Hes·so

MSE | MASTER OF SCIENCE IN ENGINEERING

DATA COLLAR

Master of Science HES-SO in Engineering

Computer Science (CS)

06.02.2026

Produced by

Adrien Rey

Under the supervision of Alexandra Andersson - HES-SO Valais Wallis
In collaboration with the ECS group - HES-SO Valais Wallis

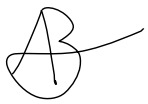
Information about this report

Contact Information

Author:	Adrien Rey
	Master Student
	HEI-Vs
E-Mail:	adrien.rey@hevs.ch

Declaration of honor
I, undersigned, hereby declare that the work submitted is the result of a personal work. I certify that I have not resorted to plagiarism or other forms of fraud. All sources of information used and the author quotes were clearly mentioned

Place, date	Sion, 05.02.2026

Signature 

Contents

Figures

Tables

1 Acknowledgments

I would like to express my gratitude to my project advisor, Alexandra Andersson, for his guidance, expertise and support throughout this project.

I am also grateful to the ECS team for their support. Special thanks to Medard Rieder and Patrice Rudaz for providing materials and assisting with firmware development.

I also thank the electronics workshop team for their work and for helping with PCB's debugging.

Finally, I want to thank my family and friends for their encouragement throughout this project.

2 Abstracts

Studying primate behavior in their natural environment is key to understanding how they live and interact. Direct observation is difficult and often affected by the surroundings. To help with this, the *nRF Monkey* audio recording collar was developed by the Embedded Communicating Systems research group at HES-SO Valais/Wallis to automatically monitor vervet monkeys in South Africa. The first version could record audio for 12 days straight, save it to an SD card, include mobile app control via Bluetooth Low Energy, as well as a remote release feature for retrieval, all of this is managed by Zephyr RTOS.

This thesis describes the design and development of the second prototype. The main goal was to improve battery life by upgrading the system's architecture. The second objective was to add Bluetooth Low Energy proximity detection to monitor the interactions of the monkeys. The new version uses the *Nordic Semiconductor nRF54L15* System-on-Chip and includes a *TDK T5848* MEMS microphone with Acoustic Activity Detection.

The approach focused on minimizing power consumption through an intelligent trigger mechanism. Rather than recording continuously, the system stays in deep sleep and only turns on when sound levels exceed a set threshold.

The results show a significant improvement in efficiency. In real-world use, the intelligent trigger enables the V2 prototype to consume only about one tenth the power of the first version. Hardware and firmware improvements also cut current use by half during recording. This allows the battery life to increase from ~20 days with an equivalent battery to over 200 days (with the V1 battery).

This tool gives scientists a reliable way to study how primates communicate, while causing little disturbance to the animals.

Keywords : *Bio-logging*, *Embedded system*, *Zephyr RTOS*, *BLE location*, *Low-power design*

3 Introduction

3.1 Context

Ethology is the study of animal behavior and relies on gathering data in natural environments. For primates such as the Vervet monkey, researchers must observe them over long periods to learn about their social lives and vocal communication. However, direct observation is challenging due to weather, human impact on animal behavior and the effort needed for manual tracking.

In collaboration with Pre Erica van de Waal (UNIL) and Pre Benedetta Franceschiello (HEI), the *Monkeycall* project was established to meet the challenge of obtaining high-quality recordings of monkeys. As part of this project, the Embedded Communicating Systems research group at HES-SO Valais/Wallis has developed the *nRF Monkey*, a bio-logging collar.

Figure 1: *nRF Monkey* on vervet monkey in South Africa

The device records audio onto an SD card automatically and can be managed with a mobile app using Bluetooth Low Energy. It also features a remote release system, allowing scientists to retrieve the collar without recapturing the animal. The system allows to record continuously for 12 days.

3.2 Problems

While the V1 prototype demonstrated the feasibility of the system, it is not yet suitable for extended scientific campaigns. The feedback from the field tests identified critical shortcomings in three main areas: power consumption, mechanical reliability, and functional scope. The field report is available in the repository (see Section AA).

It leads to several important problems:

- **Power consumption :** The most critical issue is the system's energy efficiency. The V1 architecture relies on continuous recording. Furthermore, to meet the system requirements, the battery needed is disproportionately large compared to the other components.
- **Data overload :** The system generates massive amounts of data, the majority of which is silence or irrelevant ambient noise. This complicates the post-processing phase.
- **Lack of social context :** The current system only records audio. It is therefore complicated to link the interactions between the monkeys.
- **Mechanical issues :** The collar's dimensions and weight are excessive. Furthermore, the system's waterproofing was flawed.

3.3 Preliminary works

To address the issues identified in V1, preliminary projects were conducted prior to this master's thesis, as part of my master's program. They laid the groundwork for the hardware selection and mechanical design.

3.3.1 PI *MobileSens* 2024

The interdisciplinary project *MobileSens* focused on miniaturization and robustness. Its primary goal was to reduce the collar's weight and size to minimize the impact on the monkeys' natural behavior. [1]

- **Mechanics :** It proposed two new housing variants and release mechanisms to improve comfort and reliability.
- **Electronics :** A new PCB form factor was designed to fit the smaller housing, with improved SD card retention to withstand vibrations.
- **Features :** It explored the feasibility of adding a GNSS module and developed a *BLE Booster* (long-range remote) to extend the communication range with the collar.

3.3.2 PA *Data collar* 2025

The application project *Data collar* focused on evaluating a positioning technologies and develop a custom power profiling tool. [2]

- **GNSS validation :** It tests the possibility to use GNSS as a positioning technologies. The consumption, integration and performance are evaluated.
- **Validation tools :** It develops a custom power profiling tool to accurately measure consumption in the field, as standard laboratory equipment is not portable.

3.4 Objectives

The main goal of this project is to update the monkey collar system to a new low-power design using the *nRF54L15* SoC. This means moving the current firmware to the new board and handling any hardware-specific challenges. One important task is adding the *T5848* MEMS microphone, which requires firmware to handle I2S audio streams and to set up its Acoustic Activity Detection features. To ensure the system records the correct vocalizations, field recordings from the V1 collar will be analyzed and used to adjust detection settings. The power use will be monitored and improved to help the battery last longer.

A major new feature is the addition of proximity detection to track social interactions between monkeys. Current BLE proximity solutions will be reviewed to find a low-power method that fits the needs. Each collar will act as both a broadcaster and an observer, enabling devices to detect one another. The system will save proximity data to the SD card and filter out devices that are not part of the project. To get the best balance of detection and battery life, different advertising and listening cycles will be test.

The final phase focuses on validating the V2 prototype's performance improvements and reliability. This includes verifying the electrical conformity of the new electronic boards and conducting comprehensive power consumption profiling across different operational states. By measuring current draw across distinct modes and comparing these measurements with the V1 prototype, the project can calculate theoretical improvements in battery life and validate that efficiency gains.

4 Analysis

4.1 Low-power embedded systems considerations

Small, low-power devices have to face several challenges. During the development of one of these systems, particular attention must be paid to these points. [3]

4.1.1 Mechanical challenges

- **Battery constraints :** The battery's size affects how long the device can run, so the software must be optimized accordingly.
- **Packaging density :** Making devices smaller means using multi-layer PCBs and tiny parts. This makes routing and repairs more difficult.
- **Environmental protection :** Small wearables often need to be sealed to stay waterproof. This can trap heat and make it hard to access connectors like USB or JTAG without breaking the seal.

4.1.2 Performance challenges

- **Processing vs. power :** Running the CPU uses power. To save energy, the system usually stays in sleep mode whenever possible.
- **Memory limits :** Low-power chips have limited RAM and flash. This means there is less space for data storage and for complex algorithms on the device. [4]

4.1.3 Cost challenges

- **Integration cost :** A small integrated device will be more difficult and expensive to produce in mass.
- **Battery technology :** High-density batteries are significantly more expensive per watt-hour than standard battery types.

The design of such a system is a constant balancing act between performance vs. battery autonomy, integration vs. productibility, and size vs. cost.

4.2 Existing technologies

Because the project requirements are very specific, it is difficult to find systems that meet all the needs. However, similar systems have been analyzed.

4.2.1 Edic-mini Tiny A31

This device is a spy recorder. [5]

His main characteristics are:
- **Small size and weight :** 29x12x15 mm, 6 grams.
- **Low power consumption :** Up to 30 hours in record mode or 140 hours with the voice-activating system.
- **Integrated memory :** Up to 8 GB of built-in memory.
- **Recording time :** Up to 1200 hours of recording time, limited by the memory size.

Source: https://ts-market.com/products/models/1198/

Figure 2: Edic-mini Tiny A31

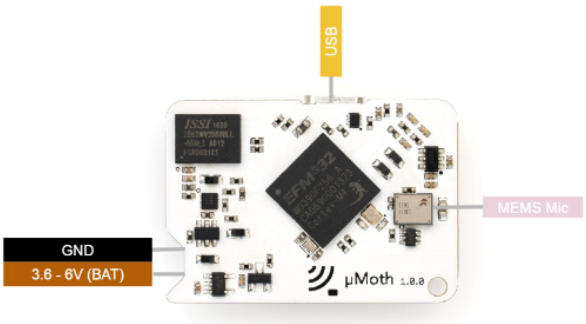
Edic-mini Tiny A31 was not originally designed to record wildlife activity, but it can easily serve as a base for developing a recording collar. Studies have validated the use of commercially available *Edic-mini Tiny* for biological monitoring. A 24-hour high-quality capture has been achieved, capturing calls of animals as small as chipmunks. [6]

4.2.2 MicroMoth

The *MicroMoth* is the open-source conservation tool. It is a bare printed circuit board with a MEMS microphone and microSD slot. It runs open-source firmware. [7]

The main performances are :

- **Small size:** 36x25x5 mm.
- **Recording time** :: 5 to 7 days on 3.7V Li-Po battery.
- **Configurable :** Can easily reach 10+ days with a schedule.
- **Memory :** Integrated SD card port.
- **Triggering :** *AudioMoth Labs* firmware allows for magnetic triggering or amplitude threshold recording.



Source: https://ts-market.com/products/models/1198/

Figure 3: MicroMoth

As for the previous device, its initial purpose is not wildlife recording. But as this is an open source project, it can be adapted to the requirements. This module has already enabled the offer of a programmable, energy-efficient platform to researchers. Large-scale sensor networks have been deployed for comprehensive biodiversity and environmental monitoring. [8]

4.3 Microphone

Sound is a vibration that travels as a pressure wave. Humans and many other species use sound to communicate or as an alarm system. It also provides information about the environment. This thesis focuses on atmospheric sound, but sound can move through both fluids and solids. [9]

Sound can be converted into electrical signals using a microphone. To understand this process, it is important to consider the different transduction mechanisms.

4.3.1 Microphone technologies

There are several types of microphone technologies:

- **Piezoresistive :** The diaphragm changes its electrical resistance when mechanically deformed by sound pressure. It then produces an output voltage.
- **Optical :** The intensity or the phase of the light is modified by the diaphragm. The transduction mechanism is produced by sensing changes in light intensity. They have the advantage of being resistant to electromagnetic interference and humidity.
- **Capacitive :** It works like a variable capacitor, in which sound waves move a flexible diaphragm toward or away from a fixed backplate.
- **Dynamic :** It works by electromagnetic induction. Diaphragm movement in a magnetic field induces a current related to sound pressure.
- **Ribbon :** A thin, corrugated metal ribbon is suspended between two magnetic poles. When sound waves move the ribbon in a magnetic field, an electrical voltage is generated along the ribbon.
- **Carbon :** Based on variable resistance. Sound pressure on a diaphragm compresses carbon granules, changing resistance and causing current fluctuations.
- **MEMS :** Typically uses a capacitive design on silicon chips. Sound pressure changes capacitance, producing a voltage signal.

The microphone was provided at the beginning of the project (*TDK T5848*). Its selection was made prior to this thesis by a member of the HES-SO Valais/Wallis. The following discussion will then focus specifically on the operation of a MEMS microphone.

4.3.2 MEMS technology

Micro-Electro-Mechanical Systems is a technology that combines electrical and mechanical parts on a microscopic scale. Standard silicon processing methods, like those used for integrated circuits, are usually used to make MEMS devices. These devices can range from a few microns to several millimeters.

New processes enable the production of many mechanical structures alongside complex electronic circuits on the same silicon chip. These devices are valued for their small size, low power consumption and lower manufacturing costs. MEMS technology helps make modern electronics smaller.

4.3.3 MEMS microphone operation

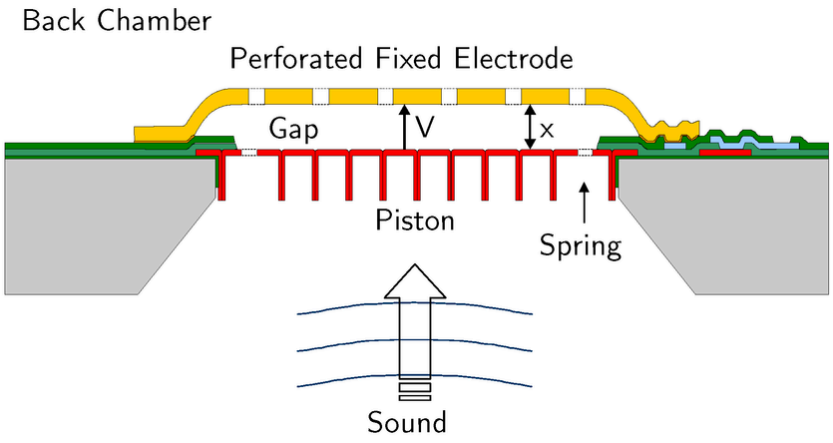
The vast majority of MEMS microphone operate using the capacitive transduction principle.

Core principle

A capacitive microphone consists of two main components that form a capacitor:

- **A movable diaphragm (or membrane) :** Acts as one plate of the capacitor, is free to move (red part in the Figure 4)
- **A rigid backplate :** Acts as the second plate of the capacitor, is fixed (yellow part in the Figure 4)

The diaphragm is designed to be acoustically sensitive. When sound waves strike the diaphragm through holes in the backplates, it vibrates. The movement changes the distance between the diaphragm and the backplate, which changes the capacitance.



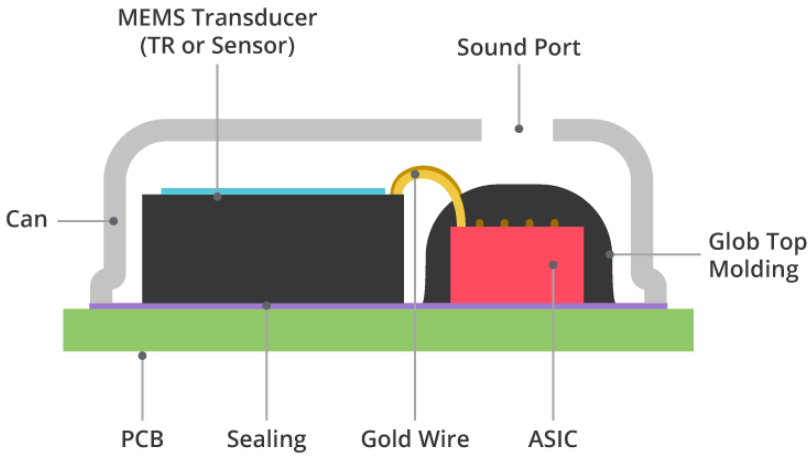
Source: *https://www.researchgate.net/figure/Basic-structure-and-working-principle-of-a-MEMS-microphone_fig2_326006102*

Figure 4: MEMS transducer cross-section view

Transduction mechanism

The movement of the sound waves is converted into an electrical signal by the transduction principle. The principal elements are detailed below. [10]

- **Capacitance change :** The capacitance C of the system is inversely proportional to the distance x between the plates. As the diaphragm moves in response to sound, the capacitance changes dynamically.
- **Signal conversion :** This fluctuating capacitance is then read by on-chip Application-Specific Integrated Circuit, which is usually integrated into the same MEMS package, as in the Figure 5.
- **Output generation :** The ASIC converts the capacitance change into an electrical signal (voltage or digital stream) that is proportional to the actual acoustic pressure. This integrated circuitry is essential, not only for signal amplification and conditioning, but also for converting the analog signal into a digital Pulse Density Modulation (PDM) or I²S signal.



Source: https://www.edn.com/how-does-mems-microphone-work/

Figure 5: MEMS microphone cross-section view

Advantages

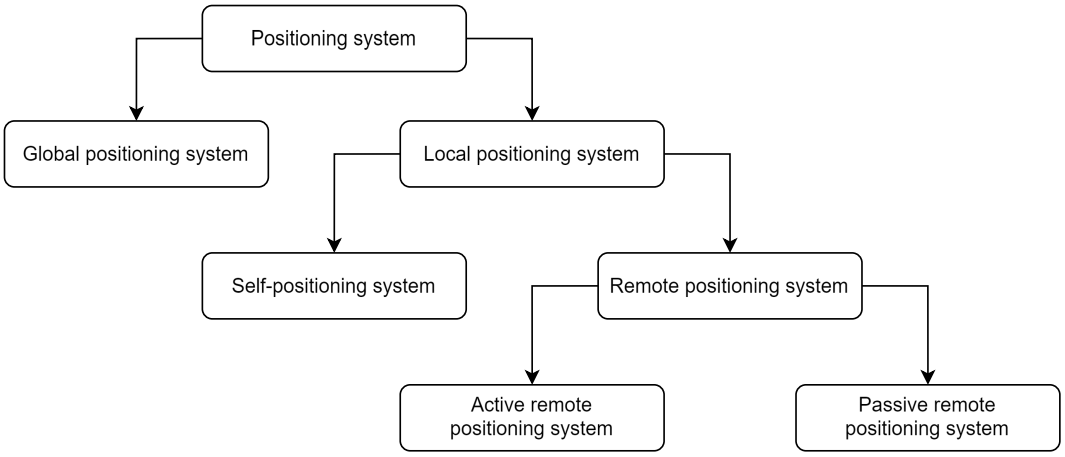
- **Small footprint and miniaturization :** MEMS fabrication allows the entire microphone system (diaphragm, backplate, and ASIC) to be packaged into a single device.
- **High immunity to RF interference :** Many MEMS microphone offer digital outputs.

Disadvantages

- **Requires power :** As capacitive devices, MEMS microphone require integrated preamplification circuitry and a polarization voltage to operate. They are active devices and therefore always require power to function.
- **Built-in noise floor limitation :** Due to their small physical size, the signal-to-noise ratio is often constrained by the diaphragm's dimensions.
- **Vulnerability to moisture and particulate contamination :** The extremely small gap between the diaphragm and the backplate can be sensitive to moisture and microscopic particles.

4.4 Positioning

There are several technologies for tracking animal locations, each having its own level of accuracy. These positioning systems are generally divided into two groups: range-free systems (which estimate position using network connections or relative localization without direct measurements), and range-based systems (which calculate position using features or signal strength).



Source: [11]

Figure 6: Positioning system classification

The literature shows that these positioning methods fall into two types, as shown in Figure 6:

- **Global positioning :** Each device finds its location worldwide, and local positioning, which is relative and is described below.
- **Local positioning :** This method is relative and includes two types: self-positioning and remote positioning. In self-positioning, each person or item can find its own position compared to a fixed point. In a remote positioning system, each node can locate other nodes within its coverage area. The target can be passive or active.

High-precision technologies can pinpoint the exact locations of individual animals, enabling studies of how they relate to one another in space. For these studies to be useful, the location data needs to be accurate enough.

Some technologies are designed to detect when an animal is near certain structures and can sometimes estimate distances. While these systems are less precise and depend on pre-installed equipment, careful placement can provide good coverage of the study area.

Some technologies enable animals to detect each other directly, without the need for external equipment. Although this method does not give exact locations, it shows which animals are close to each other, which is very useful for studying social behavior. By combining this proximity data with information from base stations, researchers can learn about both animal movements and group interactions. This type of technology is often called *Indoor positioning* in scientific literature.

Indoor Positioning System applications are usually divided into three main categories:

1. **Range-based positioning :** This method estimates a device's location by measuring distances or angles from known reference points.
2. **Fingerprinting :** This method compares real-time signal signatures to a database of signal maps to find the location.
3. **Proximity-based positioning :** This method finds location by detecting which reference point is closest or currently connected.

4.4.1 Positioning technologies

The following sections review existing solutions, outlining their pros and cons in the context of this project to help choose the best option. The evaluation and selection are done in the Section 5.

4.4.1.1 GNSS

Global Navigation Satellite System refers to a network of satellites that transmit positioning and timing data to GNSS-enabled receivers on Earth. These receivers use signals from several satellites to determine their precise location (latitude, longitude, and altitude), speed, and time. The GNSS system in the United States is the most well-known GPS, while other systems include BeiDou (China), Galileo (EU), and GLONASS (Russia). Modern GNSS modules improve accuracy, dependability, and fix times by allowing them to support multiple constellations simultaneously, particularly in difficult-to-reach places like dense forests. [11]

GNSS tags determine location by calculating position within the tag itself. GNSS tags need line-of-sight to at least 4 satellites and the tags use this information to calculate a time and location. GNSS tags without any transmission ability must be retrieved by researchers to obtain data. [12]

Advantages

- **Global coverage and independence :** operates worldwide without requiring any field infrastructure like base stations or receivers.
- **High positioning accuracy :** provides excellent spatial resolution (5-10 meters), suitable for detailed movement studies.
- **Autonomous operation :** eliminates reliance on external networks that could fail, ensuring reliability for long-term studies.

Disadvantages

- **High energy consumption :** leads to shorter battery life or heavier collars compared to other tracking technologies.
- **Poor performance in obstructed environments :** requires clear line-of-sight to satellites, fails in dense canopy or underground.
- **Data retrieval challenges :** store-on-board tags must be physically recovered to access data, risking total loss if the animal is not recaptured.

4.4.1.2 Argos satellite system

The Advanced Research and Global Observation Satellite system is a satellite-based network for location and data collection run by an international group. It uses a group of

polar-orbiting satellites flying about 850 km above Earth. These satellites pass over the poles and cover the whole planet as Earth turns.

Animal-mounted Argos transmitters (Platform Transmitter Terminal) send out radio signals at 401.65 MHz, which include an ID code and sometimes sensor data. When a satellite is overhead, it picks up these signals, records them, and measures the Doppler shift, the change in frequency caused by the movement between the satellite and the transmitter. By observing how the frequency changes as the satellite moves closer and then farther away, ground stations can triangulate the transmitter's location.

The satellite sends the data to ground stations, which use the Doppler information to estimate the transmitter's location. Unlike GNSS, where the tag receives signals from satellites to find its own position, Argos tags only send signals. This makes them simpler and uses less power. Still, getting a good location depends on the tag sending a signal when a satellite is nearby. Accuracy can vary based on factors like how many messages are sent during a pass, the angle of the satellite's path, and how steady the signal is. [13]

Advantages

- **True global coverage :** satellites provide coverage across the entire planet, including poles and oceans where no other infrastructure exists.
- **No device retrieval required :** automatically transmits data to satellites, ensuring data recovery even if the animal dies or is lost.
- **Long battery life for transmission :** consume less power than GNSS receivers by only transmitting brief packets, allowing for smaller batteries.

Disadvantages

- **Lower location accuracy :** doppler shift calculations provide only approximate locations (250m to km), unsuitable for fine-scale studies.
- **Intermittent and unpredictable data transmission :** depends on satellite overpasses and clear line-of-sight, creating irregular data gaps.
- **High operational costs :** requires expensive ongoing subscription fees based on data volume and satellite usage.

4.4.1.3 Geolocation by Light

Geolocation by Light devices, also called Global Location Sensor or light-level loggers, are small tools that estimate an animal's location using ambient light instead of satellite signals. They work on a simple idea: sunrise and sunset times change in a predictable way with longitude and latitude. The GLS device has a light sensor and a clock that records light levels throughout the day. [14]

By identifying the precise times when light levels transition from dark to daylight and back again, the device estimates the animal's location. Day length is linked to latitude. Solar noon helps show longitude, since local noon happens at different times depending on how far east or west the logger is. After the tag is collected, special algorithms turn the light data into geographic coordinates. These sensors are not very precise (usually accurate within 100-200 km) because weather and animal behavior can affect the readings. Even with lower accuracy, GLS are very small and light (often under 1 gram), so they work well for animals that are too small for GNSS or satellite tags. [15]

Advantages

- **Extremely lightweight and low cost :** simple components allow for tags weighing less than 1g with no subscription fees.
- **Very long battery life :** passive recording of light levels consumes minimal power, enabling multi-year deployments.
- **Infrastructure-free solutions :** operates autonomously in any environment without external equipment.

Disadvantages

- **Poor location accuracy :** coarse positioning (100-200 km) makes it unsuitable for detailed habitat or movement analysis.
- **Mandatory tag retrieval :** data is stored internally, requiring physical recapture of the animal to access information.
- **Environmental and behavioral interference :** weather, vegetation, or burrowing behavior can compromise light readings and position estimates.

4.4.1.4 LoRaWAN

Long Range Wide Area Network is a low-power wireless protocol made for Internet of Things applications. The system has two main parts: LoRa transmitters attached to animals and fixed LoRa gateways placed around the study area. LoRa tags use a special modulation method called Chirp Spread Spectrum (CSS), which enables long-range communication with low power consumption. The tags send data packets at regular intervals, including location, sensor readings, and device ID. LoRaWAN can also be used for range-based positioning, allowing the LoRa network to determine location without external positioning systems. [16]

In this approach, LoRaWAN tags transmit radio signals that are received by multiple fixed LoRa gateways in the study area. With this method, LoRaWAN tags transmit radio signals that are received by several fixed LoRa gateways in the study area. The system determines the animal's position by analyzing the details of these signal transmissions.

There are two main ways to do this: Received Signal Strength Indicator, which estimates distance by how much the signal weakens, and Time Difference of Arrival, which works by measuring the exact time a signal reaches each synchronized gateway. TDoA is more accurate but needs gateways with precise timing and known locations. The system uses trilateration or multilateration algorithms, which are similar to GNSS but work in reverse. [17]

Advantages

- **Exceptional battery life :** extremely low power consumption enables deployment periods of years on small batteries.
- **Leverages existing or shared infrastructure :** can utilize existing networks in agricultural or smart city regions without new equipment.
- **Real-time data and no tag retrieval :** transmits data continuously when in range, allowing immediate monitoring without physical recapture.

Disadvantages

- **Mandatory dense infrastructure deployment :** requires expensive installation and maintenance of gateway networks in remote areas.
- **Limited positioning accuracy :** TDoA/RSSI methods provide coarse resolution (20-200m) compared to GNSS.
- **Complete tracking failure outside coverage :** animals moving beyond gateway range become untraceable with no data collection.

4.4.1.5 Radio telemetry

Radio telemetry using Very High Frequency signals is a reliable way to track wildlife. Small VHF radio transmitters are attached to animals, sending out regular radio pulses or continuous signals at unique frequencies for each animal. [18]

Researchers find the tagged animals by using portable directional antennas connected to radio receivers that detect the signals. There are two main ways to track the animals:

- Manual tracking involves moving through the field with handheld equipment, turning the antenna to find the strongest signal, and then using readings from different spots to figure out the animal's location.
- Automated fixed station tracking uses stationary antenna arrays set up at key spots in the study area. These antennas continuously listen for tagged animals that come within range.

Fixed stations can be equipped with data loggers that record signal presence, strength, and timing, creating detection histories. The technology is robust, with transmitters simple, lightweight, and capable of operating for months or years on small batteries because they emit only radio pulses rather than performing complex calculations or satellite communications. [19]

Advantages

- **Extremely long battery life and lightweight :** simple electronics allow for tiny tags that operate for years.
- **Excellent signal penetration in dense habitats :** VHF signals work well in forests, terrain, and burrows where satellite signals fail.
- **Low cost and no operational fees :** inexpensive equipment with no subscription costs.

