the signals detected at 150.170 MHz and 150.172 MHz would be shadow signals of the 150.171 MHz signal. The shadow signals are

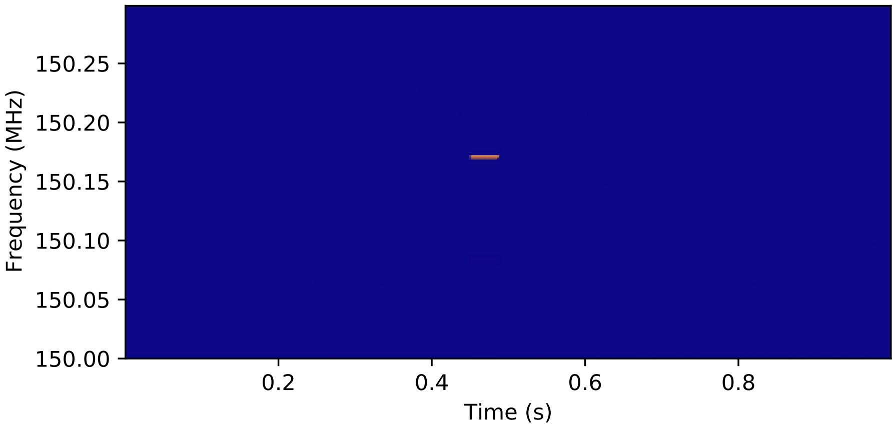


Figure 5.15: Power spectral density (PSD) of samples computed via Short-time Fourier Transform (STFT).

removed and the detected primary signals are added and written to disk, published via MQTT and sent to *pyradiotracking's* main process for signal matching and data presentation.

To improve reliability, a direct control component is introduced. The `librtlsdr` library used to retrieve data from an SDR works in such a way that as soon as requested data is available, a callback method is called. If the system load is too high and the callback method takes longer than the acquisition of the next samples, individual samples are omitted. Hardware and library-specific errors may lead to no callbacks at all. The first problem is monitored by comparing the actual received samples with the expected number of samples using the system clock. In this way, dropped samples can be detected, even accumulated over longer periods of time. The second problem is solved by (re-)setting a periodic alarm, comparable to a dead man's switch. If the callback method is not called in time, an alarm is triggered. This terminates the analysis process, which is then restarted by *pyradiotracking's* main process.

Signal Matching

The detected and filtered signals of multiple antennas are consumed by the signal matcher, which works as follows. In a list, all currently active signal groups are held. When a new signal is detected, it is compared to each of the active signal groups in time, duration, and frequency. The SDR devices used in the project do not work synchronously and use individual quartz crystals as their clock sources, hence time and frequency mismatches are likely to happen. If all parameters of an active signal group are within the configured thresholds, the signal is added to the corresponding group. If no corresponding group is found, a new active signal group with this signal is created and added to the list. After a certain timeout, the active signal groups are removed from the list and the key features are written to disk and published.

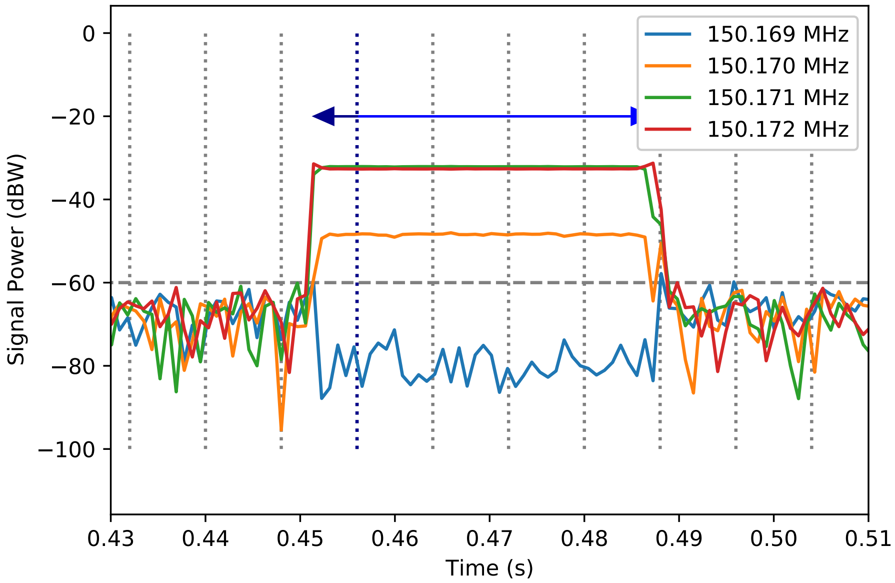


Figure 5.16: Power spectral densities (PSDs) of selected frequencies, minimal signal power threshold, and signal power sampling points.

Data Publishing

Detected signals are published directly to disk in CSV format and via MQTT in the CBOR format, which is a binary format and introduces smaller overheads compared to text-based formats. The MQTT broker running on a *tRackIT station* can be configured to forward published signals to other brokers, such as a central server via a cellular network or other IP-based networks.

Field	Accuracy	Size (bit)
Time	ms of current min	16
Frequency	offset to 150 MHz in kHz	9
Duration	ms	6
Signal Availability	flags	4
Signals	3-decimals	$[1-4] \times 17$
		$52-103$

Table 5.7: *tRackIT station*'s LoRa matched signal payload: fields, accuracy, and sizes.

In addition to this IP-based data publishing, LoRa can be used to publish signals. LoRa is a physical layer protocol based on *chirp spread spectrum* (CSS) modulation, that is robust against channel noise, multi-path fading, and the Doppler effect. This allows transmission ranges of a few kilometers in urban environments and up to 15 kilometers in rural areas, with minimal power requirements but also only low data rates between 300 bps and 50 kbps [102, 160, 187].

The LoRa publishing service of *tRackIT OS* receives signals through the local MQTT broker, converts the data in a custom data-saving binary format, and sends it via LoRa. Table 5.7 shows the fields used for a matched signal's payload, including accuracy and required size in bits. A matched signal contains a minimum of one and a maximum of four signals, depending on the number of antennas that received the signal, hence the final payload size is 52 up to 103 bits. Zeros are appended to the payload to reach the required byte boundaries, resulting in messages of 7, 9, 11, and 13 bytes, depending on the number of the contained matched signals. Compared to the already compact representation in CBOR of a 4-component matched signal (56 bytes + overhead), a reduction of up to 77% is achieved (13 bytes). In the most robust LoRa settings (SF:12, BW:250 kHz, CR:4/8), such a shortened message would require 594 ms Time-on-Air (ToA) (using Implicit Header mode with a 1-byte sender ID, the total length of the packet is 14 bytes). Following the duty cycle regulation of a maximum utilization of 1% (10%) per band, a message could be sent every 59 (5.9) seconds. While these settings do not allow continuous monitoring of individuals, sparse reporting of single observations are still of value, when trying to detect tags fallen off or with empty batteries. For stations in closer physical proximity to the receiving gateway, less robust settings may be chosen. Using a less robust LoRa setting (SF:8, BW:250 kHz, CR:4/8), the ToA drops down to 45.5 ms, allowing messages to be sent in an interval of 4.6 (0.46) seconds. From previous measurements in the Marburg Open Forest, signals could be reliably transmitted over 600 meters using this setting.

5.3.4 Experimental Evaluation

In this section, we evaluate *tRackIT OS* in benchmarking scenarios and in field experiments. The data of all experiments is publicly available at `https://github.com/Nature40/hoechst2021tRackIT-eval`.

Experimental Scenario

To evaluate *tRackIT OS* in a realistic manner, we use a system setup in the *Marburg Open Forest*, consisting of 15 *tRackIT* stations; 5 of them are used in our evaluation described below. The experiments are carried out twice: (a) with the most recent *tRackIT OS 0.7.0* and (b) using the most recent stable operating system version of the *radio-tracking.eu*[18] project [80], called *paur 4.2*. We activated a test tag and carried it around in the area of the selected *tRackIT stations* together with a GPS receiver to receive ground truth data. The experiment took place over the course of 0:51:10 h with a VHF sender of 600μW power, 20 ms duration and an interval of one signal per second, which results in 3,193 sent signals. In Figure 5.17, the GPS trace of the conducted experiment is presented; stations are marked by the white circles, and the trace is colored to indicate the time component of the experiment.

Our observations using *paur* in two seasons of 2019 and 2020 indicated high numbers of falsely detected signals. We were not able to distinguish between true and false positives through the information available after signal detection. Thus, we used a power signal threshold. The *paur* experiments conducted in this section showed the same low precision, hence all detected signals with a power lower than -78 dBW were removed for further processing. Using *tRackIT*

[18] `https://radio-tracking.eu`

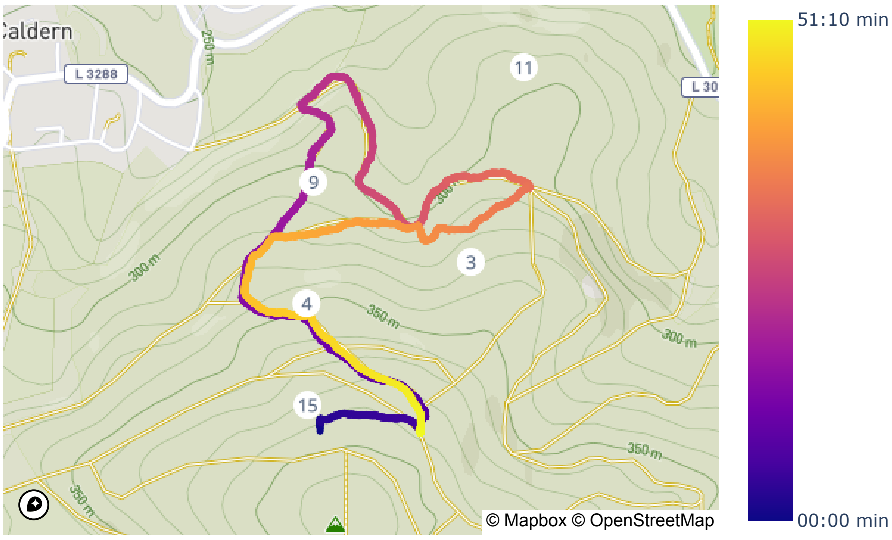


Figure 5.17: GPS trace of the experimental evaluation track and the corresponding *tRackIT* stations.

OS, this threshold is not required, since we observed very low numbers of falsely detected signals.

Signal Delay

A second observation from our previous field seasons in 2019 and 2020 is a delay in signal detection using *paur* in the order of seconds to minutes. In Figure 5.18, an example of observed signal delay is visualized. The dots show the received signal strength measured on multiple antennas of the same *tRackIT station*. Every antenna received a series of signals with low variance in signal strength that appear to be a straight line, indicating that the tag is not moving. However, these straight lines on the individual receivers are offset in time from each other, which makes further processing of the data difficult and and leads to worse to unusable bearing calculation. In the experiments of this section, we measured a delay in signal detection of 8 seconds in *paur* and no recognizable delay in *tRackIT OS*.

Signal Detection

In the *tRackIT OS* experiments, the selected five *tRackIT stations* detected 30,507 signals, each on potentially four antennas, resulting in an average of 1,525 signals (47.8%) detected per antenna. Signal detection depends on various factors, such as geographical and topographical conditions, the orientation of the antenna, the height of the transmitter above the ground, air humidity, and forest cover. Figure 5.19 shows the numbers of detected signals by station and antenna. Due to the positioning of the stations, the orientation of the antennas, and the selected test area, some of the antennas receive only a very small amount, others a large amount of the test signals. On the north antenna of station 11, only 95 (3.0%) signals could be detected, while 2,611 (81.8%) signals where detected on south antenna of station 9.

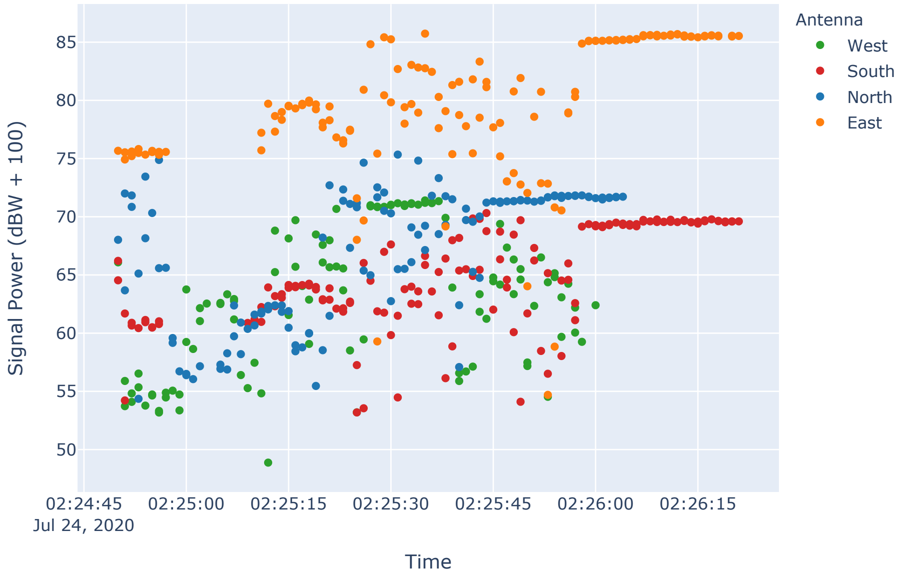


Figure 5.18: Example of signal delay among different receivers observed in the 2020 field season using *paur*.

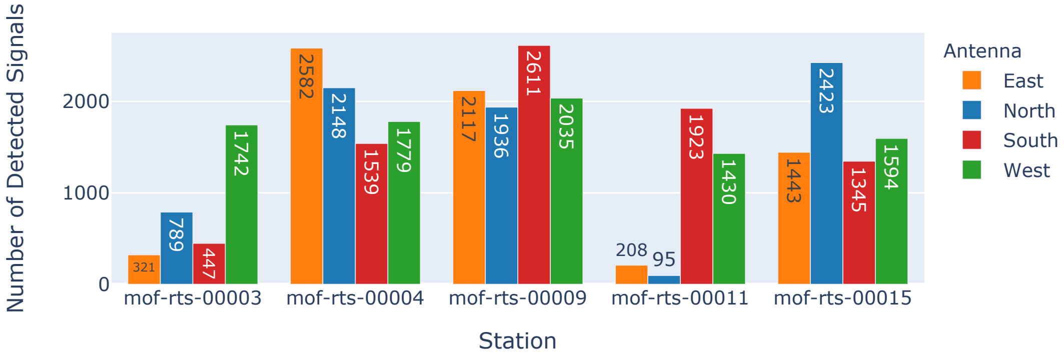
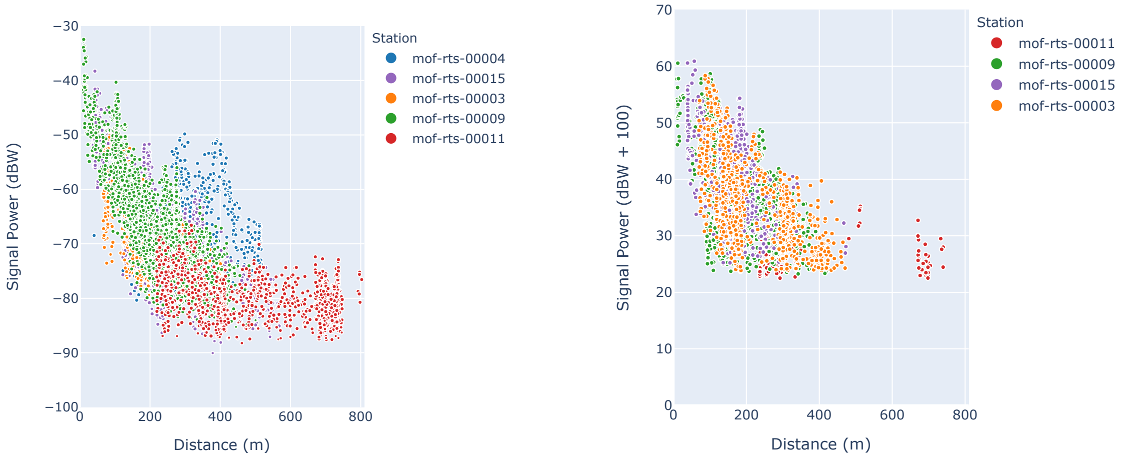


Figure 5.19: Detected signals on *tRackIT stations* in the experimental scenario.

In addition to this quantitative analysis of signal detection, we evaluated the distances between the test tag and the *tRackIT stations*. Figure 5.20 shows the distance of the tag and stations measured via GPS and the power of the detected signal. While most stations can detect signals at distances of up to 400 meters, stations 4 and 11 detect signals up to 800 meters away. While the correlation of signal strength and measured distance is straightforward, a high variance can be observed from the data and signal strength alone, hence this is not a suitable estimator for distance in the presented experiment. Initially, the overall performance of the two systems appears comparable, especially for signals with high signal strength. While *paur* received 2,728 signals usable for bearing calculations, *tRackIT OS* received 4,438 such signals, an increase of 62.7%, when applying the same -78 dBW threshold. In addition, *tRackIT OS* received 1,108 signals of lower signal strength, which corresponds to an effective increase of 103.3% compared to *paur*.



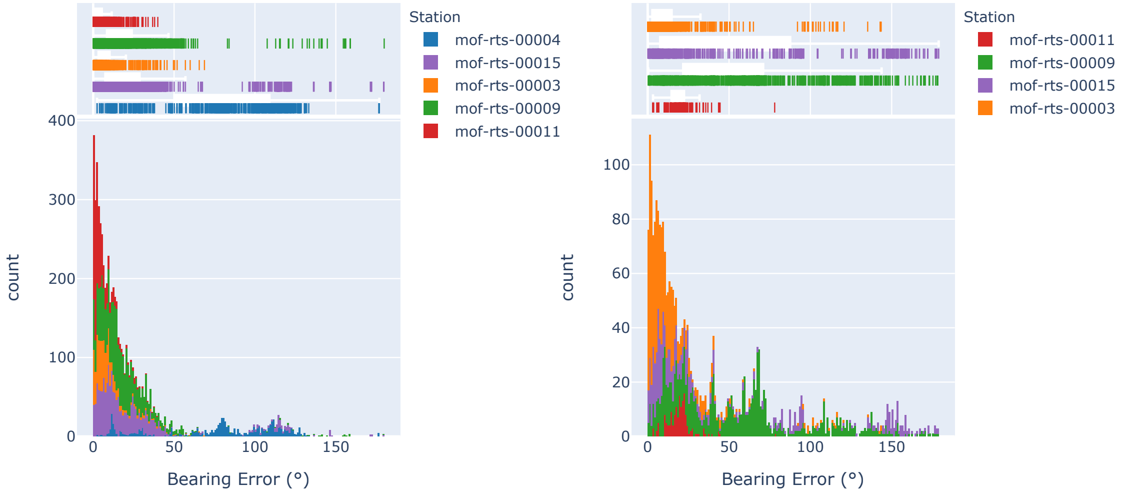
a) tRackIT OS

b) paur

Figure 5.20: Signal power and distance to a receiving station.

Bearing Calculation

To reach the goal of signal triangulation, signals detected on multiple antennas of a station are used to calculate bearings. We use the method proposed by Gottwald et al. [80] to produce comparable results for our bearing calculation. First, the pair of neighboring antennas with the highest and second highest signal strength are selected (s_l, s_r) and the relative gain difference δg is computed using the maximum signal strength difference δm: $g = \frac{s_l - s_r}{\delta m}$. Second, the signal strength is used to calculate the bearing between the antennas following the formula derived from the cosine theorem $\omega = \frac{\pi}{90} \times arccos(\delta g)$.



a) tRackIT OS

b) paur

Figure 5.21: Histogram of bearing errors.

Figure 5.21 shows a histogram of bearing errors in *tRackIT OS* and *paur*. Due to an error on station 4 which was not resolved automatically, signal detection failed on this station in the *paur* experiment run, hence no data is presented in the histogram. While *tRackIT OS* has a mean bearing error of 23.7° and a standard deviation of 30.7°, *paur* not only has a lower total bearing count, but also results in 38.9° mean bearing error with 42.6° standard deviation. These results indicate that *tRackIT OS* is superior to *paur* that represents the current the state of the art in this field.

Power Requirements

To operate stations autonomously and to monitor and transmit data, a stable power supply is necessary. To get realistic values for the required power, we measured a *tRackIT* station at the 12 volts input using a Monsoon High Voltage Power Monitor[19].

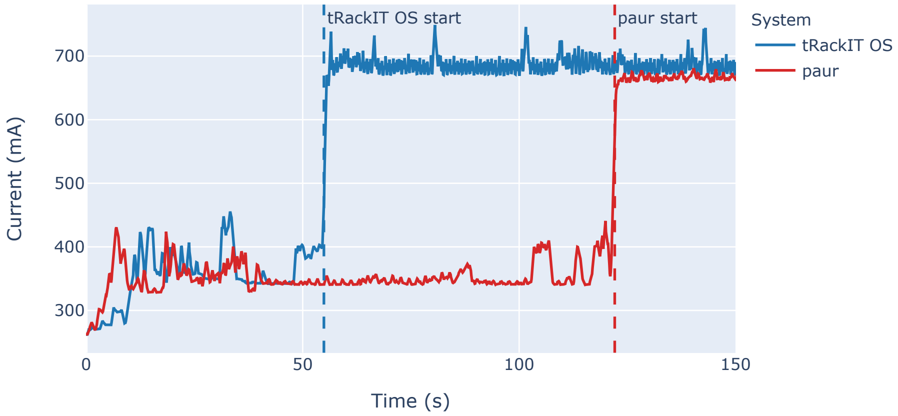


Figure 5.22: Power measurements of *tRackIT OS* and *paur* in default settings.

Figure 5.22 shows the power demands of *paur* and *tRackIT OS*. In contrast to *tRackIT OS*, *paur* does not start signal detection automatically. After all SDRs and signal analysis threads are running, *tRackIT OS* consumes an average of 8.23 W (684 mA), while *paur* consumes an average of 8.03 W (667 mA), which is an overhead of 2.55%. We also carried out experiments with varying sample rates (225 kHz – 300 kHz), but did not observe varying power demands. The systems used in the Marburg Open Forest use 12 V batteries with a capacity of 120 Ah (1440 Wh) of which only 80% should be used to limit wear, which allows a maximum theoretical runtime of 140 hours, or roughly 5.5 days. To allow a continuous operation, a 300 Watts peak solar panel is connected via a solar charger that even works during cloud cover. The presented results show only a slight increase in power consumption of *tRackIT OS* compared to *paur*, i.e., *tRackIT OS* meets the power requirements for continuous operation of the system.

[19] Monsoon Solutions Inc. High Voltage Power Monitor: `https://www.msoon.com/online-store/High-Voltage-Power-Monitor-p90002590`

5.4 Classifying Activity States of Small Vertebrates Using Automated VHF Telemetry

5.4.1 Introduction

The behaviour of an animal can be fundamentally divided into active and passive states [89], with the former requiring a much higher energy expenditure [199]. Quantifying the distribution of activity periods throughout the day provides important insights into species' responses to their environment, foraging strategies, bioenergetics, and adaptations [240]. Moreover, temporal segregation of species that share the same niche is one recognized mechanism that can facilitate stable species coexistence [172].

Detailed analysis of individual activity patterns requires high-resolution observations [173], which are often difficult to obtain. The observer's presence may influence animal behaviour and thus bias conclusions [41], and continuous observation of elusive or highly mobile species in habitats with dense vegetation is close to impossible [155]. Information on medium to large-sized species can be obtained using camera traps, GPS transmitters, and accelerometers [123], as demonstrated by investigations of dynamic habitat and resource use [263], behaviour [67], and migration and dispersal [250]. However, these devices are of limited use for small animals (<100 g), due to low detection probabilities, the trade-off between transmitter size and weight, battery life, and data-collection intensity [92, 93, 257]. Newer technical solutions such as the ATLAS system [173] or the Wildlife Biologging Network (WBN) [195] allow the tracking of small animals with high temporal and spatial resolution, but the required installation effort and costs remain high.

Very high frequency (VHF) telemetry has been employed in wildlife tracking since the 1960s [36], with the ongoing miniaturisation of VHF transmitters (< 0.2 g) allowing the tracking of small taxa (body mass < 5 g), ranging from large insects to small vertebrates [171]. Some studies take advantage of the fact that even small movements of tagged animals result in discernible variations in the strength of the received signal [128] that reflect changes in the angle and distance between the transmitter and receiver 5.23. However, collecting reasonable amounts of data on activity bouts using manual radio-telemetry requires an enormous amount of fieldwork, which implies a high level of wildlife disturbance [126], and the risk of missing critical events in the life of the tagged individuals is high [138].

