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

Jonas Höchst[1], Jannis Gottwald[2], Patrick Lampe[1], Julian Zobel[3], Thomas Nauss[2], Ralf Steinmetz[3], Bernd Freisleben[1]

Abstract: We present *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 and are built from low-cost commodity-off-the-shelf hardware. *tRackIT OS* provides software components for VHF signal processing, system monitoring, configuration management, and user access. In particular, it records, stores, analyzes, and transmits detected VHF signals and their descriptive features, e.g., to calculate bearings of signals emitted by VHF radio tags mounted on animals or to perform animal activity classification. Furthermore, we provide results of an experimental evaluation carried out in the *Marburg Open Forest*, the research and teaching forest of the University of Marburg, Germany. All components of *tRackIT OS* are available under a GNU GPL 3.0 open source license at `https://github.com/nature40/tRackIT-OS`.

Keywords: Radio Tracking; Marburg Open Forest; Open Source; Movement Ecology

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 [KA20]. Understanding the impact of altered habitats on the spatial distribution of species [Sa09], the effects of human infrastructures such as roads [Ho15, As19], and reasons for increased mortality of endangered species [Le19] 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 [Wy18, Ca10, Wa18].

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

[1] Department of Mathematics & Computer Science, University of Marburg, Germany,
{hoechst,lampep,freisleb}@informatik.uni-marburg.de

[2] Department of Geography, University of Marburg, Germany,
{gottwal5,nauss}@staff.uni-marburg.de

[3] Multimedia Communication Lab (KOM), Technical University of Darmstadt, Germany
{julian.zobel,ralf.steinmetz}@kom.tu-darmstadt.de

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 [Co99], and results in spatial and temporal data with poor resolution, which might not be sufficient for meaningful scientific analyses [Mo10].

Automated radio telemetry systems can minimize many of these disadvantages [Ka11, We16]. 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 [Go19] as part of the open source project *radio-tracking.eu*[4]. 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 paper, we present *tRackIT OS*, an open-source operating system distribution for reliable VHF radio tracking of small animals. *tRackIT OS* runs on a *tRackIT station*; its basic hardware design is due to Gottwald et al. [Go19]. 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*,
- a comparative evaluation of *tRackIT OS* compared to the state-of-the-art.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 discusses requirements, design decisions and implementation details, followed by experimental results in Section 4. Section 5 concludes the paper and outlines areas of future work.

2 Related Work

Ripperger et al. present a comprehensive overview of existing systems for localizing small animals using different technologies [Ri20]. The most recent projects on automated VHF

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

transmitter tracking are ARTS [Ka11], Atlas [We16], and Motus (also called SensorGnome) [Ta17].

ARTS consisted of towers with a height of 40 meters and top-mounted antenna arrays [Ka11], 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 [We16]. 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[5] is a globally operating network of VHF receiver stations hosted by different collaborators and supporting researchers [Ta17]. Despite its open source character, an implementation of Motus at US$ 3000 for a single SensorGnome[6] receiver with three 9-element Yagi antennas, and US$ 7500 for a Lotek SRX800 receiver station with four 9-element Yagi antennas is costly [LN18], 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 [Ta17], 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.

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*.

[5] Motus Wildlife Tracking System: `https://motus.org`
[6] SensorGnome Project: `https://sensorgnome.org`

3.1 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 [He20], fast data availability is highly important. Thus, direct data transmission from the field with the shortest possible delay between recording and transmission is desirable.

3.2 *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. [Go19]. 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[7]) expansion board (LoRa HAT), as shown in Figure 1.