Disadvantages

- **Infrastructure dependency for automated tracking :** continuous data requires installing and maintaining fixed antenna stations.
- **Labor-intensive manual tracking :** requires physically traveling to the field and triangulation, yielding only periodic snapshot data.
- **No precise positioning data :** provides relative bearings rather than coordinates, introducing significant positioning error.

4.4.1.6 Radio-Frequency Identification (RFID)

Radio-frequency identification (RFID) is a wireless technology using electromagnetic fields to identify and track tags. An RFID system consists of small tags and fixed readers. There are two main types of tags: The first type, passive tags, does not have a battery.

Instead, they draw power from the electromagnetic field generated by the reader when they are nearby. In contrast, active tags have a battery, which allows them to be detected from farther away.

When a RFID reader sends out radio waves, any tag in range replies with its unique identification code. The reader records this detection and the time. This technology uses different frequency bands.

- **Low frequency** (125-134 kHz) for very short ranges and good penetration through water/tissue.
- **High frequency** (13.56 MHz) covers moderate distances with good data rates.
- **Ultra-high frequency** (860-960 MHz) reaches longer distances but struggles with biological materials.

RFID provides presence/absence data rather than continuous positioning—researchers know when and where an animal passed a reader station, but have no information about movements between stations. The system is especially useful for monitoring specific behavioral events, such as nest visits, use of wildlife corridors, visits to feeding or watering stations, tunnel passages, or entries/exits from burrows or roosting sites. [20]

Advantages

- **Extremely lightweight and long-lasting tags :** passive tags require no battery, active tags last years with minimal power use.
- **Automated, high-resolution temporal data :** continuously records precise time-stamps for every detection event without human presence.
- **Low operational costs and maintenance :** autonomous operation with no subscription fees and robust hardware.

Disadvantages

- **Mandatory infrastructure at detection points :** requires fixed readers and power sources at every location of interest.
- **Extremely limited detection range :** requires animals to pass within centimeters or a few meters of the reader.
- **No spatial positioning data :** provides only presence/absence at specific points with no information on movement between them.

4.4.1.7 Biologgers with dead reckoning

Biologgers that use dead reckoning are self-contained devices that track movement by combining data from several onboard sensors, without needing external positioning signals. Dead reckoning is a navigation method that finds the current position by starting from a known point and updating it based on measured speed, direction, and time. These devices usually include tri-axial accelerometers, tri-axial magnetometers, gyroscopes, and, sometimes, depth, pressure, or speed sensors. The logger collects data from these sensors at high rates, capturing every movement, turn, and change in posture. Algorithms then use this information to map out the path in three dimensions. One challenge with dead reckoning is cumulative error: small measurement errors accumulate over time, so the estimated path can drift from the true position. [21]

Advantages

- **Extremely high temporal and spatial resolution :** captures fine-scale movement and behavioral details invisible to other systems.
- **Functions in GPS-denied environments :** operates independently of external signals, working underground or underwater.
- **Detailed behavioral classification :** multi-sensor data allows for automatic classification of specific activities and behaviors.

Disadvantages

- **Mandatory tag retrieval :** stores data internally, requiring physical recovery of the device.
- **Cumulative positioning error (drift) :** inherent sensor inaccuracies compound over time, degrading location accuracy without external correction.
- **Complex data processing requirements :** requires sophisticated algorithms and expertise to convert raw sensor data into movement paths.

4.4.1.8 Bluetooth/BLE

Bluetooth Low Energy tags are short-range wireless tracking devices that use energy-efficient Bluetooth technology to send identification to nearby receivers. BLE tags, also known as beacons, are small battery-powered transmitters attached to animals. They regularly send out radio signals with a unique identifier and, sometimes, additional sensor data. These signals operate in the 2.4 GHz frequency band and can usually be detected from a few meters to 30-100 meters in open areas. However, the range drops a lot in dense vegetation or when there are obstacles. [22]

The system uses fixed BLE receivers placed around the study area, or more often now, smartphones and mobile devices with special apps that can detect and report BLE tag signals. When a receiver picks up a tag within range, it records the tag's unique ID, the time, and the received RSSI. Some systems use the received RSSI to estimate how close the tag is to the receiver, but this method is not very accurate. With Bluetooth® 6.0, the Bluetooth SIG introduced Channel Sounding, which improves how Bluetooth Low Energy devices measure distance and detect presence.

Advantages

- **Extremely low power and lightweight :** energy-efficient design allows for tiny tags with long battery life.
- **Leverages existing infrastructure and crowd-sourced detection :** utilizes smartphones and existing networks for low-cost detection coverage.
- **Low cost and easy deployment :** inexpensive hardware using standard protocols with no subscription fees.

Disadvantages

- **Severe infrastructure dependency and range limitations :** requires dense receiver networks due to short detection range (10-50m).
- **Unreliable detection in field conditions :** signals are easily blocked by vegetation and terrain, creating frequent data gaps.
- **Only presence/absence data with poor spatial resolution :** reveals proximity to receivers but lacks precise positioning or trajectory data.

4.4.1.9 WiFi

WiFi-based tracking uses small WiFi-enabled transmitters that communicate with WiFi access points to detect and monitor locations. Each WiFi tag has a radio transmitter that works on the 2.4 GHz or 5 GHz frequency bands, a microcontroller, and a battery. These tags regularly send out WiFi probe requests and beacon frames, or connect to nearby WiFi access points, sharing their unique MAC address and sometimes additional sensor data.

Fixed Wi-Fi receivers or access points deployed throughout the study area detect these signals when tags come within range. The system records the tag's unique identifier, timestamp, received RSSI, and the detecting access point's known location. More refined systems use Wi-Fi triangulation or trilateration, where multiple access points detect the same tag simultaneously and use RSSI measurements or ToF calculations to estimate the tag's position with accuracy ranging from 1-15 meters, depending on infrastructure density as well as algorithms. Wi-Fi positioning can also leverage existing Wi-Fi infrastructure in urban areas, campuses, research facilities, or visitor centers. Some systems integrate with existing network infrastructure for data transmission and power, while others use standalone battery-powered or solar-powered Wi-Fi receivers. [23]

Advantages

- **Leverages ubiquitous existing infrastructure :** utilizes extensive networks already present in urban or developed areas.
- **Moderate positioning accuracy with triangulation :** offers better accuracy (1-15m) than simple beacons when multiple access points are available.
- **Bidirectional communication capability :** allows for remote configuration and data retrieval without physical recapture.

Disadvantages

- **Extremely high power consumption :** complex protocols drain batteries quickly, limiting deployment to days or weeks.
- **Critical infrastructure dependency and limited range :** signal attenuates heavily in natural habitats, requiring dense infrastructure.
- **Signal interference and reliability issues :** crowded frequency bands in developed areas cause detection failures and reduced range.

4.4.1.10 Ultra Wide Band

Ultra-Wideband is a short-range, high-precision wireless technology that sends data over a very wide frequency range, usually from 3.1 to 10.6 GHz, using low power and short pulses. UWB tracking systems use small UWB tags and several fixed UWB anchors placed at known locations. Unlike narrowband technologies such as VHF or Wi-Fi, which transmit continuous signals on a single frequency, UWB sends billions of very short pulses across a wide range of frequencies.

This pulse-based approach enables highly accurate ToF measurements: anchors detect tag transmissions and precisely measure the time it takes radio signals to travel from tag to anchor. Since radio waves travel at the speed of light, these timing measurements can be converted to distances with centimeter-level accuracy. When three or more anchors detect a tag simultaneously, the system uses trilateration to calculate the tag's precise

three-dimensional position in real time, typically achieving accuracy of 10-50 centimeters under optimal conditions. UWB signals have excellent penetration through obstacles and are highly resistant to multipath interference relative to narrowband systems, making positioning more reliable indoors and in cluttered environments. The technology supports very high data rates and can track numerous tags simultaneously with minimal interference. [24]

Advantages

- **Exceptional positioning accuracy :** provides centimeter-level precision via Time-of-Flight measurements.
- **High update rates and real-time tracking :** captures rapid movements and behavioral sequences with high frequency.
- **Robust signal performance and multi-tag capacity :** resistant to interference and multipath effects, allowing simultaneous tracking of many tags.

Disadvantages

- **Extreme infrastructure dependency and deployment complexity :** requires dense networks of precisely synchronized anchors.
- **Very limited detection range :** restricted to small, controlled areas (10-50m) like enclosures or indoor facilities.
- **High power consumption and cost :** expensive equipment and high energy use limit battery life and increase tag size.

5 Firmware design

The firmware design phase allows for a comprehensive view of how the system works and its interactions. It is a technical blueprint that bridges the gap between theoretical requirements and physical implementation. This chapter details the architecture of this thesis.

> As a reminder, this project is a second version of an existing collar. The existing firmware will serve as the basis for adding new features. Similarly, the hardware provided for this thesis is based on the first version.

5.1 Requirements specification

In this part, the requirements will be specified. The final system must fulfill these specific requirements. Some of them are already satisfied in the first version, some requirements are totally new and finally, some will require updates to the existing system.

5.1.1 Functional requirements

These define the core actions the system must perform.

1. **Acoustic recording :** The system must capture audio using a I2S MEMS microphone (TDK T5848). ➡️ New requirement
2. **Intelligent triggering :** The system shall utilize Acoustic Activity Detection to wake the processor and initiate saving sound samples only when sound exceeds a configurable threshold. ➡️ New requirement
3. **Continuous recording :** Once the first wake signal is received, the system must record until the sound levels fall below the trigger levels. ➡️ New requirement
4. **BLE proximity detection :** The device shall passively scan for nearby collars and log their informations to log files filtering it using manufacturer data. ➡️ New requirement
5. **Start recording when device nearby :** The detection of a nearby device should start the recording of the audio stream. ➡️ New requirement
6. **Persistent storage :** All audio captures and system logs must be stored on a removable SD card using the FATFS file system. ➡️ Already satisfied
7. **Remote release mechanism :** The system must include a GPIO-controlled actuator that can be triggered remotely via Bluetooth to release the collar. ➡️ Already satisfied
8. **Wireless monitoring :** The system shall provide a custom GATT service to allows an mobile app to monitor status, trigger the collar release and change the settings. ➡️ Requires existing system update

5.1.2 Performance requirements

These specify the quality and timing constraints of the system.

1. **Audio quality :** Audio must be sampled at a frequency of 16 kHz in a 16 bits resolution. ➡️ Requires existing system update
2. **Field autonomy :** The system must be capable of autonomous operation for a minimum of 12 days. ➡️ Requires existing system update
3. **Power efficiency :** The system shall maintain the microphone in a sleep mode and the MCU in deep sleep during periods of silence. ➡️ New requirement
4. **Tail recording :** The system shall continue recording for a defined tail period after the last detected sound that exceeds the limit. ➡️ New requirement

5.1.3 Interface requirements

These define how the system interacts with hardware and external firmware.

1. **BLE GATT service :** The system must implement the proprietary Speak No Evil Service with characteristics for commands, status updates, and parameter configuration. ➡️ Requires existing system update
2. **Debugging interface :** The firmware must support debugging and logging via the SEGGER Real Time Transfer protocol. ➡️ Requires existing system update
3. **Hardware configuration :** The system must support run-time configuration of microphone sensitivity via BLE. ➡️ New requirement

5.1.4 Operational requirements

These define the conditions under which the system must remain reliable.

1. **Automatic fail-safe :** The system must automatically halt recording and enter a *Power Saving* mode if the battery is low or the SD card is full. ➡️ Already satisfied
2. **State persistence :** Some of the important data must be stored in protected RAM sections to survive firmware reboots. ➡️ Requires existing system update
3. **Error handling :** The system must detect and report hardware issues. ➡️ Requires existing system update
4. **Recoverability :** The collar release command must remain functional even after the system has reached a *Disk Full* or *Low Battery* state. ➡️ Already satisfied

5.2 System overview

The *nRF Monkey* system is an audio recorder designed for primates monitoring. It integrates audio capture, BLE communication, and proximity detection into a single low-power embedded device.

The Figure 7 allows for a global view of the system, these components and their interactions. The nRF monkey system contains as main component the nRF54L15 System-

on-Chip, the T5848 MEMS microphone, a SD card and the release system. All components interact with the SoC. Thy system interacts with the mobile control app and with nearby devices using Bluetooth Low Energy.

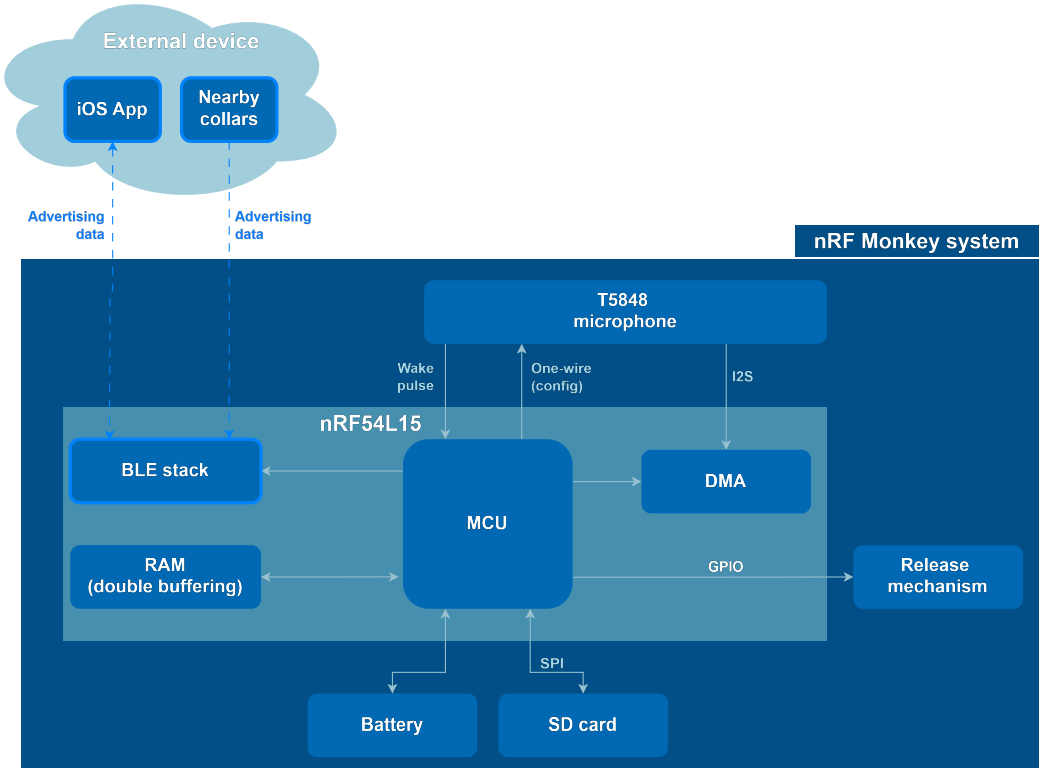


Figure 7: *nRF Monkey* system overview

The main features are explain below :

- **System core:** Built on the nRF54L15 System-on-Chip using Zephyr RTOS to manage concurrent tasks.
- **Audio capture:** Use a TDK T5848 digital MEMS microphone. It captures 16 kHz audio via the I2S protocol. It employs Acoustic Activity Detection on the microphone to wake the processor from deep sleep only when sound level meets the requirements.
- **BLE proximity:** Passively scanning for nearby collars to log social interactions. It alternates these scanning phase with advertising phases to allow other devices to detect it (see Section 5.6.1 for localization technology choice explanation) and to be able to connect to it for monitoring using the mobile application.
- **Data integrity:** Uses double-buffering for audio and proximity logs to ensure zero data loss during SD card write cycles.
- **Storage:** Audio and proximity data are stored in FATFS format on an SD card via SPI.
- **Power management:** Optimizes battery life by keeping the microcontroller in deep sleep while the microphone monitors the environment.

- **Remote control:** A custom GATT service allows an mobile app to monitor status and trigger the collar release mechanism.
- **Release mechanism:** A mechanical release mechanism allows to remotely removing the collar using a BLE command. A nylon thread holds the collar strap closed. It is then burned using a copper wire. This avoids having to sedate the monkey to remove the collar once the recording session is over.

The Figure 8 represent a simplified use case diagram of the recording procedure.

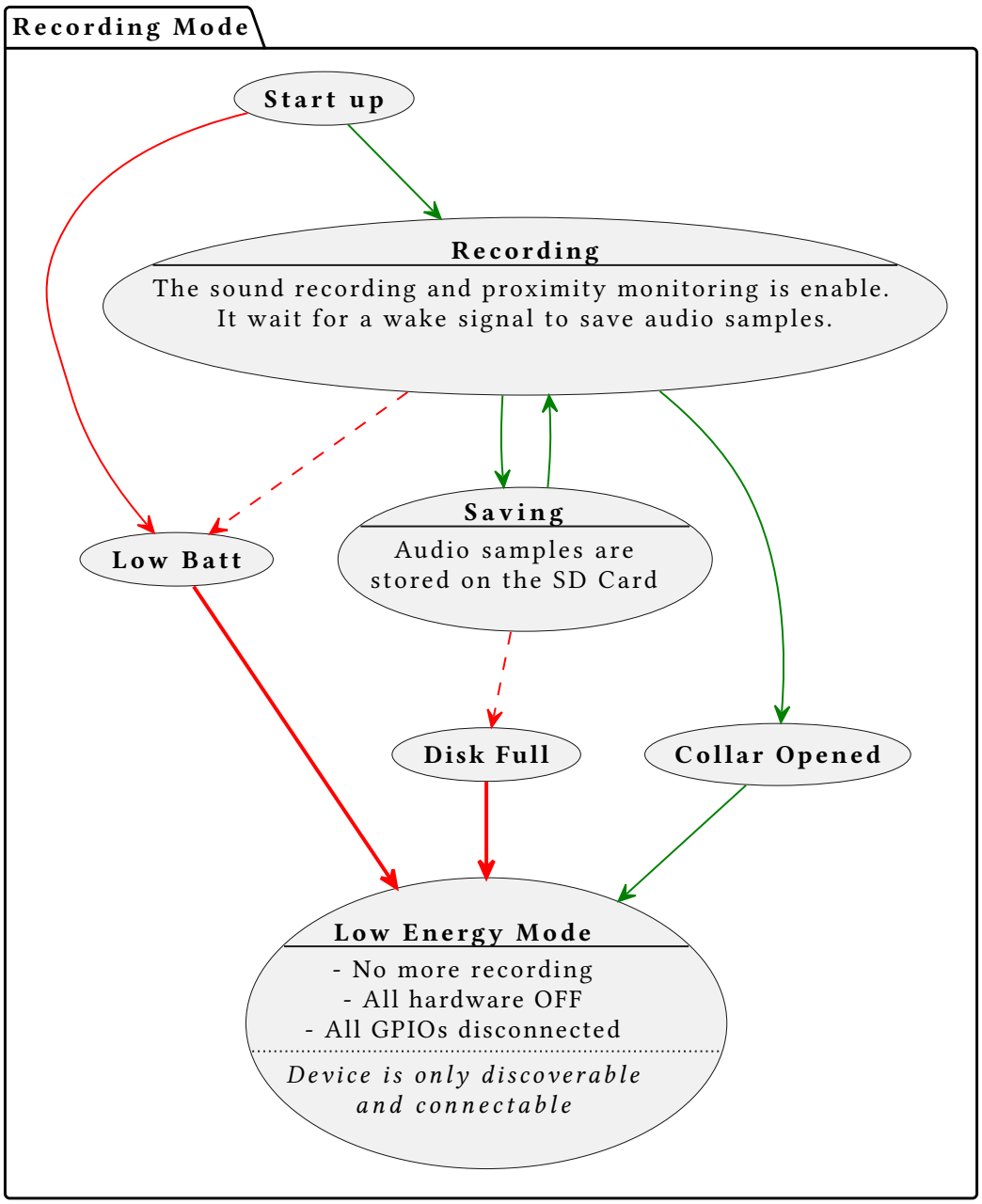


Figure 8: Record use case diagram

In the scope of this project, the terms recording and saving are used to talk about the recording phases of audio samples.

- **Recording :** This is the standby mode. The system is logically in record mode. It is waiting for a wake signal from the microphone to start saving data.
- **Saving :** This is the active capture phase. Once the wake signal is received, the microphone starts. The audio data flows through the I2S interface, and the SoC actively writes that data to the SD card.

5.3 System decomposition

The system has been split in functional block. The system decomposition is :

1. *Power supply* block
2. *Recorder* block
3. *Storage* block
4. *BLE* block
5. *Programmer* block

5.3.1 Power supply block diagram

The power supply block manage the power for the device. It regulates the battery voltage to a stable 3.3V for the digital logic and monitors the battery level in real-time.

Figure 9 illustrates the block diagram.

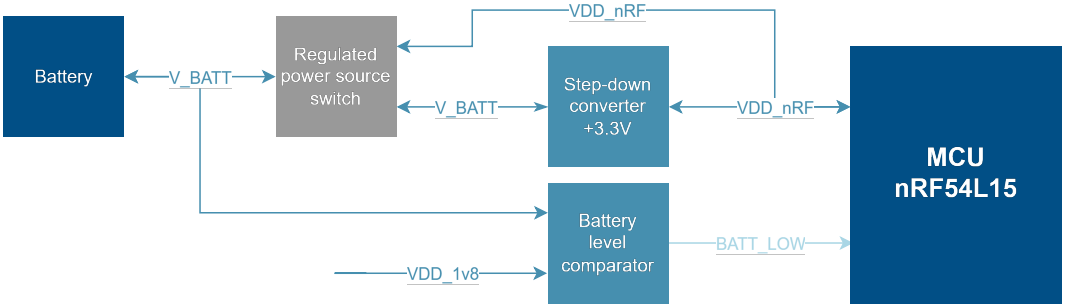


Figure 9: Power supply block diagram

The power supply block implement the following features :

- **Field autonomy :** Manages the 3.3V regulation to ensure the system operates for at least 12 days.
- **Automatic fail-safe :** Uses the battery comparator to trigger the low power mode if voltage is too low.

5.3.2 Recorder block diagram

The recorder block is responsible for the capture of audio data. It manages the 1.8V power requirements of the T5848 microphone and handles the level shifting necessary for communication with the 3.3V microcontroller.

Figure 10 illustrates the block diagram.

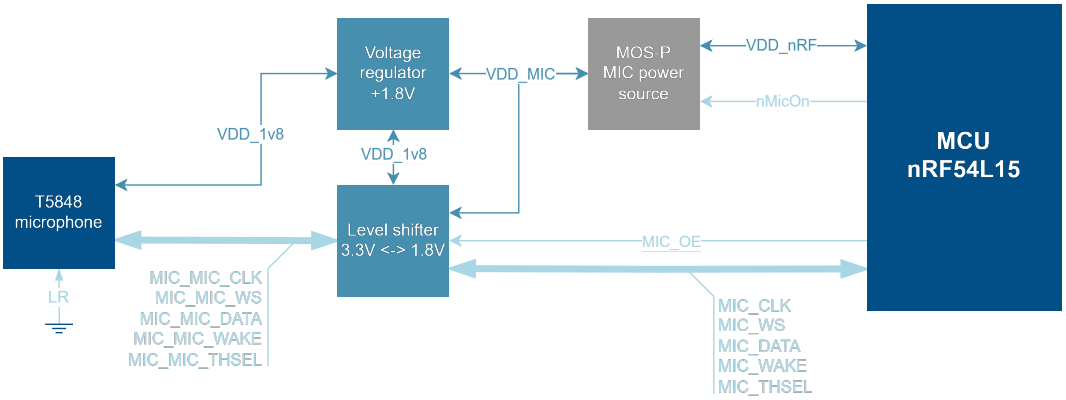


Figure 10: Recorder block diagram

The recorder block implement the following features :

- **Acoustic recording :** Captures high-quality audio at 16 kHz and 16-bit resolution via I2S.
- **Intelligent triggering :** Utilizes the AAD feature to wake the SoC only when a sound threshold is met.
- **Continuous & tail recording :** Manages the logic to keep recording after a trigger until the tail period expires.
- **Power supply :** Handles 1.8V level shifting to configure microphone sensitivity at runtime.

5.3.3 Storage block diagram

The storage block handle all the system storage : SD, RAM and flash.

Figure 11 illustrates the block diagram.

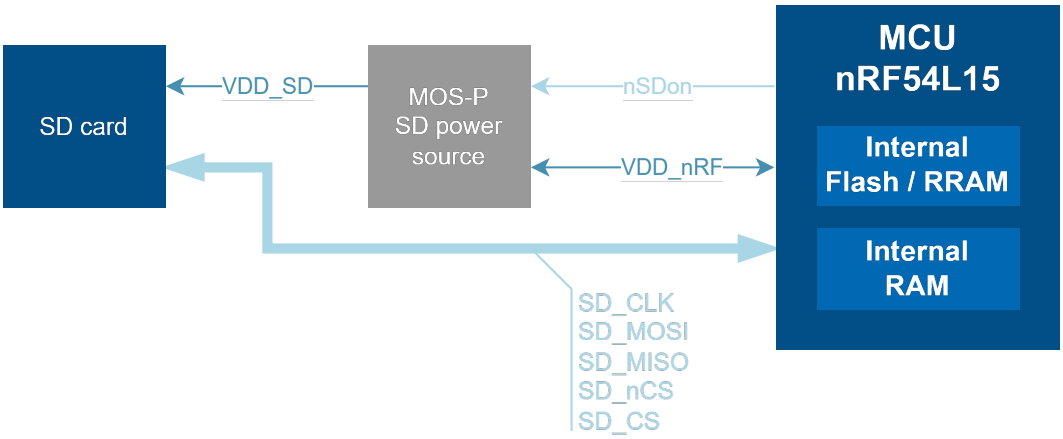


Figure 11: Storage block diagram

The storage block implement the following features :

- **Persistent storage :** Writes audio and proximity files to the SD card via SPI.
- **State persistence :** Uses protected RAM and flash to keep session data safe during resets.

- **Data integrity :** Manages the double-buffering mechanism to prevent data loss during SD write cycles.

5.3.4 BLE block diagram

The BLE block handles all wireless interactions, including proximity detection and remote monitoring. It manages the 2.4 GHz radio to scan for nearby collars using manufacturer data filters and hosts a custom GATT service (SNES) for communication with a mobile application.

Figure 12 illustrates the block diagram.

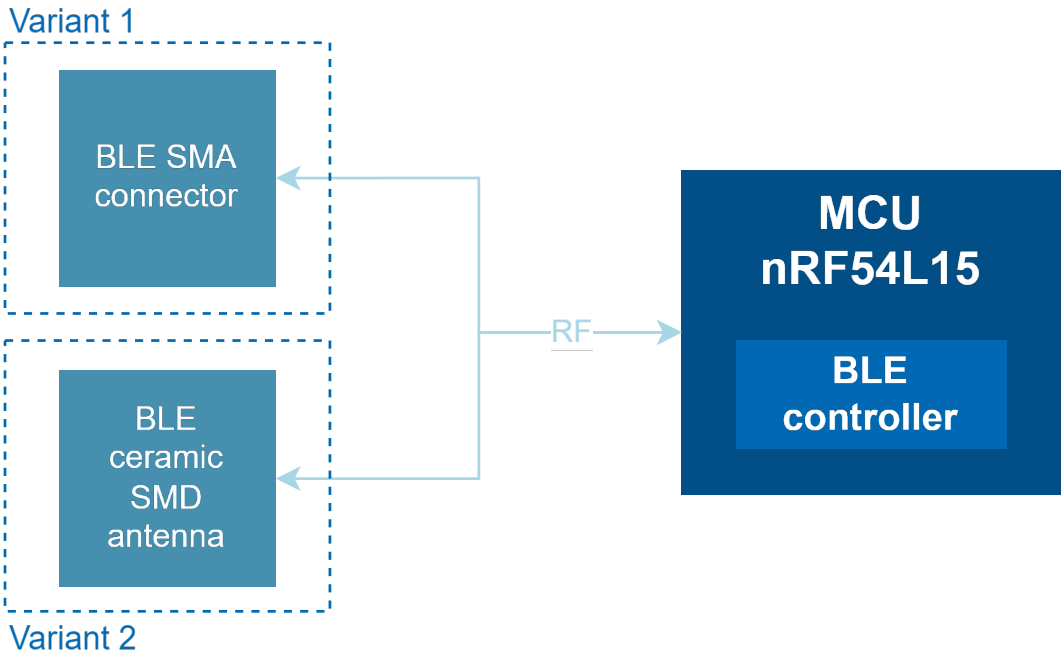


Figure 12: BLE block diagram

The BLE block implement the following features :

- **Proximity detection :** Scans for nearby devices and initiates recording when a peer is detected.
- **Wireless monitoring :** Implements the SNES GATT service for status checks and remote configuration.
- **Remote release :** Ensures the collar release command works even if the SD card is full or battery is low.