Kays et al. [124] proposed a method for automatically classifying active and passive behaviour based on a threshold in the difference in the signal strength of successive VHF-signals recorded by a system commercially available at the time. Schofield et al. [210] applied a similar system to investigate the activity pattern of 241 individuals out of three migrating songbird species during stopover. They found that a threshold of 2.5 dBm optimally separates active from passive behaviour. However, the applied systems could only track one tag at a time, resulting in a low temporal resolution (i.e., a few seconds of observations approximately every 10 min due to switching through frequency channels). High-throughput tracking systems (<10-s data interval, many individuals at a time) are now widely available and enable ground-breaking research in animal behaviour, evolution, and ecology [173]. In recent years, with the ongoing development of low-cost open-source solutions, automatic VHF radio-tracking now enables such high resolution capacities. Systems such as the Motus Wildlife Tracking System [231] or

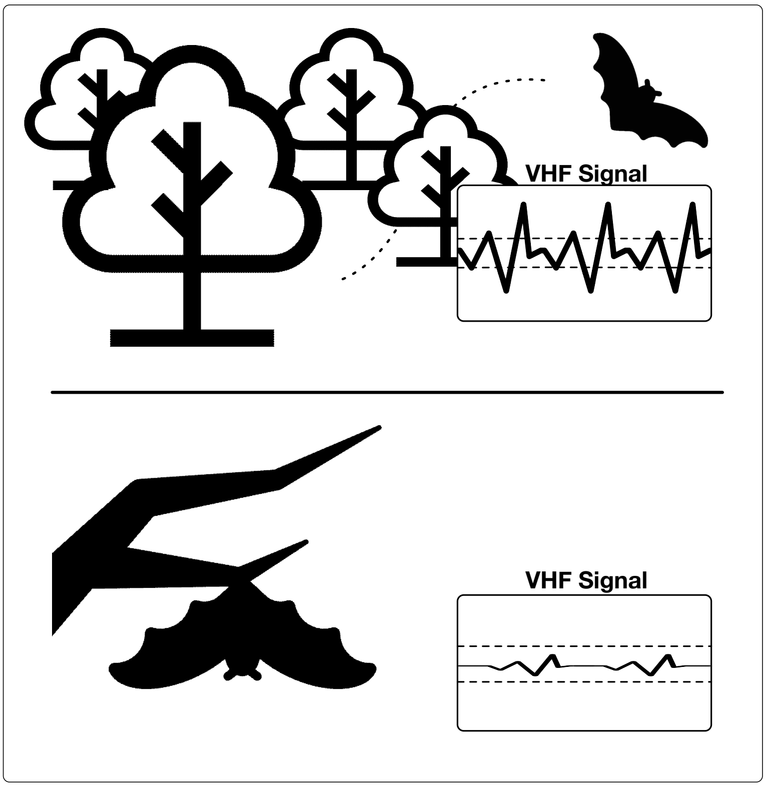


Figure 5.23: Principle of activity-recognition-based on very high frequency (VHF) signal patterns. Top: flying bat; bottom: resting bat. The amplitude and variation of the signal strength over time increase when the tagged individual is moving.

the tRackIT-System [104] allow the tracking of many individuals simultaneously and with a very high temporal resolution (seconds) over the complete tagging period. Continuous, high-resolution recording of the VHF-signals makes the entire signal pattern available for subsequent data analysis.

In this work, we build on the methodology of Kays et al. [124] by calibrating a machine-learning (ML) model based on millions of data points representing the behaviours of multiple tagged individuals of two temperate bat species (Myotis bechsteinii, Nyctalus leisleri). Many machine-learning algorithms are optimised for the recognition of complex patterns in a dataset and may be more robust against factors that influence signal propagation, such as changes in temperature and humidity, physical contact with conspecifics and/or multipath signal propagation [3] than a rule-based approach relying on a single separation value. ML approaches may therefore provide substantial improvements in the accuracy of individual activity states classification compared to threshold-based approaches.

Although deep learning methods have been successfully applied to several ecological problems where large amounts of data are available [34], we chose a random forest model due to the following reasons: (a) developing a (supervised) deep learning method requires considerable effort for selecting an appropriate neural network architecture, choosing an appropriate framework to implement the neural network, training, validating, testing, and refining the neural network [34], (b) our classification tasks resolve to a simple binary classification of active/passive states based on tabular data. In this setting, tree ensemble methods such as random forests seem to have clear advantages - they are less computationally intensive, easy to implement, robust, and at least as performant as deep learning [215], and (c) in a large study comparing 179 classifiers applied to the 121 classification data sets of the UCI repository, random forests are the best classifiers in over 90% of the cases [60].

Our random forest model was used in conjunction with recent developments in automated radio-telemetry [80, 104] to develop a toolset that allows researchers to record the activity patterns of even very small species (body mass < 5 g) in their natural habitat and with high resolution. The method was tested by applying it to independent data from bats, humans, and birds recorded in a densely vegetated and hilly area and then comparing the results with those obtained by the threshold-based approach of Kays et al. [124] using the separation value suggested by Schofield et al. [210].

In our method, activity states are recognised with high temporal resolution (< 10 s) and high accuracy. In the following, we provide detailed information on the application of the random forest model and its validation using data on the behaviour of tagged bat and bird individuals generated with an open-source multi-sensor tool [81].

In a case study, we demonstrate the use of the approach to detect differences in activity patterns between those of the two forest dwelling bat species Myotis bechsteinii and Nyctalus leisleri.

In the next sections we detail our process for developing and validating a random forest model to classify VHF-signals based on activity data gathered on bats, birds and humans. We showcase the possible insights in wildlife monitoring that our approach may bring by providing an ecological case study focusing on the comparison of activity patterns between two bat species. Detailed information on data processing and analysis is provided, along with an R package, example scripts and data in a hope to promote broad application in wildlife monitoring and ecology.

Parts of this section have been published in Jannis Gottwald, Raphael Royaute, Marcel Becker, Tobias Geitz, Jonas Hoechst, Patrick Lampe, Lea Leister, Kim Lindner, Julia Maier, Sascha Roesner, et al. “Classifying the Activity States of Small Vertebrates using Automated VHF telemetry.” In: *submitted; under review* ().

5.4.2 Field Methods

Study Area

The study was conducted in the Marburg open Forest (MOF), Hesse, Germany, a densely vegetated mixed forest of 200 ha, dominated by European beech (Fagus sylvatica) with some clearings and a relatively strong relief for low mountain ranges (lowest Position 200m, highest 400m) 5.24. The forest is home to 13 species of bats and 43 species of birds.

Tagging of Bats and Birds

Every year, we caught and then tagged bats and birds with customised VHF-transmitters of different sizes and weights (V3+, Dessau Telemetrie-Service; 0.3g-1g). Tag weights were always <4% of the body mass of the tagged individual (see S2 for technical details, methods and permits). For the ecological case study on bats, we captured and tagged 91 bat individuals from two focus species (66 M. bechsteinii and 25 N. leisleri). For the evaluation of our approach (see "transferability to small diurnal flying vertebrates" section) we used data of 19 bird individuals tagged in another study conducted in parallel (1 Leiopicus medius, 3 Cyanistes caeruleus, 3 Erithacus rubecula, 3 Garrulus glandarius, 3 Parus major, 3 Sylvia atricapilla, 3 Turdus merula). The frequency separation between transmitters used simultaneously was at least 3 kHz.

Radio-tracking

From 2018 to 2021, we operated a network of 15 custom-designed automatic radio-tracking stations (henceforth 'tRackIT-stations'; [80, 104]) distributed over the MOF 5.24. The stations recorded signal frequency, duration, and strength as well as the timestamp of the signal of all individuals tagged at a given time simultaneously and automatically.

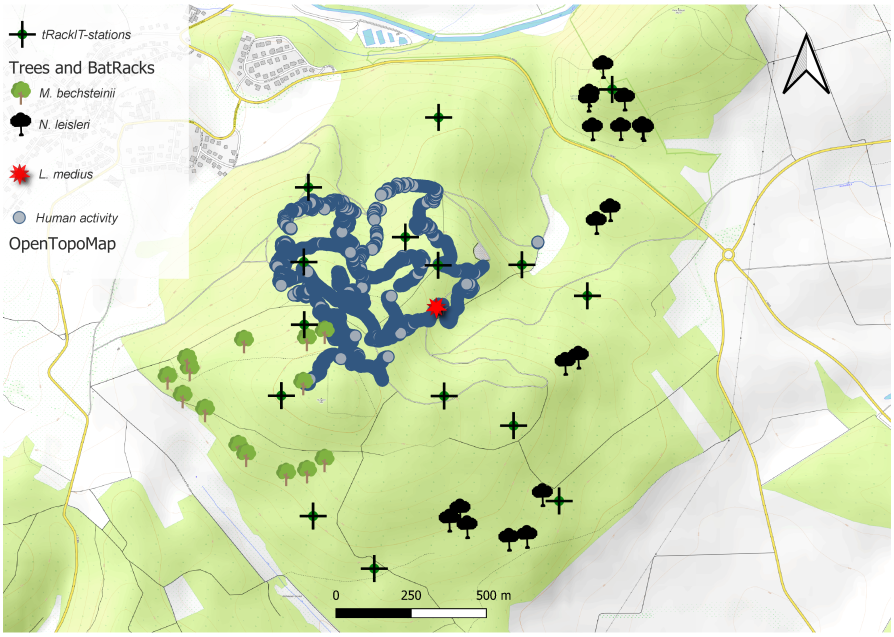


Figure 5.24: The Marburg Open Forest in Hesse, Germany. The map shows the locations of the tRackIT-stations [80, 104], the roost trees of bats (M. bechsteinii, N. leisleri) observed by BatRack multi-sensor stations [81], the breeding site of a woodpecker (Leiopicus medius) and the GPS track (shown in blue) of the activity simulation used to test the transferability of the classification method to birds and humans. (Map data from OpenStreetMap)

Each tRackIT-station consisted of four directional antennas with moderate directivity (HB9CV-antenna). While this antenna design reduces the reception range to <1000m in hilly and vegetated terrain, it guarantees overlapping radiation patterns of neighbouring antennas which was necessary for bearing calculation and subsequent triangulation as described in Gottwald et al. [80]. However, tracking of positions of tagged individuals is not part of this study. The towers of the stations had a height of approximately 8m and antennas were oriented north, east, south, and west. We permanently monitored a frequency range of 150.000-150.300 MHz.

From 2018 to 2020, we used the paur 4.3 software developed by the open-source project radio-tracking.eu [80] but switched to the tRackIT-operating system [20] in 2021 due to high amounts of noise and frequent software failures that often went unnoticed [104]. The tRackIT-system enables live transmission of parameters to assess the health of the stations as well as transmission, processing and visualisation of VHF-signals. The former greatly reduces maintenance time and the latter enables tracking of activity, positions and body temperature in near real time. For a detailed description of the hardware and software, please see Gottwald et al. [80] and Höchst & Gottwald et al. [104].

The VHF data was filtered by tag frequency +/- 3 kHz and signal duration +/- 5 milliseconds according to settings given by the manufacturer. For the data recorded with the radio-tracking.eu software, we had to visually assess the success of the filtering procedure and in some cases remove recordings below a station- and frequency-specific threshold in dBW due to high amounts of electromagnetic noise. In total, we used data from 72 individuals (M. bechsteinii: NID = 52, NObs = 577,977; N. leisleri: NID =20, NObs = 204,443) monitored for an average of 19 days (according to battery power) to distinguish active from passive states.

5.4.3 A Random Forest Model to Classify Activity States based on VHF Signals

Groundtruth

We used the patterns in the strength of the recorded VHF-signals together with a supervised ML algorithm to classify the activity of the tagged individuals. Supervised ML requires training and test data for implementation. We monitored 23 out of the 72 tagged bat individuals (6 N. leisleri and 17 M. bechsteinii) using a multi-sensor tool [81] to supply the random forest model with periods of known activity and inactivity.

First, the roost trees of tagged bats were located via manual radio-telemetry between June 9, 2020 and July 26, 2020 and between May 10, 2021 and August 18, 2021. We then set up custom-made video recorder ('BatRack') units to automatically record videos of tagged individuals ([81]; [21]). BatRacks consist of a VHF-antenna and an infrared video unit connected to a Raspberry Pi single board computer. We installed the cameras with a focus on the roost entrance and its surrounding area (40-m radius), which allowed the motion of tagged individuals to be captured on the video tracks. The infrared camera unit was automatically triggered by the VHF-signal of the bat transmitters and started recording if the VHF-signal strength exceeded a threshold of -60 dBW, i.e., when a tagged bat flew close to the roosting tree and the BatRack system.

[20] https://github.com/Nature40/tRackIT-OS

[21] https://nature40.github.io/BatRack/ (vid. 2)

We manually reviewed the video tracks recorded by BatRack units in conjunction with the VHF-signal, and classified the observed behavioural sequence into the categories swarming, passing, entering or emerging from the roost. Sequences that showed swarming, passing or emerging were classified as active, and the time between entering and emerging from the roost as inactive. In addition to the sequences recorded on video, we classified periods of time as active if an individual was recorded in short time intervals on widely separated VHF-receivers (tRackIT-stations and BatRacks). From the three (2020) to nine (2021) BatRacks set in front of a total of 30 roosting trees of 6 N. leisleri and 17 M. bechsteinii individuals 5.24, 723 h of behaviour were recorded. For these periods of known activity type, we assigned a passive or active label to the VHF-data recorded by one or more of the 15 tRackIT-stations.

Predictor Variables

We calculated 29 predictor variables thought to capture the patterns in the signal strengths over time by applying rolling windows of ±10 data entries, corresponding to an approximate time window of 20 s, to each observation of the classified VHF-data recorded by the tRackIT-stations. We chose the window size to capture the dominant signal strength pattern without smoothing out even short changes of the activity state. To prevent averaging over longer periods, the dataset was split into 5 minute bins per station before applying the rolling windows. For each bin, we selected the receiver with the most data entries, i.e. the best data coverage. We only evaluated bins with at least 60 observations, i.e. three times the window size. This procedure ensures that only stations and receivers with relatively good reception are considered for classification.

To smooth out noise or potentially distracting fluctuations in the signal, we calculated a Hampel filter, in which data points that differ from the window median by more than three standard deviations are replaced by the median [94]. We also applied a mean and a max filter on the raw data of the main receiver whereby the respective data point was replaced with the mean or max of the rolling window. Next, we calculated the variance, standard deviation, kurtosis, skewness, and sum of squares for both the raw and the smoothed data, to capture the variability and shape of the data distribution within the window.

Only one antenna is necessary to classify VHF-signals into active vs. passive states. However, agreement between receivers of the same station provides additional information and can improve the reliability of the classification. This is especially likely if the individual is relatively close to the station (< 400 m in our scenario). When data were available from more than one receiver at the same station, we calculated the variance of signal strength between the receiver with the most and the receiver with the second most observations, together with the correlation coefficient and the covariance of signal strength in a rolling window of ±10 data entries. All variables are described in Supplement S1.

Training and Test Data

To give equal weight to each class and to avoid overoptimistic accuracy metrics caused by a comparably well-detected majority class, we balanced the ground truth dataset by randomly down-sampling the activity class with the most data to the amount of data contained by the

Setup	Active data	Passive data	Total data points	Balanced active	Balanced passive
1 receiver	588,880	2,654,873	3,243,753	294,440	294,440
2 receivers	249,796	1,469,674	1,719,470	124,898	124,898

Table 5.8: Characteristics of the test and training data obtained from 723 h of video observation on 23 tagged individuals.

class with the least data. We then split these balanced data sets into 50% training data and 50% test data for data originating from one receiver. We used the same procedure for data derived from the signals of two receivers, resulting in two training and two test datasets. From a total of 3,243,753 VHF-signals, we assigned 249,796 signals to train the two-receiver model and 588,880 signals to train the one-receiver model 5.8.

Model Tuning

We used a random forest model as our classification method because it tends to outperform other classifiers, as shown in an extensive comparative study [60]. This model type is also robust against multicollinearity in predictor variables, especially when used with feature selection procedures [84], as used in our approach. Since not all variables are equally important to the model and some may even be misleading, we used 50% of the data recorded by either one or two receivers to perform a forward feature selection as implemented in the "CAST" package [162]. This resulted in two random forest models, for data collected by one receiver and two receivers, respectively.

Groundtruth for Controlled Walks with Human Subjects

We conducted a series of 61 controlled walks with a human volunteer to test the reliability of the trained models when applied to various activity patterns and tag positions. This was achieved by moving the VHF transmitters at two different heights, 15 cm above the ground at the ankle and 4 m above the ground, on a pole, around the tRackIT-stations. We simulated inactive states standing still and movements on a small spatial scale were simulated by walking and hopping back and forth over an area of about 1 m^2. We simulated movements at a medium spatial scale by walking within areas of 40 m^2, and multiple back and forth displacements and displacements of at least 200 m were used to simulate large-scale movements. We performed each movement type for 3–10 min at different positions within the north-western part of the study area, which is characterised by a diverse topography and complete forest coverage 5.24. We recorded the beginning and end times of each sequences and all signals simultaneously recorded by one or more of the 15 tRackIT-stations and then manually assigned the known activity type (active or passive). The human activity dataset consisted of 32,175 data points (26,133 active, 6,042 inactive).

Model Validation

We applied the trained random forest models to the 50% of the data withheld for testing to evaluate their performance in classifying bat activity. The same trained models were applied to the human activity datasets. In a first step, we calculated the sensitivity $\frac{truepositives}{truepositives+falsenegatives}$ and specificity $\frac{truenegatives}{truenegatives+falsepositives}$ based on a comparison of the observed data with the activity class attributed by the random forest models for both dataset. Additionally, we calculated the F-score as the harmonic mean of the precision $\frac{truepositives}{truepositives+falsepositives}$ and sensitivity, the AUC and the Kappa index, which takes the probability distribution of each class into account. Values vary between 0 and 1 (<0 and <1 for Kappa), with values close to 1 indicating that the model shows an almost perfect agreement [33, 142].

Results of Model Validation

The trained random forest models performed equally well, with F-scores of at least 0.96 and sensitivities and specificities no less than 0.95, when applied to the validation data of bats and the human activity-simulation 5.9. Whether the tag was positioned 15 cm or 4 m above the ground during the human activity-simulation had no impact on the classification accuracy. The four activity levels simulated by a human were detected similarly well, with sensitivities between 0.95 and 0.97.

Comparison to a Threshold-based Approach

We compared the results of the ML-based approach with those of a threshold-based approach by calculating the difference in the signal strength between successive signals for the test datasets of bats and humans (for methods and results on the bird data see "transferability to small diurnal flying vertebrates" section). We applied a threshold of 2.5 dB, which was deemed appropriate to optimally separate active and passive behaviours in previous studies [210]. In addition, we used the optimize-function of the R-package stats (R Core Team, 2021) to identify the value of the signal strength difference that separated the training dataset into active and passive with the highest accuracy (i.e. 1.08 dB) and applied it to the test datasets. We calculated the same metrics as described above, except for AUC, which requires probabilities for each classification.

Regardless of the method used, all F-scores values were < 0.9 (2.5 dB threshold, bats: 0.63, humans: 0.74; 1.08 dB threshold, bats: 0.74, humans: 0.88) and Kappa values < 0.60 (2.5 dB threshold, bats: 0.37, humans: 0.3; 1.08dB threshold, bats: 0.46, humans: 0.53), which correspond to a moderate to fair agreement [142]. These values remain well below those obtained from our ML models however.

Dataset	n passive	n active	F1	AUC	Sensitivity	Specificity	Precision	Kappa
Bats 1 receiver	294172	294172	0.96	0.99	0.96	0.97	0.97	0.93
Bats 2 receivers	110273	110273	0.98	1.0	0.98	0.98	0.98	0.95
Human activity	6150	26504	0.98	1.0	0.97	0.95	0.99	0.90

Table 5.9: Performance metrics of the test datasets classified by the trained random forest model

5.4.4 Ecological Case Study: Comparison of Activity Patterns in Two Bat Species

In the following, we present an ecological case study to highlight the advantages of the fine-scale classification of activity states at a 1-min rate for two species monitored over four consecutive years. Both M. bechsteinii and N. leisleri are protected species (Habitats Directive 92/43/EEC) endemic to Eurasian forests but they differ substantially in their foraging habits. N. leisleri feeds on ephemeral insects that occur in large numbers, but only for short periods at dusk and dawn [14, 202] while M. bechsteinii partially collects its prey from the vegetation [47] and is thus generally less dependent on the timing of insect flight activity [202].

We focused on the following questions: 1) Do M. bechsteinii and N. leisleri differ in their overall probability of activity? 2) Do M. bechsteinii and N. leisleri differ in their timing of activity over the course of their circadian rhythms? To answer these questions, we compared the timing of the onset and end of activity periods, the timing of maximum activity and the overall duration of night-time activity bouts using the data processed with the random forest model.

The data were processed using the corresponding tRackIT R package `https://github.com/Nature40/tRackIT`. Example data processing routines with a small dataset [22] can be found at the package GitHub page [23].

Statistical Analyses

All analyses were conducted with R v. 4.1.2 [233], using the mgcv package for additive models (Wood, 2011). Reproducible scripts are available [24].

We used hierarchical generalised additive models (HGAM) to compare differences in the overnight activity patterns of M. bechsteinii and N. leisleri. These classes of models can be applied to estimate non-linear relations between responses while allowing for a variety of error terms and random effect specifications [186]. In this study, we modelled activity over the course of the 24-h cycle as follows:

$$P(activity)i = f(time)_i + \zeta_{ID} + \zeta_{DATE} \tag{5.2}$$

[22] `http://dx.doi.org/10.17192/fdr/104`

[23] `https://nature40.github.io/tRackIT/`

[24] `https://nature40.github.io/tRackIt_activity_ecological_case_study/`

where the probability of activity for observation i is modelled as a binomial variable (0: inactive, 1: active) as a function of the time of day (centered around sunset to account for seasonal shifts in daylight). We used a circular cubic spline with 120 equally-spaced knots to constrain the beginning and end of the 24-h cycle so that they matched. Individual identity and date were added as random effects to account for individual, seasonal and yearly effects. Given the volume of data (> 700,000 observations), all models were fitted through the bam() function for faster model estimation.

Given the short timespan between observations, our models had highly autocorrelated residuals ($\rho > 0.50$). While there are no strict guidelines for accounting for autocorrelation with binomial data in HGAMs, the residual autocorrelation was not influenced by the choice of the error family specified (gaussian vs. binomial). We therefore set the autocorrelation manually at a value equal to that of the first lag (ρ = 0.57) using the start_value_rho() from the itsadug package [246]. Next, we refitted with the estimated autocorrelation value with an AR1 structure. This procedure successfully accounted for autocorrelation, as evidenced by the decrease in the median autocorrelation to −0.13 in the refitted model. Visual inspection of the autocorrelation confirmed that ρ remained $< |0.15|$ at all lags.

We compared the activity patterns of the two species by contrasting the Akaike information criterion (AIC) values for a model in which species did not vary in their daily activity patterns (Model 0) against one in which the effect of time of day varied between species (Model 1, using the “by = species” argument to specify a time × species interaction). To visualize the fine-scale difference in activity patterns between M. bechsteinii and N. leisleri, we calculated the difference in spline functions, δ f(time). This more precisely revealed the period of the day when the two species were most likely to differ in their probability of activity (negative value: $P(activity)_{Bechstein} < P(activity)_{Leisler}$, positive value: $P(activity)_{Bechstein} > P(activity)_{Leisler}$). We further characterised the activity patterns of the two bat species by calculating the following metrics based on the predicted values for Model 1:

- Onset and end of activity periods, defined as the first and last time of day when the probability of activity was larger than chance (i.e. p(activity) > 0.5).
- Time of peak activity, calculated as the time of the day when the probability of activity was maximal.
- Activity duration, defined as the duration of the activity period during a 24h period weighted by the average probability of being active (in hours). This metric was calculated as the area under the curve between the onset and end of the activity period.

Species Comparisons of Circadian Activity

Nyctalus leisleri and M. bechsteinii showed pronounced differences in the shapes of their activity curves and these species differences were also supported by AIC model selection ($\delta AIC = 15092$, 5.10; 5.25). While both species appeared to synchronise their onset of activity with sunset, N. leisleri was active an average of 19 min earlier than M. bechsteinii. N. leisleri also reached peak activity earlier, but its activity markedly declined as soon as M. bechsteinii became highly active.