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 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[8] is used. For receiving and forwarding *tRackIT stations*, the Dragino PG1301 LoRa Concentrator is used[9]. The basic hardware of a *tRackIT station* costs a total of about 200 €, consisting of 35 € for the Raspberry Pi 3B+,

[7] Semtech: `https://www.semtech.com/lora/`
[8] Dragino SX127X: `https://www.dragino.com/products/lora/item/106-lora-gps-hat.html`
[9] Dragino PG1301: `https://www.dragino.com/products/lora/item/149-lora-gps-hat.html`

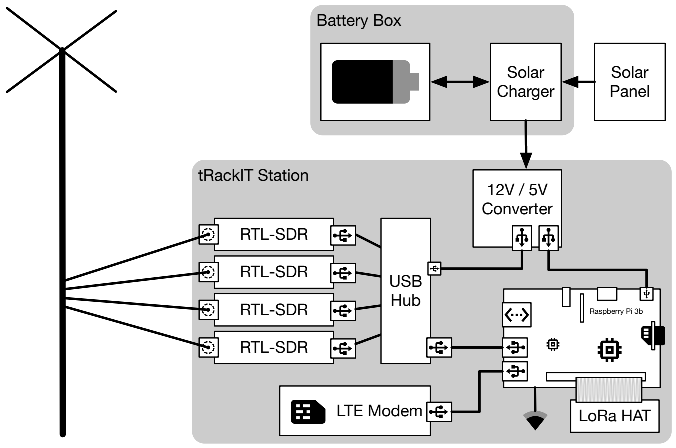


Fig. 1: The hardware components of a tRackIT station.

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.

3.3 *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 *pyradiotracking*, 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 [Hö20b], 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[10].

In Figure 2, 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 [Li17] for message distribution. It is configured such that incoming messages are forwarded to remote MQTT brokers for further processing. These brokers are also responsible for detecting and resolving connection failures.

[10] tRackIT OS, available online `https://github.com/Nature40/tRackIT-OS`

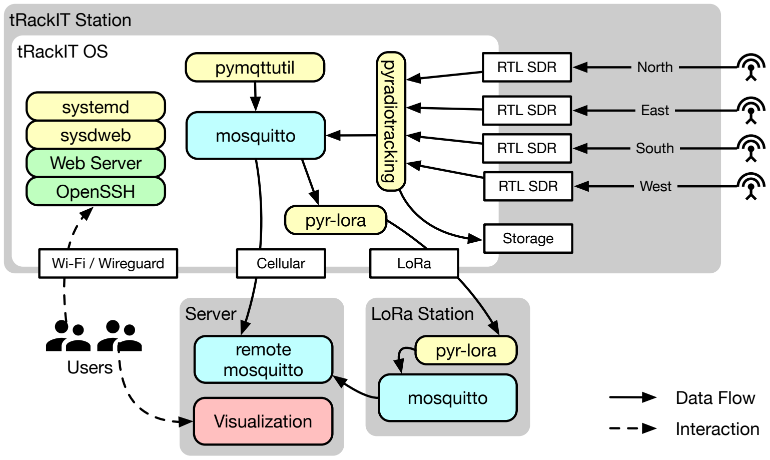


Fig. 2: Overview of the main software components of a tRackIT OS distribution.

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 3.4 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[11]. 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*[12] 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).

[11] pymqttutil, available online: `https://github.com/Nature40/pymqttutil`
[12] sysdweb, available online: `https://github.com/Nature40/sysdweb`

3.4 Signal Detection

The signal detection algorithm is implemented in the *pyradiotracking* Python package, which is released under a GPL 3.0 license[13]. In Figure 3, 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.