5.3.5 Programmer block diagram

The programer block contains all the interface required to flash and debug the system. A user LED is also available.

Figure 13 illustrates the block diagram.

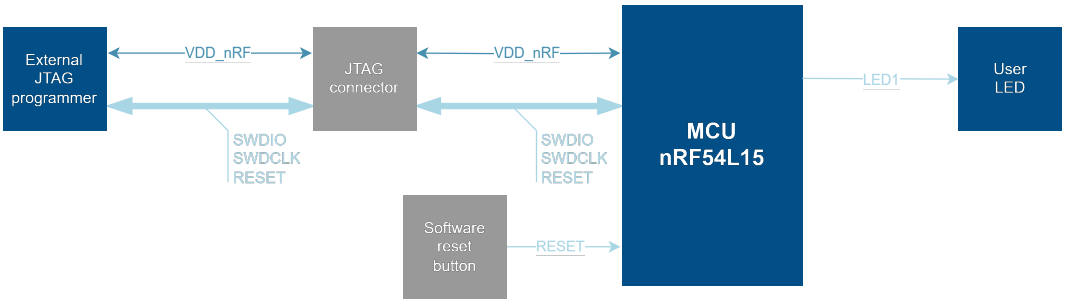


Figure 13: Programmer block diagram

The programmer block implement the following features :

- **Debugging interface :** Provides the JTAG and RTT interface for real-time firmware tracing.
- **Error handling :** Allows to monitor hardware status and detect firmware issues.

5.4 User interface

In the production collar version, there will be no physical user interface, such as a screen, LED, or buttons. To monitor and control the system, a wireless user interface is implemented via a custom Bluetooth Low Energy service. This allows a mobile application or another device to act as a remote console for monitoring system status, adjusting recording parameters, and triggering the collar release. This application already exists. The figure Figure 14 represents the mobile application.

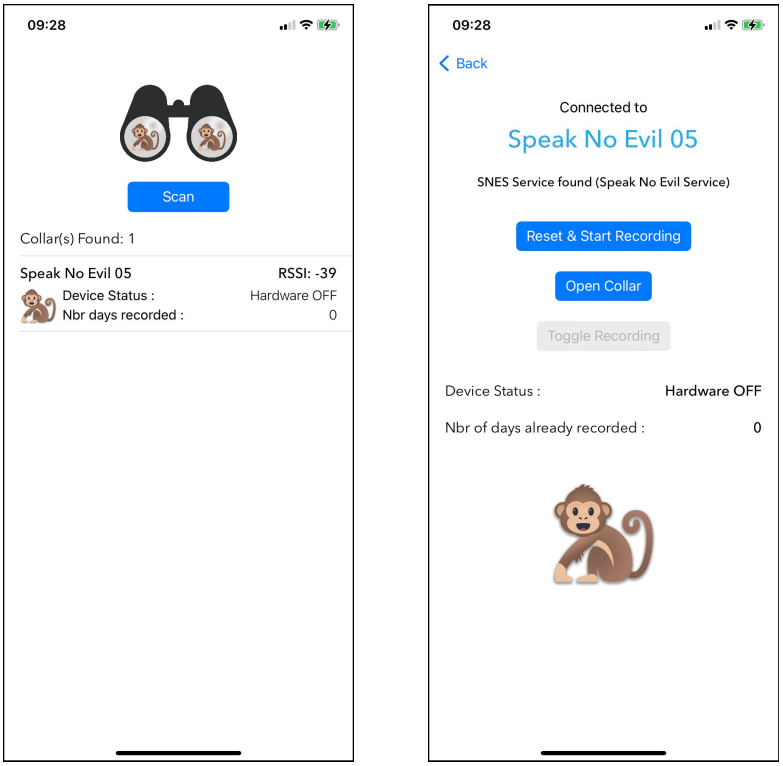


Figure 14: iOS app preview in scanning mode and connected to a collar.

5.5 Firmware architecture

The system is built on Zephyr RTOS. Multiple threads are used to make this system work. It has 6 different threads. They are described in the Figure 15.

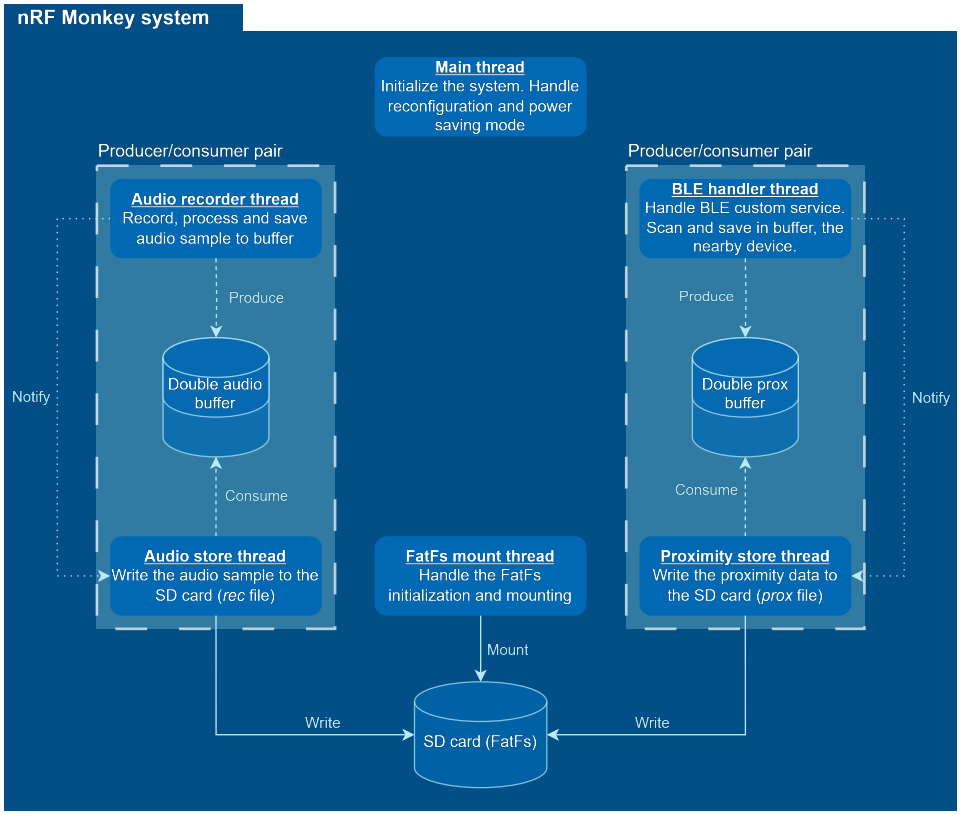


Figure 15: Threads overview

Synchronization between threads is performed using semaphores. For audio and device proximity recording, a producer-consumer pattern is used. This pattern, which uses two buffers (ping-pong), allows the producer to record without interruptions. Writing to memory is delegated to the consumer. This pattern is explained in detail in the Section 8.3.8.3.

Interrupts are used for all asynchronous events, particularly for capturing the microphone's wake signal. During the recording phase, the system is in deep sleep. It is awakened by a pulse wake from the microphone. It will then begin saving data onto the SD card until the configured audio conditions are no longer met. The recorded data is temporarily stored in RAM before being written to the SD card.

The BLE handler provides a custom GATT profile (SNES) designed specifically for this project. It acts as the communication bridge between the collar and the mobile application. This serves as the User Interface.

5.6 Technology and components selection

Many of the components or technologies are the same as in version 1. They will not be discussed here. The majority of the components were chosen before the start of this project by a collaborator from the HES-SO but the choice of the main components and technologies is discussed in this chapter.

5.6.1 Positioning technology selection

Ten technologies were analyzed, showing their advantages and disadvantages in the Section 4.4. To allow objective comparison and selection of the best technology for this project, specific criteria and evaluation metrics were defined. These criteria and metrics are detailed in the following section. A summary table of the evaluated technologies against these criteria was created to facilitate the selection of the most suitable technology.

To be able to further choose the best solution for this project, nine criteria have been defined based on the project's needs. A weight has also been assigned for each criterion to refine the selection based on the importance of the criteria. All those criteria and weights are evaluated on a scale of 1 (poor) to 5 (excellent). For criterion 6, 5 corresponds to yes and 1 to no. For the last criterion, the value is 5 if it does not need new hardware and 1 if it needs new hardware.

1. **Localization accuracy :** The precision with which the system can determine an animal's position or proximity. Weight = 3
2. **Power consumption :** The energy required by the device to run continuously. Weight = 5
3. **Weight and size :** Total mass and volume of the device, including battery. Weight = 5
4. **Cost :** Total cost per tag and per base station. Weight = 3
5. **Infrastructure requirements :** The amount and type of external infrastructure (e.g., base stations, gateways) needed. Weight = 3
6. **Interaction detection capability :** Ability to detect proximity between animals without infrastructure. Weight = 3
7. **Robustness and environmental resistance :** Device performance under field conditions (weather, vegetation, terrain). Weight = 2
8. **Data storage and processing capacity :** Onboard memory and processing capacity for offline operation. Weight = 2
9. **Requires new hardware to function :** Can the technology be used with existing hardware? Weight = 4

The Table 1 evaluates for each technology every criterion based on the existing literature. In the last column, the overall weighted score is computed using the formula : score = $\frac{\sum_{k=1}^{9} w_i * c_i}{\sum_{k=1}^{9} w_i}$. It allows for comparison of the different technologies across all criteria.

Technology	1 Location accuracy	2 Power consumption	3 Weight and size	4 Cost	5 Infrastructure requirements	6 Interaction detection	7 Robustness & environ	8 Data storage & processing	9 Requires new hardware	Overall weighted score
Weighting	3	5	5	3	3	3	2	2	4	-
BLE	●●●○○	●●●●●	●●●●●	●●●●○	●●●○○	●●●●●	●●●○○	●●●○○	●●●●●	**4.2**
RFID	●○○○○	●●●●●	●●●●●	●●●●○	●●●○○	●○○○○	●●●●○	●●●○○	●●●●●	**3.6**
Geolocation by light	●○○○○	●●●●●	●●●●●	●●●●●	●●●●●	●○○○○	●●●○○	●●●●●	●○○○○	**3.5**
Argos satellite system	●●●●○	●●●○○	●●●●○	●●○○○	●●●●●	●○○○○	●●●●●	●●●●●	●○○○○	**3.2**
LoRaWAN	●●●○○	●●●●○	●●●●○	●●●●○	●●●○○	●○○○○	●●●●○	●●●●●	●○○○○	**3.2**
UWB	●●●●○	●●●○○	●●●●○	●●●○○	●●●○○	●●●●●	●●●○○	●●●○○	●○○○○	**3.2**
GNSS	●●●●●	●●○○○	●●●○○	●●●○○	●●●●●	●○○○○	●●●●●	●●○○○	●○○○○	**2.8**
Radio telemetry	●●●○○	●●●○○	●●●●○	●●●○○	●○○○○	●○○○○	●●●●●	●●●●●	●○○○○	**2.8**
Biologgers with dead reckoning	●●●●○	●●●○○	●●●○○	●●○○○	●●●●●	●○○○○	●●●●○	●●●○○	●○○○○	**2.8**
WiFi	●●●●○	●●○○○	●●●○○	●●●●○	●●○○○	●○○○○	●●●○○	●●●○○	●○○○○	**2.5**

Table 1: Comparison of location technologies with weighted scoring

5.6.1.1 Conclusion

Fours technologies stands out from the crowd :

❌ **GNSS :** The most logical solution would have been to use GNSS to locate the animals. This solution was evaluated during preliminary work [2]. It was found that embedding a GNSS module into the collar would be challenging both mechanically (positioning relative to the satellite, antenna integration and system protection) and in firmware (managing the module's operation and power consumption). Tests showed that the module consumed 4 to 17% of the current collar's battery capacity to acquire only four positions per day. By applying battery optimization strategies, it would be possible to reduce the module's power consumption. However, four positions per day are insufficient to properly study the relationships and interactions among individuals. The results of this preliminary study are reflected in the score this technology achieved in the comparative design.

❌ **Geolocation by light :** The Geolocation by light could have been a good solution. It consumes little power and the sensors are very light and compact (~1g). They are also inexpensive and require no additional infrastructure. However, all these advantages mask a major drawback. These sensors have very low accuracy (~100-200 km). This lack of precision makes it impossible to use this data to study interactions between individuals.

❌ **RFID :** The RFID method could be a good solution for this project. The operating costs are limited, even though it requires installing readers in the field to locate the animals. However, the solution currently available in the collar can only emulate a RFID chip, it cannot read. Therefore, it would not be possible to detect interactions between individuals, only their passage near a reader. Since the range of a RFID chip is very short, the animal would have to pass very close to the reader (less than 0.5m). This makes implementing this system in the wild difficult due to this limited range. This solution would have been suitable for indoor use or for monitoring a passageway such as a burrow or nest.

✅ **BLE :** The Bluetooth Low Energy solution allows for the detection of interactions between individuals and their positions relative to beacons placed in the field. The distance from these anchors can be estimated using the Received Signal Strength Indicator or Channel Sounding. This solution incurs additional costs for the field anchors that need to be installed. However, the collar already uses BLE for control via the application, so no hardware changes are required. The signal range can be affected by field conditions. During the first field use of the collar, a range of 10 to 15 meters was observed. This solution also allows for operation in low-power mode since it does not need to run continuously.

As shown by the evaluation score, this solution is the most suitable for the project's needs. The BLE will be used to locate the monkeys and analyze their interactions.

5.6.2 SoC selection

On the first collar version, an *nRF5340* was used as the microcontroller. It was decided to update the microcontroller. The chosen one is an *nRF54L15*, a newer-generation chip. This chapter will detail the advantages and disadvantages of this change.

The nRF54L Series is the latest generation successor to the nRF52 series. Within the context of this project, this capability is particularly interesting. One of the major objectives of this project is to offer the lightest and most compact collar. This implies optimizing power consumption to minimize battery size.

The Table 2 compares the old microcontroller to the new one based on the most important features. It helps to understand the criteria that led to this model's selection.

Feature Category	nRF5340	nRF54L15
Architecture	Dual Arm Cortex-M33 (App & Network cores)	Single Arm Cortex-M33 (128 MHz) + RISC-V coprocessor
Max Memory	1.25 MB Flash 576 KB RAM (Combined)	1.5 MB NVM (RRAM) 256 KB RAM
Deep Sleep	$\sim 1.1\mu$A (System OFF)	$\sim 0.6\mu$A
Radio TX (0 dBm)	$\sim$ 3.2 mA	$\sim$ 4.8 mA
Radio RX (1 Mbps)	$\sim$ 2.6 mA	$\sim$ 3.2 mA
Bluetooth	Bluetooth 5.4	Bluetooth 6.0
LE Audio	Yes	No
Low-Power Feature	Standard Real-Time Counter	Global RTC for wake-up from System OFF

Source: https://www.nordicsemi.com/Products/Wireless/Bluetooth-Low-Energy

Table 2: Comparison between *nRF5340* and *nRF54L15*

The 54L series is selected for this project's low-power application focus and required processing performance. The *nRF5340*, with its dual-core design, could offer more efficiency for concurrent tasks. However, this difference is not expected to impact the specific requirements and constraints of the current project.

The *nRF54L15* has a larger combined non-volatile memory and uses RRAM. *nRF5340*, on the other hand, has more total RAM (576 KB vs 256 KB). Dealing with limited RAM requires a very strict approach. The stack and heap must be used correctly to reduce RAM usage. The following strategies can be applied during the implementation phase to control and reduce the risk of memory issues :

- **Data type minimization :** Choose the smallest integer type that can hold your required value.
- **Struct packing :** Organize structure members to eliminate compiler-added padding bytes.
- **Constant data storage :** Store unchanging data in flash, not RAM.
- **Minimize dynamic allocation :** Avoid malloc/free during runtime to prevent memory fragmentation.

- **Use static/pre-allocated buffers :** Define buffers globally once, and pass them to functions instead of creating them locally.
- **Limit recursion :** Keep function call depth shallow to prevent stack overflow.
- **Analyze the map file :** Use the linker map file to find and target the largest RAM consumers.

On the radio side, the *nRF5340* has a slightly better consumption for both transmission and reception. It also supports LE audio, but this feature is not a requirement for this project, as it does not need to broadcast audio. A main advantage is that the *nRF54L15* supports the latest Bluetooth features, such as Channel Sounding, for highly accurate distance measurement, thanks to Bluetooth 6.0.

Despite some drawbacks compared to the *nRF5340*, the new microcontroller chosen should enable significant improvements in power consumption. Therefore, an excellent candidate for this project.

5.6.3 Microphone selection

To reduce system consumption, it needs to limit the volume of data written to the SD card. Consequently, the previous microphone was replaced by the *TDK T5848*. The fundamental shift in this new architecture is the transition from continuous recording to an event-based system. While the previous version required the system to remain constantly active to avoid missing acoustic events, the *T5848* features an internal level-detection system that triggers recording only when a set sound threshold is reached.

5.6.3.1 Comparative analysis

The following table summarizes the technical improvements between the V1 and V2 microphones:

Feature	V1 (SPH0645LM4H-B)	V2 (TDK T5848)
Interface	32 bit I^2S (16 relevant bit)	24 bit I^2S (16 relevant bit) + 1-Wire config
Active current	600μA	120μA (low power mode)
Sleep mode current	3μA	20μA (with detection)
Wake-up logic	Always ON	Hardware interrupt
Dimensions	$3.50 \times 2.65 \times 0.98$ mm	$3.50 \times 2.65 \times 0.98$ mm

Table 3: Comparison between *Syntiant SPH0645* (V1) and *TDK T5848* (V2)

5.6.3.2 Physical and internal architecture

The *T5848* microphone is a multi-mode, low-noise digital MEMS microphone contained within a small surface-mount package, measuring $3.5 * 2.65 * 0.98$ mm. It is a bottom-port device with an I^2S digital output. It integrates a MEMS microphone element, an impedance-converter amplifier, an analog-to-digital converter, decimation and anti-aliasing filters, power management and an industry-standard 24-bit I^2S interface. The

I^2S interface allows direct connection to digital processors, eliminating the need for an external audio codec.

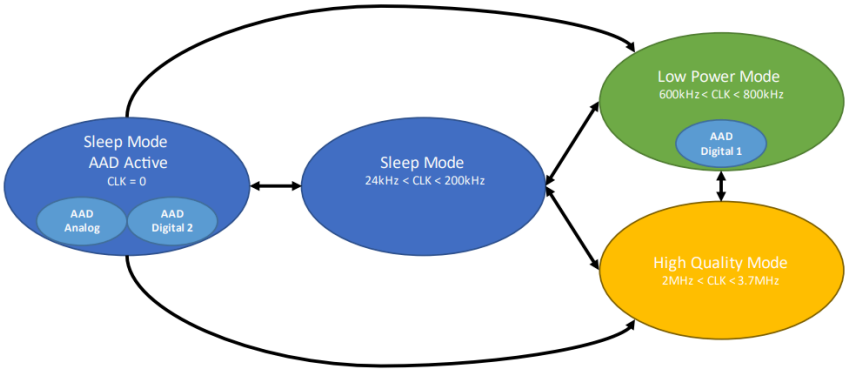
Source: https://invensense.tdk.com/products/digital/t5848/

Figure 16: TDK T5848 microphone

5.6.3.3 Operation logic

Operation modes

The microphone has multiple modes, pilot driven by the I2S clock frequency. The Figure 17 represents these modes and their transitions.



Source: https://invensense.tdk.com/products/digital/t5848/

Figure 17: T5848 mode transitions

The Table 4 summarizes the graph information along with the expected consumption.

Mode	*CLK* frequency	Data output	Typical current
High Quality Mode (HQM)	2.0 to 3.7 MHz	32 bits/channel, 24-bit precision	310 μA
Low Power Mode (LPM)	600 to 800 kHz	24 bits/channel, 16-bit precision	120 μA
Sleep Mode	0 Hz	No output	20 μA

Table 4: Operation modes

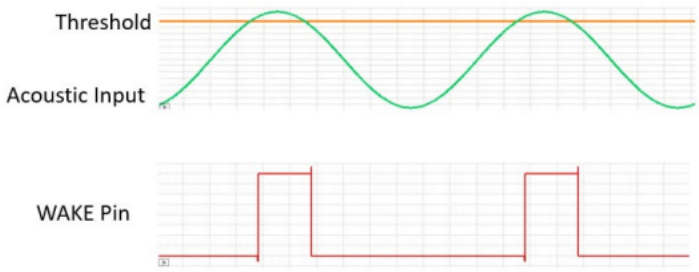
AAD

It also features Acoustic Activity Detection, which are parallel processing features. The processing of these AAD modes determines if acoustic activity has occurred or not. It records sound only when the acoustic activity meets the set conditions. There are three different types.

AAD analog

The AAD analog mode works by analyzing the analog signal from the MEMS element immediately after the pre-amplifier. This signal is measured against two preset conditions:

1. **Absolute Threshold :** Sets a fixed analog voltage level the signal must exceed to trigger the detection logic.
2. **Low Pass Filter (LPF) cutoff frequency :** Defines the upper frequency limit for detection.



Source: *https://invensense.tdk.com/products/digital/t5848/#documentation*

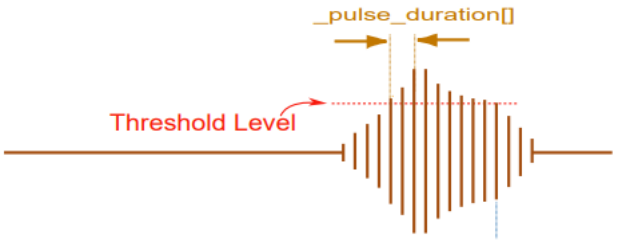
Figure 18: AAD A Threshold level and WAKE pin activation

The system will trigger the *WAKE* pin to a HIGH state only if the signal is both above the absolute threshold and below the LPF cutoff. The *WAKE* pin remains HIGH as long as these acoustic criteria are met. It automatically revert to a LOW state when the signal falls below the required levels. To ensure the operation of the AAD A mode, the main clock must be OFF.

AAD digital 1

AAD digital 1 functions as an enhancement to the Low Power Mode. In this mode, the digital bitstream is continuously analyzed by the dedicated AAD digital logic. This design determines if the acoustic signal meets the preselected configurable conditions:

1. **Absolute threshold :** Defines a fixed acoustic pressure level that the incoming signal must exceed to be considered a potential event.
2. **Relative threshold :** Sets a dynamic limit based on the ambient noise floor. It trigger an interrupt only when the signal rises a specific amount above the background environment.
3. **Minimum pulse duration :** Specifies the minimum continuous time the signal must remain above the thresholds.



Source: https://invensense.tdk.com/products/digital/t5848/#documentation

Figure 19: AAD D parameter visualization

When these criteria are satisfied, the *WAKE* Pin transitions high and stays high while the conditions are maintained, returning low when the signal falls below the threshold.

AAD digital 2

AAD digital 2 functions like AAD digital 1 by evaluating the digital bitstream against identical, configurable criteria.

However, AAD D2 delivers superior power savings at both the microphone and system levels, utilizing an internal *CLK* and eliminating the need for an external *CLK*.

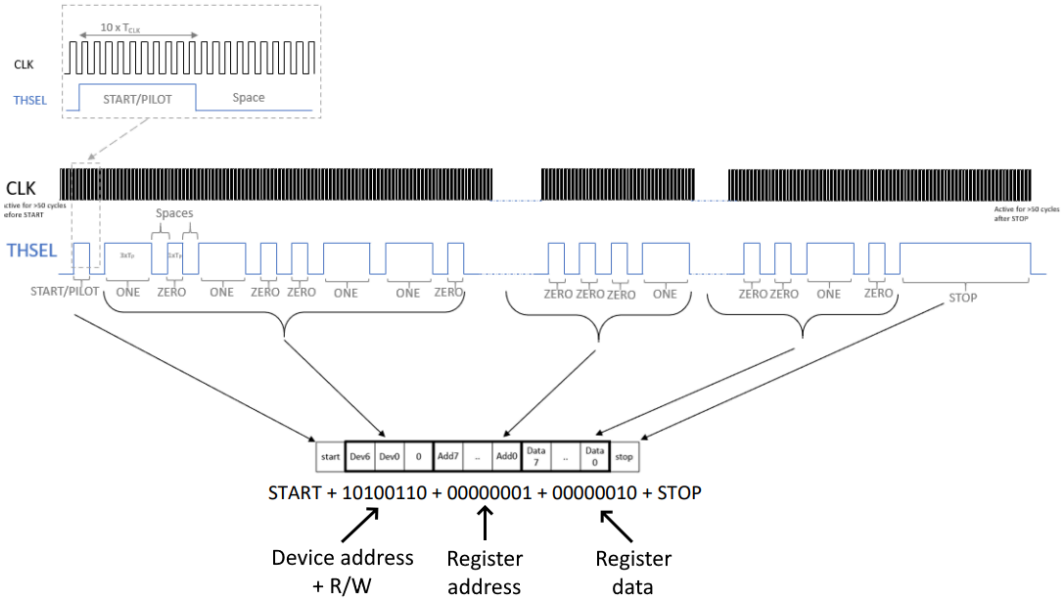
5.6.3.4 Configuration

The Acoustic Activity Detection modes on the *T5848* microphone are configured and activated via a serial one-wire protocol on the *THSEL* (Threshold selection) pin. For configuration, the I^2S clock must be running at a speed between 50 and 200 k Hz. The protocol uses pulse-width modulation on the *THSEL* pin, proportional to the number of *CLK* cycles, to encode specific symbols for logic zeros and ones. It comprises the device address, the register address and the command data. The symbols are specified in the Table 5.

Symbol name	Description	THSEL condition	Calculation
Start/Pilot	Start/Pilot which indicates start of write and defines logic pulse widths	HIGH	$10 * T_{\text{CLk}} = T_P$
Zero	Single bit Zero	HIGH	$1 * T_P$
One	Single bit One	HIGH	$3 * T_P$
Stop	Stop Signal	HIGH	$> 128 *$ CLK period
Space	Separate individual symbols above	LOW	$1 * T_P$

Table 5: One wire symbol

The overall write sequence begins with a START/PILOT symbol, followed by 24 bits of payload (device address + R/W, register address and register data) and concludes with a STOP symbol. This process is illustrated in further detail in Figure 20.



Source: https://invensense.tdk.com/products/digital/t5848/#documentation

Figure 20: AAD write sequence example

This mechanism ensures that AAD configuration settings are correctly transmitted to the microphone. The configuration writing is confirmed on the *WAKE* pin with a 12 μS pulse.