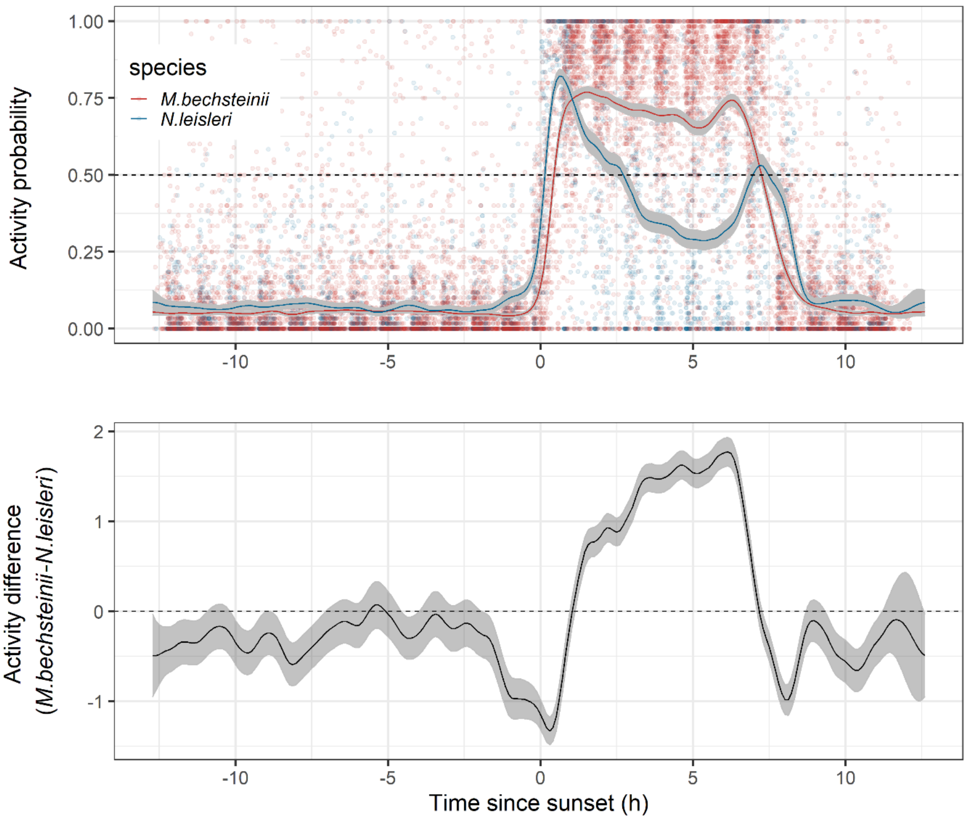


Figure 5.25: Nyctalus leisleri was consistently active sooner than M. bechsteinii, but the latter species had longer periods of continuous activity. Top panel: the points represent the activity probability calculated over 1-h intervals, and the solid lines the predicted values from the best HGAM model. The dashed line indicates the times when the population was equally likely to be detected as active or passive. Bottom panel: difference in the activity probability calculated from the best HGAM model. Positive values indicate a larger activity probability for M. bechsteinii than for N. leisleri.

The latter species was highly active throughout most of the night, as indicated by a significantly higher activity duration (area under the curve when p(activity) > 0.5 [95% CI]; M. bechsteinii: 4.70 [4.56; 4.83]; N. leisleri: 3.42 [3.23; 3.62]). However, M. bechsteinii reached the end of its activity period an average of 12 min sooner than N. leisleri (Table 4).

Transferability of the Models to Diurnal Flying Vertebrates (Birds)

Our previous section shows that the tRackIT-system can provide important insights into ecological differences between bat species with which the model was trained on. We know focus on the broader application of this method to other flying vertebrates.

Table 5.10: Model coefficients (β) and standard errors (SE) and test statistics (z, p) for the linear portion of the additive models (intercept) along with smoothed parameters for nonlinear terms (edf: estimated degrees of freedom, chi-squared and p-values).

	Model 0 ($AIC = 263401.1; R^2 = 0.45$)							Model 1 ($AIC = 248309.1; R^2 = 0.47$)				
Linear terms	β	**SE**	**z**	**p**			Linear terms	β	**SE**	**z**	**p**	
Intercept	-1.74	0.13	-12.98	<2 x 1016			Intercept	-1.74	0.13	-13.29	<2 x 1016	
Smoothed terms	**edf**	**df**	$\chi 2$	**p**	**% Variance**		Smoothed terms	**edf**	**df**	$\chi 2$	**p**	**% Variance**
time	51.84	118	4061287	<2 x 1016	17.42		time:Leisler	45.82	118	188033	<2 x 1016	8.83
							time:Bechstein	47.23	118	3571852	<2 x 1016	22.46
Random effects							Random effects					
ID	67.07	71	1976322	<2 x 1016	10.18		ID	66.32	71	1824189	<2 x 1016	4.78
DATE	280.26	306	3084452	<2 x 1016	17.44		DATE	282.01	306	2421809	<2 x 1016	10.73

Metric	*N. leisleri*	*M. bechsteinii*
Activity onset (h)	00:12	00:31
Time of peak activity (h)	00:37	01:33
Activity end (h)	07:23	07:11
Peak P(activity)	0.82 [0.80; 0.84]	0.70 [0.75; 0.79]
Activity density	3.42 [3.23; 3.62]	4.70 [4.56; 4.83]

Table 5.11: Activity metrics of N. leisleri and M. bechsteinii. Wake-up and sleep times were calculated as the first and last time when the probability of activity was > 0.5. All times are presented as hours since sunset. The time of peak activity represents the time of day when the probability of activity was maximal. Activity duration was calculated as the area under the curve between wake-up and sleep times.

To test the reliability of the model on birds, we attached a transmitter to the back of a middle spotted woodpecker (L. medius), and placed a daylight variant of the BatRack ("BirdRack") in front of its nesting tree for 4 consecutive days. The tree was located on a steep and completely forested slope of a small valley (Fig.2). A typical recorded sequence consisted of flying, hopping up the stem, and a very short feeding sequence during which the bird remained motionless at the entrance of its breeding cavity. Since the feeding sequence was usually shorter than three consecutive VHF-signals (2.5 s), we classified all recorded signals within such a sequence as active. To generate sufficient inactive sequences, 2,200 random data points were sampled from signals recorded by tRackIT-stations each night between 0:00 h and 2:00 h, while the woodpecker was presumably asleep, over four consecutive nights. The dataset of the woodpecker, based on the 75 observed activity sequences, consisted of 17,541 data points (8,741 active, 8,800 inactive).

We applied the two random forest models to all recordings of the tagged woodpecker and calculated the same performance metrics as for bats and human activity for the sequences of known activity. We used the entire woodpecker data set as well as the activity classifications of six additional bird species, each represented by 3 individuals, to assess the transferability of the model to birds of different size and movement habits. Since there are no actual observations for the latter and only partial observations for the woodpecker, we visually compared the classified activity of the woodpecker to patterns expected for diurnal vertebrates. Then, we calculated activity probability in relation to the time after sunset for three individuals from each of six small to medium-sized bird species (Table 5) using methods comparable to those of the ecological field study for bats.

Performance metrics for the sequences of known activity type of the woodpecker were in line with those for bats and human activity (F1 = 0.97; AUROC = 1, Sensitivity = 0.95; Specificity = 1; Precision = 1, Kappa = 0.94). Note that a threshold-based approach also behaved poorly on this dataset (2.5 dB threshold, F-score = 0.62, Kappa = 0.38; 1.08 threshold, F-score = 0.79, Kappa = 0.58).

Visual assessment of the active / passive sequences for the woodpecker showed typical patterns of high activity during the day, starting around sunrise (05:12) and ending around sunset (21:30; 5.26). The activity probability in relation to time after sunrise of the six additional bird species also correspond to the expected patterns for diurnal birds 5.26. Even though no actual observations were available, these patterns suggest a successful classification of the activity of different bird species.

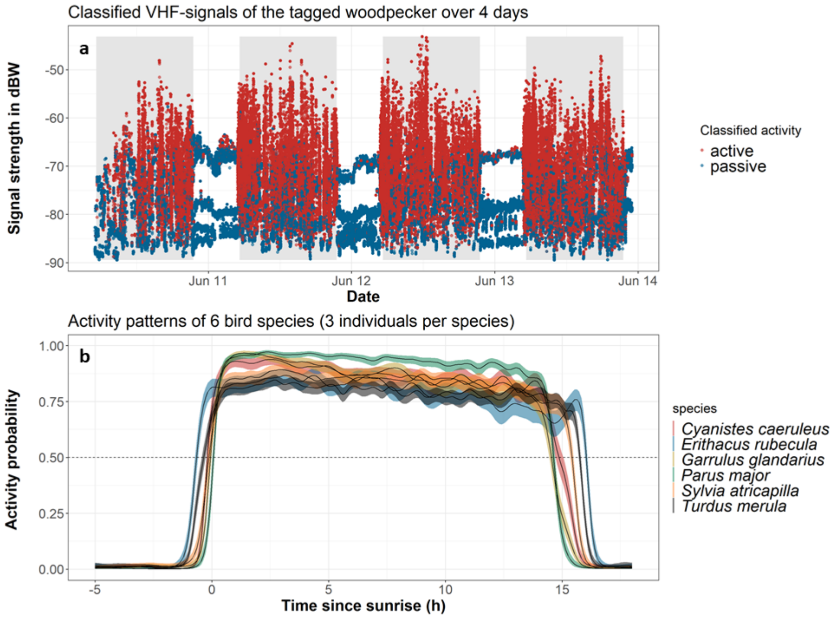


Figure 5.26: Signal strength [dBW] from a woodpecker tagged over four consecutive days and nights and the corresponding classification of the bird's activity into active (N = 146,962) and passive states (N = 303, 802). b) Probability of activity in relation to time since sunrise of six bird species calculated from activity classifications of three individuals per species (see ecological case study for methods). Periods of high activity are consistent with the diurnal activity patterns expected for these species.

The trained models are available [25]. The workflow for model tuning, evaluation and comparison with a threshold-based approach is available [26].

5.4.5 Discussion

Using a large dataset consisting of the observed behaviour of tagged bat individuals, we trained two random forest models to classify novel data from the same species into fundamental behaviour, and with high precision and high temporal resolution (1 sec interval). Our approach outperformed previous methods based on a threshold-based approach even when using a value calibrated with a large ground truth dataset. Although not inadequate, the threshold-based approach had generally lower and more variable performance metrics compared to our ML model. We also achieved similar precision when applying the ML models to ground truth data from other species (woodpecker and controlled human walks). The activity probability estimates of 18 additional bird individuals out of six species also matched expected activity patterns for diurnal vertebrates. This strongly suggests that our method generalises well and could be applied to a variety of vertebrates with similar accuracy (e.g. down to a body-mass of 4g with 0.2g transmitters [171]).

The high precision and high temporal resolution of our approach together with the easy accessibility of the developed methods may open new research avenues on the variations in the activity patterns among and within species in their response to the environment.

A more in-depth analysis of activity bouts as a function of abiotic factors or the detection of changes in patterns indicating, for example, breeding has not been conducted here, but such studies are likely to be feasible. Whether specific behaviours can be recognised (i.e., foraging, parental care, grooming), as is possible with accelerometers, also remains to be determined. The fact that the variance in the signal pattern depends less on the intensity of the movement than on the signal path remains an issue, however. While the amplitude of the measurements from accelerometers can be directly related to different behaviour classes [123], the amplitude of stationary recorded VHF-signals also changes due to the distance to the radio tracking station. The spatial context of the receiving stations as well as the localisation algorithms presented in Gottwald et al. [80] could provide additional information, such as distance to the station and direction of movement. However, for localisations, at least two radio-tracking stations are necessary and the spatial accuracy and reliability of position tracking when operating in cluttered environments such as forests is still under investigation.

The tRackIT system can currently record up to 90 individuals at a time within the same spatial context, but technology that allows for higher numbers is under development. Given the relatively low costs of the transmitters (130 €) and tRackIT-stations (1,500 €), the monitoring of an entire community of small vertebrates at high temporal resolution becomes possible with this system. For instance, a study investigating the activity states of an entire temperate forest bird community is currently conducted in the Marburg Open Forest. The tRackIT system now allows activity classification in real-time, which opens several exciting research avenues. For example, it is now being used to narrow down the time of death of chicks in meadow-breeding

[25] https://doi.org/10.17192/fdr/79

[26] https://nature40.github.io/tRackIT_activity_classification_model_tuning_and_evaluation/

birds to subsequently reduce error bars in nest survival models [27]. Personal observations during the bird-breeding season also showed clear shifts in the frequency and regularity of activity periods during the transition from the non-breeding to the breeding season (J. Gottwald). Future applications may also help automatically determine the (species-specific) onset of the breeding season in songbird communities.

Over the four years of the study, we collected data with two different software designs (radio-tracking.eu and tRackITOS) that show significant differences in data quality. We also covered a range of suboptimal recording conditions caused by topography and vegetation, which leads us to the assumption that the approach presented here is not exclusively applicable to data recorded with tRackIT-stations. Other open-source systems such as Motus (sensorgnome) [231], but also commercial systems such as the Lotek SRX/DX series receivers, record comparable data that may be used with the functionalities and models presented here. However, this was not tested as part of this study.

[27] https://www.audi-umweltstiftung.de/umweltstiftung/de/projects/greenovation/telemetry-technology.html

5.5 *BatRack*: An Open-source Multi-sensor Device for Wildlife Research

5.5.1 Introduction

Many of the important findings and principles of ecology and conservation biology have been derived from behavioural observations [35], but these are difficult to obtain for small wildlife [125]. To further close knowledge gaps, the constraint of ecological surveys between grain and extent must be further resolved. This requires automatic, cost-effective and data-efficient (i.e. triggered) observation systems that provide comprehensive sensor combinations and enable automatic observation at the individual level.

Video recordings are often used to observe the behaviour of individuals [27], but for small species camera traps are effective only over short distances [194]. The observation of bats is particularly difficult due to their nocturnal lifestyle in often richly structured habitats. Instead, recordings of echolocation calls are frequently used to monitor the presence/absence of bats [163]. These acoustic signals, with their comparatively long range [54], can serve as triggers for visual sensors. The combination of audio and video can additionally support the interpretation of the data [24].

However, recognizing individuals on images, particularly small and nocturnal species or species that lack unique visually detectable features, is challenging [198], as is the recognition of individuals based on acoustic recordings [225]. By contrast, the automatic tracking of bats using lightweight VHF radio transmitters offers several advantages [80, 124, 231]: (a) VHF signals can be used as triggers for other sensors and (b) they may support the recognition of individuals in video sequences based on comparisons of the VHF signal patterns with the movements observed in the video.

To combine the desirable features of audio and video monitoring, BatRack was developed, a modular observation system that integrates audio, video and automatic VHF radio tracking in a single unit. The three recording technologies can be used separately, simultaneously or in mutual trigger mode, and the corresponding configuration scheduled and switched automatically. BatRack's hardware is assembled from off-the-shelf components and its design and the required software have been published under a GNU GPL 3.0 license.

In the following, BatRack's hardware and software is presented, the suitability of its audio and VHF sensors in triggering the camera is evaluated, and the potential of VHF recordings in the identification of individuals in videos, in a case study of the dawn-swarming behaviour [135] of Bechstein's bat Myotis bechsteinii is tested. To date, only a few studies have examined this behaviour in detail [170]. Using BatRack's combined sensor approach, new insights based on individual-related information about the reproductive state and an individual's decision to change roost sites during the night can be provided.

Parts of this section have been published in Jannis Gottwald[28], Patrick Lampe[2], Jonas Höchst, Nicolas Friess, Julia Maier, Lea Leister, Betty Neumann, Tobias Richter, Bernd Freisleben, and

[28] shared first authorship

Thomas Nauss. “BatRack: An Open-Source Multi-Sensor Device for Wildlife Research.” In: *Methods in Ecology and Evolution* (July 2021), pp. 1867–1874.

5.5.2 Materials and Methods

The BatRack System

BatRack combines a core computation component with three sensor units (audio, video and VHF) and tailored analysis modules. Scientists and practitioners can easily assemble, configure and extend the system. Moreover, BatRack is inexpensive (650€ without a power supply), easily repaired using commodity off-the-shelf (COTS) components (Figure 5.27), and uses free and open source software (FOSS). In addition, it is configurable with respect to the attached sensors as well as their recording ranges, time-based scheduling and mutual trigger mode. A detailed description of the hardware and software modules, including product specifications and blueprints, can be found at the BatRack webpage `https://nature40.github.io/BatRack/`.

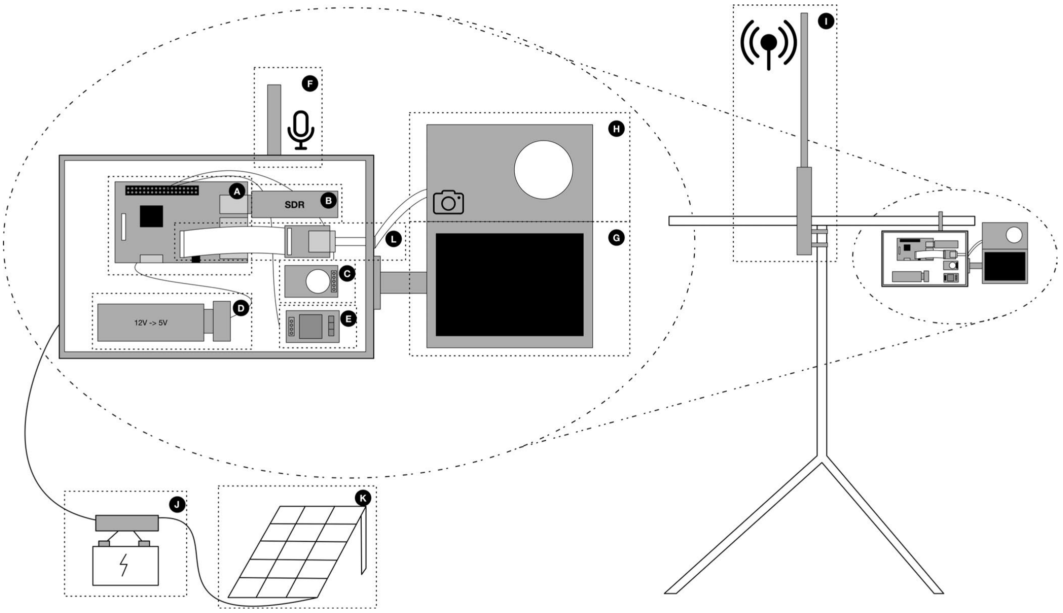


Figure 5.27: Hardware components of BatRack: (a) Raspberry pi mini computer, (b) rtl-sdr dongle, (c) real time clock or LTE stick (d) 12V to 5V converter with USB power supply, (e) KY-019 relay, (f) ultrasonic microphone, (g) IR spotlight, (h) Raspberry pi camera (NoIR or HQ-camera with removed IR filter), (i) omnidirectional antenna, (j) 12 V battery, (k) solar panel.

For easy deployment, the software comes as a customized Raspberry Pi OS image bundle called BatRackOS `https://github.com/Nature40/BatRackOS/releases/`, which was built using PIMOD [105].

The audio module (Figure 5.28a) is implemented using *pyaudio* for audio data retrieval and *numpy* for further audio processing and bat call detection. The sampling rate depends on the hardware (e.g. 384 kHz for Dodotronic Ultramic 384k). The camera analysis module (Figure 5.28b) uses the RPi Camera Web Interface software `https://elinux.org/RPi-Cam-Web-Interface`, which allows fast shutter speeds, concurrent camera access and automated exposure settings. Camera recordings are obtained in single image or continuous mode (max 90 frames/s) depending on user-defined settings.

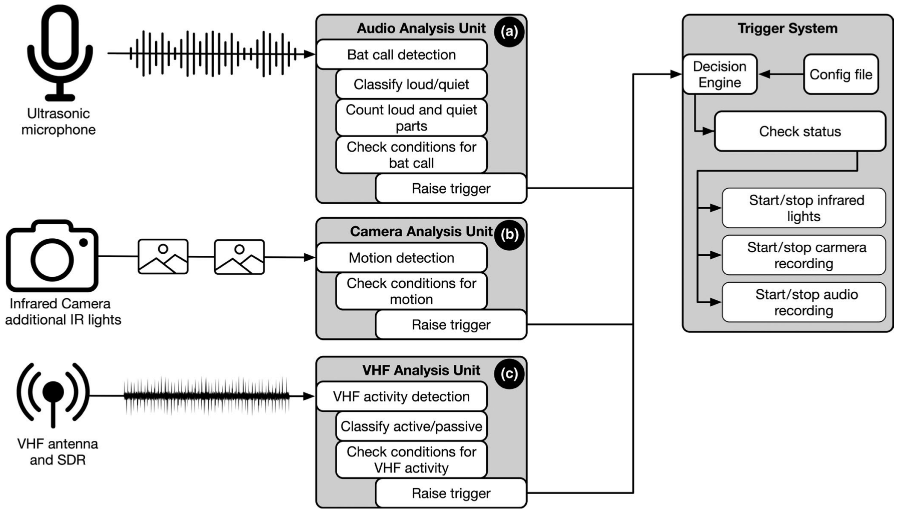


Figure 5.28: Analysis units of BatRack: (a) audio analysis unit (AAU), (b) camera analysis unit (CAU), (c) VHF analysis unit (VAU).

The VHF analysis module (Figure 5.28c) uses the signal detection algorithm described by Gottwald et al. [80]. When a new signal is received, its strength and duration are evaluated such that remote and noisy signals are filtered out. All other signals are classified as active (i.e. flying) or inactive (i.e. resting; [124]). A bat is inactive if a standard deviation in signal strength <2 is detected over at least 30 s, and active otherwise (Figure 5.29). If the VHF signals are used to trigger audio or video recordings, only time periods with active signals are written to memory, thus saving storage space and reducing the number of recordings that must be analysed. The operational modes of the analysis modules can be scheduled and configured individually.

Audio-triggered Video Recordings

The suitability of passive ultrasonic audio observations for triggering video recordings of bats was tested by placing BatRack in front of a known roost of Bechstein's bats Myotis bechsteinii for one night. The use of highly sensitive settings (10 dB, 15 kHz) resulted in the triggering of video and audio recordings for 2 s approximately every 10 s. Audio recordings were visualized using BatScope [178] and classified as Bechstein's bats, other bats or no bats. Video sequences were manually screened for bats, and the ratio of simultaneous audio and video detections of bats served as the test variable.

To optimize the trigger parameters for the detection of Bechstein's bats, the recorded audio files were postprocessed using the trigger algorithm of the audio analysis unit. All possible

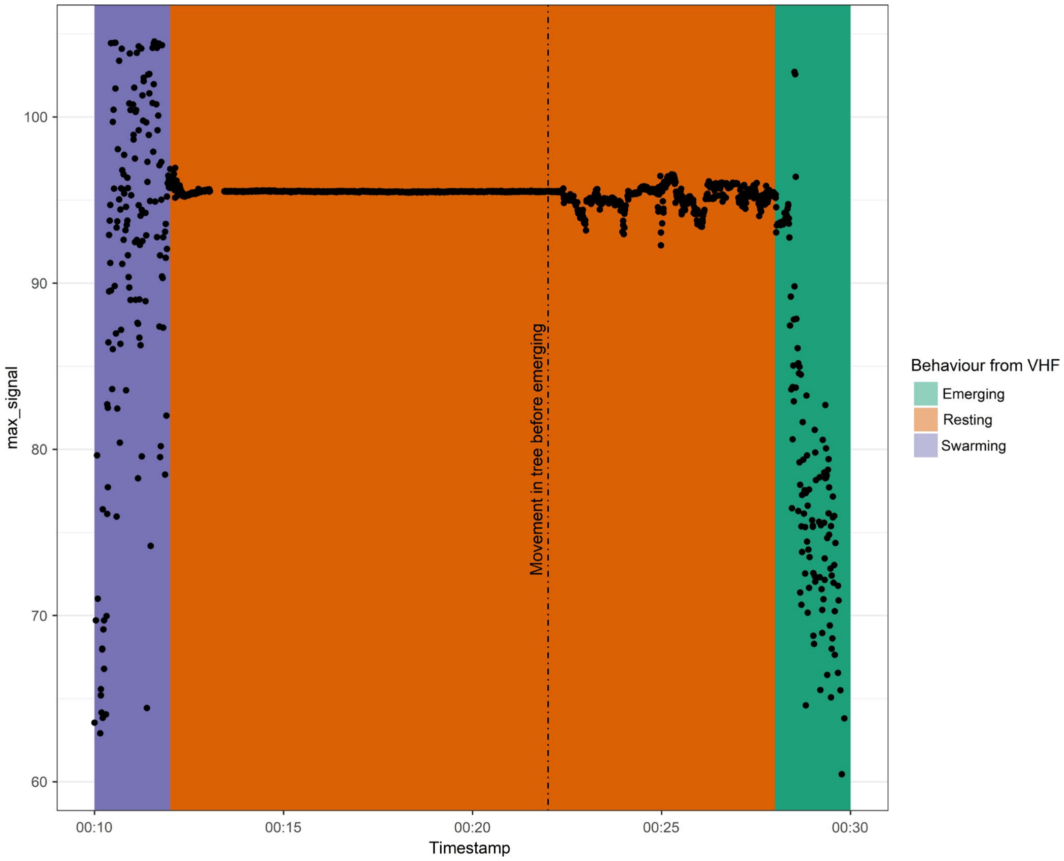


Figure 5.29: VHF signal patterns in relation to the different modes of behaviour. Swarming (purple), passive (orange), emerging from roost (green).

combinations in the range of 15–50 kHz for the frequency threshold and 15–50 dB for the sound-pressure threshold were tested. All audio recordings classified as Bechstein's bat calls were treated as true positives; all other bat calls were excluded from the training dataset. The maximum F1 score, which is the harmonic mean of precision and recall, was used to select the parameter combination that best minimized false positives while correctly identifying most true positives.