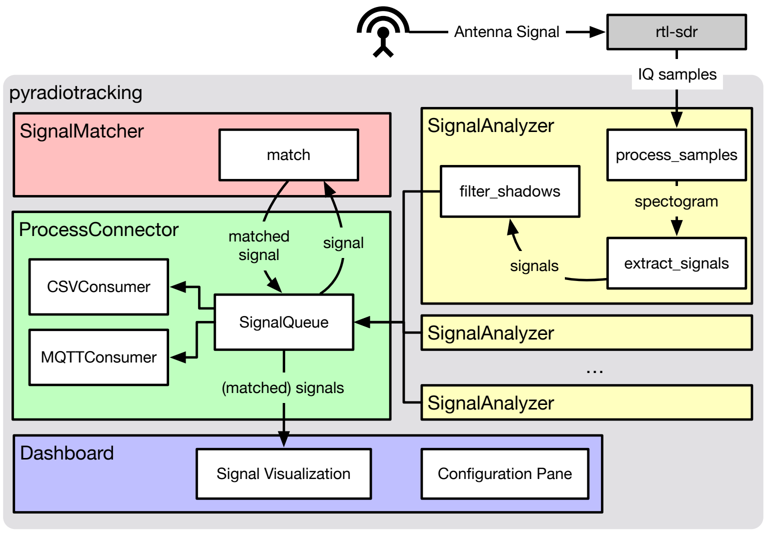


Fig. 3: Signal analysis stages implemented in pyradiotracking.

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 4, 5, and 6, three stages of signal processing are visualized.

Figure 4 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.

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

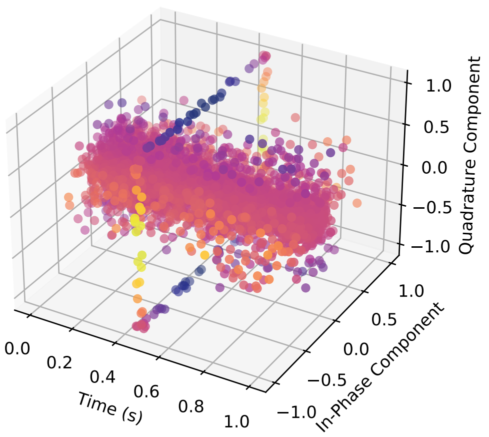


Fig. 4: IQ samples of one second, as received by RTL-SDR.

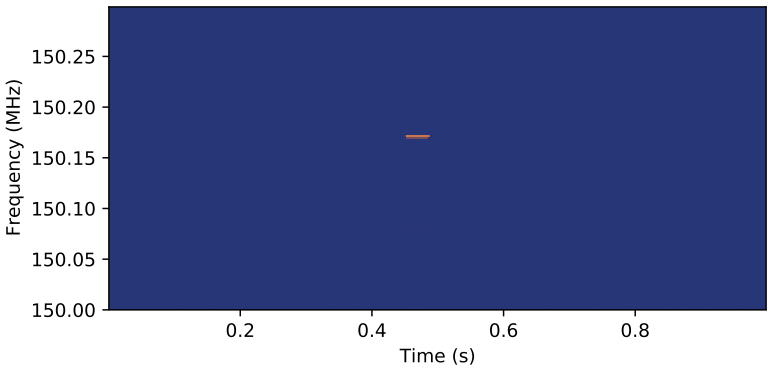


Fig. 5: Power spectral density (PSD) of samples computed via Short-time Fourier Transform (STFT).

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) [Al77] to the data. Figure 5 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 6, 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

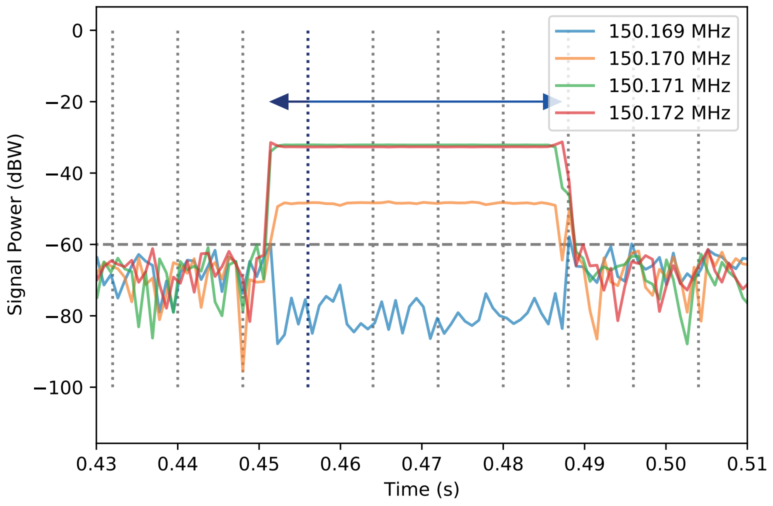


Fig. 6: Power spectral densities (PSDs) of selected frequencies, minimal signal power threshold, and signal power sampling points.