5.6.3.4.1 Implementation details

For this project, the microphone will alternate between two different modes and Acoustic Activity Detection. This will be done by changing the I2S clock. The Figure 21 shows the transition between the two used modes.

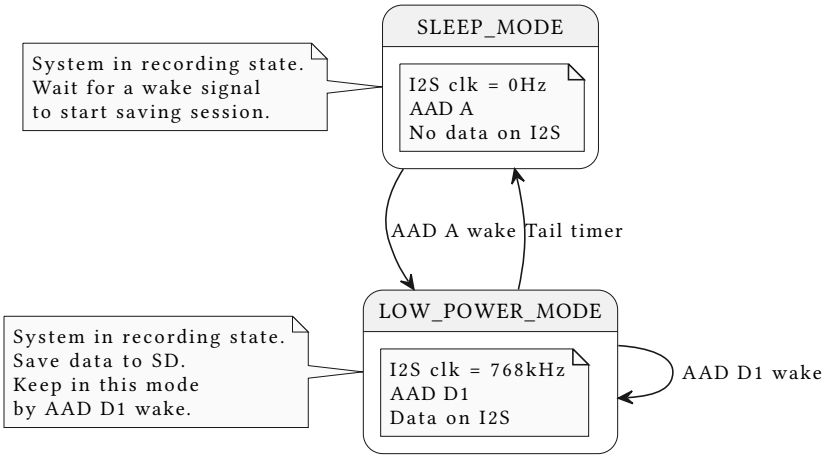


Figure 21: Microphone state transition

The microphone will then need a configuration for the analog and digital activity detection modes. The details will be provided in the implementation.

6 Hardware design

For this project, two electronic boards were designed, based on the first version by a HES-SO Valais/Wallis collaborator. They were printed by *EuroCircuits* and mounted by the electronic workshop. These boards are prototypes and are therefore larger. Thanks to numerous measurement points, it enables easy debugging of the card and its firmware. The boards have been provided at the beginning of the thesis. Testing these boards was part of this project.

The two boards form the complete system as described in Figure 7. The following two points detail the main features of each boards. Some part of the schematics will be presented. The complete ones are in the project repository (see Section AA).

6.1 nRF54L15 development board

nRF54L15 development board contains the microcontroller and all and all are associated electronics.

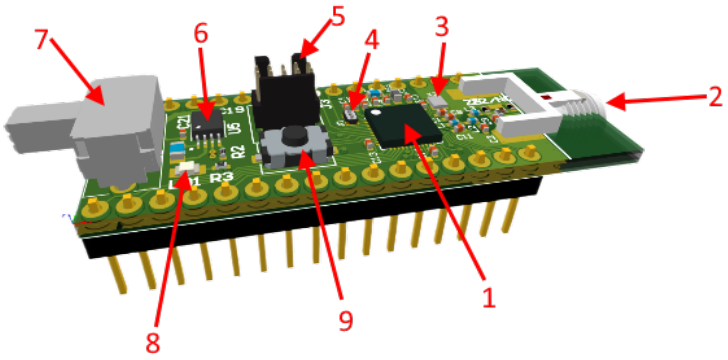


Figure 22: nRF Monkey development board

In the Figure 22, the main component are pointed. They are explained below.

1. **Microcontroller (*nRF54L15 SoC*) :** It was designed based on the nRF54L15-QFAA reference design [25]. It is connected to all these peripherals using the two rows of pins.

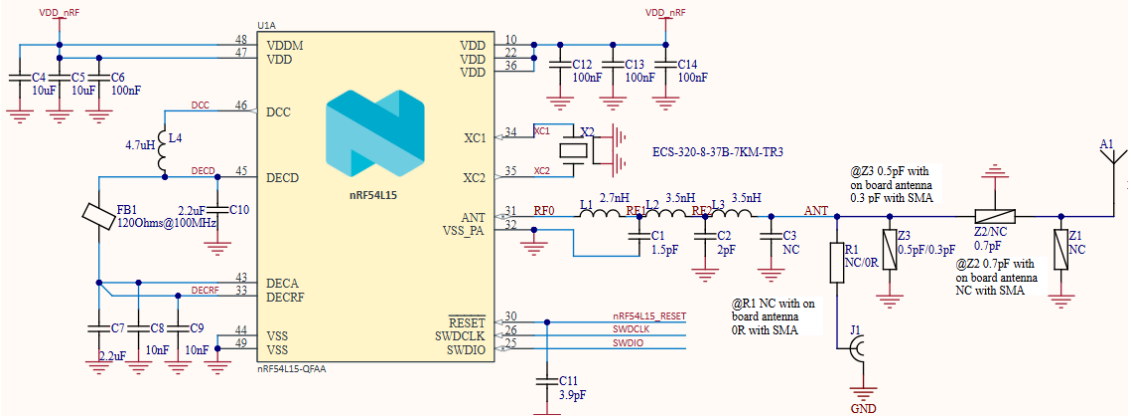


Figure 23: Microcontroller

2. **BLE antenna :** Two antenna options are available: a SMA connector or a ceramic chip antenna (*2450AT43F0100*). A universal matching network optimizes the antenna signal (see Figure 24).

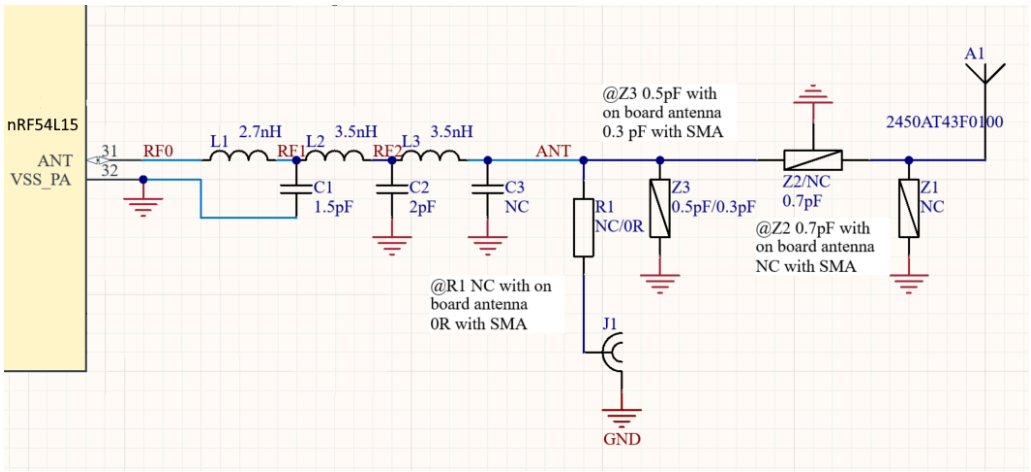


Figure 24: BLE antenna matching

The components directly following the *nRF54L15* pins (L1, C1, L2, C2, L3, C3) form a multi-stage low-pass filter and matching network. It matches the output impedance of the SoC to the standard 50 Ω transmission line.

To use the ceramic antenna, the resistor R1 is not connected. Shunt components Z2 and Z3 are added. The signal terminates at the chip antenna.

To use an external antenna or measurement equipment via the SMA connector, a 0Ω resistor bridges the signal to the J1 connector. To minimize signal loss on this path, Z2 is not connected and Z3 is reduced to 0.3 pF.

3. **Crystal 32.768kHZ (*ABS06-32.768KHZ-1-T*) :** It provides a 32.768 kHz signal to the System-on-Chip. This provides the timing for the low-power clock domain. It allows the system to maintain accurate timing while in deep sleep modes.

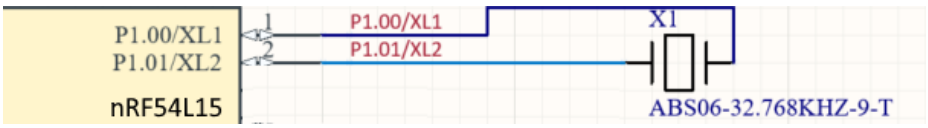


Figure 25: 32.768kHZ crystal

4. **Crystal 32MHZ (*CX2016DB*) :** Provide a 32 MHz signal to the System-on-Chip. It is the primary clock source for the microcontroller's internal operations. It is required for the Bluetooth Low Energy radio.

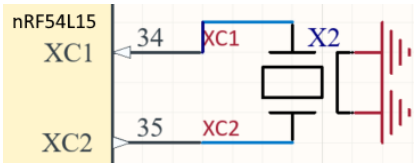


Figure 26: 32MHZ crystal

5. **Programming pins :** Allows programming the microcontroller via the SWD. It has to be connected to the nRF54l15-DK's DEBUG OUT pins or via a ST-LINK to flash a new firmware.

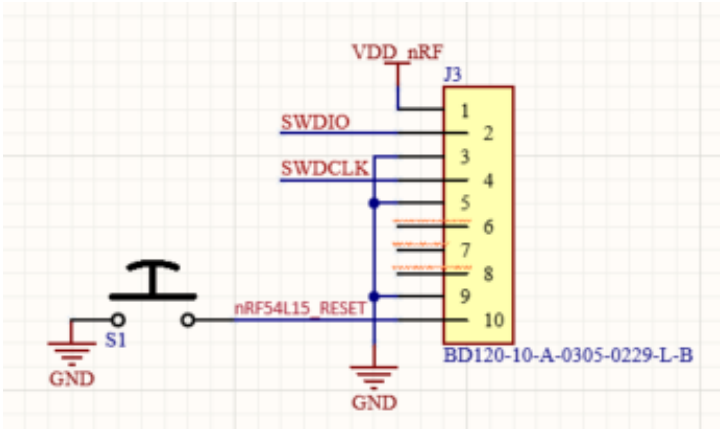


Figure 27: Programming pins

6. **3.3V step down convertor (*TPS62840DGR*) :** It provides a stable alimentation of 3.3V to the microcontroller by regulating the battery voltage.

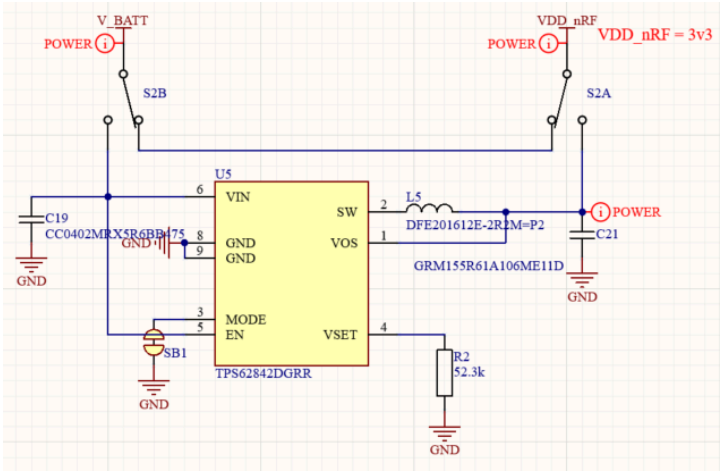


Figure 28: Step down converter

7. **Alimentation source switch :** Select the system alimentation source: direct from the battery or through the step-down convertor (see Figure 28).

8. **User led :** It allows the creation of a simple user interface.

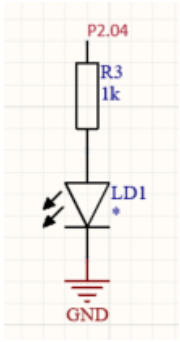


Figure 29: User led

9. **Reset button :** Performs a hardware reset. It is directly to the *nRD54L15* reset pin (see Figure 27)

A table containing the SoC pin assignments is available in the appendix (Section AB).

6.2 MIC/SD board

The MIC/SD board is a shield for the development board. It plugs into the latter. All the system's peripherals and their electronics are mounted on this board: *T5848* microphone, SD card and the release system electronics.

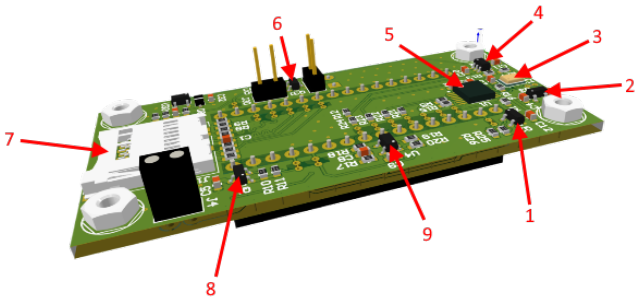


Figure 30: nRF Monkey MIC/SD board

In the Figure 22, the components are pointed and explained below.

1. **Microphone power supply enable (*DMP2035U-7*):** It allows the microcontroller to enable/disable the microphone power supply.

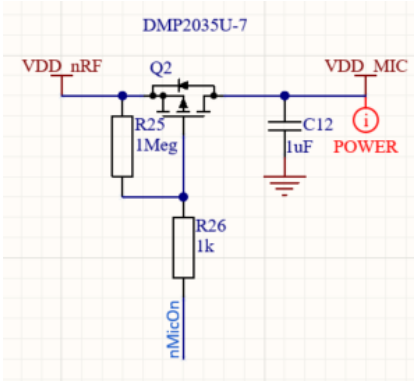


Figure 31: Microphone power supply enable

2. **Microphone voltage regulator (*TSL9A12V18CX RFG*) :** It produces a 1.8V for the microphone using the 3.3V from the development board.

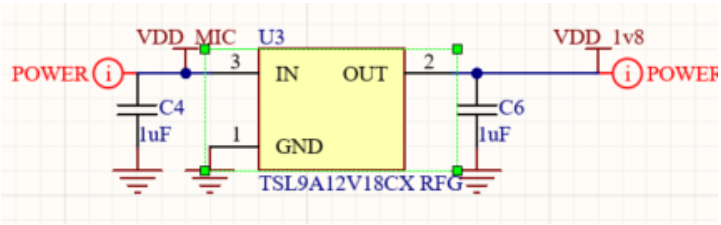


Figure 32: Microphone 1.8V voltage regulator

3. **Microphone (*TDK T5848*) :** The *T5848* microphone is soldered directly onto the board. A hole in the PCB, under the microphone, exposes the membrane to the air. It operates at 1.8V.

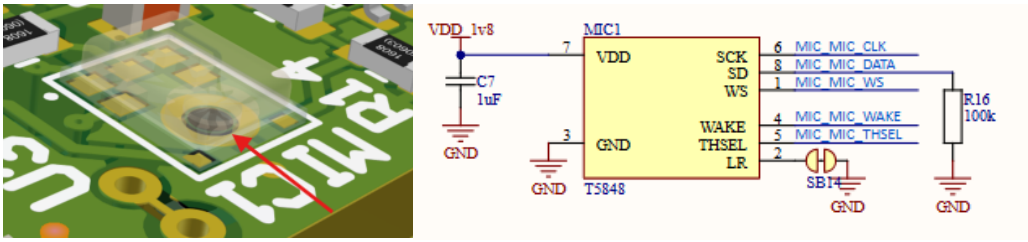


Figure 33: Microphone hole and schematic

4. **Microphone threshold control level shifter (*TXU0102*) :** It specifically manages the voltage translation and signal conditioning for the threshold configuration line. It shifts between the 3.3V SoC and the 1.8V microphone voltage.

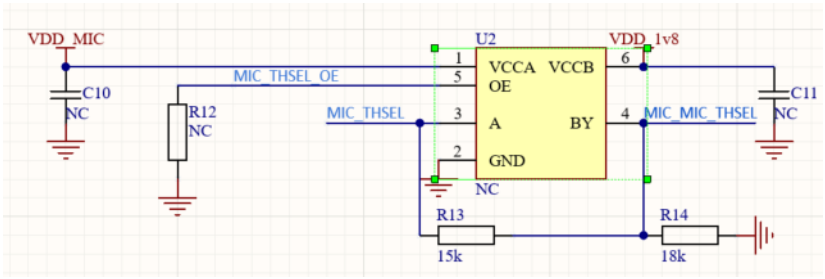


Figure 34: Microphone threshold control level shifter

5. **Audio signal level shifter (*TXU0204PW*) :** It specifically manages the voltage translation for all audio signals, from 3.3V to 1.8V and vice versa.

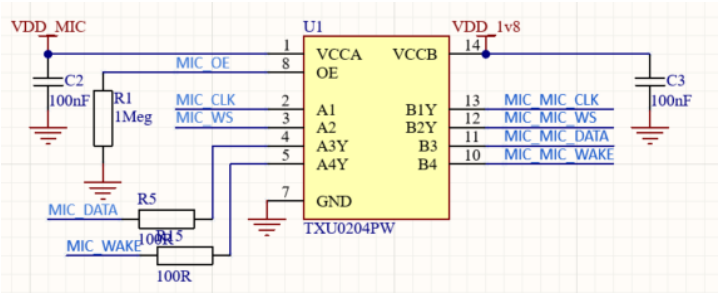


Figure 35: Audio signal level shifter

6. **Burn wire command (*DMN2056U-7*) :** The SoC triggers the release process by pulling the BURN signal high. It allows the current to flow from a high-capacity source through the burn wire. The nylon connection is then burned and the collar is released from the monkey's neck.

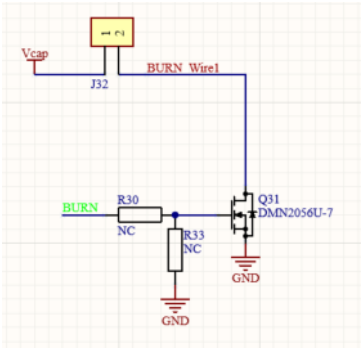


Figure 36: Burn wire command

7. **SD card connector (*Molex 47352-1001*) :** The card is interfaced using the SPI protocol. Pin 9 (DETECT) is pulled up to VDD_SD via resistor R8 to indicate when a card is inserted or removed.

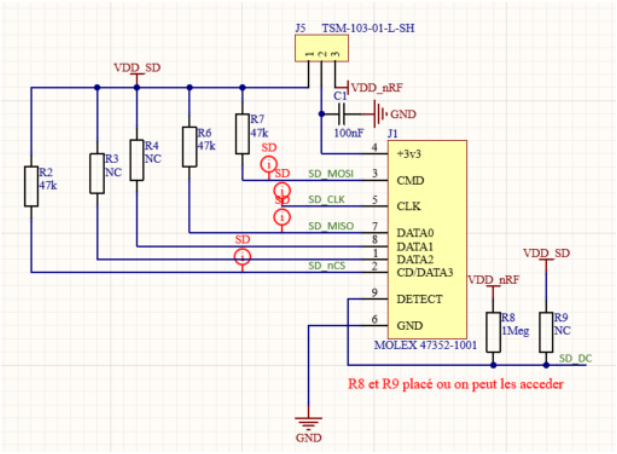


Figure 37: SD schematics

8. **SD power supply enable (*DMP2035U-7*) :** It allows a microcontroller signal to enable/disable the power supply for the SD card connector.

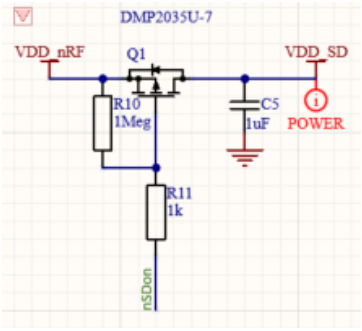


Figure 38: SD power supply enable

9. **Battery monitoring circuit (*TLV7031*) :** This circuit provides a hardware alert when the battery voltage drops below a safe operating level. The circuit is a voltage comparator with hysteresis. When the battery is full, the comparator output is LOW.

   The battery voltage V_{batt} required to trigger the low alarm is calculated using the input divider (R_{17}, R_{18}):

$$V_{\text{ref,low}} = V_{\text{ref}} * \frac{R_{20}}{R_{19} + R_{20}} = 1.8\ \text{V} * \frac{10\ \text{M}\Omega}{0.51\ \text{M}\Omega + 10\ \text{M}\Omega} = 1.7126\ \text{V} \quad (1)$$

$$V_{\text{batt,low}} = V_{\text{ref,low}} * \frac{R_{17} + R_{18}}{R_{18}} = 1.7126\ \text{V} * \frac{9M\Omega + 10M\Omega}{10M\Omega} = 3.27\ \text{V} \quad (2)$$

   When the alarm is active, the output is HIGH. The feedback resistor connects to 1.8 V, making the reference voltage effectively 1.8 V. The battery voltage required to cross this threshold is:

$$V_{\text{batt,high}} = 1.8\ \text{V} * \frac{R_{17} + R_{18}}{R_{18}} = 1.8\ \text{V} * \frac{9M\Omega + 10M\Omega}{10M\Omega} = 3.44\ \text{V} \quad (3)$$

   It is representing the safe voltage required to consider the battery good.

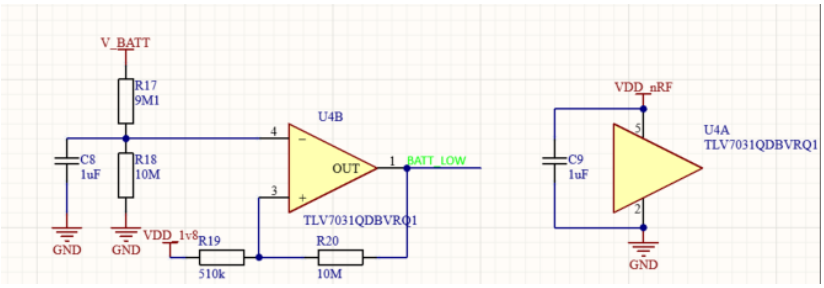


Figure 39: Battery monitoring circuit

7 BLE positioning method validation

The Bluetooth Low Energy has been chosen as the positioning method. This technology proposes two methods to detect proximity and evaluate the distance:
- RSSI-based method
- CS-based method

Before this evaluation, a reminder about BLE technology is given. Each method has advantages and disadvantages. To allow for an objective choice, both methods must be evaluated. This chapter, therefore, provides this evaluation by presenting the tests performed, the results obtained and the conclusions drawn.

7.1 BLE basics

The main goal of Bluetooth Low Energy wireless communication technology is to use as little power as possible. It works in the 2.4 GHz ISM band, which does not require a license. Out of the 40 channels that BLE creates in this band, three are used mainly for advertising. BLE devices can act as either a central or a peripheral. To help other devices find them, peripherals broadcast data to nearby devices. Centrals detect these advertising packets and can connect to the peripherals.

When a peripheral sends a packet to other devices, this is called advertising. These packets go out on the main advertising channels. The central device scans these three channels regularly to find advertising data. Depending on the settings, these channels may allow or block connections. An advertisement packet usually contains a small amount of data, the device name and the transmission power. The peripheral can send these packets at intervals from 20 milliseconds up to 10 seconds. Shorter intervals help the device get detected faster, but they use more power. [26]

After a central finds a peripheral by scanning for advertising packets, it can send a connection request. The connection is made once the central gets a response. BLE devices use attributes, called characteristics and services, to share data. Characteristics hold specific data values, while services group one or more characteristics to support management and discovery. Services usually collect similar attributes together. Different protocols handle the connection and data transfers. Bluetooth Low Energy uses a generic profile called Generic Access Profile, which is the base for all BLE devices. It sets the basic requirements for communication. GAP also provides security using cryptographic algorithms and the Security Manager and it defines the roles that BLE devices can have.

The ATT protocol serves as the basis for the higher-level Generic Attribute Profile protocol. It gives a standard way for devices to find, read, write and share information through features and services. Attribute Protocol manages data sharing by defining two roles: the client, which reads and modifies data and the server, which stores data. A device can be both a client and a server simultaneously. This protocol organizes data into characteristics, each with a server-side handle that serves as a unique ID and represents a piece of information. Each characteristic also has a Universally Unique Identifier, which can be 128 bits long for custom UUID or 16 bits long for SIG-adopted UUID.

Each attribute also has specific permissions:

- **Read :** Permission to read characteristic values from the remote device.
- **Write :** Permission to write data to a feature on the remote device.
- **Notify :** Permission for the device to send notifications to the connected device when the value of a characteristic changes without confirmation.
- **Indicate :** Permission for the device to send indications to the connected device when the value of a characteristic changes, with acknowledgement required from the connected device.Finally, BLE helps devices save energy by enabling them to adjust their communication range to suit the application's needs. The technology's open, free standards also enable fast communication. [27]

7.2 BLE positioning methods

Bluetooth Low Energy has become a key technology for Indoor Positioning System and proximity-based applications because of its low power consumption, wide adoption and ease of integration. Accurate distance estimation between devices is fundamental to enabling BLE-based positioning, ranging from simple presence detection to more advanced real-time positioning.

This work studies and compares the principles, advantages and limitations of RSSI and CS methods, establishing the basis for evaluating BLE positioning performance within diverse scenarios.

7.2.1 RSSI method

The working principle of Received Signal Strength Indicator in Bluetooth Low Energy relies on the propagation characteristics of radio waves as represented in Figure 40.

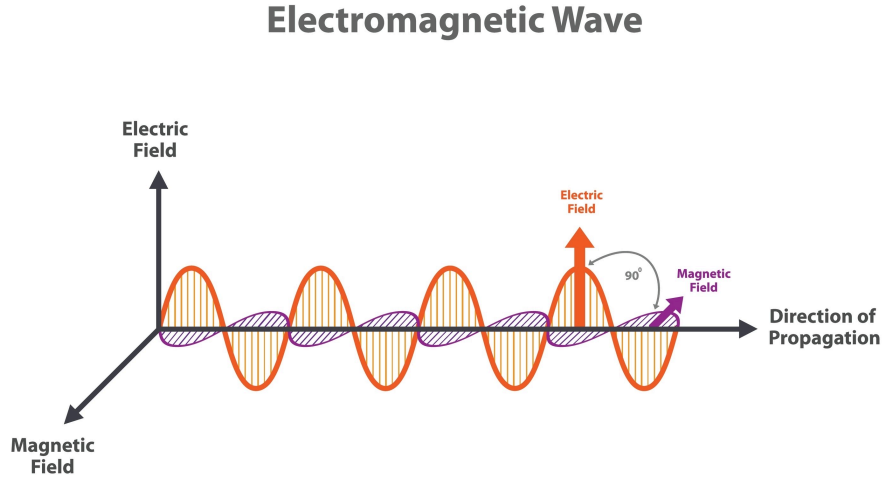


Figure 40: Electronic wave

As the transmission distance increases, the signal's energy gradually attenuates, leading to a reduction in the received signal strength. By measuring this variation in signal strength, RSSI enables the estimation of the distance between the transmitter and the receiver. The distance can be computed with the following formula. [28]

$$\mathrm{d} = 10^{\frac{\mathrm{RSSI}_{d_1} - \mathrm{RSSI}_d}{10 * c}} \tag{4}$$

With :

- RSSI_d : Measured signal strength at distance
- RSSI_{d_1} : Reference signal strength measured at 1 meter.
- c : Path loss exponent (propagation constant), representing environmental interference.

The RSSI value is influenced not only by distance but also by several environmental and hardware factors, including multipath effects, physical obstructions, antenna orientation and device performance. Despite these limitations, BLE has adopted RSSI as a energy efficient method for proximity detection and coarse distance estimation. [29]

Indoor Positioning System massively use this method to provide location data in closed environments. The academia and industry have completed many significant studies relevant to IPS.

Advantages

- **Efficiency:** Utilizes native wireless chip data with no additional hardware or processing, ensuring minimal power draw and low cost.
- **Simplicity:** Provides instantaneous proximity detection with negligible communication overhead.
- **Compatibility:** Works across all BLE standards and devices without requiring specialized hardware or updates.

Disadvantages

- **Low precision:** Highly sensitive to extern factors, resulting in significant noise and distance errors.
- **Environment sensitivity:** Accuracy depends on the path loss index (c), which requires constant recalibration for different physical environments.