VHF-triggered Video Recordings and Individual-related Behavioural Patterns

To test the suitability of the VHF recordings in camera control and of the VHF signal strength in inferring behaviour, three pairs of Bechstein's bats from the same maternity colony were captured with mist nets between June and July 2020 and fitted with VHF tags. All females, except individual h172498, were in the expected reproductive state at the time of capture (Table 5.12). To monitor the bats, three BatRacks were placed in front of known roosting trees at a distance of 5–15 m for a total of 30 nights between June and August 2020.

To determine the suitability of VHF receptions in triggering video recordings of tagged individuals, the ratio of expected captures based on VHF patterns to manually screened, actual video

ID	Reproductive state	Pair	Capture date	VHF frequency
h146480	Pregnant	Pair1	08.06.2020	150.187
h172494	Pregnant	Pair1	08.06.2020	150.128
h172498	Not reproducing	Pair2	23.06.2020	150.172
h146482	Lactating	Pair2	18.06.2020	150.199
h146486	Postlactating	Pair3	15.07.2020	150.199
h146488	Postlactating	Pair3	15.07.2020	150.156

Table 5.12: Studied female Bechstein's bat individuals and their pair assignments

captures of at least one visible bat was used as the test variable. To investigate the potential of individual measurements to infer behavioural patterns, in this case swarming and emerging, the VHF data were analysed manually. Swarming was defined as both a signal pattern indicating an active bat and a signal strength above a threshold of -20 dBW for at least 30 s (Figure 5.29, purple). This corresponded to the continuously flying of a tagged individual in close proximity to the sensor unit. Emerging was defined as a resting phase immediately followed by a strong signal, which in turn dropped off very quickly and did not stabilize immediately (Figure 5.29, green).

The observed behaviour of the bats was manually labelled as swarming if the recorded video sequences revealed an individual that moved back and forth in the area of the roost, if the individual briefly approached the tree, or if it left the roost after a short entry. Exits that were not directly followed by re-entry or swarming were classified as emerging.

To determine whether the video-captured bats could be identified as the tagged individuals, the VHF signal patterns and the corresponding movement patterns in the video were compared. Exemplary VHF data and video frames are shown in Figure 5.30. The full video sequence and animated VHF data are provided at the BatRack webpage.

Unless otherwise stated, all analyses were performed using R [193].

5.5.3 Results

Audio Trigger Performance

During the test night, 3,317 audio-triggered audio and video recordings with an average length of 2 s were collected. Bats could be manually identified on 170 video (5.1%) and 663 audio (20%) recordings. From the latter, 272 recordings (41%) most likely originated from Bechstein's bats. For 166 of the 170 (97.6%) videos, a bat call was recorded simultaneously; in 160 cases (94.1%), the call was classified as that of a Bechstein's bat.

After all files with calls that could not be assigned to Bechstein's bats were removed and files without bat calls retained as true negatives, the remaining 2,929 files were processed to determine the optimal trigger parameters. The optimal combination of frequency and volume

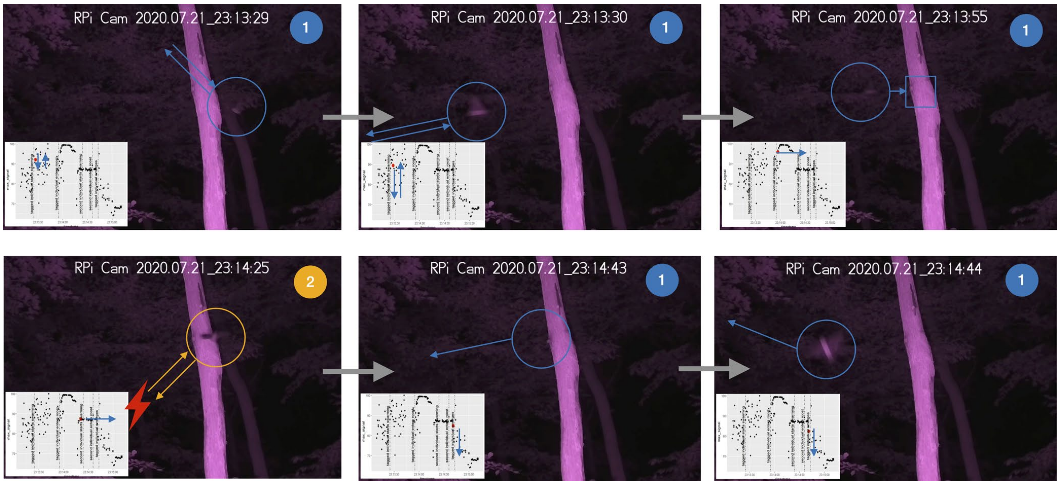

RPi Cam 2020.07.21_23:13:29
1
RPi Cam 2020.07.21_23:13:30
1
RPi Cam 2020.07.21_23:13:55
1
RPi Cam 2020.07.21_23:14:25
2
RPi Cam 2020.07.21_23:14:43
1
RPi Cam 2020.07.21_23:14:44
1

Figure 5.30: Identification of a tagged individual. The VHF signal shows strong fluctuations during swarming (up left and mid). The signal fluctuations decrease significantly after the bat enters the tree (up right, down left). Shortly after a second individual enters the tree (down mid), the tagged bat emerges from the tree (down right).

threshold based on the maximum F1 score (0.91) was 15 dB and 38 kHz. With this combination, 235 out of 274 files containing Bechstein's bat calls (sensitivity = 0.858) were correctly identified; only 5 out of 2,655 negatives were falsely identified as positives (specificity = 0.998).

VHF Trigger Performance

In total, 205 video recordings were captured that matched the VHF sequences classified as swarming or emerging. Of these, 130 (63%) were considered to show swarming and 75 (37%) emerging. Manual screening of the footage revealed one or more bats on 170 of the 205 (83%) sequences. Swarming was successfully detected in 93% of the sequences in which swarming was expected (122 out of 130), and emerging in 65% of the sequences (49 out of 75).

From the 170 (91%) video detections of a bat, in 155 the bat could be identified as the tagged individual with a very high probability, based on comparison of the movement pattern with the VHF signal strength. Among the 49 emerging and 130 swarming events, this was the case for 47 (96%) and 108 (83%), respectively.

Individual Behaviour Patterns

Pregnant and postlactating bats (Figure 5.31; pairs one and three) did not show any apparent differences in their swarming and resting patterns, either between individuals of a pair or between pairs. All four individuals showed a higher swarming frequency on the night of the roost change and on the following night. During the latter, repeated swarming sequences and resting phases at the abandoned tree occurred. The lactating female (Figure 5.31; pair two, h146482), showed a higher frequency of swarming behaviour and resting periods than observed in the nonreproducing female (Figure 5.31; pair two).

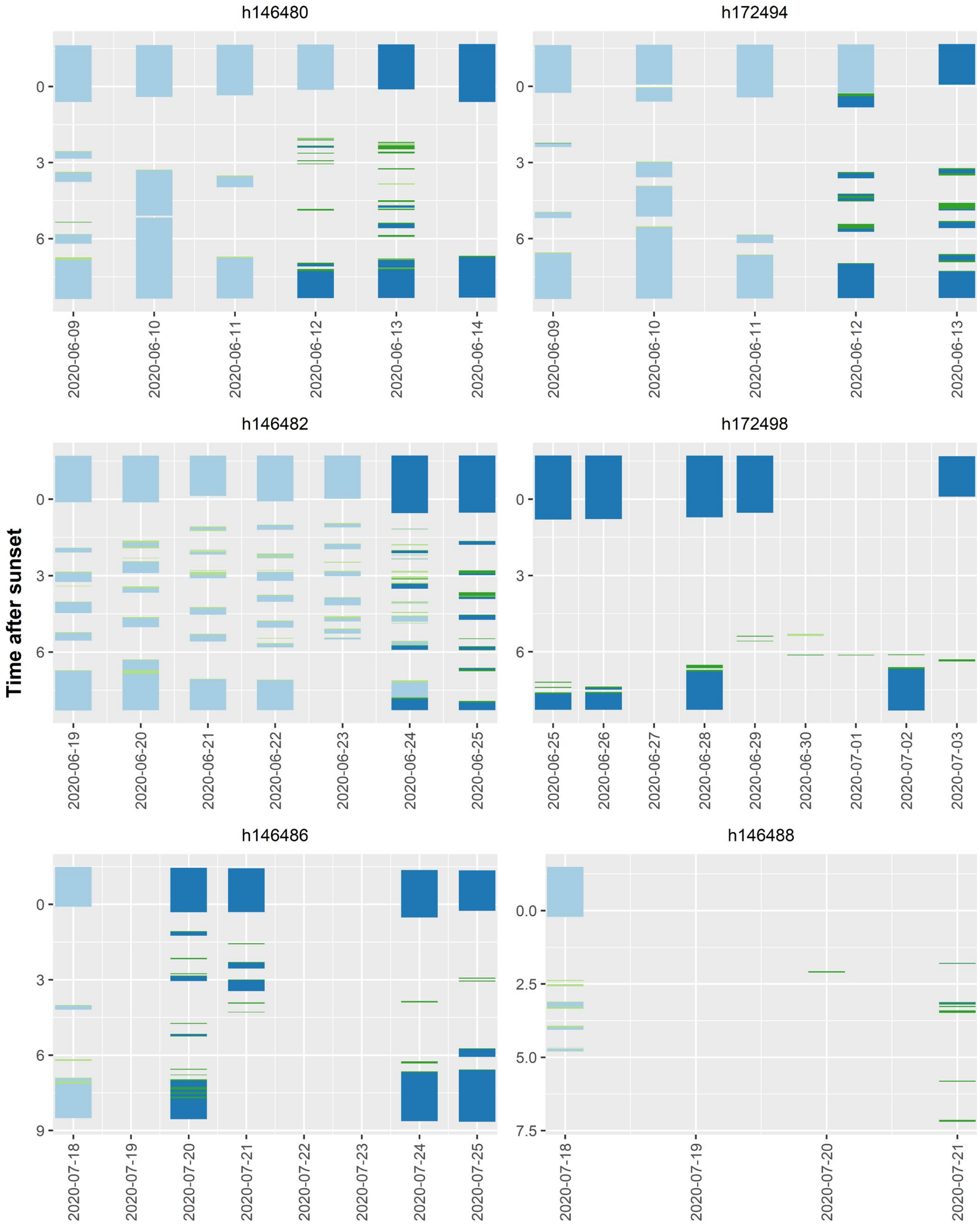


Figure 5.31: VHF-signal-derived behavioural patterns of Bechstein's bat pairs. Blue = inactivity, green = swarming. Gradations in the respective colour scale indicate the roost used for resting (Tree A = lighter blue, Tree B = darker blue) or for swarming (Tree A = lighter green, Tree B = darker green).

5.5.4 Discussion

A prerequisite for understanding dynamic natural environments as socio-ecological systems is data collected using highly automated monitoring systems. Recent developments have shown that the integration of (multiple) sensors and technologies in data acquisition and analysis can provide deep insights into the ecology of different species [85, 195, 207, 239].

Our case study on the behaviour patterns of Bechstein's bats illustrates the potential of BatRack. The observations provide first anecdotal indications of an association of increased swarming activity with a change of roost (pair one, three) and weening of the pup (pair two). However, the advantage of information acquired at the individual level comes at the price of having to tag the animals. Furthermore, the identification of an individual is more difficult if several tagged individuals with similar levels of activity are recorded simultaneously.

The application possibilities of BatRack are manifold. Observations that were previously only possible in the laboratory can be obtained in natural habitats. BatRack is best suited for studies where the observation of bats is linked to a specific and small area (e.g. hibernation roosts, maternity colonies, specific resource occurrences). The focus here is on the study of social behaviour between animals or the use of resources. The mobility of BatRack also makes it possible to complement laboratory studies with field experiments (e.g. changes in resource availability). Moreover, while behavioural contexts can often be inferred from vocalizations, reference recordings are missing for many species [234]. This deficiency can be addressed by BatRack, which can be used to collect visual ground truth of the behavioural significance of vocalizations. Thus, BatRack is a promising building block to close knowledge gaps regarding bat behaviour and to develop and evaluate conservation measures.

5.6 Summary

This chapter presented work on bat monitoring. In particular, the following contributions were presented:

Bat Call Recognition

We presented a novel self-attention approach for detecting bat echolocation calls and recognizing bat species in audio spectrograms. It is based on pre-trained data-efficient image transformer models used as components in a workflow that we developed to process audio spectrograms of recorded bat echolocation calls. We showed that it outperforms state-of-the-art CNN-based approaches for bat call detection as well as for bat species recognition on several publicly available data sets, yielding up to 90.2% average precision for detection and up to 88.7% accuracy for recognition, respectively. Furthermore, we demonstrated that the training and test set distribution are essential for determining recognition accuracy.

Bat@Edge

We presented a system design based on *Bird@Edge*, which is described in Section 4.5. The new hardware had to find its way into the edge computing system to overcome the challenges of higher sampling rates. Furthermore, the streaming approach of the microphone node is changed to a file-based approach. Here, a more powerful microphone node can analyze live audio data and only store and transmit relevant parts to the *Bat@Edge Station*. This approach enables live evaluation of multiple *Bat@Edge Mics* at a single *Bat@Edge Station*.

tRackIT OS

We presented *tRackIT OS*, open-source software for reliable VHF radio tracking of small animals in their wildlife habitat. *tRackIT OS* is an operating system distribution for *tRackIT stations* that receive signals emitted by VHF tags mounted on animals. *tRackIT OS* encompasses components for VHF signal processing, system monitoring, configuration management, and user access. We evaluated and compared *tRackIT OS* against a previous operating system distribution (called *paur*), in an experimental field evaluation carried out in the *Marburg Open Forest*. Our experimental results showed that compared to *paur*, *tRackIT OS* (a) enables reliable VHF signal detection for bearing calculation, (b) increases the number of usable signals by 103.3%, (c) improves the mean bearing calculation error from 38.9° to 23.7°, and (d) introduces only a slight overhead in power consumption of 2.55% or 0.2 W. *tRackIT OS* has the potential to substantially improve the quality of habitat usage studies and/or environmental assessments in the context of anthropogenic interventions in the environment while massively reducing the time required for fieldwork.

Activity State Classification

We presented a method to classify the activity state of an animal equipped with a VHF transmitter. This approach is written in R and published as open-source software. In the ecological case study, we demonstrated that our approach enables the detection of even subtle differences in the timing of activity according to a species' ecological preferences (differences in activity onset of < 20 min). Specifically, we were able to show distinct activity patterns for these two species, characterized by a slight shift in their timing of activity and significantly lower activity of N.leisleri during the night. Given that these species have evolved to occupy different ecological niches, these patterns are much more likely due to synchronisation of activity peaks with prey abundance rather than to an avoidance of competition [200]. Nyctalus leisleri, like other aerial hawking bats, has likely evolved to exploit insect emergence at dusk and dawn, thus avoiding the greater predation risk that may occur at higher light levels [202]. By contrast, M. bechsteinii and other gleaning bats are less constrained to flying insects as a food source such that an onset of activity comparable to that of N. leisleri would not bring substantial additional benefit.

Our findings are generally in line with previous observations of the activity patterns of N. leisleri [200, 214]. No comparable studies exist for M. bechsteinii, but in acoustic studies with results reported at the genus level all-night activity was determined for Myotis [185]. However, our study is the first to investigate the overlap of these two species within the same study area. Our approach also allows to detect changes in activity probability according to the reproductive status of individuals and indicates that these shifts are species specific.

The scientific insights that can be expected from automatic radio-tracking-based activity studies have the potential to deepen our understanding of the ecology and behaviour of small animal species in unprecedented ways [173]. With the recent advances in open-source automatic radio-tracking [80, 104, 231] together with the trained models and data-processing functionalities of the tRackIT R-package, the scientific community is now equipped with an accessible toolset that allows the activity patterns of small animals to be analysed and classified at high temporal resolution.

BatRack

BatRack offers a highly promising solution for the observation of bats. With its modular COTS design, *BatRack* can be readily built and easily maintained, allows individual configurations and extensions, and enables both flexible scheduling and the combination of measurements. *BatRack* yields reliable occurrence information based on audio or VHF recordings. The latter can be used in the retrieval of basic behavioral information even from a single, non-directional VHF receiver. Especially for small bat species, the use of either VHF or audio to trigger a video unit results in more energy- and storage-efficient video capture than allowed by purely schedule-based recording. A high detection probability and a substantial reduction in false positives are ensured by applying targeted trigger parameters to the audio unit. Triggering based on the VHF signal results in an even better performance with bats captured almost all the time when one of the tagged bats triggered the recording. In our study, valuable video recordings were obtained even for bats flying up to 15 m away from the sensor.

6 Software Design of a Bat Monitoring System

In this chapter, the software design of a bat monitoring system with the integration of the projects of Chapter 4 and Chapter 5 is presented. First, a design principles for a concrete system are presented, and after that, the design of the system is presented.

First of all, scalability is essential to enable a comprehensive coverage. For this, the components used (hardware and software) must be available, and the costs to be incurred must be as low as possible. In addition, the system should be easy to build, configure, and operate to make nationwide deployment possible in the first place. Another critical point is the accessibility in the field, which enables the sensor nodes to send status updates and to intervene in a short time in case of malfunctions. This ability to communicate is the only way to ensure permanent and closely timed monitoring, since it is possible to react and correct the error or problem in the event of a malfunction or failure. To guarantee a fast insight into the recorded data, automatic processing of the monitoring data is necessary. This dependency can also have the advantage that pre-processing can already be implemented in the field, so only the results need to be transmitted. Pre-processing the data on the sensor node or an edge device would drastically reduce the transmitted data volume and should therefore be considered in the system design, if possible. Since pre-processing steps are often more costly and energy-intensive in the field, the researcher should not lock a system for ecological monitoring into these pre-processing steps in the field. In addition, there should be strict interfaces for the data and the pre-processing routines to remain interchangeable. The processing methods must also be usable in the backend to achieve the same knowledge if the data reaches the backend without further pre-processing.

6.1 Design Principles

The following design principles consists of six steps to build a networked wireless sensor system for bat monitoring.

6.1.1 Using COTS Components to Offer Scalability

To offer scalability, it is necessary to build new sensor nodes with available components. COTS components are, most of the time, affordable and available. Because of that, using COTS components is a solid ground for the sensor node design and the design of a scalable networked wireless sensor system. However, building a robust sensor node surviving and working in the field with no stable energy source and self-made protection against the weather is still challenging. The design principle is to use relevant COTS components for the desired use case, extract the requirements beforehand, and design the sensor system along the requirements.

6.1.2 Clever Measurements Using Domain Knowledge

Instead of measuring all the time in the same interval, the sensor node can change the interval length over time. The sensor node can do this without losing information in case the scheduling is done with the domain knowledge of the researcher. So, the data rates are decreased, and storing the data is not as space intensive as without a schedule. An additional strategy to save space and transmission, which the sensor node can use, is to observe the signals from a sensor and only capture data if the data is relevant. This design principle could lead to more energy consumption because the sensor that is observed is read out all the time, so the consideration is between, on the one hand, energy and, on the other hand, storage and bandwidth. This consideration is especially relevant for all-time running sensors, which are less energy-efficient. This approach could benefit multi-sensor systems if the triggering sensor consumes less energy than the other sensors and all other sensors are not read out until a trigger arises.

6.1.3 Data Reduction at the Sensor Node or at the Edge

The sensor node could also reduce the captured data by pre-processing the monitoring data. This reduction could be a simple compression of the captured data with known compression algorithms, and for better compression rates, multiple measurements can be compressed and sent together. Also, for ecological measurements like temperature or humidity data, it could be beneficial to only store every n-th value and for every other value the offset to this. In this case, the offset is, in most cases, smaller than the value itself and can be stored in a less storage-hungry data type, or in case the offset is zero no data has to be stored. For image or video data, post-processing on the device or the network edge saves most of the storage space if it is reasonable only to store the result from the post-processing. In the case of image or video processing by machine learning models, the learned labels have to be stored and they could require significantly less storage space than the original data.

Moreover, recording images or videos of people without permission to do so in public places (photo or video sensor nodes) could be legally challenging. Thus, detecting this data and deleting it could also be an improvement. The end device should reduce this type of data to delete faces at the nearest location to the collection point, in the best case directly while recording. For the other parts of data reduction, the execution of the models and algorithms could take place on the end device or edge. If not enough energy is produced in the field, the backend can also execute the models for the higher price of transmission bandwidth.

6.1.4 Robust and Timely Data Transmission with Low Overhead

Prompt data transmission is desirable to detect errors, discrepancies, and failures. For this, the data must be available in the form to be sent to the node with an uplink. To complete the sending process, the pre-processing must be completed, and the resulting data must be transmitted over the network. The time needed for pre-processing stands against prompt data transmission to the backend but can be, in some cases, compensated or partly compensated by the lower transmission time. This appraisal depends on the individual case and cannot be stated in general terms. Furthermore, the data should be transferred with as little overhead as necessary. Protocols that can be used for transmitting status information, such as MQTT, are particularly suitable for these requirements. With other widely used methods, such as REST, the overhead would be much higher, and MQTT also offers the advantage of push-based processing of the data in the backend as soon as new data reaches the backend.

6.1.5 Automatic Generation of Knowledge from Data

To generate knowledge of the data, metadata is necessary to interpret the arriving data. For example, the system must know at which location the sensor node collects data and with which hardware or hardware version this data was collected. Calibration data could also be a major point to store and use for the processing steps. For this purpose, it must be possible to record this metadata in a form that is easy to use. This metadata recording also includes further data on, e.g., bird or bat transmitter; here, it must be noted for evaluation at what time which transmitter with which frequency was stuck on which animal.

With all this metadata, the system can generate insights in the backend, and the system can merge data from different types of sensor nodes. This data fusion enables a broader picture of the situation and generates further insights. The incoming data must be considered as a time series data stream, and the results must be available again for further procedures.

Here, the temporal component is essential since the proposed system can sort all collected sensor data using a time stamp since the data become almost worthless without temporal and spatial information. This procedure also creates a general interface for the data so that other researchers can add further processing steps here. New processing steps can retrospectively process data again with these steps.

6.1.6 Visualization of Data and Knowledge

The last part of the system is to visualize the data and findings so that a researcher can quickly correct failures and recognize connections and correlations in the data. For this purpose, the system should be based on a time series database and a visualization engine to visualize the data as time series. Thus, the sensor system uses the time series database as a general interface for the data. Data storage, processing, and visualization should use the same interface and data definitions to ensure comparability.

In case of low bandwidth of the uplink node or insufficient data volume, visualization is also beneficial in case the health data of the sensor nodes are transmitted, and the raw data is stored on the system. In this case, a researcher can get insights into the sensor node by observing the health data and collecting data from the sensor node, if necessary. Most of the other benefits are lost in this case. Still, the scalability could be maintained because the data collection is the only manual part, and the processing steps are automatically done after the manual data collection. So, the system must be able to upload the collected data and initialize the processing routine.

6.2 Bat Monitoring System

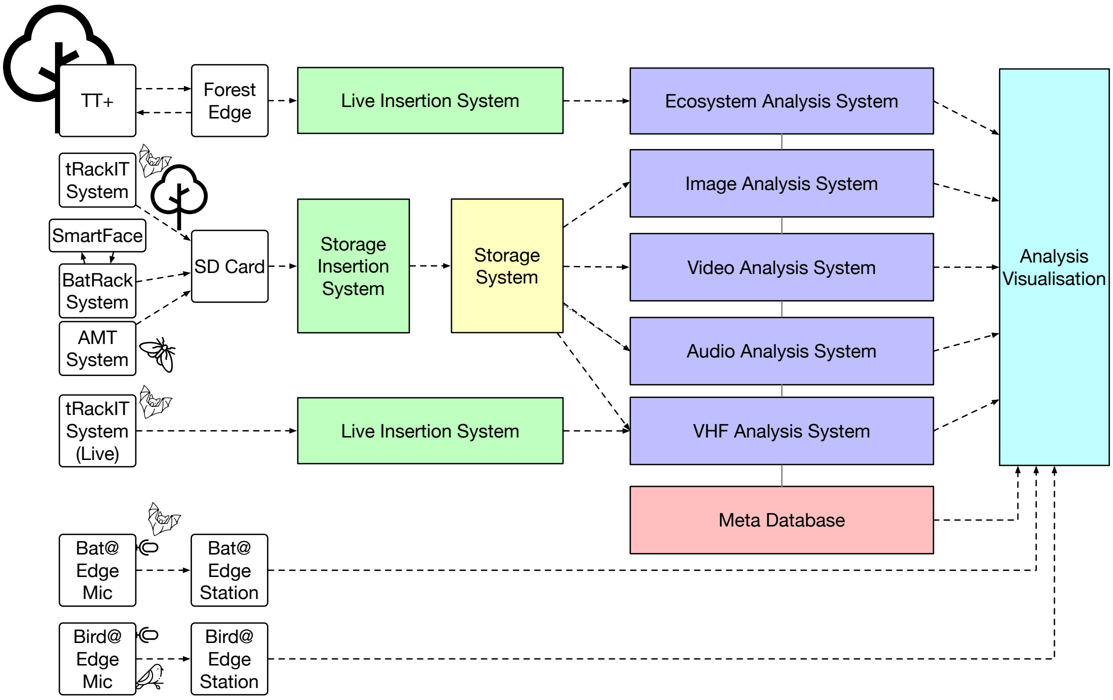


Figure 6.1: System design of a bat monitoring system

In addition to the publications presented in Chapter 4 and Chapter 5, a concrete implementation of a bat monitoring system must also build on several other components. Here, the automated processing and the triggering of the following process steps are essential. In addition, however, the obtained results must be stored in a meaningful way and made available for further processing steps. Furthermore, additional information about the technology used must be stored in a meta-database. And incoming data must be enriched with this metadata to ensure a correct processing result. In Figure 6.1, a software design with most of the presented components is shown. On the left side, a sensor node collects data and uses all the described design principles of Section 6.1. In the middle, the collected data is transmitted and stored; on the right side, the analysis components process the data to generate knowledge.