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 6, 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 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.

3.5 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.

3.6 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	$[1-4]\times 17$
		$52-103$

Tab. 1: 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 [Md17, Pe17, Hö20a].

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 1 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.

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`.

4.1 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*[14] project [Go19], 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

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

ms duration and an interval of one signal per second, which results in 3,193 sent signals. In Figure 7, 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.

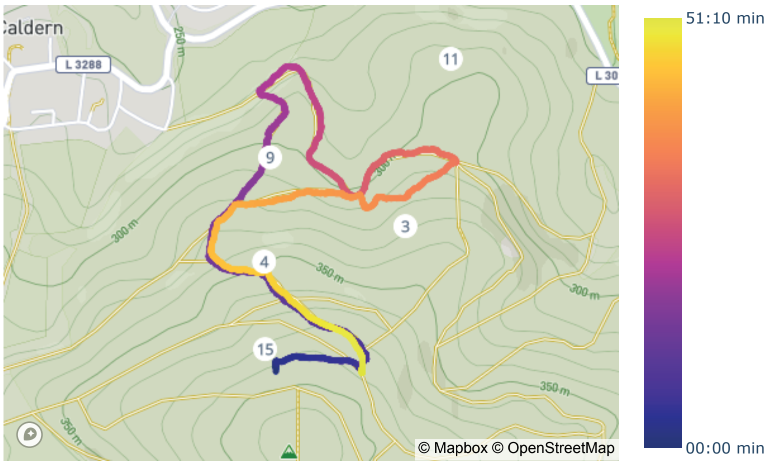


Fig. 7: GPS trace of the experimental evaluation track and the corresponding tRackIT stations.

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 paper showed the same low precision, hence all detected signals with a power lower than -78 dBW were removed for further processing. Using *tRackIT OS*, this threshold is not required, since we observed very low numbers of falsely detected signals.

4.2 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 8, 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 paper, we measured a delay in signal detection of 8 seconds in *paur* and no recognizable delay in *tRackIT OS*.

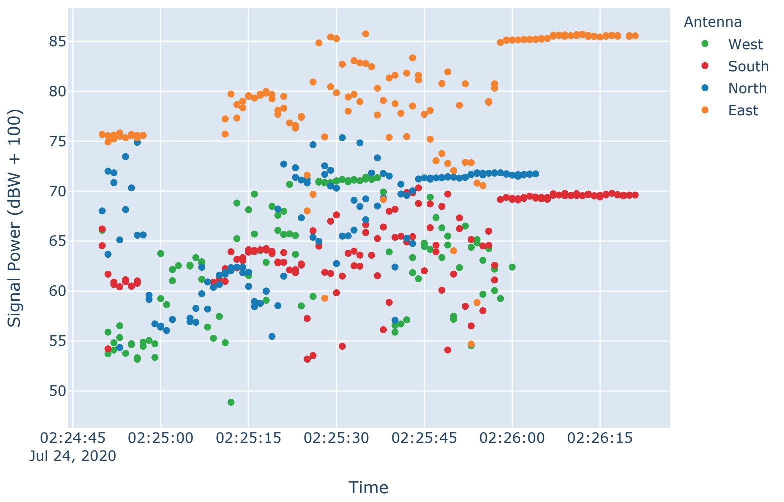


Fig. 8: Example of signal delay among different receivers observed in the 2020 field season using paur.

4.3 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 9 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.