7.2.2 Channel sounding method

Introduced by Bluetooth® 6.0, Channel Sounding enhances how BLE devices measure distance and detect presence. To achieve this, Bluetooth® Channel Sounding uses two proven ranging methods, Phase-Based Ranging and Round-Trip Time. The CS uses a more sophisticated channelization technique with 80 channels, each 1 MHz wide, while BLE uses 40 channels, each 2 MHz wide. The channels that overlap with BLE advertising channels 37, 38 and 39 have been specifically identified as useless in order to reduce interference with primary BLE functions. As a result, only 74 of the 80 established channels are being used for Channel Sounding.

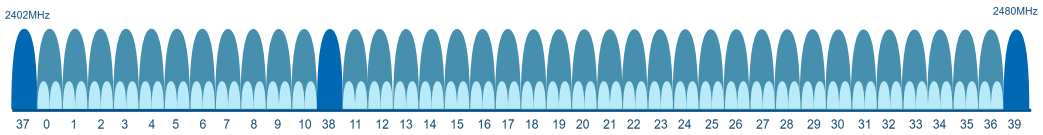
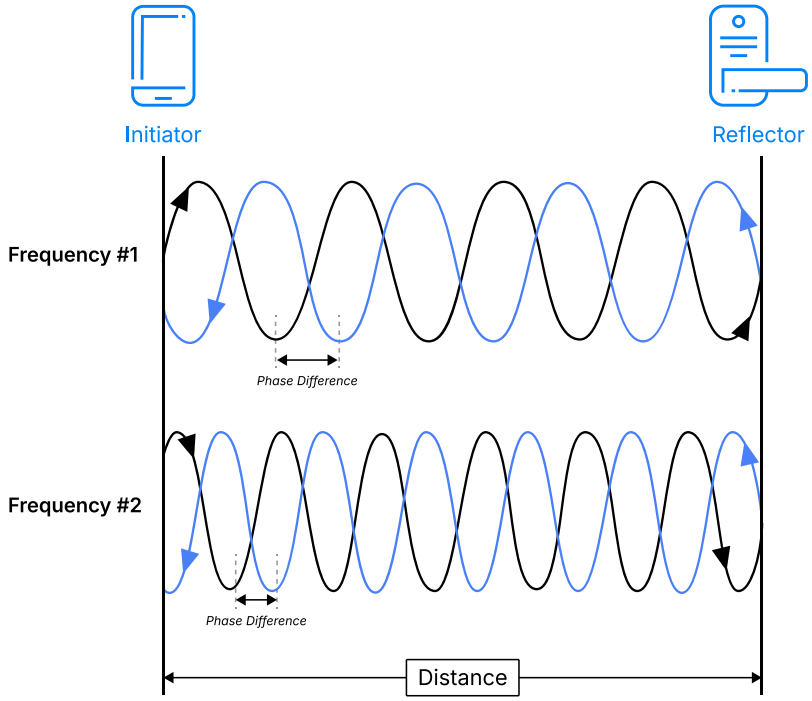


Figure 41: Bluetooth Low Energy ISM band spectrum

The Figure 41 shows both 1 MHz and 2 MHz channel widths. Dark blue channels indicate the primary BLE advertisement channels, turquoise channels represent the secondary data channels. Finally, the new spectrum allocation for CS is highlighted in light blue. [30]

7.2.2.1 Phased-Based Ranging

This first ranging method is used to measure the distance. In Phase-Based Ranging, an initiator device sends a signal to a reflector device, which returns the signal. This process is repeated across multiple frequencies. The distance between the devices is calculated from the phase differences between the transmitted and received signals at those frequencies.

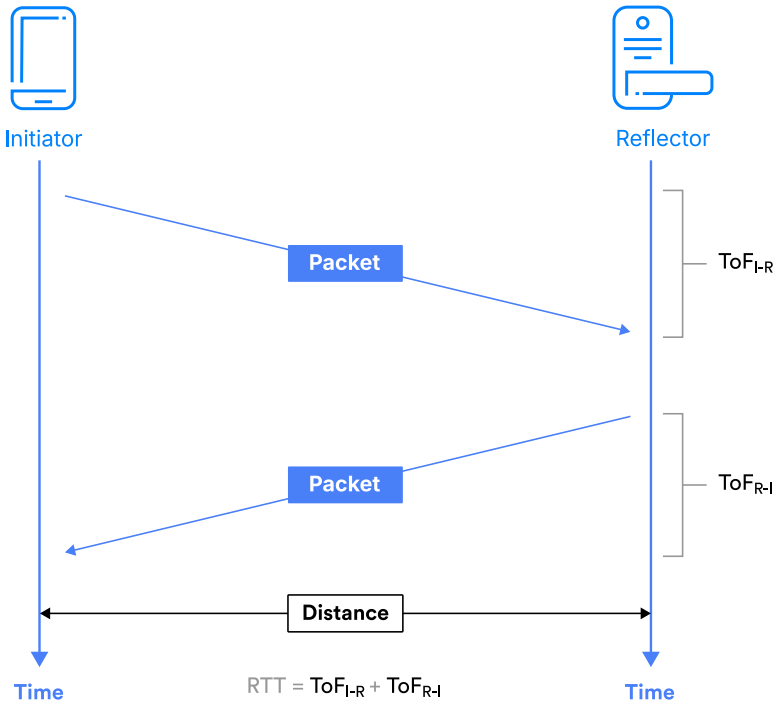


Source: https://www.bluetooth.com/learn-about-bluetooth/feature-enhancements/channel-sounding/

Figure 42: Phased-Based Ranging principle

7.2.2.2 Round-Trip Time

This second ranging method provides a countermeasure against sophisticated Man-in-the-Middle relay attacks. An initiator device sends cryptographically scrambled packets to a reflector device, which then returns them. The distance between the devices is then calculated based on the time it took for the packets to travel back and forth.



Source: https://www.bluetooth.com/learn-about-bluetooth/feature-enhancements/channel-sounding/

Figure 43: Round-Trip Time principle

RTT serves as a secure distance-bounding technique, providing an independent distance measurement to cross-check the PBR measurement. By combining PBR and RTT, Bluetooth® Channel Sounding enables secure and accurate distance measurements between devices.

Advantages

- **Precision:** Uses PBR to mitigate multipath effects, providing significantly higher accuracy than RSSI.
- **Security:** Employs RTT with cryptographically scrambled packets to prevent relay and Man-in-the-Middle attacks.
- **Reliability:** Cross-checks PBR and RTT data to ensure distance measurements are both precise and trustworthy.

Disadvantages

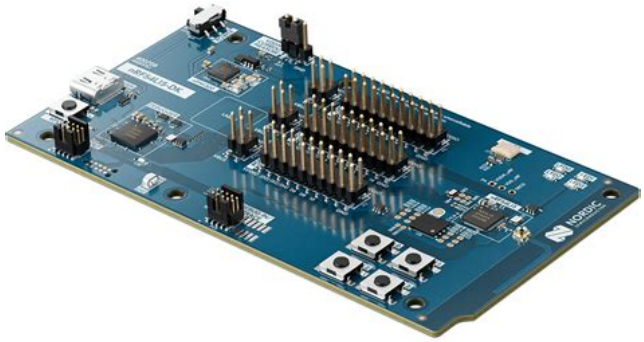
- **Compatibility:** Requires Bluetooth® 6.0 and specialized hardware, making it incompatible with older BLE devices.
- **Resource intensive:** Sophisticated signal processing across multiple channels increases power consumption and computational overhead.
- **Low maturity:** As an emerging technology, the development ecosystem and troubleshooting resources are currently limited compared to RSSI.

7.3 Methods evaluation

The RSSI-and CS-based methods have been evaluated to help select the most suitable method for the project. The major evaluation criteria are:

1. Accuracy
2. Detection range
3. Power consumption
4. Maximum numbers of measure

A test will be carried out for each of these criteria. To perform these tests, two nRF54L15-DK (v0.9.2) will be used. These kits use Bluetooth® 6.0, enabling the use of the Channel Sounding.



Source: https://www.nordicsemi.com/Products/Development-hardware/nRF54L15-DK

Figure 44: nRF54L15-DK development board

RSSI

The Received Signal Strength Indicator measurement will be tested using two devices configured as follows:

- **Central :** Acts as the BLE scanner. It periodically reads the RSSI value from the received advertising packet. These packets are filtered using the manufacturer's data. The distance are then calculated.
- **Peripheral :** Acts as the BLE advertiser. Advertises its presence periodically.

To compute the distance based on the measured RSSI, the Equation 4 was used.

CS

To test the Channel Sounding, specific nRF CS samples were selected for their relevant roles in the Channel Sounding procedure. The following samples were chosen:

- **Bluetooth - Channel Sounding Initiator with Ranging Requestor :** acts as a GATT Ranging Requestor client and configures the Channel Sounding initiator role. Regular Channel Sounding procedures are set up, local sub-event data is stored and peer ranging data is fetched.
- **Bluetooth - Channel Sounding Reflector with Ranging Requestor** : exposes the GATT Ranging Responder Service and configures the Channel Sounding reflector role. When Channel Sounding Ranging Data is generated by the controller, it is automatically

stored by the Ranging Service and can be queried by the Ranging Requestor at any time. The Channel Sounding Ranging Data can then be used by the peer device to estimate distance.

The samples were modified to print the results in the desired format. They are available in the project repository (see Section AA).

7.4 Accuracy evaluation

The accuracy of both methods has been evaluated. Two different tests have been performed :

- Indoor environment test
- Outdoor environment test

a) Indoor test setup b) Outdoor test setup

Figure 45: Indoor and outdoor measurement configurations

The measures were performed at 1 meter from the ground, as shown in Figure 46. It has a direct view between the two devices (Figure 45). The *Central* or *Initiator* device stays steady at the zero meter mark. The *Peripheral* or *Reflector* moves gradually by 1m from 0m until it reaches 10m, scanning 20 samples at each position.

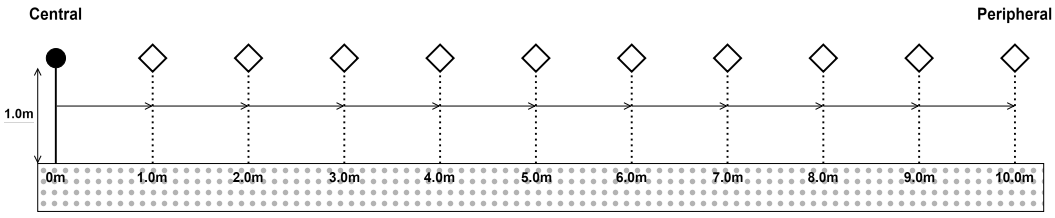


Figure 46: Distance between the beacon and scanner in the measurements

7.4.1 Indoor environment test

The indoor tests were performed in a laboratory room containing offices equipped with desktop computers. It also contains storage cabinets. The Figure 47 shows the indoor test environment.

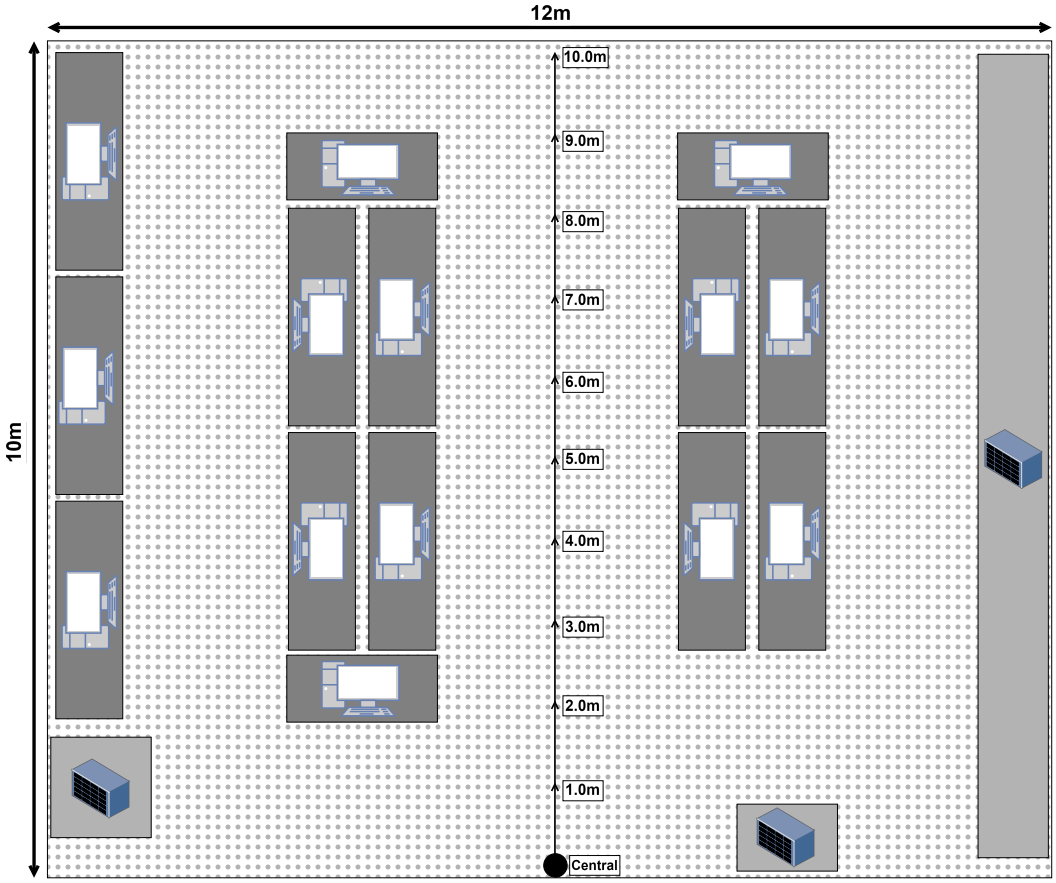


Figure 47: Indoor measurement environment overview

7.4.2 Outdoor environment test

The outdoor tests were performed along the main HEI building. The Figure 48 represents the outdoor situation.

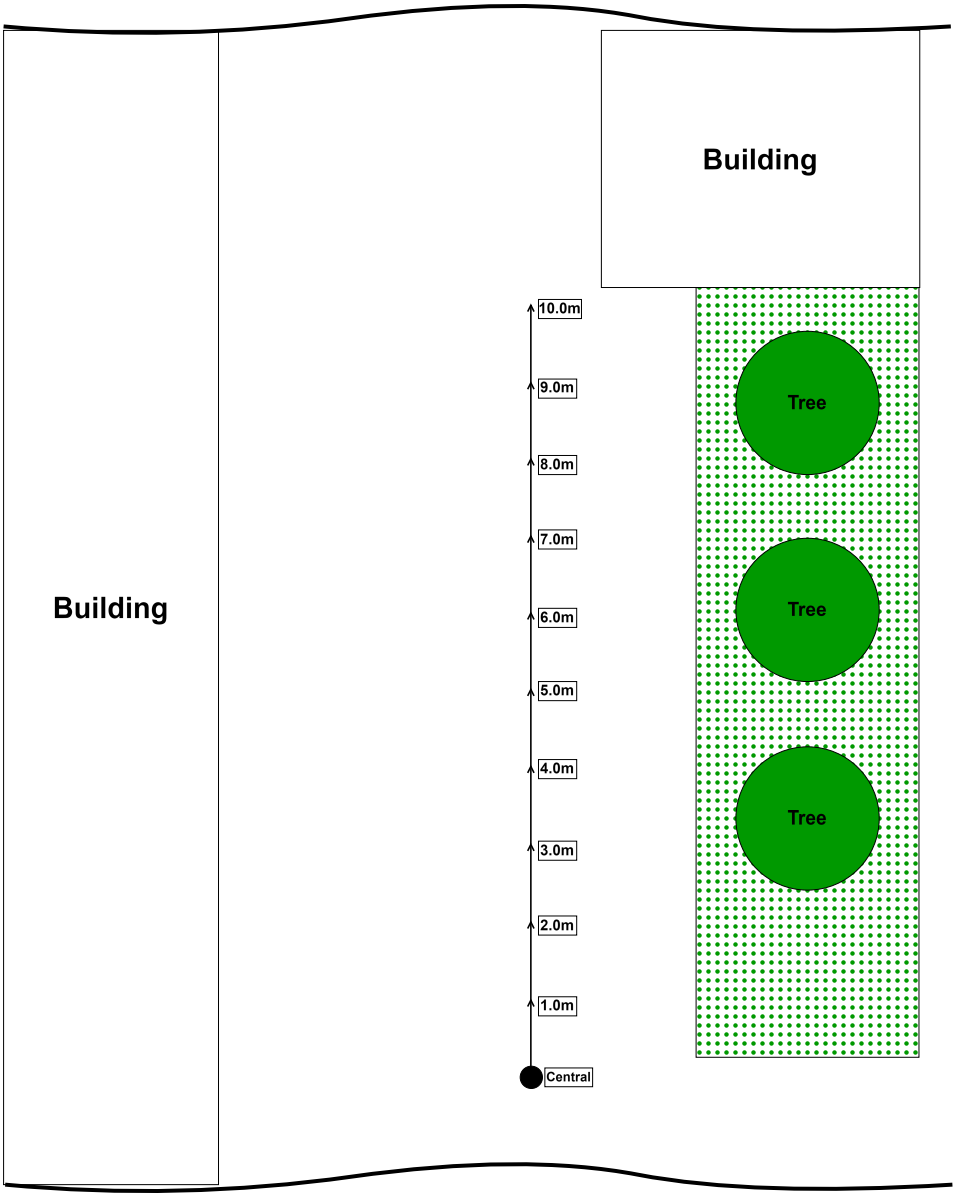


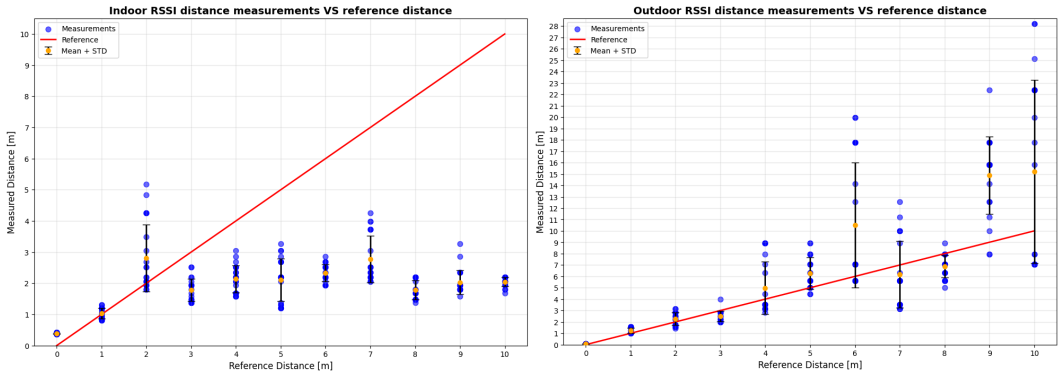
Figure 48: Outdoor measurement environment overview

7.4.3 Results

In this section, the test results will be analyzed and discussed. In the Figure 49 and Figure 51, the results of the four tests have been plotted separately. The X axis represents the reference distance, while the Y axis represents the measured distance. The red line serves as a reference. It helps compare results across different tests. The samples are represented as blue dots in a scatter plot. The orange dot represents the mean value, along with a black line that represents the standard deviation.

7.4.3.1 RSSI-based method

The Figure 49 plots the accuracy results for both indoor and outdoor tests.



a) RSSI indoor b) RSSI outdoor

Figure 49: Indoor and outdoor test results for RSSI method

- **Indoor performance :** Accuracy is poor due to signal multipath and reflections within the room. Even with a calibrated path loss constant ($c = 3.5$), environmental interference prevents reliable distance estimation.
- **Outdoor performance :** Results are acceptable only up to a range of 5 meters. Beyond this, signal variance increases significantly, leading to imprecise mean values.
- **Calibration sensitivity :** Distance calculations depend heavily on the accuracy of the 1-meter reference (RSSI_{d_1}) and the path loss exponent (c).
- **Requirement :** Precise power measurements at exactly 1 meter are essential to minimize calculation errors.

Parameters tuning

Knowing the reference distances allows optimizing the value of the part loss index. This allows the measurement calculation to be fine-tuned to the operating environment. It can be tuned to minimize the average error or the standard deviation. It is also possible to make compromises and minimize both variables. Optimal performance at a specific distance can be achieved. However, this will affect accuracy at other distances.

To illustrate this possibility for improvement, the parameter value was subsequently optimized to minimize the average error using the measured RSSI values. A Python function was written for this purpose (see Listing 1).

```
# Part loss index optimisation function
def optimize_path_loss_exponent(values):
   ref_dist = list(range(11))
   best_n = None
   min_error = float('inf')
   error = np.array([])
   c = np.linspace(2, 5, 25)

   for c_val in c:
      for rssi in values:
```

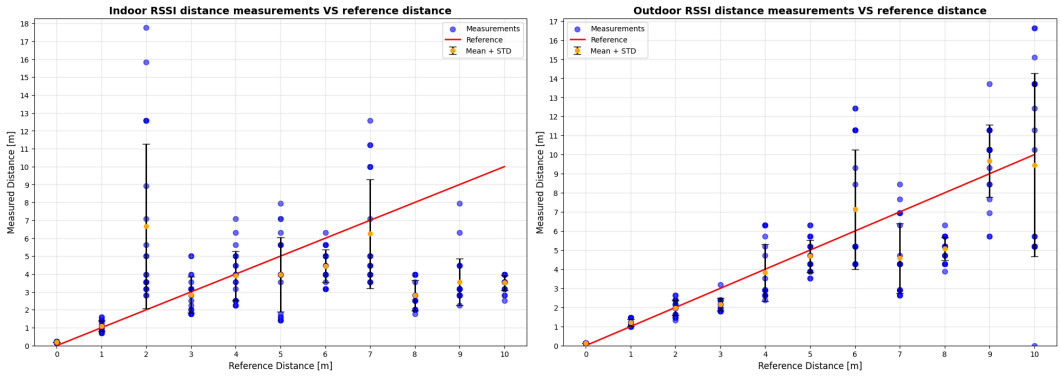
```
        dist = calculate_distance(-40, rssi, c_val)
        error = np.append(error, abs(dist - ref_dist[values.index(rssi)]))

    if np.mean(error) < min_error:
        min_error = np.mean(error)
        best_c = c_val

return best_c, min_error
```

Listing 1: Part loss index optimization function

The values tested range from 2 to 5 in increments of 0.125 for c. The distance and error are calculated with the average Received Signal Strength Indicator value for each reference distance and c values. Then the best c value is saved along with the corresponding error. These optimizations have improved the overall accuracy of the system, as shown in the Figure 50.



a) Optimized RSSI indoor b) Optimized RSSI outdoor

Figure 50: Indoor and outdoor test results for RSSI optimized values

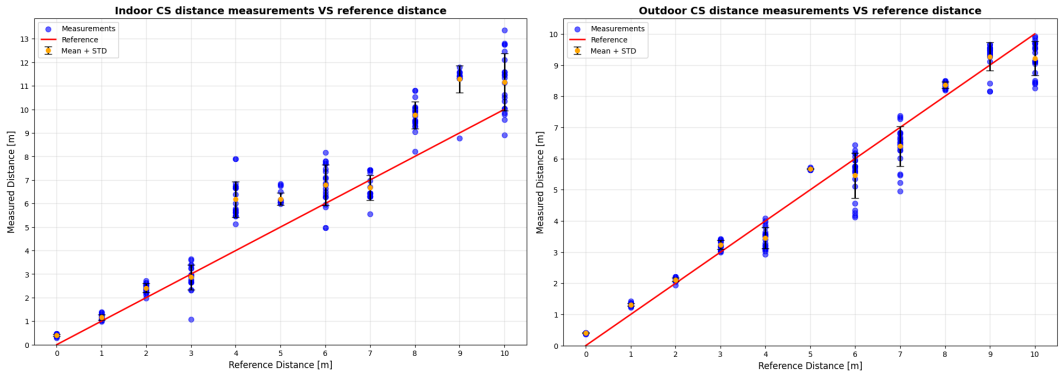
While optimization improves overall accuracy, it introduces significant deviation in indoor environments, demonstrating that single-parameter optimization can lead to undesirable behavior.

If optimization is performed, this should be kept in mind and adapted to the needs of the project.

Optimization improves accuracy, but it needs at least one known reference distance, so it cannot happen automatically in real time in the field. To use this method, researchers must set up test installations and update collar parameters by hand. Another option is to take one field measurement and use it to apply optimized parameters later during data analysis, moving distance calculations from the edge device to the analysis stage.

7.4.3.2 CS-based method

The Figure 51 shows the accuracy result for the CS method.



a) CS indoor b) CS outdoor

Figure 51: Indoor and outdoor test results for both CS methods

The Channel Sounding method demonstrated better precision. The measured values closely match the reference line and maintain minimal deviation from it.

- **Enhanced accuracy :** Measurements show significantly lower noise levels compared to previous methods, indicating that Channel Sounding effectively mitigates multipath and environmental interference.
- **Optimal environment :** Outdoor tests yielded the most consistent results, confirming the robustness of the phase-based measurement approach in open spaces.

7.4.4 Analysis

The results summarized in the Table 6 highlight clear performance differences between the RSSI-based and Channel Sounding distance measurement methods across indoor and outdoor environments. It also shows the difference between non-optimized and optimized RSSI distance calculation.

Test	Mean error [m]	Mean standard deviation [m]
Indoor RSSI test	3.29	0.42
Optimized indoor RSSI test	2.34	1.45
Indoor CS test	0.98	0.51
Outdoor RSSI test	1.90	2.35
Optimized outdoor RSSI test	0.85	1.41
Outdoor CS test	0.44	0.28

Table 6: Mean error and standart deviation of the 11 measures points

- **Standard RSSI :** Shows significant errors (3.29 m indoors, 1.90 m outdoors) and high instability due to multipath effects and interference.
- **Optimized RSSI :** Improved accuracy (2.34 m indoors, 0.85 m outdoors) through parameter tuning, though signal behavior remains inconsistent at larger distances.

- **Channel Sounding (CS) :** Demonstrates superior performance with minimal error (0.98 m indoors, 0.44 m outdoors) and high precision due to robust signal processing.

7.4.5 Conclusion

While RSSI is sufficient for simple proximity detection, Channel Sounding is the far more reliable technique for precise distance measurements. A hybrid approach using RSSI for initial coverage and CS for accurate positioning is the most effective balance of power and precision.

7.5 Detection range

The range is a important feature for proximity detection and distance measurement. This should allow for proper sizing and tuning of the developed system. To now the theoretical maximum range, both methods have been pushed to the limit in a outdoor test. The central has been place on a fix point, then the peripheral device was gradually moved away, while maintaining the line of sight, in an attempt to reach the detection limit.

7.5.1 Results

The maximum length reach is 100 meters. At this distance, both centrals were still detecting the peripheral device and providing distance estimates. The detection limit was not reached because the testing environment did not allow for achieving a greater distance with a clear line of sight. However, the distance achieved is significant and can serve as a basis for developing the system.

	Reference distance [m]	**Measured distance [m]**	**Error [m]**	**Standard deviation [m]**
RSSI	100	209.24	109.24	83.13
CS	100	94.03	5.97	0.08

** mean of 20 samples*

Table 7: RSSI and CS range results

In the Table 7, the mean measured distance and error are compiled, as well as the standard deviation. Unsurprisingly, the error and the standard deviation for the RSSI are important, but it still detects the peripheral. The CS always shows a consistent standard deviation and an accuracy of over 100 meters.

7.5.2 Conclusion

These tests validate the functionality of these two methods over a distance of 100 meters. They corroborate and support the observations made in the accuracy test. However, it is important to keep in mind that these tests are not representative of real-world conditions. They provide a theoretical maximum distance.