6.2.1 Database for Metadata of VHF Data

The VHF data is recorded by a so called *tRackIT* station, i.e., the hardware described by Gottwald et al. [80] with *tRackIT OS* as its operating system, as presented in Section 5.3. The database for metadata is used for later analysis. The data to be stored differs for each sensor node. Using the example of the *tRackIT* station, we explain which data must be recorded in order to be able to perform processing. To record data in the *tRackIT* station, bats must be equipped with VHF transmitters, and the metadata of these transmitters must also be recorded. These are:

- Date of transmitter attachment - Start date from when the transmitter was attached to the bat and the start date of processing.
- Expected end date of the transmitter - this date can be used to minimize errors and show the user that this transmitter is overdue for processing
- End date - the time when the transmitter can no longer be used for evaluation, either because it has been dropped or because it no longer has energy
- Frequency of the transmitter - to assign specific VHF signals to a unique animal, the incoming signals must be divided according to frequencies.
- Frequency Fluctuation - the expected fluctuation range of the received signals
- Minimum Duration - the minimum length of a signal; signals shorter than this are not recognized as signals by a transmitter.
- Maximum Duration - the maximum length of a signal analogous to the minimum
- Individual ID - the ID of the individual transmitted by the transmitter
- Components of temperature curve of transmitters - some transmitters are temperature dependent, and for calculation of temperature components for the equation are needed.

In addition to the data from the transmitters, data from bat individuals must be collected so that it is straightforward to evaluate which individuals were involved. The metadata stored includes:

- Species - the type of bat captured and transmitter used to make evaluations between species or within species.
- Size - the size of the individual must be able to be re-recorded for each capture to map a change in size over time
- Weight - the weight of the bat, which can also vary over time
- Sex - sex of the bat to record differences between sexes.
- Ring number - this is attached to recognize the bat when it is first captured and should, therefore, be recorded.
- Age - the age of the animal if known exactly, otherwise if sexed or not.

Furthermore, metadata about a *tRackIT* station must be recorded. Additional activities can be performed in this way, for example, calculating triangulation or filtering data. The metadata stored for a *tRackIT* station include:

- Lat - Latitude of the station and, in connection with the longitude, the station's location.
- Lon - Longitude of the station
- Orientation 1-4 - The measured deviation towards 0 degrees north for antennae 1, 2, 3, and 4

- Calibration 1-4 - The measured value for the calibration of antenna 1, 2, 3, and 4

With this information, further processing steps can be performed. To collect the data in a user-friendly way, a user interface for the meta-database is provided, which allows saving and changing the data. Researcher are given a central location where different people can collect, modify, and access the data.

6.2.2 Data Flow of tRackIT Data

To enrich the incoming VHF signal of a *tRackIT* station with metadata for analysis, the signals recorded by the *tRackIT* station are transmitted to the backend. In the backend, this data is matched with the meta-database in the data collector, see Figure 6.2, and assigned to the incoming signal to a transmitter and thus also to a bat individual. This matching makes it easy to filter the signals by an individual in all further processing steps and to view the time series data per individual. Triangulation can then be performed on this data, accessing the metadata of the stations, explicitly the location of the station and the orientation and calibration values of the antennas. The interaction provides a centralized way to calibrate the stations and let this process impact the results. The triangulations are written to the time series database for further processing and can be retrieved here for further processing or graphical processing in a defined form.

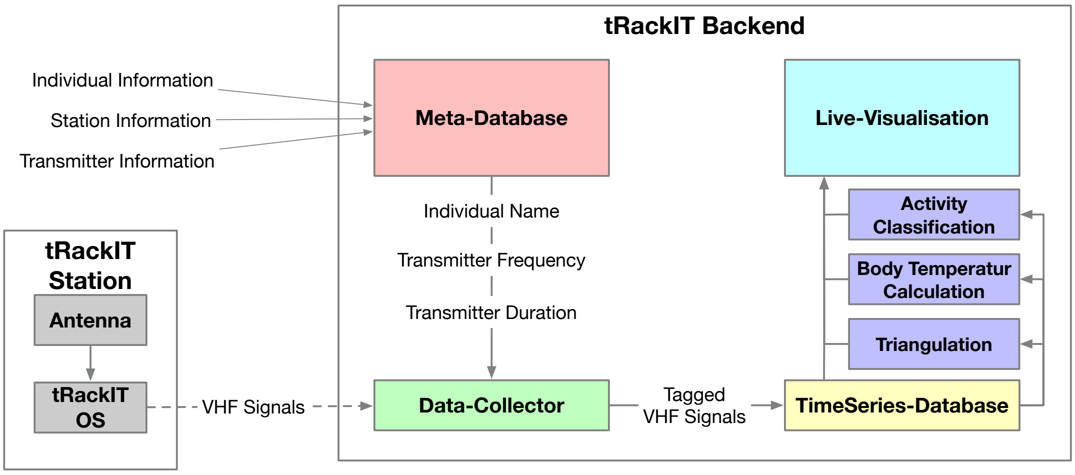


Figure 6.2: The data flow of VHF signals in combination with the metadata and data tagging

Furthermore, body temperatures for transmitters with variable transmission times are calculated on this time series data. Here, it is easy to say that if the bat's body temperature decreases, the distance between signals becomes longer. If the temperature increases, the distance between the two signals becomes shorter. These differences allow the body temperature to be calculated using a transmitter-specific curve, which is also obtained from the database for metadata. The temperatures are, in turn, written to the time series database, making them easy to find in a central location and available in a precise data format.

For activity classification, the signals are processed by the individual, and thus further knowledge is created here, which is written to the time series database after processing. Here, the apparent accessibility and interface are also considered advantageous and enable defined access.

The data stored in the time series database can thus be accessed in a defined manner, in a known format, and displayed in a live visualization. This visualization allows the user to view the data almost in real-time and to draw conclusions from the data on time. On the other hand, errors can be detected at an early stage and corrected in the field or remotely.

6.2.3 Data Flow of AMT / BatRack / tRackIT Data

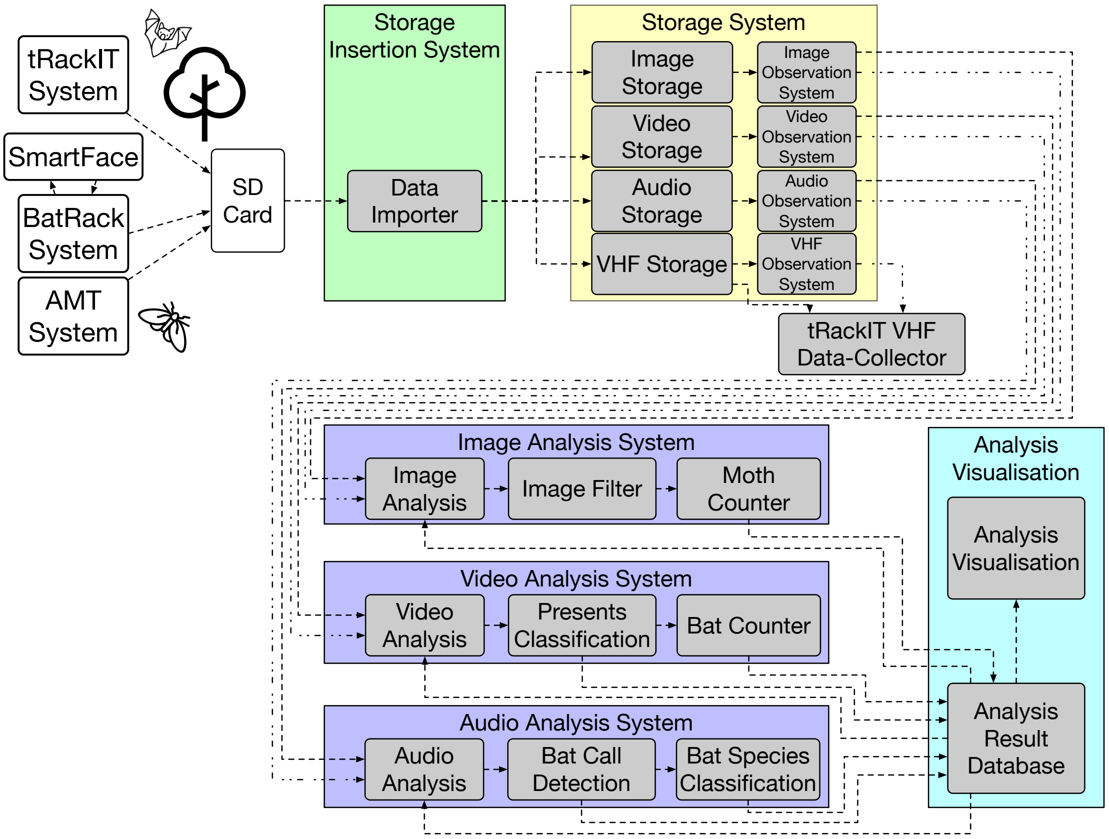


Figure 6.3: The data flow of the video, audio, image, and VHF data in combination with the data processing and result store. The tRackIT VHF Data Collector is the same as in Figure 6.2, and the data passes through this processing chain

In contrast to the live transmitted data of a tRackIT station, the sensor data is transmitted via SD card for fallback purposes or in case of a poor transmission rate. The state and remote access to the sensor nodes are unaffected and continue to be performed live and remotely. In this case, the data is read out manually and uploaded to an online service, e.g., via FTP. The data then reaches the storage insertion system, which imports the data. As shown in Figure 6.3, the storage insertion system is accessed by several monitoring systems that monitor the processing status and trigger processing when new data is received. Here, the data are made available to the processes in the network, and the individual processing steps can be divided on different machines. This availability in the network achieves data parallelism and, thus, scalability.

The uploaded VHF signals are forwarded to the processing machines as described in the previous chapter. In contrast, the image, audio, and video data are processed separately. The process is always divided into a pre-processing step in which relevant parts of the data are extracted and detailed processing of the areas detected in the previous step.

In the case of the image data, empty images of the AMT are first detected and sorted out. On the non-empty images, a count of the number of insects present is performed. This two-step process has the advantage that the detection of empty images is less energy intensive than counting moths. This two-step process results in a more efficient structure, similar to *SmartFace*. The collected results are added to a database and presented to the user.

For video data, the pre-processing in presence and absence is even more critical since the more considerable the data, the more computationally intensive the processing. Here, the processing is again triggered with new data, and a two-stage process is initiated. In the end, the results are also written in a defined form in a database and can be evaluated. Furthermore, the data is available for further processing and can be merged with other data.

The first step for the audio data on bat calls is to select the bat calls and mark them for further processing. Similar to *SmartFace*, the specific configuration of the call detection is essential for the data quality in the further processing steps. This is true for multi-step processing in general. Thus, the next step, species recognition, also depends on the quality of the delivered calls. Also, in this subarea, the results are fed back into the central result database, thus enabling access to the results.

6.2.4 Data Flow of ForestEdge

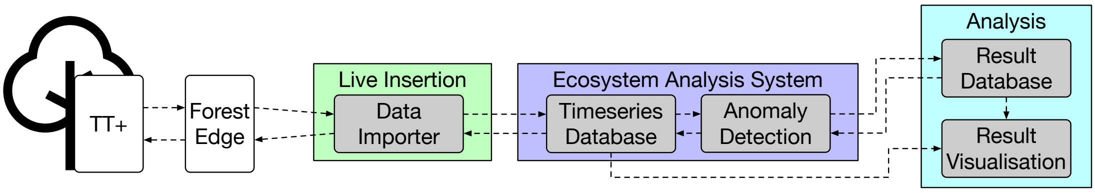


Figure 6.4: The data flow of the ForestEdge data with anomaly detection and live visualization

As described in Chapter 4.4, *ForestEdge* is deployed in the field and intercepts LoRa messages and answers messages from the *TT+*. After the data is captured and simple anomaly detection for this *TT+* is made on the device in the field, the data is transmitted to the backend. At this point, the data from all *TT+* is collected, and an additional anomaly detection for the hole network is done. With this second layer, research area-wide phenomena could be detected. Also, the visualization and the storing of the processed result are done in a defined way so that further processing steps can be applied in the future. With this concept, another sensor can be added in the same way, and the concept of *Unobtrusive Mechanism Interception* can also be applied.

6.2.5 Data Flow of Bat@Edge and Bird@Edge

The data flow for *Bat@Edge* and *Bird@Edge* differ from those described before. Here, data analysis takes place in the field on an edge computing node. With this, the only data which should be transmitted is the recognized labels of the machine learning model. This can be seen in the system design diagram 6.1.

6.3 Summary

This chapter presented six design principles that can help build a networked sensor system from scratch. After that, the data flow of a concrete implementation of such a networked sensor system was presented. This part highlights the processing steps as the metadata enriches the sensor data for further processing. In conclusion, a structure for a networked sensor system was presented and was filled with life to monitor the behavior of bats in their natural habitat. The used sensor nodes and parts of the processing were published before and presented in other chapters of this thesis.

7 Conclusion

An effective approach to fighting climate change requires extensive and reliable data. This work provides methods to acquire such data in a cost- and energy-efficient way, especially for analyzing bat behavior. For other species, the proposed approaches can be adapted. First, a summary of the given work is presented. Afterwards, we look at possible future research.

7.1 Summary

This thesis presented work on developing networked sensor systems for ecological monitoring. The particularly challenging environment of a forest was chosen. Problems arising in this environment were explained. The presented sensor nodes and analysis methods can solve a majority of these problems. In the following, the specific contributions of this thesis are presented.

In Chapter 4, novel sensor node designs were presented to monitor the ecosystem surrounding bats.

EcoSense is a system design built on static and moving nodes to monitor the environment. Here, the two node types have different requirements, and *EcoSense* gives hints for using separate sensor or computing units based on their abilities and power consumption.

An automated moth trap (*ATM*) was presented with an automatic camera and a light that takes photos of insects using a predefined schedule. With this data, we can measure how many insects are present, and we presented a comparative study between the ATM and a conventional catching method. We showed that trends are the same for the ATM and the conventional method. In addition, with this method, the ATM can also monitor trends in a single night without additional effort.

A novel approach and real-world use case was presented with *Unobtrusive Mechanism Interception* and *ForestEdge*. Here, we added features to a legacy device (*TreeTalker*) and converted the *TreeTalker* to a flexible and responsible sensor node. With this novel approach, many additional sensor nodes can be integrated, and researchers can add new features.

With *Bird@Edge*, an edge networking design was shown to process the recorded data at the edge of the network. This approach is suitable for live evaluation, error handling, and saving bandwidth in remote areas.

Finally, a new approach to filtering faces from images with *SmartFace* was presented, and this could be used to delete pictures with faces and detect human disturbances in the research area.

In Chapter 5, novel approaches were presented to get insights into a bat's life.

First, approaches to perform bat call recognition for European bats to tag recorded audio automatically were presented.

With the approach of *Bird@Edge* and a bat call recognition neural network, *Bat@Edge* was made possible. *Bat@Edge* supports the preprocessing of ultrasonic audio files in the field, which results in fewer transmissions.

We presented *tRackIT OS* as a solution to the scalable acquisition of a bat position and the low-maintenance provision of this service.

A classification method for VHF signals into active and passive animals, which is used to get an insight into activity graphs over the day and night of bats, was presented.

BatRack is a multisensor device that records audio, video, and VHF signals. With this device, it was possible to get matching data from the different sensors, so labeling VHF data with the correlating behavior was possible. Also, recording real-world audio with matching videos has been made possible by this device.

Finally, in Chapter 6, we presented six design principles that can help build a networked sensor system from scratch. Furthermore, the software design of a bat monitoring system with the integration of the projects of Chapter 4 and Chapter 5 was presented.

7.2 Future Work

There are several topics for future research.

EcoSense

Future possibilities of direct device-to-device communication via cheap LoRa add-ons for smartphones should be used more extensively. These add-ons open possibilities for new apps on mobile devices, such as infrastructure-less messaging or using a phone for sensing and remote control tasks. Furthermore, the usefulness of DTN software directly on the MCUs based on miniDTN should be explored. Finally, integrating energy harvesting solutions would also increase the system's flexibility when deployed in remote places.

AMT

Changes in species occurrences throughout the night can also be monitored. Additionally, the non-lethal nature of the system favors its use in long-term deployments in regions with species conservation restrictions [50]. Reducing the illumination time per hour would limit disturbance, in contrast to the light pollution resulting from the continuous operation of light traps, which might have adverse effects on moths [20, 45] and other animals and thus on their inventory. Different settings should be tested in further studies to find a reasonable compromise between a comprehensive assessment of night-flying insect fauna and keeping the light periods as short as possible. Due to their flexibility in scheduling, AMTs could also be attractive for studies on the dispersal of moths. In addition to moths, other light-attracted insects, such as adult caddisflies [226], could be monitored in a similar, non-lethal fashion using AMTs. Finally, the modularity of the AMT allows for its easy adjustment to specific needs, e.g., the use of brighter UV LEDs, depending on whether the aim is to compare demarcated locations or obtain qualitative assessments over a wider area.

Unobtrusive Mechanism Interception and ForestEdge

Our approach requires manual work for each application to identify the system, environment, mechanisms, and possible interceptors. Providing adequate tools to reduce manual work would help support our approach's adoption. Furthermore, apart from adding a single mechanism as part of an interceptor to enhance a single proprietary component, future work should consider supporting multiple mechanisms per system and systems per interceptor. Finally, when an interceptor replaces or modifies an existing mechanism or adds a new mechanism, the old mechanism is still available, although not used. Transitioning between multiple mechanisms could add the benefit of using the mechanism that achieves the best results for a given situation.

Bird@Edge

Self-supervised learning could leverage the vast amount of unlabeled data and improve the recognition quality of the target domain. Furthermore, continual and federated learning of machine learning models at the edge are interesting future research topics. Finally, several Bird@Edge deployments should be tested to identify potential problems in harsh environments.

SmartFace

There are several areas for future work, such as (a) exploring the use of visual concept detection algorithms as filters to determine whether humans are present in an image; the most computation-intensive parts of *SmartFace* can then be skipped if no persons are present, (b) improving the runtimes of *SmartFace* on mobile devices by utilizing multiple CPU cores or GPUs, and (c) applying *SmartFace* to non-scenario specific image sets (e.g., LFW or FDDB) for a general evaluation of our optimizations.

Bat Call Recognition

First, the detection performance could be improved by considering different call durations and types of calls, like social calls and feeding buzzes, and using a particular object detection network architecture. Second, the generalization capabilities of the trained neural network models should be further investigated by considering, for example, different hardware devices and recording environments. Finally, the classification of bat behavior and the recognition of individual bats based on different echolocation calls are exciting research directions in the future.

Bat@Edge

Bat@Edge could be beneficial for the execution time and the amount of data that can be processed during a day. Data with a low prediction probability could be cached and loaded into the backend for retraining to improve CNN's prediction. For this purpose, an uplink with sufficient bandwidth would have to be available.

tRackIT OS

Calculating exact bearings can be challenging since signals are affected by factors such as vegetation, the topology of the surrounding area, humidity, and rainfall. While bearings can be directly calculated based on a simple model, higher quality can be achieved by using data from multiple stations and further context information, such as a topology model and/or a calibration for the specific area of operation. Furthermore, transmitting all detected signals under the bandwidth limitations of the LoRa protocol is pretty challenging. A coordinated selection and transmission approach for detected signals should be developed to increase the efficiency of stations connected via LoRa. Finally, the continuous preparation and further processing of the collected data is the next primary task in creating a user-friendly and widely applicable animal tracking system for generating ecological knowledge.

Activity State Classification

The behavior of bats consists of more than active and passive phases. The active phases can be divided into more classes of activity, for example, hunting, traveling to hunting grounds or back to the roost, or deciding which roost is the one for the night (swarming). A method with more than two classes must be developed to detect all of these behaviors automatically.

BatRack

BatRack uses multiple sensors to detect, track and record the behavior of bats. With an additional thermal camera, the observations can be more precise, and this sensor could also be a good trigger for the recordings.

List of Figures

Chapter 5: Bat Monitoring 109

List of Tables

Bibliography

[1] Karl Aberer, Saket Sathe, Dipanjan Chakraborty, Alcherio Martinoli, Guillermo Barrenetxea, Boi Faltings, and Lothar Thiele. “OpenSense: Open Community Driven Sensing of the Environment.” In: *ACM SIGSPATIAL Int. Workshop on GeoStreaming*. ACM. 2010, pp. 39–42 (cit. on p. 36).

[2] T Mitchell Aide, Carlos Corrada-Bravo, Marconi Campos-Cerqueira, Carlos Milan, Giovany Vega, and Rafael Alvarez. “Real-time Bioacoustics Monitoring and Automated Species Identification.” In: *PeerJ* 1 (2013), e103 (cit. on p. 113).

[3] Michael Olusope Alade. “Investigation of the Effect of Ground and Air Temperature on Very High Frequency Radio Signals.” In: *Journal of Information Engineering and Applications* 3 (2013), pp. 16–21 (cit. on p. 144).

[4] Jonathan Allen. “Short Term Spectral Analysis, Synthesis, and Modification by Discrete Fourier Transform.” In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 25.3 (1977), pp. 235–238 (cit. on p. 134).

[5] Bastian Alt, Markus Weckesser, Christian Becker, Matthias Hollick, Sounak Kar, Anja Klein, Robin Klose, Roland Kluge, Heinz Koeppl, Boris Koldehofe, et al. “Transitions: A Protocol-independent View of the Future Internet.” In: *Proceedings of the IEEE* 107.4 (2019), pp. 835–846 (cit. on p. 71).

[6] Fernando Ascensão, Andreas Kindel, Fernanda Zimmermann Teixeira, Rafael Barrientos, Marcello D’Amico, Luís Borda-de-Água, and Henrique M Pereira. “Beware that the Lack of Wildlife Mortality Records can Mask a Serious Impact of Linear Infrastructures.” In: *Global Ecology and Conservation* 19 (2019), e00661 (cit. on p. 129).

[7] Helen Balinsky, David Subiros Perez, and Steven J. Simske. “System Call Interception Framework for Data Leak Prevention.” In: *IEEE 15th Int. Enterprise Distributed Object Computing Conference*. 2011, pp. 139–148 (cit. on p. 69).

[8] Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. “Fitting Linear Mixed-effects Models using lme4.” In: *arXiv preprint arXiv:1406.5823* (2014) (cit. on p. 60).

[9] Lars Baumgärtner, Paul Gardner-Stephen, Pablo Graubner, Jeremy Lakeman, Jonas Höchst, Patrick Lampe, Nils Schmidt, Stefan Schulz, Artur Sterz, and Bernd Freisleben. “An Experimental Evaluation of Delay-Tolerant Networking with Serval.” In: *IEEE Global Humanitarian Technology Conference (GHTC)*. Seattle, USA: IEEE, Oct. 2016, pp. 70–79 (cit. on p. 37).

[10] Lars Baumgärtner, Paul Gardner-Stephen, Pablo Graubner, Jeremy Lakeman, Jonas Höchst, Patrick Lampe, Nils Schmidt, Stefan Schulz, Artur Sterz, and Bernd Freisleben. “An Experimental Evaluation of Delay-tolerant Networking with Serval.” In: *IEEE Global Humanitarian Technology Conference (GHTC)*. Seattle, USA, Oct. 2016, pp. 70–79 (cit. on p. 6).

[11] Lars Baumgärtner, Stefan Kohlbrecher, Juliane Euler, Tobias Ritter, Milan Stute, Christian Meurisch, Max Mühlhauser, Matthias Hollick, Oskar von Stryk, and Bernd Freisleben. “Emergency Communication in Challenged Environments via Unmanned Ground and Aerial Vehicles.” In: *Global Humanitarian Technology Conference (GHTC).* IEEE. 2017, pp. 1–9 (cit. on p. 38).

[12] Lars Baumgärtner, Patrick Lampe, Jonas Höchst, Ragnar Mogk, Artur Sterz, Pascal Weisenburger, Mira Mezini, and Bernd Freisleben. “Smart Street Lights and Mobile Citizen Apps for Resilient Communication in a Digital City.” In: *IEEE Global Humanitarian Technology Conference (GHTC).* Seattle, USA, Oct. 2019, pp. 1–8 (cit. on p. 6).