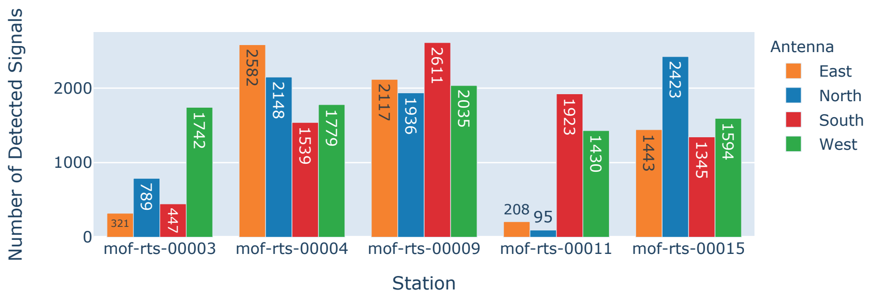
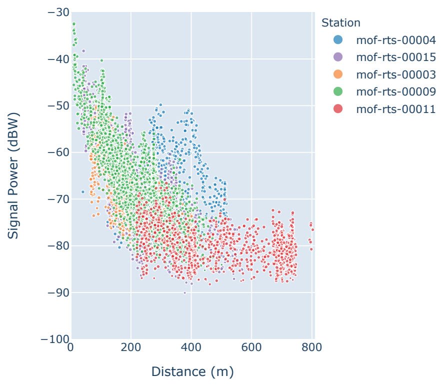
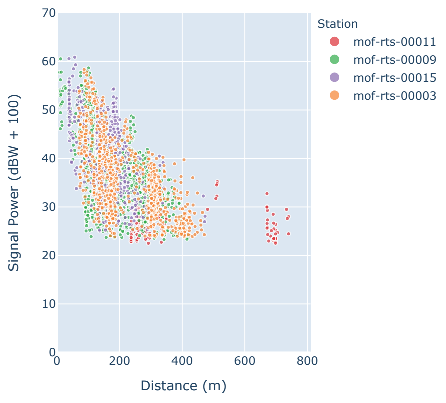


Fig. 9: Detected signals on tRackIT stations in the experimental scenario.

In addition to this quantitative analysis of signal detection, we evaluated the distances between the test tag and the *tRackIT stations*. Figure 10 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



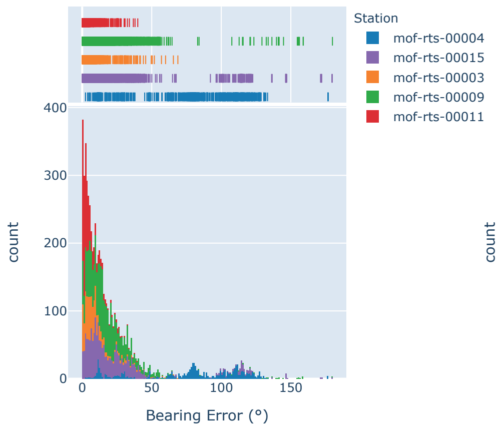
(b) paur

Fig. 10: Signal power and distance to a receiving station.

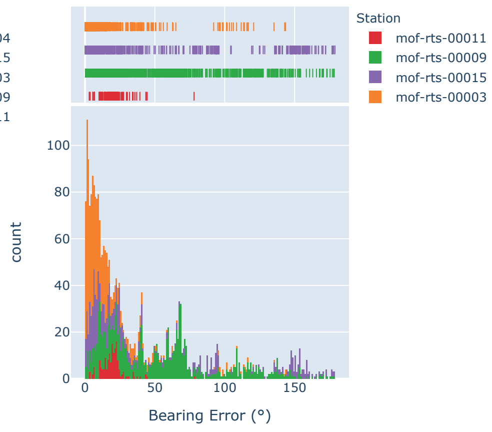
4.4 Bearing Calculation

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. [Go19] 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 $\omega = \frac{\pi}{90} \times arccos(\delta g)$.

Figure 11 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



(a) tRackIT OS



(b) paur

Fig. 11: Histogram of bearing errors.

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.

4.5 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[15].

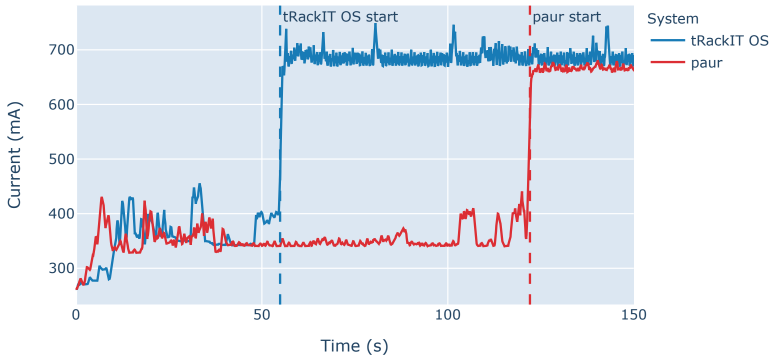


Fig. 12: Power measurements of tRackIT OS and paur in default settings.