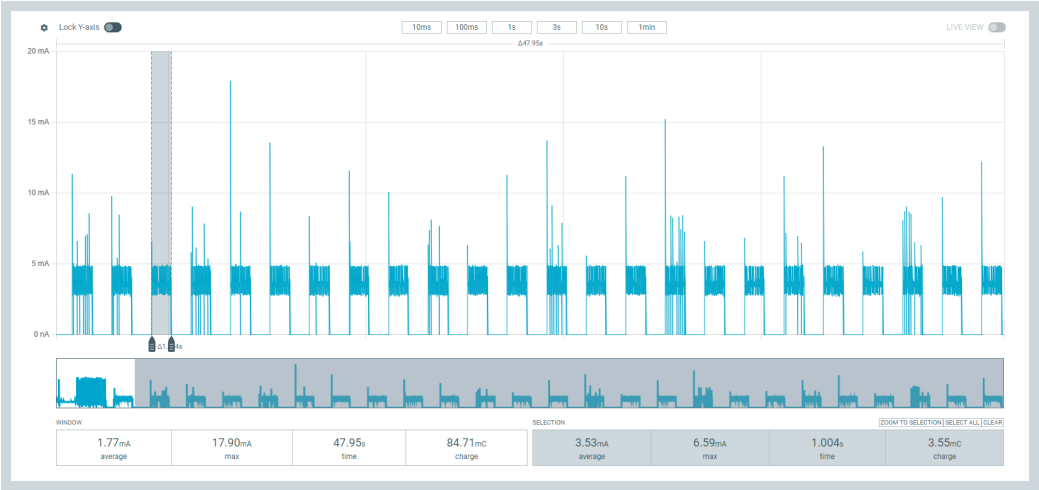
7.6 Power consumption

The goal of the power consumption test is to measure and compare the active radio time and energy per measurement for RSSI-based and Channel Sounding-based ranging. This test has been performed in an indoor environment. The devices were positioned a meter apart. A *Nordic Semiconductor Power Profiler Kit II* was used to measure current consumption from the central and peripheral components.

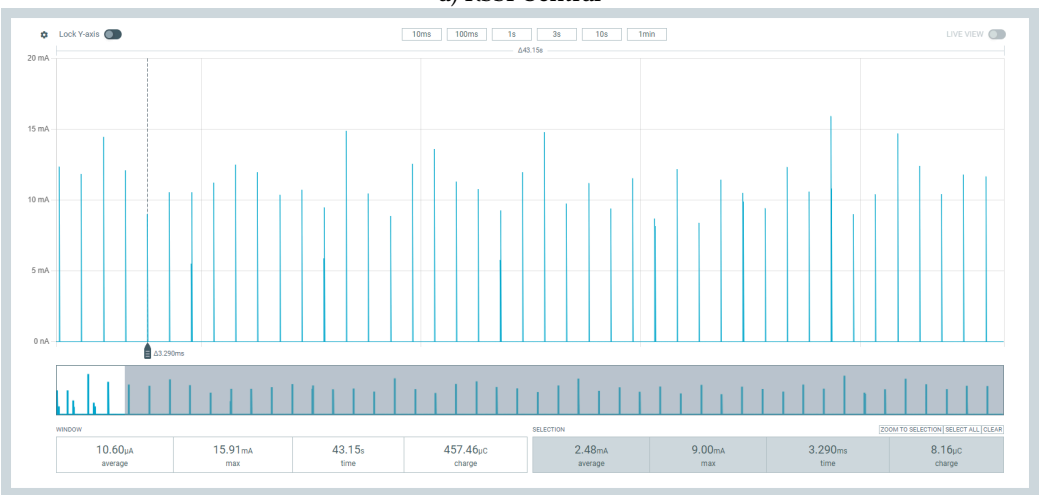
For both methods, a measurement is taken, then the device sleeps for a second. Only the time spent awake is relevant. It will allow to compare the consumption over the same time interval.

7.6.1 RSSI-based method

The Figure 52 shows the current consumption for both devices in RSSI mode. The central is scanning for a second and then sleeping. The peripheral is advertising every second. The central draws a current of 3.53mA for 1 second. The peripheral consume 2.43mA for 2.29ms. The energy used is linked to the active radio time.



a) RSSI Central



b) RSSI Peripheral

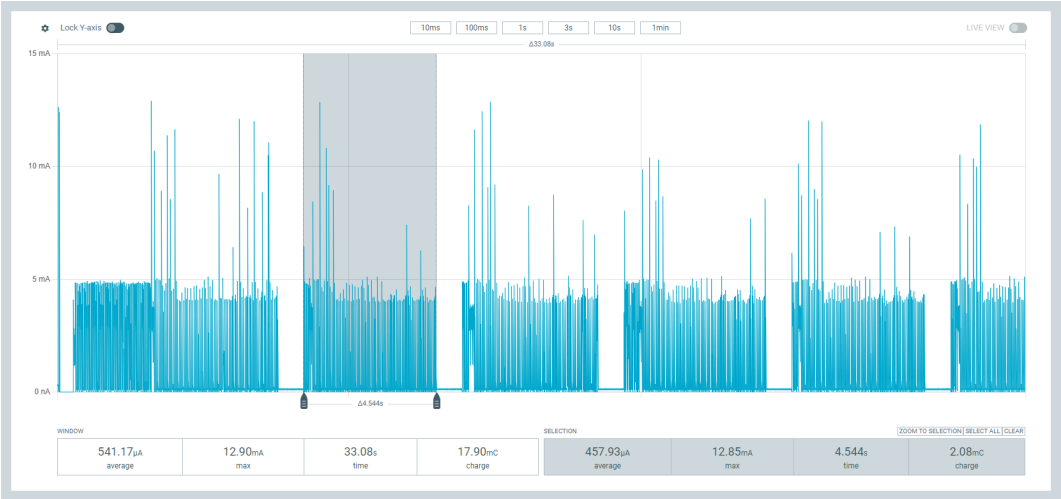
Figure 52: RSSI power consumption results

The RSSI-based method offers power predictability due to its passive nature.

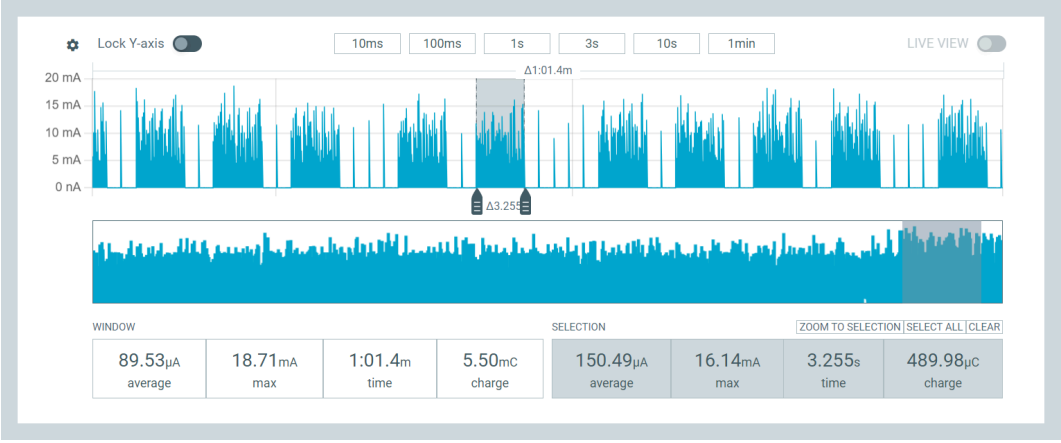
- **Passive interaction :** Devices do not require a formal connection to measure signal strength.
- **Constant power draw :** Energy consumption remains identical whether multiple devices are nearby or none are in range, as confirmed by current measurements in both scenarios.
- **System integration :** Since the collar already advertises every second to communicate with the mobile app, implementing RSSI only adds the overhead of the scanning phase to the existing power budget.
- **Efficiency :** This predictable consumption is a advantage for meeting the project's low-power requirements.

7.6.2 CS-based method

The Figure 53 shows the current consumption for the Channel Sounding. The initiator tries to connect to the reflector every second to perform a distance measurement. The connection and distance measure procedure takes almost 5 seconds.



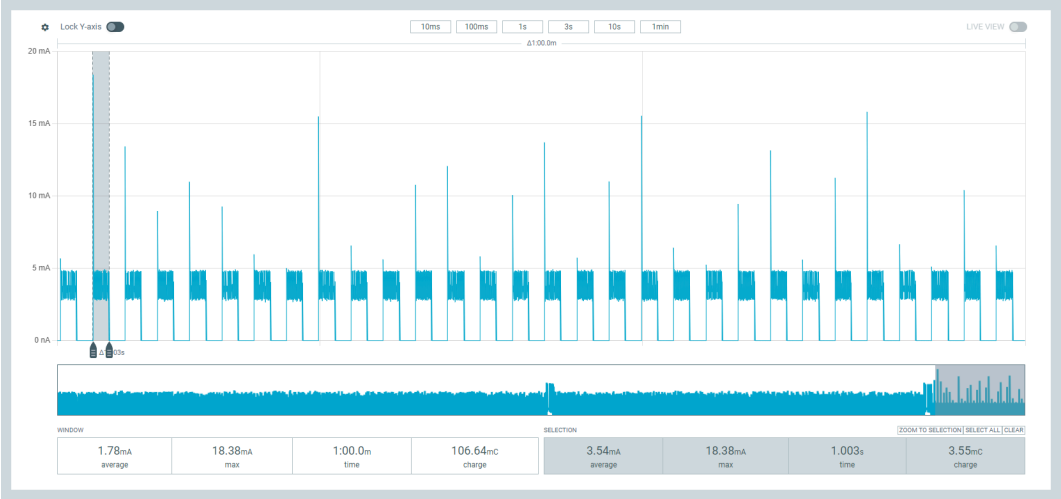
a) CS Initiator



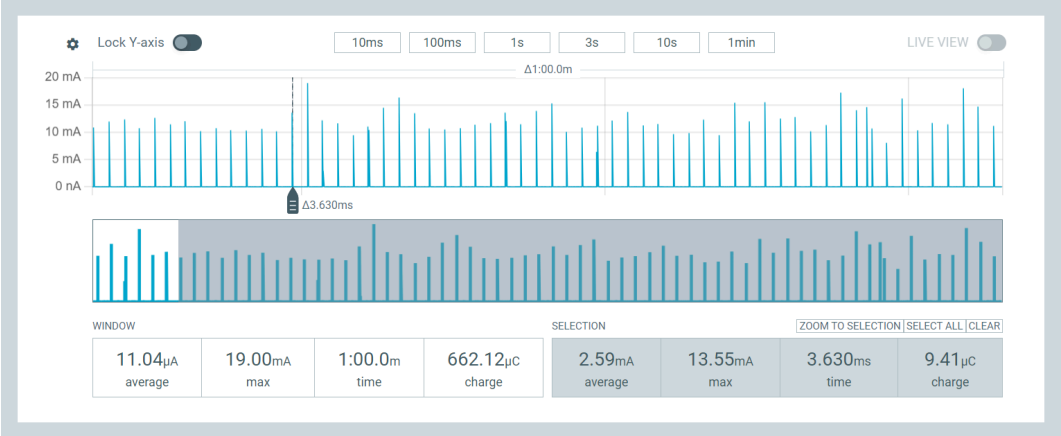
b) CS Reflector

Figure 53: CS power consumption results

During measurement period, the initiator consume a mean value of 457.93μA for 4.54 seconds when it make a measurement. The reflector use only 150.49μA for 3.26s. As the measurement require a active connection, the test has been performed the devices isolated from each other, represented in the Figure 54.



a) Isolated CS Initiator



b) Isolated CS Reflector

Figure 54: CS power consumption results with isolated devices

The initiator scan then for 1s every second. It consume a average current of 3.54mA when the radio is up. The reflector consume 2.59mA but only for 3.63ms. When it advertise at a rate of one advertisement per second, the average is only 11.04μA. When the devices are isolated, they act as a simple scanning/advertising device as the RSSI.

- **Extended measurement window :** The procedure for connection and distance estimation is significant, lasting nearly 5 seconds.
- **Role-based asymmetry :** The Initiator carries a much heavier power burden, consuming a mean current of 457.93 muA for 4.54 seconds per measurement.
- **Isolated consumption :** When the devices are isolated, the initiator continuously consumes energy attempting to establish a link. The global charge is even higher.

7.6.3 Analysis

7.6.3.1 Active ranging

The Table 8 provides an overview of consumption tests during active ranging.

Role	Radio current	Time	Mean current
Central RSSI	3.53 mA	1 s	1.77 mA
Peripheral RSSI	2.48 mA	3.29 ms	10.60 μA
Initiator CS	457.93 μA	4.5 s	541.17 μA
Reflector CS	150.49 μA	3.3 s	89.53 μA

Table 8: Consumption during active ranging

When devices are actively measuring distance, the total dual-role consumption for a single device performing both scan/initiation and advertising/reflection roles is calculated as follows:

$$\bar{I}_{\text{RSSI}} = \bar{I}_{\text{central}} + \bar{I}_{\text{peripheral}} = 1.77\text{mA} + 10.60\mu\text{A} \approx 1.78\text{mA} \tag{5}$$

$$\bar{I}_{\text{CS}} = \bar{I}_{\text{initiator}} + \bar{I}_{\text{reflector}} = 541.17\mu\text{A} + 89.53\mu\text{A} \approx 0.63\text{mA} \tag{6}$$

- **CS efficiency :** The Channel Sounding method is significantly more power-efficient, consuming approximately 0.63 mA for a full dual-role cycle.
- **RSSI overhead :** The RSSI method requires roughly 1.78 mA, making it nearly three times more energy-intensive than CS during active use.
- **Dominant consumer :** The high cost of RSSI is driven by the Central scanning phase, which requires long active radio windows to detect advertising packets.
- **Connection advantage :** Once a CS connection is established, the data exchange and distance calculation are less energy-intensive than continuous passive scanning, despite taking more time to complete.

7.6.3.2 Isolated operation

When no other devices are in range, both methods revert to a base scanning/advertising state. The power profiles in this state are nearly identical:

Role	Radio current	Time	Mean current
Central RSSI	3.53 mA	1 s	1.77 mA
Peripheral RSSI	2.48 mA	3.29 ms	10.60 μA
Initiator CS	3.54 mA	1 s	1.78 mA
Reflector CS	2.59 mA	3.63 ms	11.04 μA

Table 9: Consumption for isolated devices

When isolated operation, performing no measurement, the mean currents are for both 1.78 mA.

- **Identical profiles :** Power consumption is nearly identical between both methods when devices are unable to perform a measurement.

- **Central/Initiator draw :** Both methods consume roughly 1.78 mA while performing a continuous 1-second scan.
- **Peripheral/Reflector draw**: Both methods maintain a very low-power state, consuming between 10.60 muA and 11.04 muA while advertising.
- **Passive validation :** These results validate that RSSI is a truly passive measurement. It consumes the same amount of energy regardless of whether a device is detected.

7.6.4 Conclusion

The power consumption analysis shows that the Channel Sounding method is significantly more efficient than the RSSI method during full dual-role ranging cycles. This major efficiency disparity is primarily driven by the long active scanning window required by the Central RSSI component, which dominates the total energy budget. While both methods exhibit nearly identical standby consumption when devices are isolated, the connection-oriented measurement phase in CS introduces much lower operational overhead than continuous passive scanning.

Finally, while both modes allow for further power optimization by adjusting scanning and advertising intervals, this creates a direct trade-off between battery longevity and system responsiveness. Reducing the radio duty cycle to save energy significantly reduces the probability of successful proximity detection.

7.7 Maximum number of measures

7.7.1 RSSI-based method

The RSSI-based method is passive. It scans advertising packets and, based on the collected data, estimates the distance. In a single scan, the device can detect a large number of advertising devices. It has no hard-coded limit in the BLE specification. However, the number of devices that can be detected is limited by practical factors such as packet collisions, scan duration, interval, or memory.

7.7.2 CS-based method

On the other hand, the CS-based method is active. The initiator makes a connection with the reflector. This procedure takes about 4.5 seconds. With the provided sample, it is not possible to do multiple connections. Therefore, only one measurement can be taken at a time. This greatly limits the number of measurements possible when many devices are nearby. It is possible to measure distance with multiple devices simultaneously. But particular attention must be paid to Channel Sounding timing events management. It must allow sufficient time for the CS timing-activity.

Furthermore, the scalability of CS is fundamentally constrained by the requirement for an active ACL connection. While a BLE scanner can process advertising packets from an indefinite number of devices in parallel (limited only by radio congestion), a CS initiator is bound by the hardware and firmware limits of the Bluetooth controller. Most commercial BLE stacks enforce a hard limit on the maximum number of concurrent connections. Even if the stack permits multiple connections, the physical radio is a shared resource that must be time-sliced. As the number of target devices increases, the available airtime for each device decreases. The initiator must act as a scheduler, interleaving CS procedures

for each connected reflector. Consequently, the update rate for distance measurements scales inversely with the number of devices. This contrasts sharply with RSSI, where the overhead is minimal and no handshake is required.

7.7.3 Conclusion

While RSSI offers estimation suitable for high-density environments, CS trades scalability for precision and security, making it optimal for one-to-one interactions rather than crowd tracking.

7.8 Conclusion

7.8.1 Methods comparison

The following table summarizes the strengths and weaknesses of each technique:

Criterion	RSSI-based Method	CS-based Method
Accuracy	High variability and error	High, precise and stable
Power	Higher overhead due to passive scanning pulses	Lower overhead per measurement cycle.
Scalability	High, can monitor many devices in parallel	Limited, serial connections create bottlenecks
Complexity	Simple, uses standard advertising packets	High, requires BLE 6.0 and ACL connections.
Best use case	High-density proximity detection	Precise positioning and secure ranging

Table 10: Comparison between RSSI and Channel Sounding methods

7.8.2 Method selection

The experimental results show that the RSSI-based method is the best option for this application. Its main advantage is that it can scale well in high-density environments.

Although this method provides less precise distance estimates and can be affected by environmental noise, it works very well for detecting proximity, the main goal of this tracking project. To keep the system running for a long time, it is important to optimize parameters and keep power use low. If more accurate tracking is needed in the future, the system can be upgraded to a hybrid approach with CS.

7.9 RSSI based strategy

To detect and be detected using Bluetooth Low Energy, each device alternates between an advertising phase (during which it can be detected) and a scanning phase (during which it can detect).

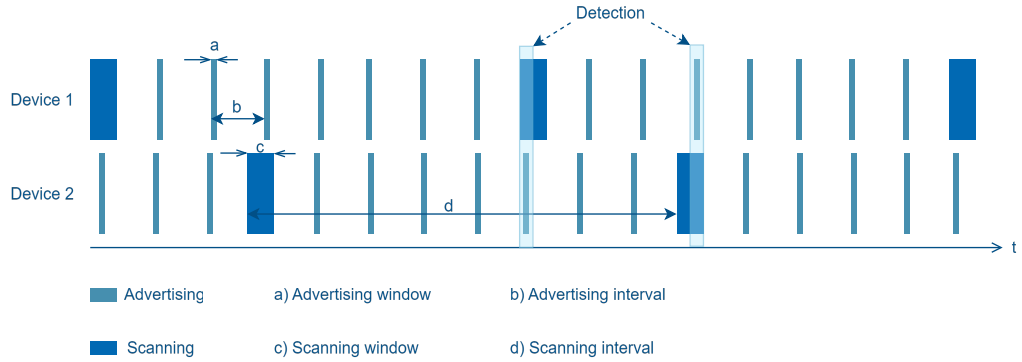


Figure 55: Devices proximity detection strategies

When advertising packets are sent during a scanning period, the scanning device performs a detection. It therefore knows that he is near a device. To ensure detection, these two periods must overlap within a reasonable timeframe. This was done by playing with these three parameters :

- **Scanning window (c)** : How long each scanning session lasts once started.
- **Scanning interval (d)** : How often a device starts a scan for nearby signals.
- **Advertising interval (b)** : How often a device broadcasts its advertisement packets.

The probabilities of encounter, based on the scanning and advertising intervals and windows, can be calculated and then confirmed through real-world tests. These calculations are detailed in Section 9.4.

8 Implementation

The following chapter details the technical implementation of the *nRF Monkey* firmware, focusing on the migration to the nRF54L15 architecture. It outlines the development environment, the multi-threaded system design under Zephyr RTOS and the configurations for the digital microphone and BLE proximity subsystems.

8.1 Development environment & tools

This section will detailed the development methodology, environment and the hardware used during this thesis.

8.1.1 Development methodology

The work to be done has been divided into work packages, each with tasks and subtasks. The work is then divided into manageable units. This approach allows to structure the project. The completed and remaining tasks can then be easily tracked. It aims to significantly improve project's overall performance.

Four main work packages have been defined :

- **WP0 - Project management & reporting :** Focuses on the administrative oversight of the project, including timeline management and the final documentation of all technical findings.
- **WP1 - Microcontroller & microphone upgrade :** Involves integrating the new MCU and microphone while optimizing the system for minimal power consumption.
- **WP2 - BLE proximity detection :** Focuses on developing low-power proximity algorithms and a command set to allow remote monitoring via Bluetooth Low Energy.
- **WP3 - Global power analysis :** Consists of testing the complete system under realistic scenarios to validate battery life and operational efficiency.

The detailed work packages are available in Section AC.

8.1.2 Version control

In addition to this approach, version control has been used. It allows for tracking and managing changes made to the firmware code. The main advantages of this process are :

1. A complete, long-term history of changes to each file.
2. Branching and merging allows an individual to work independently on separate change logs.

This work is based on an existing project that already uses Git version control. This ensures continuity and makes it easy to differentiate the work done within this project.

As the work was carried out by a single person, a simple branching model was used. In contains a *main* branch (production firmware), a *dev* branch (firmware under development) and finally *features* branches created from *dev* (adding features to the firmware under development) as shown in Figure 56.

Source: https://www.atlassian.com/git/tutorials/comparing-workflows#!workflow-gitflow

Figure 56: GIT flow

8.1.3 Development environment

The Table 11 list all the firmware and tools used during this thesis.

Software	Version	Description
VS Code	v3.107.1	IDE
nRF Connect Toolchain	v3.0.1	The collection of compiler (GCC), linkers and build tools required to cross-compile for Nordic hardware.
nRF Connect SDK (NCS)	v3.0.0	The Zephyr-based firmware framework containing libraries, drivers and samples for nRF54 development.
Zephyr RTOS	v4.0.99	The real-time operating system kernel used by the SDK.
nRF Connect for VS Code	v3.0.0	Primary IDE extension for Zephyr RTOS development, CMake build management and device flashing.
nRF Connect Desktop	v5.2.1	Central hub for Nordic tools including Bluetooth Low Energy, Programmer and Power Profiler apps.
nRF Connect for Android	v4.29.1	Generic tool that allows to scan, advertise and explore Bluetooth Low Energy device.
Audacity	v3.7.7	Open-source audio editor used for analyzing microphone samples, normalization and SPL calibration.
J-Link RTT Viewer	v8.70	Real-time terminal for viewing high-speed debug logs via the J-Link interface without stopping the MCU.
Waveform	v3.24.4	Logic analyzer or oscilloscope firmware used for hardware signal debugging (e.g., I2S/SPI lines).
Fork	2.15.3.0	Visual Git client used for version control, branching and managing the project repository.
Git	2.44.0	Version control system.
Python	v3.12.10	High-level interpreted language used for West build scripts, data processing and automation tasks.
West	v1.2.0	Zephyr's meta-tool for managing projects, dependencies and build system integration.
Ellisys Bluetooth Analyzer	v5.0.9433	Tools that provides intuitive understandings of complex protocol and RF behaviors and flexible configuration.

Table 11: Summary of the project development environment

8.2 System architecture

The system is built on Zephyr RTOS. It allows a multi-threaded execution model where tasks are isolated. A simplified state diagram that represents the firmware flow is presented below. The key elements of this design will be presented in the next chapters.

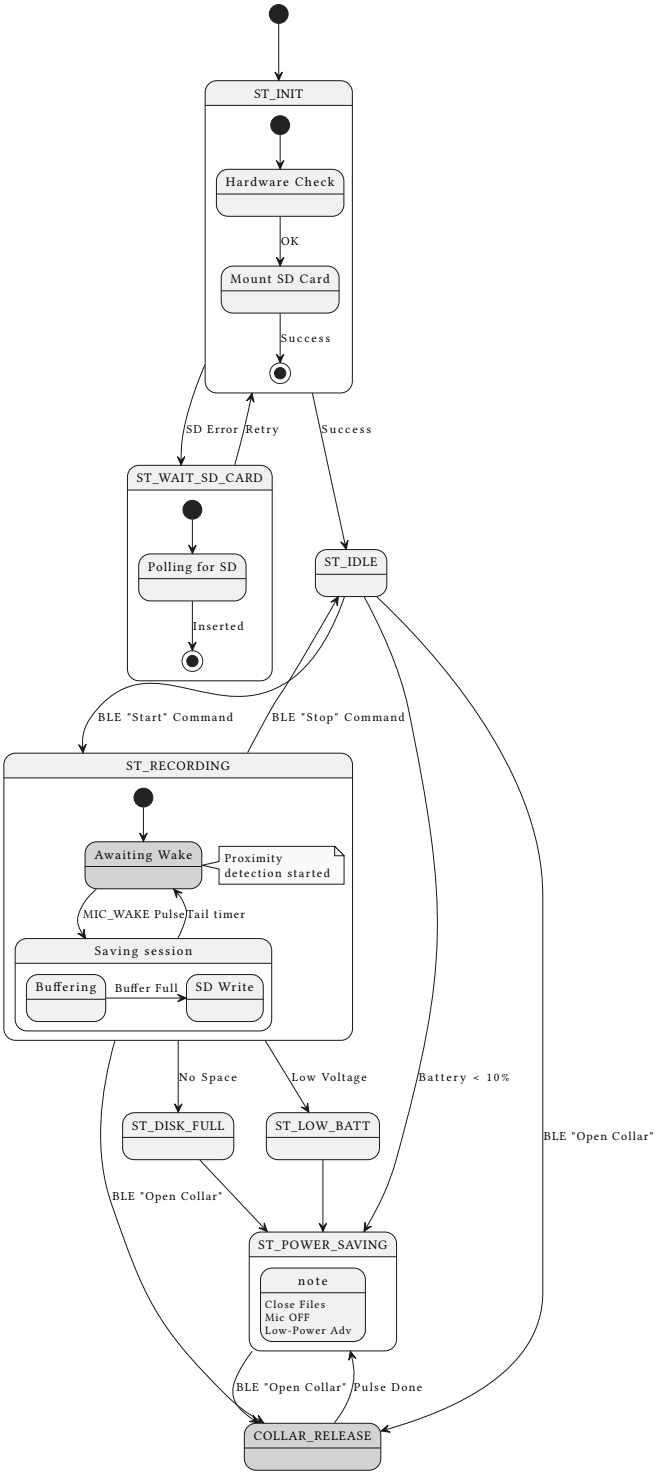


Figure 57: Global state machine

8.3 Thread architecture

Multiple threads are used to make this system work. It has 6 different threads. All the threads are detailed in the followings tables.

8.3.1 Main thread

Parameter	Details
Priority	High (5)
Stack size	4096 bytes
Purpose	Performs initial hardware/service startup.
Trigger	System boot.
Operations	Initialize the system, handle reset and power saving mode.
Sleep logic	Sleeps during the main loop while waiting for state updates.

Table 12: Main thread

8.3.2 FATFS mount thread

Parameter	Details
Priority	Low (7)
Stack size	4096 bytes
Purpose	Handles mounting and initialization of the SD Card file system.
Trigger	After initialization.
Operations	Interfaces with Zephyr Disk Access API to mount FATFS partition.
Sleep logic	Sleeps while waiting to handle SD card removal.