[13] Lars Baumgärtner, Alvar Penning, Patrick Lampe, Björn Richerzhagen, Ralf Steinmetz, and Bernd Freisleben. “Environmental Monitoring using Low-Cost Hardware and Infrastructureless Wireless Communication.” In: *IEEE Global Humanitarian Technology Conference (GHTC).* IEEE. 2018, pp. 1–8 (cit. on pp. xiii, 6, 33, 34, 36).

[14] Andres Beck. “Fecal Analyses of European Bat Species.” In: *Myotis* 32.33 (1995), pp. 109–119 (cit. on p. 151).

[15] Hicham Bellafkir, Markus Vogelbacher, Jannis Gottwald, Markus Mühling, Nikolaus Korfhage, Patrick Lampe, Nicolas Frieß, Thomas Nauss, and Bernd Freisleben. “Bat Echolocation Call Detection and Species Recognition by Transformers with Self-attention.” In: *International Conference on Intelligent Systems and Pattern Recognition.* Springer. 2022, pp. 189–203 (cit. on pp. 4, 109, 112).

[16] Oded Berger-Tal and José J Lahoz-Monfort. “Conservation Technology: The Next Generation.” In: *Conservation Letters* 11.6 (2018), e12458 (cit. on p. 2).

[17] Kim Bjerge, Jakob Bonde Nielsen, Martin Videbæk Sepstrup, Flemming Helsing-Nielsen, and Toke Thomas Høye. “An Automated Light Trap to Monitor Moths (Lepidoptera) using Computer Vision-based Tracking and Deep Learning.” In: *Sensors* 21.2 (2021), p. 343 (cit. on pp. 22, 51, 53, 67).

[18] Fabian A Boetzl, Elena Ries, Gudrun Schneider, and Jochen Krauss. “It’s a Matter of Design — How Pitfall Trap Design Affects Trap Samples and Possible Predictions.” In: *PeerJ* 6 (2018), e5078 (cit. on p. 51).

[19] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. “P4: Programming Protocol-independent Packet Processors.” In: *ACM SIGCOMM Computer Communication Review* 44.3 (2014), pp. 87–95 (cit. on p. 69).

[20] Douglas H Boyes, Darren M Evans, Richard Fox, Mark S Parsons, and Michael JO Pocock. “Is Light Pollution Driving Moth Population Declines? A Review of Causal Mechanisms Across the Life Cycle.” In: *Insect Conservation and Diversity* 14.2 (2021), pp. 167–187 (cit. on p. 181).

[21] Gunnar Brehm and Jan C Axmacher. “A Comparison of Manual and Automatic Moth Sampling Methods (Lepidoptera: Arctiidae, Geometridae) in a Rain Forest in Costa Rica.” In: *Environmental Entomology* 35.3 (2006), pp. 757–764 (cit. on pp. 51, 64).

[22] Ryan C Burner, Tone Birkemoe, Jens Åström, and Anne Sverdrup-Thygeson. “Flattening the Curve: Approaching Complete Sampling for Diverse Beetle Communities.” In: *Insect Conservation and Diversity* 15.2 (2022), pp. 157–167 (cit. on p. 51).

[23] Ryan C Burner, Tone Birkemoe, Siri Lie Olsen, and Anne Sverdrup-Thygeson. "Sampling Beetle Communities: Trap Design Interacts with Weather and Species Traits to Bias Capture Rates." In: *Ecology and evolution* 10.24 (2020), pp. 14300–14308 (cit. on p. 51).

[24] Rachel T Buxton, Patrick E Lendrum, Kevin R Crooks, and George Wittemyer. "Pairing Camera Traps and Acoustic Recorders to Monitor the Ecological Impact of Human Disturbance." In: *Global Ecology and Conservation* 16 (2018), e00493 (cit. on p. 158).

[25] Francesca Cagnacci, Luigi Boitani, Roger A Powell, and Mark S Boyce. "Animal Ecology meets GPS-based Radiotelemetry: A Perfect Storm of Opportunities and Challenges." In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 365.1550 (2010), pp. 2157–2162 (cit. on p. 129).

[26] Abigail E Cahill, Matthew E Aiello-Lammens, M Caitlin Fisher-Reid, Xia Hua, Caitlin J Karanewsky, Hae Yeong Ryu, Gena C Sbeglia, Fabrizio Spagnolo, John B Waldron, Omar Warsi, et al. "How does Climate Change cause Extinction?" In: *Proceedings of the Royal Society B: Biological Sciences* 280.1750 (2013), p. 20121890 (cit. on pp. 1, 9, 12).

[27] Anthony Caravaggi, Peter B Banks, A Cole Burton, Caroline MV Finlay, Peter M Haswell, Matt W Hayward, Marcus J Rowcliffe, and Mike D Wood. "A Review of Camera Trapping for Conservation Behaviour Research." In: *Remote Sensing in Ecology and Conservation* 3.3 (2017), pp. 109–122 (cit. on p. 158).

[28] Nuria Castell, Mike Kobernus, Hai-Ying Liu, Philipp Schneider, William Lahoz, Arne J Berre, and Josef Noll. "Mobile Technologies and Services for Environmental Monitoring: The Citi-Sense-MOB Approach." In: *Urban Climate* 14 (2015), pp. 370–382 (cit. on p. 36).

[29] Qi Chang, Hui Qu, Pengxiang Wu, and Jingru Yi. "Fine-grained Butterfly and Moth Classification using Deep Convolutional Neural Networks." In: *Rutgers University: New Brunswick, NJ, USA* (2017) (cit. on p. 51).

[30] Xing Chen, Jun Zhao, Yan-hua Chen, Wei Zhou, and Alice C. Hughes. "Automatic Standardized Processing and Identification of Tropical Bat Calls using Deep Learning Approaches." In: *Biological Conservation* 241 (2020). ISSN: 0006-3207 (cit. on pp. 112, 113).

[31] Jordan Cheney, Ben Klein, Anil K Jain, and Brendan F Klare. "Unconstrained Face Detection: State of the Art Baseline and Challenges." In: *International Conference on Biometrics (ICB)*. IEEE. 2015, pp. 229–236 (cit. on pp. 96, 98, 100).

[32] H Chenji and R Stoleru. "Delay-tolerant Networks (DTNs) for Emergency Communications." In: *Advances in Delay-tolerant Networks (DTNs): Architecture and Enhanced Performance* (2014), p. 105 (cit. on p. 95).

[33] N Chinchor. *MUC-4 Evaluation Metrics in Proc. of the Fourth Message Understanding Conference 22–29*. 1992 (cit. on p. 150).

[34] Sylvain Christin, Éric Hervet, and Nicolas Lecomte. "Applications for Deep Learning in Ecology." In: *Methods in Ecology and Evolution* 10.10 (2019), pp. 1632–1644 (cit. on pp. 144, 145).

[35] Tim Clutton-Brock and Ben C Sheldon. "Individuals and Populations: The Role of Long-term, Individual-based Studies of Animals in Ecology and Evolutionary Biology." In: *Trends in ecology & evolution* 25.10 (2010), pp. 562–573 (cit. on p. 158).

[36] WW Cochran, DW Warner, JR Tester, and VB Kuechle. "Automatic Radio-tracking System for Monitoring Animal Movements." In: *BioScience* 15.2 (1965), pp. 98–100 (cit. on p. 143).

[37] Jeffrey P Cohn. "Tracking Wildlife: High-tech Devices help Biologists Trace the Movements of Animals Through Sky and Sea." In: *BioScience* 49.1 (1999), pp. 12–17 (cit. on p. 129).

[38] Rachael A Collett and Diana O Fisher. "Time-lapse Camera Trapping as an Alternative to Pitfall Trapping for Estimating Activity of Leaf Litter Arthropods." In: *Ecology and Evolution* 7.18 (2017), pp. 7527–7533 (cit. on p. 62).

[39] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. "On the Relationship between Self-attention and Convolutional Layers." In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia*. 2020 (cit. on p. 114).

[40] Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. "Meshed-Memory Transformer for Image Captioning." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 10575–10584 (cit. on p. 113).

[41] Margaret C Crofoot and Richard W Wrangham. "Intergroup Aggression in Primates and Humans: The Case for a Unified Theory." In: *Mind the gap*. Springer, 2010, pp. 171–195 (cit. on p. 143).

[42] Kevin Darras, Péter Batáry, Brett Furnas, Antonio Celis-Murillo, Steven L. Van Wilgenburg, Yeni A. Mulyani, and Teja Tscharntke. "Comparing the Sampling Performance of Sound Recorders Versus Point Counts in Bird Surveys: A Meta-analysis." In: *Journal of Applied Ecology* 55.6 (2018), pp. 2575–2586 (cit. on p. 82).

[43] Quentin De Coninck and Olivier Bonaventure. "Tuning Multipath TCP for Interactive Applications on Smartphones." In: *IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE. 2018, pp. 1–9 (cit. on p. 69).

[44] Quentin De Coninck, François Michel, Maxime Piraux, Florentin Rochet, Thomas Given-Wilson, Axel Legay, Olivier Pereira, and Olivier Bonaventure. "Pluginizing QUIC." In: *ACM Interest Group on Data Communication*. ACM. 2019, pp. 59–74 (cit. on p. 69).

[45] Tobias Degen, Oliver Mitesser, Elizabeth K Perkin, Nina-Sophie Weiß, Martin Oehlert, Emily Mattig, and Franz Hölker. "Street Lighting: Sex-independent Impacts on Moth Movement." In: *Journal of Animal Ecology* 85.5 (2016), pp. 1352–1360 (cit. on p. 181).

[46] Raphael K Didham, Yves Basset, C Matilda Collins, Simon R Leather, Nick A Littlewood, Myles HM Menz, Jörg Müller, Laurence Packer, Manu E Saunders, Karsten Schönrogge, et al. "Interpreting Insect Declines: Seven Challenges and a Way Forward." In: *Insect Conservation and Diversity* 13.2 (2020), pp. 103–114 (cit. on pp. 50, 51).

[47] Markus Dietz and Jacques B Pir. *Distribution, Ecology and Habitat Selection by Bechsteins Bat (Myotis bechsteinii) in Luxembourg*. Laurenti, 2011 (cit. on p. 151).

[48] Simone Disabato, Giuseppe Canonaco, Paul G Flikkema, Manuel Roveri, and Cesare Alippi. "Birdsong Detection at the Edge with Deep Learning." In: *IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE. 2021, pp. 9–16 (cit. on p. 83).

[49] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." In: *9th Int. Conference on Learning Representations, ICLR 2021, Austria*. 2021 (cit. on pp. 82, 111, 113, 114).

[50] Eleanor Drinkwater, Elva JH Robinson, and Adam G Hart. "Keeping Invertebrate Research Ethical in a Landscape of Shifting Public Opinion." In: *Methods in Ecology and Evolution* 10.8 (2019), pp. 1265–1273 (cit. on pp. 50, 181).

[51] Alexander Druffel and Kris Heid. "Davinci: Android App Analysis Beyond Frida via Dynamic System Call Instrumentation." In: *Int. Conference on Applied Cryptography and Network Security*. Springer. 2020, pp. 473–489 (cit. on p. 69).

[52] Paul Eggleton. "The State of the World's Insects." In: *Annu. Rev. Environ. Resour* 45 (2020), pp. 61–82 (cit. on p. 50).

[53] David Ely, Stefan Savage, and David Wetherall. "Alpine: A User-Level Infrastructure for Network Protocol Development." In: *3rd USENIX Symp. on Internet Technologies and Systems*. Vol. 3. 2001, pp. 15–23 (cit. on p. 69).

[54] Hiroto Enari, Haruka S Enari, Kei Okuda, Tetsuya Maruyama, and Kana N Okuda. "An Evaluation of the Efficiency of Passive Acoustic Monitoring in Detecting Deer and Primates in Comparison with Camera Traps." In: *Ecological Indicators* 98 (2019), pp. 753–762 (cit. on p. 158).

[55] Michael S Engel, Luis MP Cerıéaco, Gimo M Daniel, Pablo M Dellapé, Ivan Löbl, Milen Marinov, Roberto E Reis, Mark T Young, Alain Dubois, Ishan Agarwal, et al. *The Taxonomic Impediment: A Shortage of Taxonomists, not the Lack of Technical Approaches*. 2021 (cit. on p. 50).

[56] Stefano M Faccin, Carl Wijting, Jarkko Kenckt, and Ameya Damle. "Mesh WLAN Networks: Concept and System Design." In: *IEEE Wireless Communications* 13.2 (2006), pp. 10–17 (cit. on p. 15).

[57] TM Fayle, RUTH E Sharp, and MICHAEL EN Majerus. "The Effect of Moth Trap Type on Catch Size and Composition in British Lepidoptera." In: *British Journal of entomology and natural history* 20.4 (2007), p. 221 (cit. on p. 51).

[58] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. "Object Detection with Discriminatively Trained Part-based Models." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9 (2010), pp. 1627–1645 (cit. on p. 96).

[59] Hao Feng, Biao Wang, Chao Zhang, Bing Yu, Wonjun Hwang, Jae-Joon Han, Changkyu Choi, and Haitao Wang. "A fast Multi-view Face Detector for Mobile Phone." In: *IEEE Int. Conf. on Image Processing (ICIP)*. IEEE. 2016, pp. 3219–3223 (cit. on p. 96).

[60] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. "Do We need Hundreds of Classifiers to Solve Real World Classification Problems?" In: *The journal of machine learning research* 15.1 (2014), pp. 3133–3181 (cit. on pp. 145, 149).

[61] Ana Elisa Ferreira, Fernando M Ortiz, Luis Henrique MK Costa, Brandon Foubert, Ibrahim Amadou, and Nathalie Mitton. "A Study of the LoRa Signal Propagation in Forest, Urban, and Suburban Environments." In: *Annals of Telecommunications* 75.7 (2020), pp. 333–351 (cit. on p. 66).

[62] Bob Fischer and Brendon MH Larson. "Collecting Insects to Conserve them: A Call for Ethical Caution." In: *Insect Conservation and Diversity* 12.3 (2019), pp. 173–182 (cit. on p. 50).

[63] Auriel MV Fournier, Easton R White, and Stephen B Heard. "Site-selection Bias and Apparent Population Declines in Long-term Studies." In: *Conservation Biology* 33.6 (2019), pp. 1370–1379 (cit. on p. 50).

[64] R Fox, EB Dennis, CA Harrower, D Blumgart, JR Bell, P Cook, AM Davis, LJ Evans-Hill, F Haynes, D Hill, et al. "The State of Britain's Larger Moths 2021." In: *Butterfly Conservation, Rothamsted Research and UK Centre for Ecology …* (2021) (cit. on pp. 50, 61).

[65] Richard Fox. "The Decline of Moths in Great Britain: A Review of Possible Causes." In: *Insect Conservation and Diversity* 6.1 (2013), pp. 5–19 (cit. on p. 51).

[66] Juan A Fraire, Sandra Céspedes, and Nicola Accettura. "Direct-to-Satellite IoT-A Survey of the State of the Art and Future Research Perspectives." In: *International Conference on Ad-Hoc Networks and Wireless*. Springer. 2019, pp. 241–258 (cit. on p. 16).

[67] Robin Freeman, Todd Dennis, Todd Landers, David Thompson, Elizabeth Bell, Mike Walker, and Tim Guilford. "Black Petrels (Procellaria Parkinsoni) Patrol the ocean Shelf-break: GPS Tracking of a Vulnerable Procellariiform Seabird." In: *PLoS One* 5.2 (2010), e9236 (cit. on p. 143).

[68] Winifred F. Frick, Tigga Kingston, and Jon Flanders. "A Review of the Major Threats and Challenges to Global Bat Conservation." In: *Annals of the New York Academy of Sciences* 1469.1 (2020), pp. 5–25 (cit. on pp. 2, 111).

[69] Nicolas Friess, Jörg Bendix, Martin Brändle, Roland Brandl, Stephan Dahlke, Nina Farwig, Bernd Freisleben, Hajo Holzmann, Hanna Meyer, Thomas Müller, et al. "Introducing Nature 4.0: A sensor network for Environmental Monitoring in the Marburg Open Forest." In: *Biodiversity Information Science and Standards* 2 (2019) (cit. on p. 76).

[70] Norbert Fröschle. "Engineering of New Participation Instruments Exemplified by Electromobility, Particulate Matter and Clean Air Policy-Making." In: *HMD Praxis der Wirtschaftsinformatik* 54.4 (2017), pp. 502–517 (cit. on p. 36).

[71] Sarah Gallacher, Duncan Wilson, Alison Fairbrass, Daniyar Turmukhambetov, O Mac Aodha, Stefan Kreitmayer, M Firman, Gabriel Brostow, and Kate Jones. "Shazam for Bats: Internet of Things for Continuous Real-time Biodiversity Monitoring." In: *IET Smart Cities* (2021) (cit. on pp. 83, 122).

[72] Massimo Gallo and Rafael Laufer. "ClickNF: A Modular Stack for Custom Network Functions." In: *USENIX Annual Technical Conference (USENIX ATC 18)*. 2018, pp. 745–757 (cit. on p. 69).

[73] Paul Gardner-Stephen, Andrew Bettison, Romana Challans, and Jeremy Lakeman. "The Rational Behind The Serval Network Layer for Resilient Communications." In: *Journal of Computer Science* 9.12 (2013), p. 1680 (cit. on p. 95).

[74] Kevin J Gaston. "Global Patterns in Biodiversity." In: *Nature* 405.6783 (2000), pp. 220–227 (cit. on p. 64).

[75] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. "Audio Set: An Ontology and Human-labeled Dataset for Audio Events." In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 2017, pp. 776–780 (cit. on p. 114).

[76] M Gerboles, L Spinelle, A Kotsev, M Signorini, and L Srl. "AirSensEUR: An Open-Designed Multi-Sensor Platform for Air Quality Monitoring." In: *4th Scientific Meeting EuNetAir.* 2015, pp. 3–5 (cit. on p. 36).

[77] Zachariah J Gezon, Eli S Wyman, John S Ascher, David W Inouye, and Rebecca E Irwin. "The Effect of Repeated, Lethal Sampling on Wild Bee Abundance and Diversity." In: *Methods in Ecology and Evolution* 6.9 (2015), pp. 1044–1054 (cit. on pp. 12, 50).

[78] Yuan Gong, Yu-An Chung, and James R. Glass. "AST: Audio Spectrogram Transformer." In: *Interspeech 2021.* 2021, pp. 571–575 (cit. on pp. 82, 113–115).

[79] Jannis Gottwald, Raphael Royaute, Marcel Becker, Tobias Geitz, Jonas Hoechst, Patrick Lampe, Lea Leister, Kim Lindner, Julia Maier, Sascha Roesner, et al. "Classifying the Activity States of Small Vertebrates using Automated VHF telemetry." In: *submitted; under review* () (cit. on pp. 5, 109, 145).

[80] Jannis Gottwald, Ralf Zeidler, Nicolas Friess, Marvin Ludwig, Christoph Reudenbach, and Thomas Nauss. "Introduction of an Automatic and Open-source Radio-tracking System for Small Animals." In: *Methods in Ecology and Evolution* 10.12 (2019), pp. 2163–2172 (cit. on pp. 10, 21, 129, 131, 138, 141, 145–147, 156, 158, 160, 167, 172).

[81] Jannis Gottwald[1], Patrick Lampe[2], Jonas Höchst, Nicolas Friess, Julia Maier, Lea Leister, Betty Neumann, Tobias Richter, Bernd Freisleben, and Thomas Nauss. "BatRack: An Open-Source Multi-Sensor Device for Wildlife Research." In: *Methods in Ecology and Evolution* (July 2021), pp. 1867–1874 (cit. on pp. xiii, 5, 56, 66, 109, 145–147, 158).

[82] Pablo Graubner, Patrick Lampe, Jonas Höchst, Lars Baumgärtner, Mira Mezini, and Bernd Freisleben. "Opportunistic Named Functions in Disruption-tolerant Emergency Networks." In: *ACM International Conference on Computing Frontiers 2018 (ACM CF).* Ischia, Italy: ACM, May 2018, pp. 129–137 (cit. on p. 6).

[83] Pablo Graubner, Patrick Lampe, Jonas Höchst, Lars Baumgärtner, Mira Mezini, and Bernd Freisleben. "Opportunistic Named Functions in Disruption-tolerant Emergency Networks." In: *ACM Int. Conference on Computing Frontiers (ACM CF).* Ischia, Italy, May 2018, pp. 129–137 (cit. on p. 42).

[84] Baptiste Gregorutti, Bertrand Michel, and Philippe Saint-Pierre. "Correlation and Variable Importance in Random Forests." In: *Statistics and Computing* 27.3 (2017), pp. 659–678 (cit. on p. 149).

[85] Stefan Greif and Yossi Yovel. "Using on-board Sound Recordings to Infer Behaviour of Free-moving Wild Animals." In: *Journal of Experimental Biology* 222.Suppl_1 (2019), jeb184689 (cit. on p. 165).

[86] Adriano Guarnieri, Stefano Maini, Giovanni Molari, Valda Rondelli, et al. "Automatic Trap for Moth Detection in Integrated Pest Management." In: *Bulletin of Insectology* 64.2 (2011), pp. 247–251 (cit. on p. 51).

[1] shared first authorship

[87] Luis Guerra, Laura M McGarry, Victor Robles, Concha Bielza, Pedro Larranaga, and Rafael Yuste. “Comparison between Supervised and Unsupervised Classifications of Neuronal Cell Types: A Case Study.” In: *Developmental neurobiology* 71.1 (2011), pp. 71–82 (cit. on p. 17).

[88] Nathan R Hahn, Sara P Bombaci, and George Wittemyer. “Identifying Conservation Technology Needs, Barriers, and Opportunities.” In: *Scientific reports* 12.1 (2022), pp. 1–7 (cit. on pp. xiii, 2, 9, 51).

[89] Stefan Halle and Nils Christian Stenseth. *Activity Patterns in Small Mammals: An Ecological Approach; with 11 Tables.* Vol. 141. Springer Science & Business Media, 2000 (cit. on p. 143).

[90] Caspar A Hallmann, Martin Sorg, Eelke Jongejans, Henk Siepel, Nick Hofland, Heinz Schwan, Werner Stenmans, Andreas Müller, Hubert Sumser, Thomas Hörren, et al. “More than 75 percent Decline over 27 years in Total Flying Insect Biomass in Protected Areas.” In: *PloS one* 12.10 (2017), e0185809 (cit. on p. 50).

[91] Caspar A Hallmann, Theo Zeegers, Roel van Klink, Rikjan Vermeulen, Paul van Wielink, Henk Spijkers, Jurriën van Deijk, Wouter van Steenis, and Eelke Jongejans. “Declining Abundance of Beetles, Moths and Caddisflies in the Netherlands.” In: *Insect Conservation and Diversity* 13.2 (2020), pp. 127–139 (cit. on p. 50).

[92] Michael T Hallworth and Peter P Marra. “Miniaturized GPS Tags Identify Non-breeding Territories of a Small Breeding Migratory Songbird.” In: *Scientific reports* 5.1 (2015), pp. 1–6 (cit. on p. 143).

[93] Talisin T Hammond, Dwight Springthorpe, Rachel E Walsh, and Taylor Berg-Kirkpatrick. “Using Accelerometers to Remotely and Automatically Characterize Behavior in Small Animals.” In: *Journal of Experimental Biology* 219.11 (2016), pp. 1618–1624 (cit. on p. 143).

[94] Frank R Hampel. “The Influence Curve and its Role in Robust Estimation.” In: *Journal of the american statistical association* 69.346 (1974), pp. 383–393 (cit. on p. 148).

[95] Jeffrey A Harvey, Robin Heinen, Inge Armbrecht, Yves Basset, James H Baxter-Gilbert, T Martijn Bezemer, Monika Böhm, Riccardo Bommarco, Paulo AV Borges, Pedro Cardoso, et al. “International Scientists Formulate a Roadmap for Insect Conservation and Recovery.” In: *Nature Ecology & Evolution* 4.2 (2020), pp. 174–176 (cit. on p. 50).

[96] Axel Hausmann, Andreas H Segerer, Thomas Greifenstein, Johannes Knubben, Jerôme Morinière, Vedran Bozicevic, Dieter Doczkal, Armin Günter, Werner Ulrich, and Jan Christian Habel. “Toward a Standardized Quantitative and Qualitative Insect Monitoring Scheme.” In: *Ecology and evolution* 10.9 (2020), pp. 4009–4020 (cit. on pp. 50, 51, 64).

[97] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition.” In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2016 (cit. on pp. 82, 112, 113).

[98] Luke Hecht. *Methods for Studying Wild Animals’ Causes of Death.* Wild Animal Initiative, `https://www.wildanimalinitiative.org/blog/cause-of-death-2`. Accessed: 2020-11-02. Nov. 2020 (cit. on p. 131).

[99] Christof Henkel, Pascal Pfeiffer, and Philipp Singer. "Recognizing Bird Species in Diverse Soundscapes under Weak Supervision." In: *Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21-24, 2021.* Vol. 2936. CEUR Workshop Proceedings. CEUR-WS.org, 2021, pp. 1579–1586 (cit. on p. 82).