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

Figure 12 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.

5 Conclusion

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* 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 field work.

There are several areas for future work. For example, calculating exact bearings can be challenging, since signals are affected by multiple factors, such as vegetation, 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 of multiple stations and further context information, such as a topology model and/or a calibration for the specific area of operation. Furthermore, it is quite challenging to transmit all detected signals under the given bandwidth limitations of the LoRa protocol. 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 major task in creating a user-friendly and widely applicable animal tracking system for generating ecological knowledge.

6 Acknowledgements

This work is financially supported by the HMWK LOEWE program of the State of Hesse, Germany, within the LOEWE research cluster *Nature 4.0*.

References

[Al77] Allen, Jonathan: Short term spectral analysis, synthesis, and modification by discrete Fourier transform. IEEE Transactions on Acoustics, Speech, and Signal Processing, 25(3):235–238, 1977.

[As19] Ascensão, Fernando; Kindel, Andreas; Teixeira, Fernanda Zimmermann; Barrientos, Rafael; D'Amico, Marcello; Borda-de Água, Luís; Pereira, Henrique M: Beware that the lack of wildlife mortality records can mask a serious impact of linear infrastructures. Global Ecology and Conservation, 19:e00661, 2019.

[Ca10] Cagnacci, Francesca; Boitani, Luigi; Powell, Roger A; Boyce, Mark S: Animal ecology meets GPS-based radiotelemetry: a perfect storm of opportunities and challenges. Philosophical Transactions of the Royal Society B: Biological Sciences, 365(1550):2157—2162, 2010.

[Co99] Cohn, Jeffrey P: Tracking wildlife: high-tech devices help biologists trace the movements of animals through sky and sea. BioScience, 49(1):12–17, 1999.

[Go19] Gottwald, Jannis; Zeidler, Ralf; Friess, Nicolas; Ludwig, Marvin; Reudenbach, Christoph; Nauss, Thomas: Introduction of an automatic and open-source radio-tracking system for small animals. Methods in Ecology and Evolution, 10(12):2163–2172, 2019.

[He20] Hecht, Luke: , Methods for studying wild animals' causes of death. Wild Animal Initiative, `https://www.wildanimalinitiative.org/blog/cause-of-death-2`, 11 2020. Accessed: 2020-11-02.

[Ho15] Hothorn, Torsten; Müller, Jörg; Held, Leonhard; Möst, Lisa; Mysterud, Atle: Temporal patterns of deer–vehicle collisions consistent with deer activity pattern and density increase but not general accident risk. Accident Analysis & Prevention, 81:143–152, 2015.

[Hö20a] Höchst, Jonas; Baumgärtner, Lars; Kuntke, Franz; Penning, Alvar; Sterz, Artur; Freisleben, Bernd: LoRa-based device-to-device smartphone communication for crisis scenarios. In: 17th International Conference on Information Systems for Crisis Response and Management (ISCRAM 2020). Blacksburg, Virginia, USA, May 2020.

[Hö20b] Höchst, Jonas; Penning, Alvar; Lampe, Patrick; Freisleben, Bernd: PIMOD: A tool for configuring single-board computer operating system images. In: 2020 IEEE Global Humanitarian Technology Conference (GHTC 2020). Seattle, USA, pp. 1–8, October 2020.

[Ka11] Kays, Roland; Tilak, Sameer; Crofoot, Margaret; Fountain, Tony; Obando, Daniel; Ortega, Alejandro; Kuemmeth, Franz; Mandel, Jamie; Swenson, George; Lambert, Thomas et al.: Tracking animal location and activity with an automated radio telemetry system in a tropical rainforest. The Computer Journal, 54(12):1931–1948, 2011.