Table 13: FATFS mount thread

8.3.3 Audio recorder thread

Parameter	Details
Priority	Medium (6)
Stack size	4096 bytes
Purpose	Manages high-speed capture of raw digital audio from the microphone.
Trigger	Woken by AAD interrupt from the microphone.
Operations	Triggers I2S, performs 24-bit to 16-bit PCM conversion, manages double-buffering.
Sleep logic	Blocks on semaphore until AAD wake event.

Table 14: Audio recorder thread

8.3.4 Audio file store thread

Parameter	Details
Priority	Low (7)
Stack size	4096 bytes
Purpose	Offloads SD card writing process from acquisition thread.
Trigger	Buffer full signal.
Operations	Opens .WAV files and writes PCM blocks from RAM to SD card.
Sleep logic	Blocks until a full buffer of audio data is ready for storage.

Table 15: Audio file store thread

8.3.5 BLE handler thread

Parameter	Details
Priority	Medium (6)
Stack size	4096 bytes
Purpose	Initializes Bluetooth stack and manages SNES GATT services.
Trigger	After initialization.
Operations	Sets advertising parameters and handles GATT read/write events.
Sleep logic	Sleeps between scanning events

Table 16: BLE handler thread

8.3.6 Proximity detection store thread

Parameter	Details
Priority	Low (7)
Stack size	4096 bytes
Purpose	Logs peer device IDs and RSSI values to the SD card.
Trigger	Proximity buffer full or flush timer expiry.
Operations	Appends proximity records to .DAT files.
Sleep logic	Blocks until a full buffer of nearby devices is ready for storage.

Table 17: Proximity detection store thread

The treads sequence will be detailed in the following chapters.

8.3.7 Priority and scheduling strategy

The priority assignment is a critical part of a multi-threaded system running on an RTOS. This system uses a preemptive priority-based scheduling model.

Thread priority

- **High Priority (Main thread) :** It handles booting, resetting and low-power mode. It needs to preempt all other threads. The small amount of logic it contains is critical and must override the rest of the program. However, it is only executed in very specific cases. Therefore, this thread will not preempt other threads for extended periods during normal operation.
- **Medium Priority (BLE and audio recorder thread) :** They contain the heart of the program. They buffer the audio and proximity data. They are also both producers. For this reason, the need to have a higher priority than their associate consumer. If one of these threads is preempted for too long, data cannot be moved to RAM, leading to lost data that cannot be recovered.
- **Low Priority (BLE proximity, audio store and FATFs mount thread) :** Although writing to the SD card is data-heavy, these threads act as consumers in the double-buffering pattern. Because the data is already in RAM, these operations can be delayed by several milliseconds without causing data loss.

8.3.8 Inter-thread communication

The firmware is separated into distinct threads, but they interact with each other in different ways. Several mechanisms are used to achieve this.

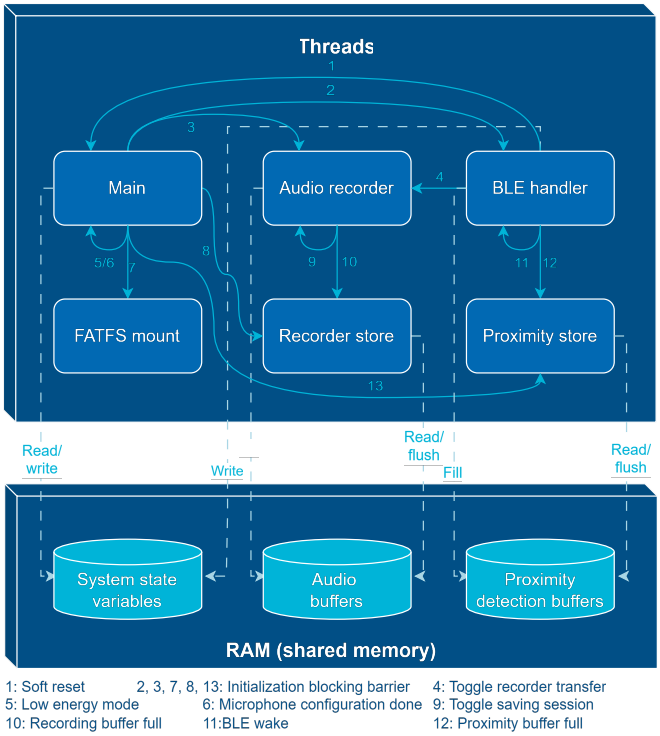


Figure 58: Threads and there interactions

The Figure 58 summarize the inter-thread communication. The plain arrows represent the semaphore. The dotted arrows represent the access to shared memory. The different mechanisms are explained in the following points.

8.3.8.1 Initialization blocking barrier

To ensure a stable startup, the main thread utilizes semaphores as a blocking barrier to prevent secondary threads from executing before the hardware is ready.

```
K_SEM_DEFINE(thread_i2s_busy_sem, 1, 1);
K_SEM_DEFINE(thread_recorder_store_busy_sem, 1, 1);
K_SEM_DEFINE(thread_ble_busy_sem, 1, 1);
K_SEM_DEFINE(thread_fatfs_busy_sem, 1, 1);
K_SEM_DEFINE(thread_proximity_store_busy_sem, 1, 1);
```

Listing 2: Initialization blocking barrier

After this initialization, these semaphores serve as a diagnostic tool to verify if a specific thread is currently running and to prevent redundant re-initializations.

8.3.8.2 Synchronization semaphore

To coordinate concurrent tasks, the system uses semaphores for synchronization. These semaphores act as event triggers, allowing threads to remain in sleep until the semaphore is released. The following categories outline the semaphores used.

Global system semaphores

- **Software reset :** Used to manage system resets, triggered by microphone configuration changes or recording toggle.

```
K_SEM_DEFINE(reset_sem, 0, 1);
```

- **Low energy mode :** Used to signal the transition into low energy mode.

```
K_SEM_DEFINE(low_energy_mode_sem, 1, 1);
```

- **Microphone configuration done :** Synchronizes the microphone initialization process, ensuring the system waits for the confirmation pulse after sending configuration registers.

```
K_SEM_DEFINE(mic_config_done_sem, 0, 1);
```

Audio subsystem semaphores

- **Toggle recorder transfer :** Use to start a recording session.

```
K_SEM_DEFINE(recorder_toggle_transfer_sem, 0, 1);
```

- **Toggle saving session :** Triggers the transition from awaiting to active saving.

```
K_SEM_DEFINE(recorder_toggle_saving_sem, 0, 1);
```

- **Audio buffer full :** Signals the audio file store thread that an audio RAM buffer is full and ready to be written to the SD card (see Section 8.3.8.3).

```
K_SEM_DEFINE(recorder_file_access_sem, 0, 1);
```

Bluetooth & proximity semaphores

- **Proximity buffer full :** Signals the proximity file store thread that an proximity RAM buffer is full and ready to be written to the SD card (see Section 8.3.8.3).

```
K_SEM_DEFINE(proximity_file_access_sem, 0, 1);
```

- **BLE thread wakeup :** Signals the BLE thread to wakeup, after an advertise data change or a low power mode (see Section 8.3.8.3).

```
K_SEM_DEFINE(ble_wakeup_sem, 0, 1);
```

8.3.8.3 Producer/consumer pattern

The nRF Monkey firmware implements a Producer/Consumer pattern to manage the data rate from peripherals. With this pattern, no data is missed during SD writing. This pattern is represented in the Figure 59.

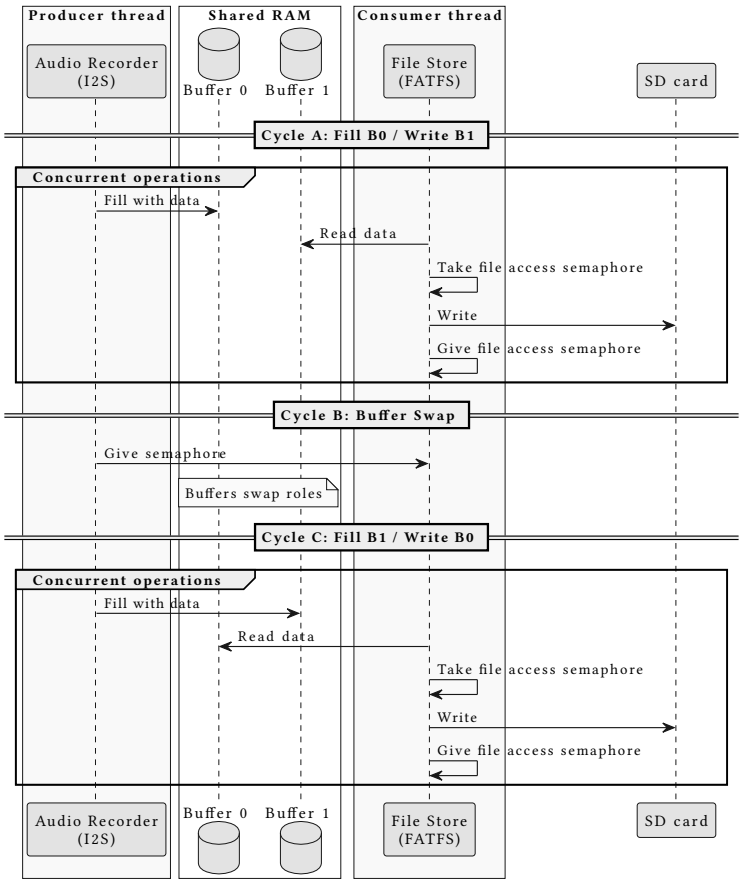


Figure 59: Producer/Consumer pattern for the sound recording

The system has two separate producer/consumer pairs : audio and proximity. However, they both write on the same SD card. To avoid data corruption, access to the SD card is protected using a mutual exclusion lock. With that, only one consumer can write to the SD card at a time. The writing process is therefore guaranteed not to be interrupted.

Lock free synchronization

To maintain uninterrupted recording, the system uses a lock-free synchronization strategy rather than mutual exclusion locks. This prevents high-latency SD card write operations from blocking the producer thread, which would otherwise result in buffer overflows and data loss.

Safety is instead guaranteed by a significant temporal margin between production and consumption. For audio recording at 16 kHz (16 bit), the data rate is:

$$\text{data}_{\text{rate}} = 16000 \text{ samples/s} * 2 \text{ bytes/sample} = 32 \text{ kB/s} \tag{7}$$

With a 32 kB buffer, the fill time is $t_{\text{fill}} = 1$ s. Given a conservative SD write speed of 10 MB/s for Class 10 cards, the write time is:

$$t_{\text{write}} = \frac{32 \text{ kB}}{10000 \text{ kB/s}} = 3.2 \text{ ms} \tag{8}$$

This results in a production/consumption ratio of approximately 320:1, providing a robust safety margin. A similar principle applies to proximity detection, even at a high density of 64 devices per minute, the 6.5 kB buffer takes almost 7 minutes to fill, while the write operation completes in less than 1 ms.

These theoretical margins ensure that buffer access conflicts are statistically negligible. The real write times are verified in the Section 9.1.

8.3.9 Interruptions

In addition to threads, the system uses hardware interrupts to manage real-time events and power transitions. All the interrupts used are listed and detailed at this point.

Interrupt	Module	Description
Mic wake	Recorder	Detects T5848 config pulse and sound triggers event to wake the SoC.
Battery low	Main	Fired by a comparator if voltage drops to trigger a safe shutdown.
Card detect	SD card	Detects the SD card presence.

Table 18: Summary of system and peripheral interrupts

System interruptions are not listed in this table.

8.4 Zephyr RTOS configuration

8.4.1 Kconfig

In the nRF Connect SDK and Zephyr RTOS, Kconfig is the system used to statically configure the software at compile time. It allows enabling or disabling specific features, drivers and libraries without modifying the source code. It brings benefits :

- **Conditional compilation :** Code for disabled features is completely excluded from the final binary. It is critical for saving memory on the project device.
- **Dependency management :** Kconfig ensures that if a high-level feature is enabled, all necessary low-level dependencies are enabled as well.
- **Visual configuration :** The Guiconfig or the nRF Kconfig GUI in VS Code can be used to browse and search for options.

A Kconfig file is therefore used in this project for the various reasons mentioned above. The configurations added for this project are listed and detailed below.

I2S & audio module

Config name	Default	Description
I2S_SAMPLE_FREQUENCY	16000	Sampling frequency in Hz.
I2S_SAMPLE_FREQ_DIVIDER	10	Divider for DMA block size.
I2S_SAMPLE_BIT_WIDTH	24	Bit width of each I2S sample.
I2S_BYTES_PER_SAMPLE	4	Memory allocation per sample in bytes.
I2S_NUMBER_OF_CHANNELS	1	Number of audio channels.
I2S_INITIAL_BLOCKS	2	Number of pre-allocated DMA blocks.
I2S_MIC_INPUT_GAIN	1	Software amplification gain.
I2S_MIC_INPUT_DIVIDER	1	Software amplification divider.
I2S_RECORD_AUTO_START	y	Automatically start recording at boot.

Table 19: I2S and audio module configuration parameters

Microphone module

Config name	Default	Description
MIC_AAD_A_SELECT	0x08	Analog Activity Detection mode.
MIC_AAD_A_LPF	0x0A	Analog LPF setting (Low Pass Filter).
MIC_AAD_A_TH	0x05	Analog threshold for activity detection.
MIC_AAD_D_SELECT	0x01	Digital Activity Detection mode.
MIC_AAD_D_ALGO_SEL	0x03	Digital algorithm (Absolute/Relative).
MIC_AAD_D_FLOOR	0xA0	Digital noise floor setting.
MIC_AAD_D_REL_PULSE_MIN	0xC8	Minimum pulse duration for relative.

MIC_AAD_D_ABS_PULSE_MIN	0x7D0	Minimum pulse duration for absolute.
MIC_AAD_D_REL_TH	0x8F	Digital Relative Threshold.
MIC_AAD_D_ABS_TH	0x1E0	Digital Absolute Threshold.
MIC_RECORDING_TAIL_MSEC	10000	Recording tail duration after trigger [ms].

Table 20: Microphone AAD configuration parameters

Storage module

Config name	Default	Description
STORAGE_SAMPLE_BIT_WIDTH	16	Bit width used when saving to SD card.
STORAGE_BYTES_PER_SAMPLE	3	Bytes used per sample for storage.
STORAGE_TIME_DIFF_THRESHOLD	3600	Seconds before opening a new file.
RECORDER_STORAGE_FILE_NAME	rec	Base filename for audio recordings.
PROXIMITY_STORAGE_FILE_NAME	prox	Base filename for proximity logs.
STORAGE_FILE_EXTENSION	dat	File extension for stored data.

Table 21: SD card and storage configuration parameters

BLE & proximity module

Config name	Default	Description
BT_MIN_CONN_INTERVAL	495	Min connection interval (618.75 ms).
BT_MAX_CONN_INTERVAL	750	Max connection interval (937.5 ms).
BT_ADV_INTERVAL_MS	300	BLE Advertising interval [ms].
BT_PROXIMITY_MONITORING	y	Enable proximity monitoring service.
BT_PROXIMITY_ENABLE_AUDIO_SAVING	n	Start saving session when device detected.
BT_PROXIMITY_ENABLE_AUDIO_RSSI_MIN	−50	Min RSSI to start saving session.
BT_SCAN_INTERVAL_MS	60000	BLE Scan interval [ms].
BT_SCAN_WINDOW_MS	1400	BLE Scan window [ms].
BT_PROX_BUFFER_SIZE	512	Entries in proximity storage buffer.
BT_PROX_FLUSH_TIMEOUT	10	Timeout in minutes to flush to SD card.

Table 22: BLE and proximity monitoring configuration parameters

All these values can then be modified in the prj.conf or used directly in the code with the structure *CONFIG_ + config name*.

8.4.2 Device tree

A device tree uses a structured data format to describe a board's hardware configuration. Instead of embedding all this information directly in C code, the device tree stores it in a hardware description file that the build process handles [31]. This separation offers several benefits:

- **Hardware abstraction and portability :** allow board changes by updating only the configuration overlay.
- **Clean application code :** No hard-coded pin numbers, addresses or constants are used. The device tree ensures those details live in a centralized location and the code simply queries them.
- **Shared configuration for SDK and drivers :** Drivers in Zephyr rely heavily on the device tree. Enabling an overlay not only informs the application but also notifies the appropriate driver subsystem. This approach ensures consistency and eliminates redundant configuration.

Application portability will be key in this project. Since the board changes from the first version of the collar (*nRF5340* to *nRF54L15*), the logic of the existing overlay can be updated for the *nRF54L15*.

8.4.2.1 Hierarchy

The device tree is hierarchical, similar to a file system. At the root level (/), the top level definitions and inside are nodes representing peripherals.

Example

```
// Root node (starting point of hardware description)
/ {
  // Child node: Groups devices with a common driver
    leds {
      compatible = "gpio-leds";

      // Label : used in code via macros like DT_ALIAS(...) (led0)
      // Node name : unique name in the hierarchy (led_0)
      led0: led_0 {
      // Properties: Key-value pairs defining configuration
      gpios = <&gpio0 13 GPIO_ACTIVE_LOW>;  // links to the GPIO controller
      label = "System Status LED";
    };
  };
```

```
  // Aliases: Standardized nicknames for the application code, allowing C code
to access hardware via a generic name like status-led
  aliases {
  status-led = &led0;
  };
};

// Overlay/reference: Using & modifies a node defined in another file, useful
for changing status or pin assignments without editing the base DTS
&uart0 {
  status = "okay";
  current-speed = <115200>;
};
```

Listing 3: Device tree example

8.4.2.2 Overlay

Overlays are the recommended method to customize a board's device tree configuration for an application. Each board includes a default .dts file that defines its hardware layout. Instead of changing this file directly, an .overlay file can be created in the application directory to adjust or extend the device tree for your project. For this project, the created overlay file is named *nrf54l15dk_nrf54l15_cpuapp.overlay*, which contains the customizations specific to the *nRF54L15* board.

8.4.3 Project overlay

This section summarizes the device tree configuration for the prototype PCBs, based on the *nRF54L15* SoC. The complete file is available in the provided code resource.

The first part of the file is dedicated to deactivating unused hardware. To minimize power consumption, all unused equipment are deactivated. This is to prevent current leakage. The LEDs, buttons, data bus, timer and NFC are disabled using the property:

```
status = "disabled";
```

SPI (SD card)
A SPI interface is used to communicate with the SD card. SPI is a high-speed, synchronous serial communication protocol used for data exchange between a master controller (SoC) and a peripheral device (SD card). It typically utilizes four wires: Clock (SCK), Master Out Slave In (MOSI), Master In Slave Out (MISO) and Chip Select (CS). It enables full-duplex data transmission synchronized by a clock signal.

The Listing 4 represents the SPI node. The following code snippet demonstrates how to configure and use the SPI interface for communication with the SD card.

```
&spi20 {
  status = "okay";
  compatible = "nordic,nrf-spim";
  pinctrl-0 = <&spi20_default>; // Pins used during active operation
  pinctrl-1 = <&spi20_sleep>; // Pins used during sleep operation
  pinctrl-names = "default", "sleep";
    cs-gpios = <&gpio2 6 GPIO_ACTIVE_LOW>; // Define the CS pin
  // Child node representing the physical SD card slot on the SPI bus
  sdhc0: sdhc@0 {
  compatible = "zephyr,sdhc-spi-slot";
  reg = <0>;
  status = "okay";
  // Virtual node for the disk driver to interface with the File System
  mmc {
   compatible = "zephyr,sdmmc-disk";
   disk-name = "SD";
   status = "okay";
  };
  spi-max-frequency = <24000000>; // Set bus speed (24 MHz)
 };
};
```

Listing 4: SPI overlay node

The communication protocol signals are mapped to the pin control. It uses the following pins :

```
&pinctrl {
  spi20_default: spi20_default {
    group1 {
        psels = <NRF_PSEL(SPIM_SCK,  1, 3)>,  // SD_SCK
                <NRF_PSEL(SPIM_MISO, 1, 4)>,  // SD_MISO
                <NRF_PSEL(SPIM_MOSI, 1, 9)>;  // SD_MOSI
    };
  };

  spi20_sleep: spi20_sleep {
    group1 {
        psels = <NRF_PSEL(SPIM_SCK,  1, 3)>,  // SD_SCK
                <NRF_PSEL(SPIM_MISO, 1, 4)>,  // SD_MISO
                <NRF_PSEL(SPIM_MOSI, 1, 9)>;  // SD_MOSI
        low-power-enable;
    };
  };
};
```

Listing 5: SPI pin control

I2S (microphone)

The microcontroller communicates with the microphone using Inter-IC Sound protocol. It is a synchronous serial communication protocol designed for transmitting digital audio data. It uses a Bit Clock (SCK) and a Word Select (LRCK) signal for channel synchronization and an input data line (SDIN) and an output data line (SDOUT) to ensure audio transfer.

The I2S node is contained in the Listing 6. The main features are explained in this code snippet.

```
i2s_rxtx: &i2s20 {
  compatible = "nordic,nrf-i2s";
  status = "okay";
  pinctrl-0 = <&i2s20_default>; // Pins used during active operation
  pinctrl-1 = <&i2s20_sleep>;   // Pins used during sleep operation
  pinctrl-names = "default", "sleep";
};
```

Listing 6: SPI overlay node

The communication protocol signals are mapped to the pin control. It uses the following pins :

```
&pinctrl {
  i2s20_default: i2s20_default {
    group1 {
      psels = <NRF_PSEL(I2S_LRCK_M, 1, 14)>,  // MIC_WS
              <NRF_PSEL(I2S_SCK_M, 1, 2)>,    // MIC_SCK  (for data transfer)
              <NRF_PSEL(I2S_SDIN, 1, 13)>;    // MIC_SDIN
    };
  };

  i2s20_sleep: i2s20_sleep {
    group1 {
      psels = <NRF_PSEL(I2S_LRCK_M, 1, 14)>,  // MIC_WS
              <NRF_PSEL(I2S_SCK_M, 1, 2)>,    // MIC_SCK  (for data transfer)
              <NRF_PSEL(I2S_SDIN, 1, 13)>;    // MIC_SDIN
      low-power-enable;
    };
  };
};
```

Listing 7: I2S pin control

As the I2S is only use to received data, no output pin is mapped in this project.

Input pins

The system used input GPIOs. They are also defined in the device tree. The three inputs are :

```
monkey_inputs {
  compatible = "gpio-keys";
  cd: cd_0 {
    gpios = <&gpio1 5 GPIO_ACTIVE_LOW>;   // SD_CD
    label = "Card Detect";
  };

  lb: low_batt_0 {
    gpios = <&gpio1 7 GPIO_ACTIVE_HIGH>;  // LOW_BATT
    label = "Low Batt status";
  };

  wake: wake_0 {
    gpios = <&gpio1 12 GPIO_ACTIVE_HIGH>; // MIC_WAKE
    label = "Mic Wake";
  };
};
```

Listing 8: Input pins

These 3 pin are used with interrupts.

Output pins

Finally, the system outputs signals using GPIO. This project has 7 output pins.

```
monkey_outputs {
  compatible = "gpio-leds";
  bc: burn_collar_0 {
    gpios = <&gpio2 1 GPIO_ACTIVE_HIGH>;  // BURN
    label = "Collar Burn Activator";
  };

  sd: sd_enable_0 {
    gpios = <&gpio1 8 GPIO_ACTIVE_LOW>;   // SD_EN
    label = "SD Card Switch";
  };

  clk: clk_0 {
    gpios = <&gpio1 2 GPIO_ACTIVE_HIGH>;  // MIC_CLK (for config)
    label = "Mic clock";
  };

  thsel: thsel_0 {
    gpios = <&gpio2 0 GPIO_ACTIVE_HIGH>;  // MIC_THSEL
    label = "Mic THSEL";
  };

  thseloe: thsel_oe_0 {
    gpios = <&gpio2 2 GPIO_ACTIVE_HIGH>;  // MIC_OE_THSEL
    label = "Mic THSEL output enable";
  };
```

```
  mic: mic_enable_0 {
    gpios = <&gpio1 10 GPIO_ACTIVE_LOW>;  // MIC_EN
    label = "Mic Switch";
  };

  oe: mic_oe_0 {
    gpios = <&gpio1 11 GPIO_ACTIVE_HIGH>; // MIC_OE
    label = "Mic output enable";
  };
};
```

Listing 9: Output pins

Aliases
All the GPIO pins have dedicated aliases. The Listing 10 shows all the aliases.

```
  aliases {
   cd0  = &cd;          // Card detect
   lb0  = &lb;          // Low battery
   bc0  = &bc;          // Burn collar
   sd0  = &sd;          // SD enable
   clk0 = &clk;         // MIC CLK
   thsel0 = &thsel;     // MIC THSEL
   thseloe0 = &thseloe; // MIC THSEL output enable
   wake0 = &wake;       // MIC wake
   mic0 = &mic;         // MIC enable
   oe0 = &oe;           // MIC output enable
  };
```

Listing 10: Device tree aliases

Now the GPIO can be accessed in the C code using their generic name, as in this code snippet.

```
#define MIC_CLK_NODE    DT_ALIAS(clk0)
struct gpio_dt_spec mic_clk_gpio = GPIO_DT_SPEC_GET(MIC_CLK_NODE, gpios);
```

Listing 11: Aliases access from C code example

8.4.4 Flash partitions

In the previous project, the flash memory partitions were defined in the device tree. Two partitions were defined :

- Storage partition
- User partition

With the previous method of referencing partitions via labels, undeclared identifier errors occur. The code was updated to use the Partition Manager static configuration and the integrated C API for the *nRF54L15* [32].

Incorrect amount of gaps error
The partition manager reported an error due to multiple gaps in the static flash configuration:

```
Partition manager failed: Incorrect amount of gaps found in static
configuration.
Gaps found (2): 0x0-0xfe000 0x100000-0x165000
```

Listing 12: Incorrect amount of gaps error

The initial configuration had *user_storage* partition at address 0xfe000, which caused:
- Overlap with existing *image-0-nonsecure* partition (see Table 23)
- Creation of multiple gaps in the flash layout

The *nRF54L15* flash memory has the following fixed constraints :

Address range	Size	Purpose
0x000000 - 0x00ffff	64KB	MCUboot bootloader
0x010000 - 0x060fff	324KB	Application image-0
0x061000 - 0x0b1fff	324KB	Application image-0-nonsecure
0x0b2000 - 0x102fff	324KB	Update image-1
0x103000 - 0x153fff	324KB	Update image-1-nonsecure
0x154000 - 0x15bfff	32KB	TF-M Reserved
0x15c000 - 0x164fff	36KB	Available for user partitions

Table 23: *nRF54L15* flash partition layout

The flash layout has to be reorganized to place all user partitions contiguously after the TF-M reserved region. This was done in the *pm_static.yaml* file (see Listing 13).

```
storage_partition:
  address: 0x15d000
  size: 0x6000
  region: flash_primary

user_partition:
  address: 0x163000
  size: 0x2000
  region: flash_primary
```

Listing 13: pm_static.yaml file

With this modification, the firmware can be built and the partition can be accessible again using the code as in Listing 14.

```
// Define which partition to use for user data
#define STORAGE_PARTITION user_partition

// Resolve memory parameters from the Partition Manager/Device Tree
#define STORAGE_PARTITION_OFFSET    FIXED_PARTITION_OFFSET(STORAGE_PARTITION)
#define STORAGE_PARTITION_DEVICE    FIXED_PARTITION_DEVICE(STORAGE_PARTITION)
```

Listing 14: Partition access from C code

8.5 Main module

The main module handle all the system initialization, firmware reset and low battery events.