[100] Jens Heuschkel, Alexander Frömmgen, Jon Crowcroft, and Max Mühlhäuser. "Virtual-Stack: Adaptive Multipath Support Through Protocol Stack Virtualization." In: *10th Int. Network Conference (INC).* 2016, pp. 73–78 (cit. on p. 69).

[101] Andrew P. Hill, Peter Prince, Jake L. Snaddon, C. Patrick Doncaster, and Alex Rogers. "AudioMoth: A Low-cost Acoustic Device for Monitoring Biodiversity and the Environment." In: *HardwareX* 6 (2019), e00073. ISSN: 2468-0672 (cit. on p. 89).

[102] Jonas Höchst, Lars Baumgärtner, Franz Kuntke, Alvar Penning, Artur Sterz, and Bernd Freisleben. "LoRa-based Device-to-Device Smartphone Communication for Crisis Scenarios." In: *17th International Conference on Information Systems for Crisis Response and Management (ISCRAM).* Blacksburg, Virginia, USA, May 2020 (cit. on p. 137).

[103] Jonas Höchst, Hicham Bellafkir, Patrick Lampe, Markus Vogelbacher, Markus Mühling, Daniel Schneider, Kim Lindner, Sascha Rösner, Dana G. Schabo, Nina Farwig, and Bernd Freisleben. "Bird@Edge: Bird Species Recognition at the Edge." In: *International Conference on Networked Systems (NETYS).* Springer. May 2022, pp. 69–86 (cit. on pp. xiv, 4, 33, 34, 82).

[104] Jonas Höchst, Jannis Gottwald, Patrick Lampe, Julian Zobel, Thomas Nauss, Ralf Steinmetz, and Bernd Freisleben. "tRackIT OS: Open-Source Software for Reliable VHF Wildlife Tracking." In: *51. Jahrestagung der Gesellschaft für Informatik, Digitale Kulturen, INFORMATIK 2021, Berlin, Germany.* LNI. GI, Sept. 2021, pp. 425–442 (cit. on pp. xiii, xiv, 5, 109, 130, 144–147, 167).

[105] Jonas Höchst, Alvar Penning, Patrick Lampe, and Bernd Freisleben. "PIMOD: A Tool for Configuring Single-Board Computer Operating System Images." In: *IEEE Global Humanitarian Technology Conference (GHTC).* Seattle, USA, Oct. 2020, pp. 1–8 (cit. on pp. 5, 56, 86, 132, 159).

[106] Jonas Höchst, Artur Sterz, Alexander Frömmgen, Denny Stohr, Ralf Steinmetz, and Bernd Freisleben. "Learning Wi-Fi Connection Loss Predictions for Seamless Vertical Handovers Using Multipath TCP." In: *44th IEEE Int. Conference on Local Computer Networks (LCN).* Osnabrück, Germany, Oct. 2019, pp. 18–25 (cit. on p. 73).

[107] Laurens Hogeweg, Theo Zeegers, Ioannis Katramados, and Eelke Jongejans. "Smart Insect Cameras." In: *Biodiversity Information Science and Standards* 3 (2019), e39241 (cit. on p. 51).

[108] M Holyoak, V Jarosik, and I Novak. "Weather-induced Changes in Moth Activity Bias Measurement of Long-term Population Dynamics from Light Trap Samples." In: *Entomologia Experimentalis et Applicata* 83.3 (1997), pp. 329–335 (cit. on p. 51).

[109] Michio Honda, Felipe Huici, Costin Raiciu, Joao Araujo, and Luigi Rizzo. "Rekindling Network Protocol Innovation with User-level Stacks." In: *ACM SIGCOMM Comp. Comm. Review* 44.2 (2014), pp. 52–58 (cit. on p. 69).

[110] Torsten Hothorn, Jörg Müller, Leonhard Held, Lisa Möst, and Atle Mysterud. "Temporal Patterns of Deer–vehicle Collisions Consistent with Deer Activity Pattern and Density Increase but not General Accident Risk." In: *Accident Analysis & Prevention* 81 (2015), pp. 143–152 (cit. on p. 129).

[111] Toke T Høye, Johanna Ärje, Kim Bjerge, Oskar LP Hansen, Alexandros Iosifidis, Florian Leese, Hjalte MR Mann, Kristian Meissner, Claus Melvad, and Jenni Raitoharju. "Deep Learning and Computer Vision will Transform Entomology." In: *Proceedings of the National Academy of Sciences* 118.2 (2021), e2002545117 (cit. on p. 66).

[112] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Tech. rep. Technical Report 07-49, University of Massachusetts, Amherst, 2007 (cit. on p. 96).

[113] Keita Imaizumi and Vasily G Moshnyaga. "Network-based Face Recognition on Mobile Devices." In: *IEEE 3rd Int. Conf. on Consumer Electronics*. IEEE. 2013, pp. 406–409 (cit. on p. 96).

[114] iNaturalist. *A Community for Naturalists*. URL: `https://www.inaturalist.org/` (cit. on p. 89).

[115] EunYoung Jeong, Shinae Wood, Muhammad Jamshed, Haewon Jeong, Sunghwan Ihm, Dongsu Han, and KyoungSoo Park. "mTCP: a Highly Scalable User-level TCP Stack for Multicore Systems." In: *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. 2014, pp. 489–502 (cit. on p. 69).

[116] Jolle W Jolles. "Broad-scale Applications of the Raspberry Pi: A review and Guide for Biologists." In: *Methods in Ecology and Evolution* 12.9 (2021), pp. 1562–1579 (cit. on pp. 1, 2).

[117] Gareth Jones, David S. Jacobs, Thomas H. Kunz, Michael R. Willig, and Paul A. Racey. "Carpe Noctem: The Importance of Bats as Bioindicators." In: *Endang Species Res* 8 (2009), pp. 93–115 (cit. on p. 111).

[118] Kate E. Jones, Jon A. Russ, Andriy-Taras Bashta, Zoltálan Bilhari, Colin Catto, István Csősz, Alexander Gorbachev, Páleter Győrfi, Alice Hughes, Igor Ivashkiv, Natalia Koryagina, Anikálo Kurali, Steve Langton, Alanna Collen, Georgiana Margiean, Ivan Pandourski, Stuart Parsons, Igor Prokofev, Abigel Szodoray-Paradi, Farkas Szodoray-Paradi, Elena Tilova, Charlotte L. Walters, Aidan Weatherill, and Oleg Zavarzin. "Indicator Bats Program: A System for the Global Acoustic Monitoring of Bats." In: *Biodiversity Monitoring and Conservation* (2013), pp. 211–247 (cit. on p. 117).

[119] Stefan Kahl, Mary Clapp, W. Alexander Hopping, Hervé Goëau, Hervé Glotin, Robert Planqué, Willem-Pier Vellinga, and Alexis Joly. "Overview of BirdCLEF 2020: Bird Sound Recognition in Complex Acoustic Environments." In: *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020*. Vol. 2696. CEUR Workshop Proceedings. CEUR-WS.org, 2020 (cit. on p. 82).

[120] Stefan Kahl, Tom Denton, Holger Klinck, Hervé Glotin, Hervé Goëau, Willem-Pier Vellinga, Robert Planqué, and Alexis Joly. "Overview of BirdCLEF 2021: Bird Call Identification in Soundscape Recordings." In: *Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21-24, 2021.* Vol. 2936. CEUR Workshop Proceedings. CEUR-WS.org, 2021, pp. 1437–1450 (cit. on p. 82).

[121] Stefan Kahl, Connor M. Wood, Maximilian Eibl, and Holger Klinck. "BirdNET: A Deep Learning Solution for Avian Diversity Monitoring." In: *Ecological Informatics* 61 (2021), p. 101236. ISSN: 1574-9541 (cit. on pp. 82, 89, 90).

[122] Todd E Katzner and Raphaël Arlettaz. "Evaluating Contributions of Recent Tracking-based Animal Movement Ecology to Conservation Management." In: *Frontiers in Ecology and Evolution* 7 (2020), p. 519 (cit. on p. 129).

[123] Roland Kays, Margaret C Crofoot, Walter Jetz, and Martin Wikelski. "Terrestrial Animal Tracking as an Eye on Life and Planet." In: *Science* 348.6240 (2015), aaa2478 (cit. on pp. 143, 156).

[124] Roland Kays, Sameer Tilak, Margaret Crofoot, Tony Fountain, Daniel Obando, Alejandro Ortega, Franz Kuemmeth, Jamie Mandel, George Swenson, Thomas Lambert, et al. "Tracking Animal Location and Activity with an Automated Radio Telemetry System in a Tropical Rainforest." In: *The Computer Journal* 54.12 (2011), pp. 1931–1948 (cit. on pp. 129, 130, 143–145, 158, 160).

[125] Marcella J Kelly. "Design, Evaluate, Refine: Camera Trap Studies for Elusive Species." In: *Animal Conservation* 11.3 (2008), pp. 182–184 (cit. on p. 158).

[126] Robert E Kenward. *A Manual for Wildlife Radio Tagging.* Academic press, 2000 (cit. on p. 143).

[127] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* 2015 (cit. on pp. 87, 119).

[128] Charles G Kjos and William W Cochran. "Activity of Migrant Thrushes as Determined by Radio-telemetry." In: *The Wilson Bulletin* (1970), pp. 225–226 (cit. on p. 143).

[129] Brendan F Klare, Ben Klein, Emma Taborsky, Austin Blanton, Jordan Cheney, Kristen Allen, Patrick Grother, Alan Mah, and Anil K Jain. "Pushing the Frontiers of Unconstrained Face Detection and Recognition: IARPA Janus Benchmark A." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2015, pp. 1931–1939 (cit. on p. 96).

[130] Florian Klingler, Fynn Hauptmeier, Christoph Sommer, and Falko Dressler. "An Open Source Approach to Field Testing of WLAN up to IEEE 802.11 ad at 60 GHz Using Commodity Hardware." In: *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS).* IEEE. 2020, pp. 1312–1313 (cit. on p. 15).

[131] Roel van Klink, Diana E Bowler, Konstantin B Gongalsky, and Jonathan M Chase. "Long-term Abundance Trends of Insect Taxa are only Weakly Correlated." In: *Biology Letters* 18.2 (2022), p. 20210554 (cit. on pp. 65, 67, 107).

[132] Anna K Knuff, Nathalie Winiger, Alexandra-Maria Klein, Gernot Segelbacher, and Michael Staab. "Optimizing Sampling of Flying Insects using a Modified Window Trap." In: *Methods in Ecology and Evolution* 10.10 (2019), pp. 1820–1825 (cit. on p. 51).

[133] Keigo Kobayashi, Keisuke Masuda, Chihiro Haga, Takanori Matsui, Dai Fukui, and Takashi Machimura. "Development of a Species Identification System of Japanese Bats from Echolocation Calls using Convolutional Neural Networks." In: *Ecol. Informatics* 62 (2021), p. 101253 (cit. on pp. 112, 113).

[134] Dimitri Korsch, Paul Bodesheim, and Joachim Denzler. "Deep Learning Pipeline for Automated Visual Moth Monitoring: Insect Localization and Species Classification." In: *INFORMATIK 2021* (2021) (cit. on p. 67).

[135] Thomas H Kunz. "Roosting Ecology of Bats." In: *Ecology of bats*. Springer, 1982, pp. 1–55 (cit. on p. 158).

[136] Thomas H. Kunz. *Ecology of Bats*. 1st ed. Springer, Boston, MA, 1982. ISBN: 978-1-4613-3421-7 (cit. on p. 111).

[137] Thomas H. Kunz, Elizabeth Braun de Torrez, Dana Bauer, Tatyana Lobova, and Theodore H. Fleming. "Ecosystem Services Provided by Bats." In: *Annals of the New York Academy of Sciences* 1223.1 (2011), pp. 1–38 (cit. on p. 111).

[138] Thomas D Lambert, Roland W Kays, Patrick A Jansen, Enzo Aliaga-Rossel, and Martin Wikelski. "Nocturnal Activity by the Primarily Diurnal Central American agouti (Dasyprocta Punctata) in Relation to Environmental Conditions, Resource Abundance and Predation Risk." In: *Journal of Tropical Ecology* 25.2 (2009), pp. 211–215 (cit. on p. 143).

[139] Patrick Lampe, Lars Baumgärtner, Ralf Steinmetz, and Bernd Freisleben. "Smartface: Efficient Face Detection on Smartphones for Wireless on-demand Emergency Networks." In: *24th International Conference on Telecommunications (ICT)*. IEEE. 2017, pp. 1–7 (cit. on pp. xv, 6, 34, 39, 46, 95).

[140] Patrick Lampe, Markus Sommer, Artur Sterz, Jonas Höchst, Christian Uhl, and Bernd Freisleben. "ForestEdge: Unobtrusive Mechanism Interception in Environmental Monitoring." In: *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. IEEE. 2022, pp. 264–266 (cit. on pp. 4, 33, 34, 76).

[141] Patrick Lampe, Markus Sommer, Artur Sterz, Jonas Höchst, Christian Uhl, and Bernd Freisleben. "Unobtrusive Mechanism Interception." In: *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. IEEE. 2022, pp. 303–306 (cit. on pp. xiv, 4, 33, 34, 68, 76, 77).

[142] J Richard Landis and Gary G Koch. "An Application of Hierarchical Kappa-type Statistics in the Assessment of Majority Agreement Among Multiple Observers." In: *Biometrics* (1977), pp. 363–374 (cit. on p. 150).

[143] Mojib Latif. *Heißzeit: Mit Vollgas in die Klimakatastrophe-und wie wir auf die Bremse treten*. Verlag Herder GmbH, 2020 (cit. on p. 1).

[144] Mihai T Lazarescu. "Design and Field Test of a WSN Platform Prototype for Long-term Environmental Monitoring." In: *Sensors* 15.4 (2015), pp. 9481–9518 (cit. on p. 36).

[145] Daniel Lees, Tom Schmidt, Craig DH Sherman, Grainne S Maguire, Peter Dann, Glenn Ehmke, and Michael A Weston. "An Assessment of Radio Telemetry for Monitoring Shorebird Chick Survival and Causes of Mortality." In: *Wildlife Research* 46.7 (2019), pp. 622–627 (cit. on p. 129).

[146] Ariel K Lenske and Joseph J Nocera. “Field Test of an Automated Radio-telemetry System: Tracking Local Space use of Aerial Insectivores.” In: *Journal of Field Ornithology* 89.2 (2018), pp. 173–187 (cit. on p. 130).

[147] Roger A Light. “Mosquitto: Server and Client Implementation of the MQTT Protocol.” In: *Journal of Open Source Software* 2.13 (2017), p. 265 (cit. on p. 132).

[148] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. “Focal Loss for Dense Object Detection.” In: *IEEE International Conference on Computer Vision (ICCV)* (Oct. 2017) (cit. on p. 87).

[149] César Llamas, Manuel A González, Carmen Hernández, and Jesús Vegas. “Open Source Hardware based Sensor Platform Suitable for Human Gait Identification.” In: *Pervasive and Mobile Computing* 38 (2017), pp. 154–165 (cit. on p. 36).

[150] Gary M Lovett, Douglas A Burns, Charles T Driscoll, Jennifer C Jenkins, Myron J Mitchell, Lindsey Rustad, James B Shanley, Gene E Likens, and Richard Haeuber. “Who needs Environmental Monitoring?” In: *Frontiers in Ecology and the Environment* 5.5 (2007), pp. 253–260 (cit. on p. 76).

[151] Manisha Luthra, Boris Koldehofe, Jonas Höchst, Patrick Lampe, Ali Haider Rizvi, and Bernd Freisleben. “INetCEP: In-Network Complex Event Processing for Information-Centric Networking.” In: *15th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS).* Cambridge, UK, Sept. 2019, pp. 1–13 (cit. on p. 5).

[152] Balz Maag, Zimu Zhou, and Lothar Thiele. “W-Air: Enabling Personal Air Pollution Monitoring on Wearables.” In: *Proc. ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2.1 (2018), p. 24 (cit. on p. 36).

[153] Oisin Mac Aodha, Rory Gibb, Kate E Barlow, Ella Browning, Michael Firman, Robin Freeman, Briana Harder, Libby Kinsey, Gary R Mead, Stuart E Newson, et al. “Bat detective—Deep Learning Tools for Bat Acoustic Signal Detection.” In: *PLoS computational biology* 14.3 (2018), e1005995 (cit. on pp. 2, 10, 113, 117, 119).

[154] Robert MacCurdy, Rich Gabrielson, Eric Spaulding, Alejandro Purgue, Kathryn Cortopassi, and Kurt Fristrup. “Automatic Animal Tracking Using Matched Filters and Time Difference of Arrival.” In: *J. Commun.* 4.7 (2009), pp. 487–495 (cit. on p. 21).

[155] Leonardo Maffei, Andrew J Noss, Erika Cuéllar, and Damián I Rumiz. “Ocelot (Felis pardalis) Population Densities, Activity, and Ranging Behaviour in the Dry Forests of Eastern Bolivia: Data From Camera Trapping.” In: *Journal of Tropical Ecology* 21.3 (2005), pp. 349–353 (cit. on p. 143).

[156] Ying Mao, Jiayin Wang, Bo Sheng, and Fan Wu. “Building Smartphone Ad-hoc Networks with Long-range Radios.” In: *34th Int. Conf. on Computing and Communications.* IEEE. 2015, pp. 1–8 (cit. on p. 36).

[157] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin

Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* 2015. URL: `https://www.tensorflow.org/` (cit. on p. 88).

[158] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. "librosa: Audio and Music Signal Analysis in Python." In: *Proceedings of the 14th python in science conference.* Vol. 8. 2015 (cit. on p. 88).

[159] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. "OpenFlow: Enabling Innovation in Campus Networks." In: *ACM SIGCOMM Computer Communication Review* 38.2 (2008), pp. 69–74 (cit. on p. 69).

[160] Afef Mdhaffar, Tarak Chaari, Kaouthar Larbi, Mohamed Jmaiel, and Bernd Freisleben. "IoT-based Health Monitoring via LoRaWAN." In: *IEEE EUROCON 17th International Conference on Smart Technologies, Ohrid, Macedonia, July 6-8, 2017.* IEEE, 2017, pp. 519–524 (cit. on p. 137).

[161] Massimo Merenda, Carlo Porcaro, and Demetrio Iero. "Edge Machine Learning for AI-enabled IoT devices: A Review." In: *Sensors* 20.9 (2020), p. 2533 (cit. on pp. 83, 122).

[162] Hanna Meyer, Christoph Reudenbach, Tomislav Hengl, Marwan Katurji, and Thomas Nauss. "Improving Performance of Spatio-temporal Machine Learning Models using Forward Feature Selection and Target-oriented Validation." In: *Environmental Modelling & Software* 101 (2018), pp. 1–9 (cit. on p. 149).

[163] Markus Milchram, Marcela Suarez-Rubio, Annika Schröder, and Alexander Bruckner. "Estimating Population Density of Insectivorous Bats based on Stationary Acoustic Detectors: A Case Study." In: *Ecology and evolution* 10.3 (2020), pp. 1135–1144 (cit. on p. 158).

[164] Diomar Cristina Mistro, Luiz Alberto Dıéaz Rodrigues, and Sergei Petrovskii. "Spatiotemporal Complexity of Biological Invasion in a Space-and Time-discrete Predator–Prey System with the Strong Allee Effect." In: *Ecological Complexity* 9 (2012), pp. 16–32 (cit. on p. 64).

[165] Jonas Mielke Moeglich[2], Patrick Lampe[1], Mario Fickus, Jannis Gottwald, Thomas Nauss, Roland Brandl, Martin Braendle, Nicolas Friess, Bernd Freisleben, and Lea Heidrich. "Automated Non-lethal Moth Traps can be used for Robust Estimates of Moth Abundance." In: *submitted; under review* () (cit. on pp. xiii, 5, 33, 34, 52).

[166] Graham A Montgomery, Michael W Belitz, Rob P Guralnick, and Morgan W Tingley. "Standards and Best Practices for Monitoring and Benchmarking Insects." In: *Frontiers in Ecology and Evolution* (2021), p. 513 (cit. on pp. 12, 50, 51).

[167] Graham A Montgomery, Robert R Dunn, Richard Fox, Eelke Jongejans, Simon R Leather, Manu E Saunders, Chris R Shortall, Morgan W Tingley, and David L Wagner. "Is the Insect Apocalypse upon us? How to Find out." In: *Biological Conservation* 241 (2020), p. 108327 (cit. on p. 50).

[168] Robert A Montgomery, Gary J Roloff, Jay M Ver Hoef, and Joshua J Millspaugh. "Can we Accurately Characterize Wildlife Resource use when Telemetry Data are Imprecise?" In: *The Journal of Wildlife Management* 74.8 (2010), pp. 1917–1925 (cit. on p. 129).

[2] shared first authorship

[169] Markus Mühling, Jakob Franz, Nikolaus Korfhage, and Bernd Freisleben. “Bird Species Recognition via Neural Architecture Search.” In: *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020.* Vol. 2696. CEUR Workshop Proceedings. CEUR-WS.org, 2020 (cit. on p. 82).

[170] Ladislav Naďo and Peter Kaňuch. “Dawn Swarming in Tree-dwelling Bats–An Unexplored Behaviour.” In: *Acta chiropterologica* 15.2 (2013), pp. 387–392 (cit. on p. 158).

[171] B Naef-Daenzer, D Fruh, M Stalder, P Wetli, and E Weise. “Miniaturization (0.2 g) and Evaluation of Attachment Techniques of Telemetry Transmitters.” In: *Journal of Experimental Biology* 208.21 (2005), pp. 4063–4068 (cit. on pp. 143, 156).

[172] Miyabi Nakabayashi, Tomoko Kanamori, Aoi Matsukawa, Joseph Tangah, Augustine Tuuga, Peter T Malim, Henry Bernard, Abdul Hamid Ahmad, Ikki Matsuda, and Goro Hanya. “Temporal Activity Patterns Suggesting Niche Partitioning of Sympatric Carnivores in Borneo, Malaysia.” In: *Scientific reports* 11.1 (2021), pp. 1–12 (cit. on p. 143).

[173] Ran Nathan, Christopher T Monk, Robert Arlinghaus, Timo Adam, Josep Alós, Michael Assaf, Henrik Baktoft, Christine E Beardsworth, Michael G Bertram, Allert I Bijleveld, et al. “Big-data Approaches Lead to an Increased Understanding of the Ecology of Animal Movement.” In: *Science* 375.6582 (2022), eabg1780 (cit. on pp. 143, 167).

[174] Stuart E Newson, Hazel E Evans, and Simon Gillings. “A Novel Citizen Science Approach for Large-scale Standardised Monitoring of Bat Activity and Distribution, Evaluated in Eastern England.” In: *Biological Conservation* 191 (2015), pp. 38–49. ISSN: 0006-3207 (cit. on p. 117).

[175] Andrew Ng et al. “Sparse Autoencoder.” In: *CS294A Lecture notes* 72.2011 (2011), pp. 1–19 (cit. on pp. 17–19).

[176] Johnny Nguyen, Karl Kesper, Gunter Kräling, Christian Birk, Peter Mross, Nico Hofeditz, Jonas Höchst, Patrick Lampe, Alvar Penning, Bastian Leutenecker-Twelsiek, Carsten Schindler, Helwig Buchenauer, David Geisel, Caroline Sommer, Ronald Henning, Pascal Wallot, Thomas Wiesmann, Björn Beutel, Gunter Schneider, Enrique Castro-Camus, and Martin Koch. “Repurposing CPAP Machines as Stripped-Down Ventilators.” In: *Scientific Reports* 11.1 (June 2021), pp. 1–9 (cit. on p. 5).

[177] James D Nichols and Byron K Williams. “Monitoring for Conservation.” In: *Trends in ecology & evolution* 21.12 (2006), pp. 668–673 (cit. on p. 10).

[178] Martin K Obrist and Ruedi Boesch. “BatScope Manages Acoustic Recordings, Analyses Calls, and Classifies Bat Species Automatically.” In: *Canadian Journal of Zoology* 96.9 (2018), pp. 939–954 (cit. on p. 160).

[179] Kenichi Ozaki, Katsuhiko Sayama, Akira Ueda, Masato Ito, Ken Tabuchi, and Teruhiko Hironaga. “Short-term, Efficient Sampling Strategies for Estimating Forest Moth Diversity using Light Traps.” In: *Annals of the Entomological Society of America* 104.4 (2011), pp. 739–748 (cit. on p. 50).

[180] Eleni Papadatou, Roger K Butlin, and John D Altringham. “Identification of Bat Species in Greece from Their Echolocation Calls.” In: *Acta Chiropterologica* 10.1 (2008), pp. 127–143 (cit. on p. 11).

[181] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition.” In: *Interspeech 2019* (Sept. 2019) (cit. on p. 118).

[182] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. “Automatic Differentiation in PyTorch.” In: *NIPS-W*. 2017 (cit. on p. 117).

[183] Sharvanath Pathak and Vivek S Pai. “ModNet: A Modular Approach to Network Stack Extension.” In: *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. 2015, pp. 425–438 (cit. on p. 69).