[KA20] Katzner, Todd E; Arlettaz, Raphaël: Evaluating contributions of recent tracking-based animal movement ecology to conservation management. Frontiers in Ecology and Evolution, 7:519, 2020.

[Le19] Lees, Daniel; Schmidt, Tom; Sherman, Craig DH; Maguire, Grainne S; Dann, Peter; Ehmke, Glenn; Weston, Michael A: An assessment of radio telemetry for monitoring shorebird chick survival and causes of mortality. Wildlife Research, 46(7):622–627, 2019.

[Li17] Light, Roger A: Mosquitto: server and client implementation of the MQTT protocol. Journal of Open Source Software, 2(13):265, 2017.

[LN18] Lenske, Ariel K; Nocera, Joseph J: Field test of an automated radio-telemetry system: tracking local space use of aerial insectivores. Journal of Field Ornithology, 89(2):173–187, 2018.

[Md17] Mdhaffar, Afef; Chaari, Tarak; Larbi, Kaouthar; Jmaiel, Mohamed; Freisleben, Bernd: IoT-based health monitoring via LoRaWAN. In (Karadzinov, Ljupco; Cvetkovski, Goga; Latkoski, Pero, eds): IEEE EUROCON 2017 -17th International Conference on Smart Technologies, Ohrid, Macedonia, July 6-8, 2017. IEEE, pp. 519–524, 2017.

[Mo10] Montgomery, Robert A; Roloff, Gary J; Hoef, Jay M Ver; Millspaugh, Joshua J: Can we accurately characterize wildlife resource use when telemetry data are imprecise? The Journal of Wildlife Management, 74(8):1917–1925, 2010.

[Pe17] Petäjäjärvi, Juha; Mikhaylov, Konstantin; Pettissalo, Marko; Janhunen, Janne; Iinatti, Jari: Performance of a low-power wide-area network based on LoRa technology: Doppler robustness, scalability, and coverage. International Journal of Distributed Sensor Networks, 13(3):1550147717699412, 2017.

[Ri20] Ripperger, Simon P; Carter, Gerald G; Page, Rachel A; Duda, Niklas; Koelpin, Alexander; Weigel, Robert; Hartmann, Markus; Nowak, Thorsten; Thielecke, Jörn; Schadhauser, Michael et al.: Thinking small: next-generation sensor networks close the size gap in vertebrate biologging. PLoS Biology, 18(4):e3000655, 2020.

[Sa09] Sawyer, Hall; Kauffman, Matthew J; Nielson, Ryan M; Horne, Jon S: Identifying and prioritizing ungulate migration routes for landscape-level conservation. Ecological Applications, 19(8):2016–2025, 2009.

[Ta17] Taylor, Philip; Crewe, Tara; Mackenzie, Stuart; Lepage, Denis; Aubry, Yves; Crysler, Zoe; Finney, George; Francis, Charles; Guglielmo, Christopher; Hamilton, Diana et al.: The Motus Wildlife Tracking System: a collaborative research network to enhance the understanding of wildlife movement. Avian Conservation and Ecology, 12(1), 2017.

[Wa18] Walton, Zea; Samelius, Gustaf; Odden, Morten; Willebrand, Tomas: Long-distance dispersal in red foxes Vulpes vulpes revealed by GPS tracking. European Journal of Wildlife Research, 64(6):1–6, 2018.

[We16] Weiser, Adi Weller; Orchan, Yotam; Nathan, Ran; Charter, Motti; Weiss, Anthony J; Toledo, Sivan: Characterizing the accuracy of a self-synchronized reverse-GPS wildlife localization system. In: 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN). IEEE, pp. 1–12, 2016.

[Wy18] Wyckoff, Teal B; Sawyer, Hall; Albeke, Shannon E; Garman, Steven L; Kauffman, Matthew J: Evaluating the influence of energy and residential development on the migratory behavior of mule deer. Ecosphere, 9(2):e02113, 2018.