[184] Y. Paumen, M. Mälzer, S. Alipek, J. Moll, B. Lüdtke, and H. Schauer-Weisshahn. “Development and Test of a Bat Calls Detection and Classification Method Based on Convolutional Neural Networks.” In: *Bioacoustics* 0.0 (2021), pp. 1–12 (cit. on pp. 10, 112, 113, 121).

[185] Tommaso Pecorella, Luca Simone Ronga, Francesco Chiti, Sara Jayousi, and Laurent Franck. “Emergency Satellite Communications: Research and Standardization Activities.” In: *IEEE Communications Magazine* 53.5 (2015), pp. 170–177 (cit. on p. 167).

[186] Eric J Pedersen, David L Miller, Gavin L Simpson, and Noam Ross. “Hierarchical Generalized Additive Models in Ecology: An Introduction with mgcv.” In: *PeerJ* 7 (2019), e6876 (cit. on p. 151).

[187] Juha Petäjäjärvi, Konstantin Mikhaylov, Marko Pettissalo, Janne Janhunen, and Jari Iinatti. “Performance of a Low-power Wide-area Network based on LoRa Technology: Doppler Robustness, Scalability, and Coverage.” In: *International Journal of Distributed Sensor Networks* 13.3 (2017), p. 1550147717699412 (cit. on p. 137).

[188] Nathalie Pettorelli, JE Baillie, and Sarah M Durant. “Indicator Bats Program: A System for the Global Acoustic Monitoring of Bats.” In: *Biodiversity monitoring and conservation: bridging the gap between global commitment and local action* (2013), pp. 211–247 (cit. on p. 10).

[189] Congduc Pham, Abdur Rahim, and Philippe Cousin. “WAZIUP: A Low-cost Infrastructure for Deploying IoT in Developing Countries.” In: *Int. Conf. on e-Infrastructure and e-Services for Developing Countries*. Springer. 2016, pp. 135–144 (cit. on p. 36).

[190] P Devi Pradeep, B Anil Kumar, et al. “A Survey of Emergency Communication Network Architectures.” In: *International Journal of u-and e-Service, Science and Technology* 8.4 (2015), pp. 61–68 (cit. on p. 95).

[191] Michele Preti, François Verheggen, and Sergio Angeli. “Insect Pest Monitoring with Camera-equipped Traps: Strengths and Limitations.” In: *Journal of pest science* 94.2 (2021), pp. 203–217 (cit. on pp. 51, 62).

[192] Jean-Francois Puget. “STFT Transformers for Bird Song Recognition.” In: *Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21-24, 2021*. Vol. 2936. CEUR Workshop Proceedings. CEUR-WS.org, 2021 (cit. on p. 82).

[193] R Core Team. *R: A language and Environment for Computing (Version 3.6. 3)[Computer software]. R Foundation for Statistical Computing*. 2020 (cit. on p. 162).

[194] Christoph Randler and Nadine Kalb. “Distance and Size Matters: A Comparison of Six Wildlife Camera Traps and Their Usefulness for Wild Birds.” In: *Ecology and Evolution* 8.14 (2018), pp. 7151–7163 (cit. on p. 158).

[195] Simon P Ripperger, Gerald G Carter, Rachel A Page, Niklas Duda, Alexander Koelpin, Robert Weigel, Markus Hartmann, Thorsten Nowak, Jörn Thielecke, Michael Schadhauser, et al. “Thinking Small: Next-generation Sensor Networks Close the Size Gap in Vertebrate Biologging.” In: *PLoS Biology* 18.4 (2020), e3000655 (cit. on pp. 130, 143, 165).

[196] Charlotte Roemer, Jean-François Julien, and Yves Bas. “An Automatic Classifier of Bat Sonotypes Around the World.” In: *Methods in Ecology and Evolution* (2021) (cit. on p. 112).

[197] Nicolas Roth, Herrmann Heinrich Hacker, Lea Heidrich, Nicolas Friess, Enrique Garcıéa-Barros, Jan Christian Habel, Simon Thorn, and Jörg Müller. “Host Specificity and Species Colouration Mediate the Regional Decline of Nocturnal Moths in Central European Forests.” In: *Ecography* 44.6 (2021), pp. 941–952 (cit. on p. 60).

[198] J Marcus Rowcliffe, Juliet Field, Samuel T Turvey, and Chris Carbone. “Estimating Animal Density using Camera Traps without the Need for Individual Recognition.” In: *Journal of Applied Ecology* 45.4 (2008), pp. 1228–1236 (cit. on p. 158).

[199] J Marcus Rowcliffe, Roland Kays, Bart Kranstauber, Chris Carbone, and Patrick A Jansen. “Quantifying Levels of Animal Activity using Camera Trap Data.” In: *Methods in ecology and evolution* 5.11 (2014), pp. 1170–1179 (cit. on p. 143).

[200] I Ruczyński, P Zahorowicz, T Borowik, and Z Hałat. “Activity Patterns of Two Syntopic and Closely Related Aerial-hawking Bat Species during Breeding Season in Białowieża Primaeval Forest.” In: *Mammal Research* 62.1 (2017), pp. 65–73 (cit. on p. 167).

[201] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. “Imagenet Large Scale Visual Recognition Challenge.” In: *International journal of computer vision* 115.3 (2015), pp. 211–252 (cit. on p. 87).

[202] Jens Rydell, Abigail Entwistle, and Paul A Racey. “Timing of Foraging Flights of Three Species of Bats in Relation to Insect Activity and Predation Risk.” In: *Oikos* (1996), pp. 243–252 (cit. on pp. 151, 167).

[203] David Sánchez-Fernández, Richard Fox, Roger LH Dennis, and Jorge M Lobo. “How Complete are Insect Inventories? An Assessment of the British Butterfly Database Highlighting the Influence of Dynamic Distribution Shifts on Sampling Completeness.” In: *Biodiversity and Conservation* 30.3 (2021), pp. 889–902 (cit. on pp. 50, 51, 60).

[204] Manu E Saunders and Gary W Luck. “Pan Trap Catches of Pollinator Insects Vary with Habitat.” In: *Australian Journal of Entomology* 52.2 (2013), pp. 106–113 (cit. on p. 51).

[205] Hall Sawyer, Matthew J Kauffman, Ryan M Nielson, and Jon S Horne. “Identifying and Prioritizing Ungulate Migration Routes for Landscape-level Conservation.” In: *Ecological Applications* 19.8 (2009), pp. 2016–2025 (cit. on p. 129).

[206] Stefano Scalercio, Pietro Brandmayr, Nino Iannotta, Ruggero Petacchi, and Luigi Boccaccio. “Correlations between Landscape Attributes and Ecological Traits of Lepidoptera Communities in Olive Groves.” In: *European Journal of Entomology* 109.2 (2012) (cit. on p. 50).

[207] Ulrike E Schlägel, Johannes Signer, Antje Herde, Sophie Eden, Florian Jeltsch, Jana A Eccard, and Melanie Dammhahn. “Estimating Interactions between Individuals from Concurrent Animal Movements.” In: *Methods in Ecology and Evolution* 10.8 (2019), pp. 1234–1245 (cit. on p. 165).

[208] Douglas C Schmidt, Michael Stal, Hans Rohnert, and Frank Buschmann. *Pattern-oriented Software Architecture, Patterns for Concurrent and Networked Objects*. John Wiley & Sons, 2013 (cit. on p. 71).

[209] Nils Schmidt, Lars Baumgärtner, Patrick Lampe, Kurt Geihs, and Bernd Freisleben. “Miniworld: Resource-aware Distributed Network Emulation via Full Virtualization.” In: *IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2017, pp. 818–825 (cit. on p. 6).

[210] Lynn N Schofield, Jill L Deppe, Theodore J Zenzal Jr, Michael P Ward, Robert H Diehl, Rachel T Bolus, and Frank R Moore. “Using Automated Radio Telemetry to Quantify Activity Patterns of Songbirds during Stopover.” In: *The Auk: Ornithological Advances* 135.4 (2018), pp. 949–963 (cit. on pp. 143, 145, 150).

[211] E Schwab, S Pogrebnoj, M Freund, F Flossmann, S Vogl, and K-H Frommolt. “Automated bat call classification using deep convolutional neural networks.” In: *Bioacoustics* (2022), pp. 1–16 (cit. on pp. 112, 113, 118–120).

[212] Sebastian Seibold, Martin M Gossner, Nadja K Simons, Nico Blüthgen, Jörg Müller, Didem Ambarlı, Christian Ammer, Jürgen Bauhus, Markus Fischer, Jan C Habel, et al. “Arthropod Decline in Grasslands and Forests is Associated with Landscape-level Drivers.” In: *Nature* 574.7780 (2019), pp. 671–674 (cit. on p. 50).

[213] Katja C Seltmann, Neil S Cobb, Lawrence F Gall, Charles R Bartlett, M Anne Basham, Isabelle Betancourt, Christy Bills, Benjamin Brandt, Richard L Brown, Charles Bundy, et al. “LepNet: The Lepidoptera of North America Network.” In: *Zootaxa* 4247.1 (2017), pp. 73–77 (cit. on p. 51).

[214] CB Shiel, RE Shiel, and JS Fairley. “Seasonal Changes in the Foraging Behaviour of Leisler’s Bats (Nyctalus Leisleri) in Ireland as Revealed by Radio-telemetry.” In: *Journal of Zoology* 249.3 (1999), pp. 347–358 (cit. on p. 167).

[215] Ravid Shwartz-Ziv and Amitai Armon. “Tabular Data: Deep Learning is not all You Need.” In: *Information Fusion* 81 (2022), pp. 84–90 (cit. on p. 145).

[216] BJÖRN M SIEMERS. “Bats in the Field and in a Flight Cage: Recording and Analysis of Their Echolocation Calls and Behavior.” In: *Bat echolocation research: tools, techniques, and analysis* (2004), pp. 107–113 (cit. on p. 10).

[217] Pavan Sikka, Peter Corke, Leslie Overs, Philip Valencia, and Tim Wark. “Fleck-A platform for Real-world Outdoor Sensor Networks.” In: *3rd. Int. Conf. on Intelligent Sensors, Sensor Networks and Information*. IEEE. 2007, pp. 709–714 (cit. on p. 36).

[218] Benno I Simmons, Andrew Balmford, Andrew J Bladon, Alec P Christie, Adriana De Palma, Lynn V Dicks, Juan Gallego-Zamorano, Alison Johnston, Philip A Martin, Andy Purvis, et al. “Worldwide Insect Declines: An Important Message, but Interpret with Caution.” In: *Ecology and Evolution* 9.7 (2019), pp. 3678–3680 (cit. on p. 50).

[219] Reinald Skiba. *Europäische Fledermäuse*. Westarp Wissenschaften, Hohenwarsleben, 2003 (cit. on p. 118).

[220] Nishant Somy, Abhijit Mondal, Bishakh Ghosh, and Sandip Chakraborty. "System Call Interception for Serverless Isolation." In: *Proceedings of the SIGCOMM'20 Poster and Demo Sessions*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 57–59. ISBN: 9781450380485 (cit. on p. 69).

[221] Tolga Soyata, Rajani Muraleedharan, Colin Funai, Minseok Kwon, and Wendi Heinzelman. "Cloud-vision: Real-time Face Recognition using a Mobile-cloudlet-cloud Acceleration Architecture." In: *IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2012, pp. 59–66 (cit. on p. 96).

[222] Talia Speaker, Stephanie O'Donnell, George Wittemyer, Brett Bruyere, Colby Loucks, Anthony Dancer, Marianne Carter, Eric Fegraus, Jonathan Palmer, Ellie Warren, et al. "A Global Community-sourced Assessment of the State of Conservation Technology." In: *Conservation Biology* (2022) (cit. on pp. xiii, 1, 2).

[223] Ian F Spellerberg. *Monitoring Ecological Change*. Cambridge University Press, 2005 (cit. on pp. 9, 10).

[224] Artur Sterz, Lars Baumgärtner, Jonas Höchst, Patrick Lampe, and Bernd Freisleben. "OPPLOAD: Offloading Computational Workflows in Opportunistic Networks." In: *IEEE 44th Conference on Local Computer Networks (LCN)*. Osnabrück, Germany, Oct. 2019, pp. 381–388 (cit. on p. 5).

[225] Dan Stowell, Tereza Petrusková, Martin Šálek, and Pavel Linhart. "Automatic Acoustic Identification of Individuals in Multiple Species: Improving Identification Across Recording Conditions." In: *Journal of the Royal Society Interface* 16.153 (2019), p. 20180940 (cit. on p. 158).

[226] SC Stuijfzand, S Engels, E Van Ammelrooy, and M Jonker. "Caddisflies (Trichoptera: Hydropsychidae) used for Evaluating Water Quality of Large European Rivers." In: *Archives of Environmental Contamination and Toxicology* 36.2 (1999), pp. 186–192 (cit. on p. 181).

[227] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the Inception Architecture for Computer Vision." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2818–2826 (cit. on p. 41).

[228] Michael A. Tabak, Kevin L. Murray, John A. Lombardi, and Kimberly J. Bay. "Automated Classification of Bat Echolocation Call Recordings with Artificial Intelligence." In: *Ecological Informatics* 68 (2022), p. 101526 (cit. on p. 112).

[229] Alejandro Talaminos-Barroso, Javier Reina-Tosina, and Laura M Roa. "Interceptor Pattern-Based Middleware for IoT Protocol Interoperability." In: *IoT and Cloud Computing for Societal Good*. Springer, 2022, pp. 221–244 (cit. on p. 70).

[230] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking Model Scaling for Convolutional Neural Networks." In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114 (cit. on p. 87).

[231] Philip Taylor, Tara Crewe, Stuart Mackenzie, Denis Lepage, Yves Aubry, Zoe Crysler, George Finney, Charles Francis, Christopher Guglielmo, Diana Hamilton, et al. "The Motus Wildlife Tracking System: A Collaborative Research Network to Enhance the Understanding of Wildlife Movement." In: *Avian Conservation and Ecology* 12.1 (2017) (cit. on pp. 21, 130, 143, 157, 158, 167).

[232] Hajime Tazaki, Ryo Nakamura, and Yuji Sekiya. "Library Operating System with Mainline Linux Network Stack." In: *Proceedings of NetDev* (2015) (cit. on p. 69).

[233] R Core Team. *R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. 2012.* 2021 (cit. on p. 151).

[234] Daniella Teixeira, Martine Maron, and Berndt J van Rensburg. "Bioacoustic Monitoring of Animal Vocal Behavior for Conservation." In: *Conservation Science and Practice* 1.8 (2019), e72 (cit. on p. 165).

[235] Telefonaktiebolaget L. M. Ericsson (Ericsson). *Ericsson Mobility Report.* Tech. rep. Nov. 2021. URL: `https://www.ericsson.com/4ad7e9/assets/local/reports-papers/mobility-report/documents/2021/ericsson-mobility-report-november-2021.pdf` (cit. on p. 69).

[236] David L Tennenhouse, Jonathan M Smith, W David Sincoskie, David J Wetherall, and Gary J Minden. "A Survey of Active Network Research." In: *IEEE Communications Magazine* 35.1 (1997), pp. 80–86 (cit. on p. 69).

[237] David L Tennenhouse and David J Wetherall. "Towards an Active Network Architecture." In: *ACM SIGCOMM Computer Communication Review* 26.2 (1996), pp. 5–17 (cit. on p. 69).

[238] Josh Thomas, Jeff Robble, and Nick Modly. "Off Grid Communications with Android Meshing the Mobile World." In: *IEEE Conference on Technologies for Homeland Security.* 2012, pp. 401–405 (cit. on p. 95).

[239] Sivan Toledo, David Shohami, Ingo Schiffner, Emmanuel Lourie, Yotam Orchan, Yoav Bartan, and Ran Nathan. "Cognitive Map-Based Navigation in Wild Bats revealed by a new High-throughput Tracking System." In: *Science* 369.6500 (2020), pp. 188–193 (cit. on p. 165).

[240] Colin J Torney, Juan M Morales, and Dirk Husmeier. "A Hierarchical Machine Learning Framework for the Analysis of Large Scale Animal Movement Data." In: *Movement ecology* 9.1 (2021), pp. 1–11 (cit. on p. 143).

[241] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. "Training Data-efficient Image Transformers & Distillation Through Attention." In: *38th International Conference on Machine Learning, PMLR* 139 (2021), pp. 10347–10357 (cit. on pp. 113–115).

[242] Viet-Hoang Tran and Olivier Bonaventure. "Beyond Socket Options: Making the Linux TCP Stack Truly Extensible." In: *IFIP Networking Conference (IFIP Networking).* IFIP, 2019, pp. 1–9 (cit. on p. 69).

[243] Devis Tuia, Benjamin Kellenberger, Sara Beery, Blair R Costelloe, Silvia Zuffi, Benjamin Risse, Alexander Mathis, Mackenzie W Mathis, Frank van Langevelde, Tilo Burghardt, et al. "Perspectives in Machine Learning for Wildlife Conservation." In: *Nature communications* 13.1 (2022), pp. 1–15 (cit. on pp. 1, 2).

[244] Silvia Liberata Ullo and Ganesh Ram Sinha. "Advances in Smart Environment Monitoring Systems using IoT and Sensors." In: *Sensors* 20.11 (2020), p. 3113 (cit. on p. 76).

[245] Riccardo Valentini, Luca Belelli Marchesini, Damiano Gianelle, Giovanna Sala, Alexey Yarovslavtsev, Viacheslav Vasenev, Simona Castaldi, et al. "New Tree Monitoring Systems: From Industry 4.0 to Nature 4.0." In: *Annals of Silvicultural research* 43.2 (2019), pp. 84–88 (cit. on pp. 13, 76).

[246] Jacolien Van Rij, Martijn Wieling, R Harald Baayen, and Dirk van Rijn. "Itsadug: Interpreting Time Series and Autocorrelated Data using GAMMs." In: (2015) (cit. on p. 152).

[247] Paul Viola and Michael J Jones. "Robust Real-time Face Detection." In: *International Journal of Computer Vision* 57.2 (2004), pp. 137–154 (cit. on pp. 96, 98).

[248] J Wolfgang Wägele, Paul Bodesheim, Sarah J Bourlat, Joachim Denzler, Michael Diepenbroek, Vera Fonseca, Karl-Heinz Frommolt, Matthias F Geiger, Birgit Gemeinholzer, Frank Oliver Glöckner, et al. "Towards a Multisensor Station for Automated Biodiversity Monitoring." In: *Basic and Applied Ecology* 59 (2022), pp. 105–138 (cit. on p. 66).

[249] David L Wagner. "Insect Declines in the Anthropocene." In: *Annual review of entomology* 65 (2020), pp. 457–480 (cit. on p. 50).

[250] Zea Walton, Gustaf Samelius, Morten Odden, and Tomas Willebrand. "Long-distance Dispersal in Red Foxes Vulpes Revealed by GPS Tracking." In: *European Journal of Wildlife Research* 64.6 (2018), pp. 1–6 (cit. on pp. 129, 143).

[251] David I Warton, Remko A Duursma, Daniel S Falster, Sara Taskinen, et al. "smatr 3-an R Package for Estimation and Inference about Allometric Lines." In: *Methods in ecology and evolution* 3.2 (2012), pp. 257–259 (cit. on p. 60).

[252] Adi Weller Weiser, Yotam Orchan, Ran Nathan, Motti Charter, Anthony J Weiss, and Sivan Toledo. "Characterizing the Accuracy of a Self-synchronized Reverse-GPS Wildlife Localization System." In: *15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE. 2016, pp. 1–12 (cit. on pp. 129, 130).

[253] David J Wetherall, John Guttag, David L Tennenhouse, et al. "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols." In: *IEEE Open Archichtecture and Network Programming*. Vol. 98. San Francisco, CA. IEEE, 1998, pp. 117–129 (cit. on p. 69).

[254] Peter JT White, Katharine Glover, Joel Stewart, and Amanda Rice. "The Technical and Performance Characteristics of a Low-cost, Simply Constructed, Black Light Moth Trap." In: *Journal of insect science* 16.1 (2016), p. 25 (cit. on p. 66).

[255] Hadley Wickham, Winston Chang, Lionel Henry, et al. *ggplot2: Create Elegant Data Visualisations using the Grammar of Graphics.[R package]; 2018*. 2021 (cit. on p. 61).

[256] Ross Wightman. *PyTorch Image Models*. `https://github.com/rwightman/pytorch-image-models`. 2019 (cit. on p. 117).

[257] Martin Wikelski, Roland W Kays, N Jeremy Kasdin, Kasper Thorup, James A Smith, and George W Swenson Jr. "Going Wild: What a Global Small-animal Tracking System could do for Experimental Biologists." In: *Journal of Experimental Biology* 210.2 (2007), pp. 181–186 (cit. on p. 143).

[258] Rachael Winfree, Jeremy W. Fox, Neal M Williams, James R Reilly, and Daniel P Cariveau. "Abundance of Common Species, not Species Richness, Drives Delivery of a Real-world Ecosystem Service." In: *Ecology letters* 18.7 (2015), pp. 626–635 (cit. on p. 65).

[259] Jan Alexander Wirwahn and Thomas Bartoschek. "Usability Engineering For Successful Open Citizen Science." In: *Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings*. Vol. 15. 1. 2015, p. 54 (cit. on p. 36).

[260] Lior Wolf, Tal Hassner, and Itay Maoz. "Face Recognition in Unconstrained Videos with Matched Background Similarity." In: *IEEE Conf. on Computer Vision and Pattern Recognition*. IEEE. 2011, pp. 529–534 (cit. on p. 96).

[261] Mirko Wölfling, Mira C Becker, Britta Uhl, Anja Traub, and Konrad Fiedler. "How Differences in the Settling Behaviour of Moths (Lepidoptera) may Contribute to Sampling Bias when using Automated Light Traps." In: *European Journal of Entomology* 113 (2016) (cit. on pp. 51, 64).

[262] Shipher Wu, Chun-Min Chang, Guan-Shuo Mai, Dustin R Rubenstein, Chen-Ming Yang, Yu-Ting Huang, Hsu-Hong Lin, Li-Cheng Shih, Sheng-Wei Chen, and Sheng-Feng Shen. "Artificial Intelligence Reveals Environmental Constraints on Colour Diversity in Insects." In: *Nature communications* 10.1 (2019), pp. 1–9 (cit. on p. 51).

[263] Teal B Wyckoff, Hall Sawyer, Shannon E Albeke, Steven L Garman, and Matthew J Kauffman. "Evaluating the Influence of Energy and Residential Development on the Migratory Behavior of Mule Deer." In: *Ecosphere* 9.2 (2018), e02113 (cit. on pp. 129, 143).

[264] Xeno-canto. *Sharing Bird Sounds from Around the World*. URL: `https://www.xeno-canto.org/` (cit. on p. 89).

[265] Yao-Yuan Yang, Moto Hira, Zhaoheng Ni, Anjali Chourdia, Artyom Astafurov, Caroline Chen, Ching-Feng Yeh, Christian Puhrsch, David Pollack, Dmitriy Genzel, Donny Greenberg, Edward Z. Yang, Jason Lian, Jay Mahadeokar, Jeff Hwang, Ji Chen, Peter Goldsborough, Prabhat Roy, Sean Narenthiran, Shinji Watanabe, Soumith Chintala, Vincent Quenneville-Bélair, and Yangyang Shi. "TorchAudio: Building Blocks for Audio and Speech Processing." In: *IEEE International Conference on Acoustics, Speech, and Signal Processing, Singapore*. 2022 (cit. on p. 117).

[266] Jun Yu, Jing Li, Zhou Yu, and Qingming Huang. "Multimodal Transformer With Multi-View Visual Representation for Image Captioning." In: *IEEE Transactions on Circuits and Systems for Video Technology* 30.12 (2020), pp. 4467–4480 (cit. on p. 113).

[267] Hubert Zimmermann. "OSI Reference Model-the ISO Model of Architecture for Open Systems Interconnection." In: *IEEE Transactions on communications* 28.4 (1980), pp. 425–432 (cit. on p. 15).

[268] Julian Zobel, Paul Frommelt, Patrick Lieser, Jonas Höchst, Patrick Lampe, Bernd Freisleben, and Ralf Steinmetz. "Energy-efficient Mobile Sensor Data Offloading via WiFi using LoRa-based Connectivity Estimations." In: *51. Jahrestagung der Gesellschaft für Informatik, Digitale Kulturen, INFORMATIK 2021, Berlin, Germany*. LNI. GI, Sept. 2021, pp. 461–479 (cit. on p. 5).

[269] Barret Zoph and Quoc V. Le. "Neural Architecture Search with Reinforcement Learning." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 2017 (cit. on p. 82).

[270] Imran Zualkernan, Jacky Judas, Taslim Mahbub, Azadan Bhagwagar, and Priyanka Chand. “A Tiny CNN Architecture for Identifying Bat Species from Echolocation Calls.” In: *IEEE / ITU International Conference on Artificial Intelligence for Good (AI4G)*. Sept. 2020, pp. 81–86 (cit. on pp. 112–114).

[271] Imran Zualkernan, Jacky Judas, Taslim Mahbub, Azadan Bhagwagar, and Priyanka Chand. “An AIoT System for Bat Species Classification.” In: *IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)*. 2021, pp. 155–160 (cit. on pp. 83, 122).