8.5.1 main_thread

8.5.1.1 Main initialization sequence

The Figure 60 represents the system initialization sequence in the *main_thread*.

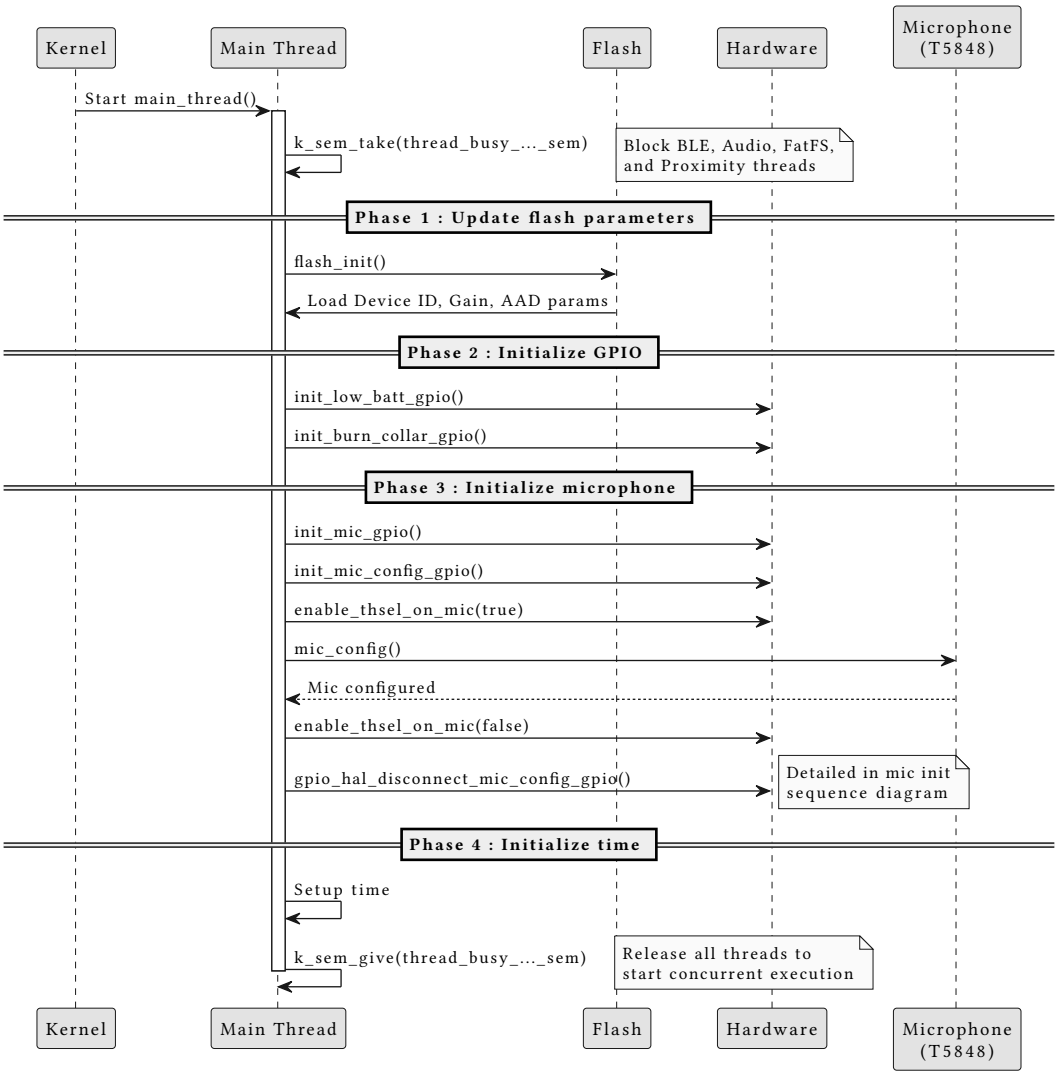


Figure 60: System initialization sequence

The microphone initialization process is detailed in the Section 8.6.2.

8.5.1.2 Operational sequence

The main thread operational sequence is represented in the Figure 61. It a fall in this loop after the initialization.

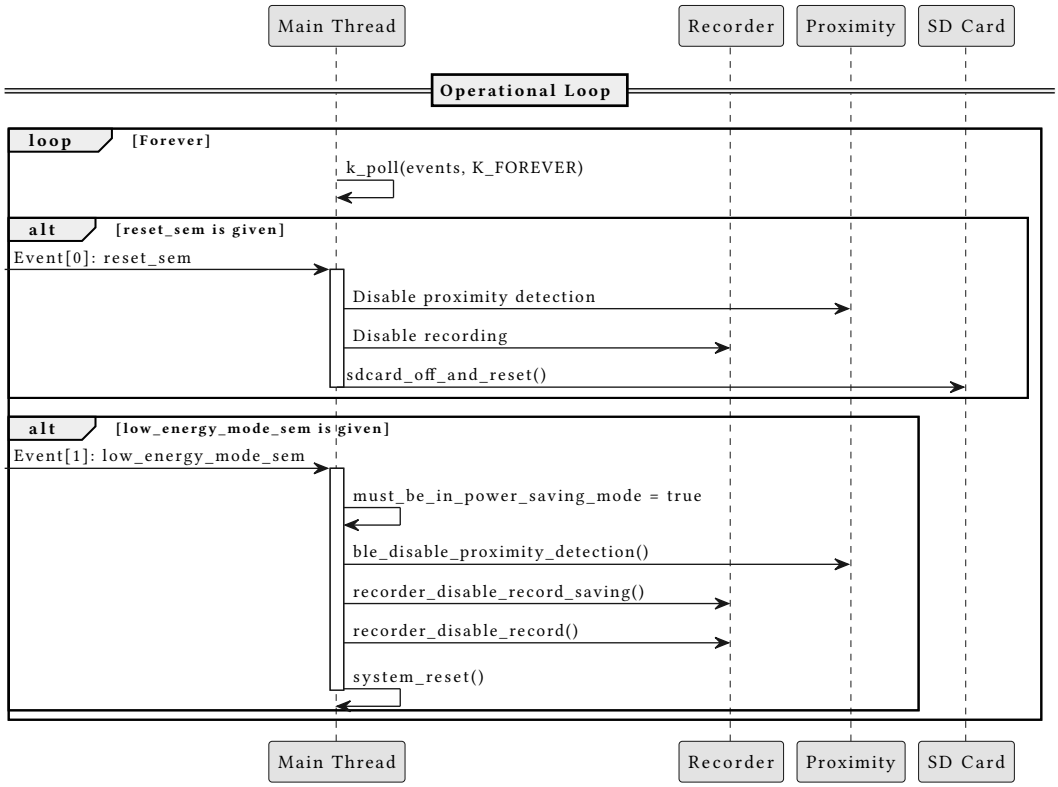


Figure 61: System operational sequence

k_poll

The operational sequence uses *k_poll(events, K_FOREVER)* to wait for both the reset and low-energy mode semaphores. *k_poll* is a Zephyr kernel mechanism that allows a thread to wait for multiple events simultaneously, such as semaphores, FIFOs or signals.

Instead of checking objects sequentially or busy-waiting, *k_poll* puts the thread to sleep until at least one of the monitored events becomes ready (or a timeout expires), ensuring efficient CPU usage.

First, an array of *k_poll_event* structures is created, defining the semaphores to monitor:

```
struct k_poll_event events[] = {
  K_POLL_EVENT_STATIC_INITIALIZER(K_POLL_TYPE_SEM_AVAILABLE,
                                  K_POLL_MODE_NOTIFY_ONLY,
                                  &reset_sem, 0),
  K_POLL_EVENT_STATIC_INITIALIZER(K_POLL_TYPE_SEM_AVAILABLE,
                                  K_POLL_MODE_NOTIFY_ONLY,
                                  &low_energy_mode_sem, 0),
};
```

The thread then blocks on the *k_poll* call, suspending execution until a semaphore is given:

```
k_poll(events, ARRAY_SIZE(events), K_FOREVER);
```

Finally, once a semaphore is available, the specific logic is triggered by checking the event state:

```
if (events[0].state == K_POLL_STATE_SEM_AVAILABLE) { ... }
```

8.6 T5848 module

The initialization of the *TDK T5848* MEMS microphone is handled in the t5848 module. The configuration is written to the microphone by the SoC using a 1-Wire protocol (see Section 5.6.3).

Initially, the idea was to use SPI to emulate the 1-Wire protocol, with MOSI as the THSEL line. But the microphone requires a configuration frequency between 50 and 200 *k*Hz and the SPI is not able to support this frequency at the moment. [33]

When the *spi-max-frequency* is set to a value under 1 MHz, the following error appears.

```
Error : spi_write_dt() failed, err: -5
```

As it is currently impossible to use SPI due to the frequency limitation, bit-banging is used to control the clock and data lines. Once configured, the microphone reacts to the I2S frequency. There is no need for this module to interact with the microphone anymore. This module is also generic. It has no direct link with the hardware. The pins are passed as arguments. It enables reusing the same function across multiple hardware instances without duplicating logic. This module is inspired by a *TDK T5838* driver [34].

8.6.1 Related pin

The following signals are used in this module to send data to the microphone.

Signal	Pin	Type	Function
MIC_CLK	P1.02	GPO	Serial clock (SCK)
MIC_THSEL	P2.00	GPO	Threshold selection (THSEL)
MIC_THSEL_OE	P2.02	GPO	Threshold output enable
nMicON	P1.10	GPO	Power enable

Table 24: T5848 interface signals

The MIC_CLK pin is shared by the 1-Wire clock and the I2S clock signal. Once the microphone is configured, the clock and threshold selection GPOs are disconnected to not interfere with the I2S communication

8.6.2 Microphone initialization sequence

The sequence diagram below represents the initialization process. The bitstream sent is detailed in the next figure.

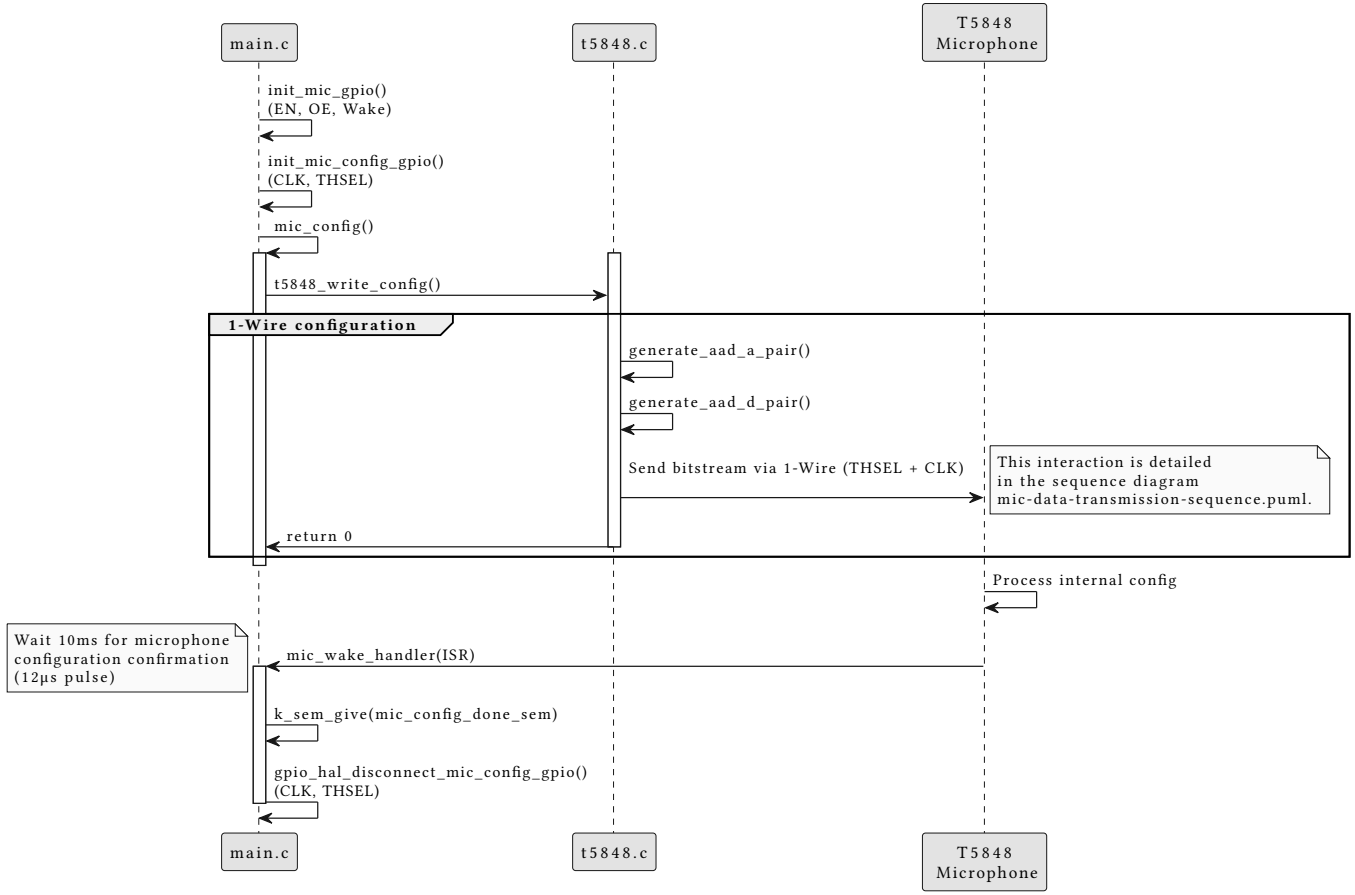


Figure 62: Microphone initialization sequence

The configuration works in pairs: a register address and a register value. The *t5848_write_config* function generates those pairs based on the given configuration. Each of these pairs is then sent to the microphone using the *t5848_reg_write* function. The operation of this function is detailed in the Figure 63. The sending procedure refers to the configuration protocol (see Section 5.6.3.4).

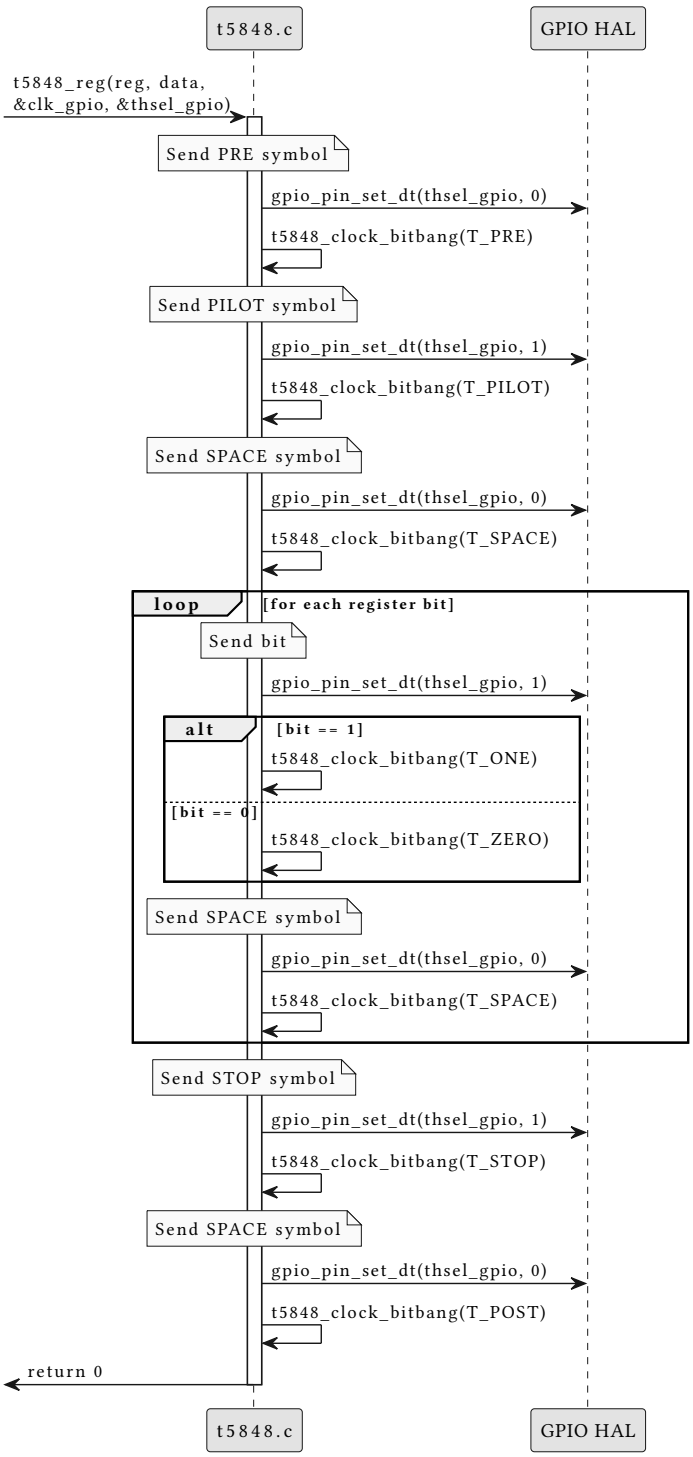


Figure 63: Microphone register write sequence

The configuration is written in fixed registers in the microphone. The register map is represented in the Figure 64

Address	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
29h	Reserved				AAD A_EN	Reserved	AAD D2_EN	AAD D1_EN
2Ah	Reserved				AAD D_FLOOR[12:8]			
2Bh	AAD D_FLOOR[7:0]							
2Ch	Reserved							
2Dh	AAD D_ALGO_SEL[1:0]		Reserved					
2Eh	AAD D_REL_PULSE_MIN[7:0]							
2Fh	AAD D_ABS_PULSE_MIN[11:8]				AAD D_REL_PULSE_MIN[11:8]			
30h	AAD D_ABS_PULSE_MIN[7:0]							
31h	AAD D_ABS_THR[7:0]							
32h	Reserved			AAD D_ABS_THR[12:8]				
33h	AAD D_REL_TH[7:0]							
35h	Reserved					AAD A_LPF[2:0]		
36h	Reserved				AAD A_TH[2:0]			

Reserved bit | AAD A bit | AAD D bit

Figure 64: Microphone register map

Due to the parameter size and the register size, some AAD D settings are written to multiple registers. Furthermore, some registers are shared by multiple values. To ensure that the written data matches the map, the transmitted configurations are manipulated during the generation of address/data pairs (in *t5848_generate_aad_d_pair* or *t5848_generate_aad_a_and_d_pair* function).

The floor setting is split between the register 2A (bits 12 to 8) and 2B (bits 7 to 0).

```
// Floor values
reg_data_pairs[i++] = (struct t5848_address_data_pair)
{T5848_REG_AAD_D_FLOOR_HI, (uint8_t)((config_d->aad_d_floor >> 8) & 0x1F)};
reg_data_pairs[i++] = (struct t5848_address_data_pair)
{T5848_REG_AAD_D_FLOOR_LO, (uint8_t)(config_d->aad_d_floor & 0xFF)};
```

Listing 15: Floor values manipulation

The algorithm selection parameter is shifted and masked to fill the three MSB from the register 2D.

```
// Algo select (0x2D)
reg_data_pairs[i++] = (struct t5848_address_data_pair){T5848_AADD_ALGO_SEL,
(uint8_t)((config_d->aad_d_algo_sel << 6) & 0xC0)}
```

Listing 16: Algorithm selection manipulation

The minimum pulse durations are distributed across registers 2E, 2F and 30.

```
 // Pulse minimums
 reg_data_pairs[i++] = (struct t5848_address_data_pair)
{T5848_REG_AAD_D_REL_PULSE_MIN_LO, (uint8_t)(config_d->aad_d_rel_pulse_min &
0xFF)};
 reg_data_pairs[i++] = (struct t5848_address_data_pair)
{T5848_REG_AAD_D_ABS_REL_PULSE_MIN_SHARED, (uint8_t)((config_d-
>aad_d_abs_pulse_min >> 4) & 0xF0) | ((config_d->aad_d_rel_pulse_min >> 8) &
0x0F)};
 reg_data_pairs[i++] = (struct t5848_address_data_pair)
{T5848_REG_AAD_D_ABS_PULSE_MIN_LO, (uint8_t)(config_d->aad_d_abs_pulse_min &
0xFF)};
```

Listing 17: Pulse minimums manipulation

Finally, the absolute threshold value is split between the registers 32 (bits 12 to 8) and 33 (bits 7 to 0).

```
// Absolute threshold
reg_data_pairs[i++] = (struct t5848_address_data_pair)
{T5848_REG_AAD_D_ABS_THR_LO, (uint8_t)(config_d->aad_d_abs_thr & 0xFF)};
reg_data_pairs[i++] = (struct t5848_address_data_pair)
{T5848_REG_AAD_D_ABS_THR_HI, (uint8_t)(((config_d->aad_d_abs_thr >> 8) & 0x1F)
| 0x40)};
```

Listing 18: Absolute threshold manipulation

8.6.3 Microphone configuration confirmation

The microphone sends a pulse to confirm the configuration writing. It sends a 12 μs pulse on the *WAKE* pin once the last register is written. The Figure 65 shows the data line in blue and the wake signal in yellow.

Figure 65: Microphone configuration confirmation pulse

The confirmation pulse occurs after the last STOP symbol (70.16 μs). It is not sent after the last POST symbol.

To catch this pulse, an interrupt is used on the *WAKE* pin (see Listing 19). The interval between the rising and the falling edge is calculated. If the calculated interval is between 8 and 16 μS, the configuration is confirmed and the main thread is released (*mic_config_done_sem* give). This semaphore wait the confirmation pulse for 10 mS.

```
void mic_wake_handler(const struct device *dev,
                      struct gpio_callback *cb, uint32_t pins) {

  uint32_t now = k_cycle_get_32();
  // Control confirmation pulse after configuration
  if (!mic_init_done) {
    // Rising edge detection
    if (gpio_pin_get_dt(&mic_wake_gpio) == 1) {
      pulse_start_cycle = now;  // Pulse start time
    // Falling edge detection
    } else {
      uint32_t pulse_stop_cycle = now;  // Pulse stop time
      if (pulse_start_cycle == 0) {
        return;
      }
      // Compute pulse duration
      uint32_t duration_cycles = pulse_stop_cycle - pulse_start_cycle;
      uint32_t duration_us = k_cyc_to_us_near32(duration_cycles);
      pulse_start_cycle = 0;
      // Check if valid confirmation pulse
      if (duration_us >= 8 && duration_us <= 16) {
        is_mic_set = true;
        mic_init_done = true;
        k_sem_give(&mic_config_done_sem);
        LOG_INF("Microphone has received config !");
      } else {
        LOG_WRN("Pulse invalid! Expected ~12us, got %d us", duration_us);
     }
    }
  // AAD wake signal
  } else {
    ...
  }
}
```

Listing 19: Microphone wake interrupt handler

The confirmation pulse indicates that the configuration has been successfully written to the registers. It does not confirm that the sent settings are correct. The microphone may not be configured correctly even if the pulse has been received.

8.7 SD card module

The SD card module serves as the system's persistent storage. It abstracts the low-level SPI communication and the FATFS file system logic. A high-level API is then available for other threads to read and write data.

8.8 sdcard_thread_fatfs_mount

The *sdcard_thread_fatfs_mount* mount the file system at boot

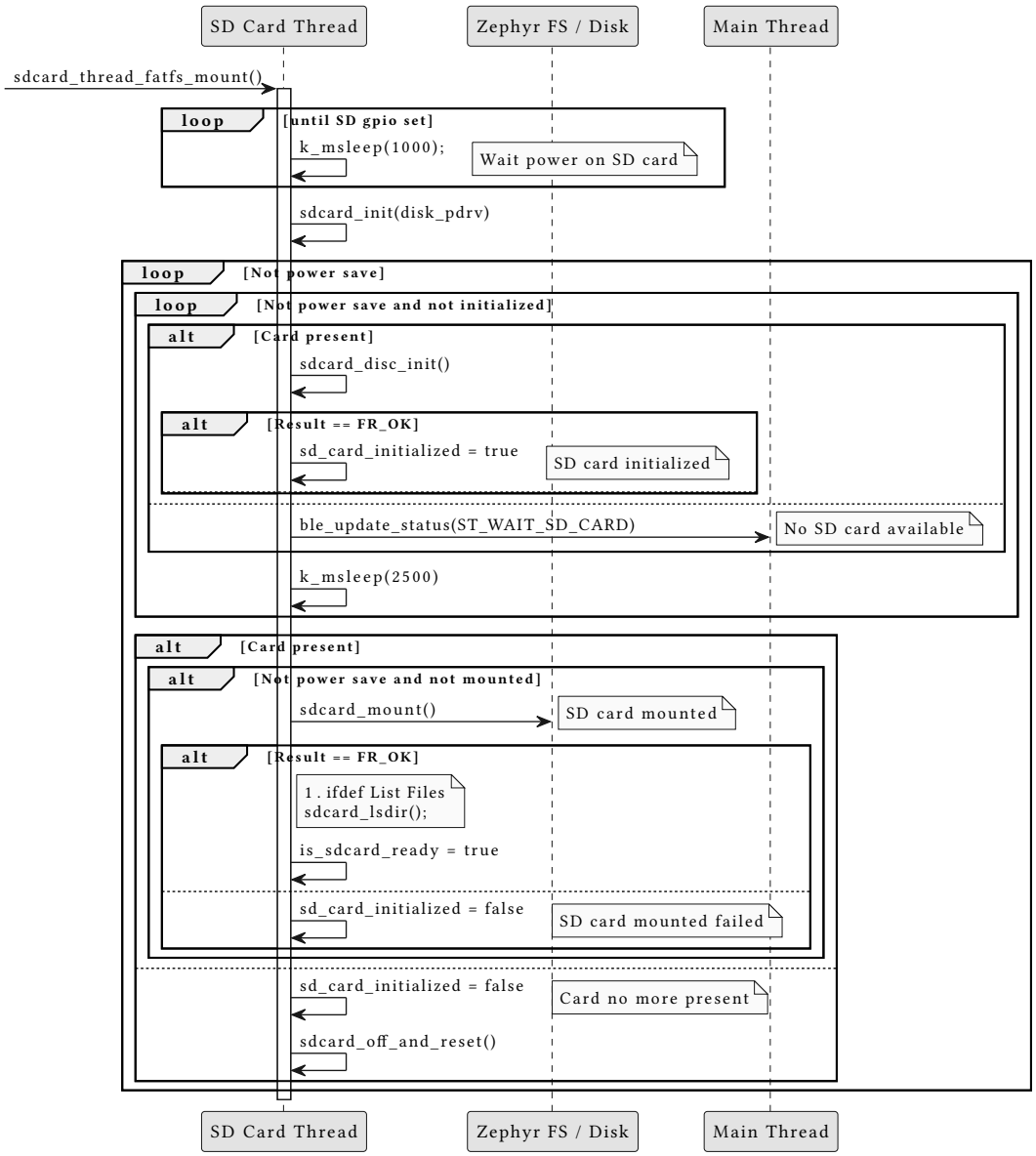


Figure 66: SD mounting sequence

8.9 Recorder module

The recorder module is responsible for recording. Two threads handle this: *recorder_thread_i2s* and *recorder_thread_store_to_file*. This module existed in version 1. It has been updated to match the project specifications.

8.9.1 Related pin

The following signals are used in this module to communicate with the SD card and received data from the microphone.

	Signal	Pin	Type	Function
Microphone pins	*nMicOn*	P1.10	GPO	Microphone power control
	MIC_OE	P1.11	GPO	Audio level shifter enable
	MIC_WAKE	P1.12	GPI	Acoustic wake interrupt
	MIC_CLK	P1.02	Clock	I2S serial clock SCK
	MIC_DATA	P1.13	GPI	I2S serial data input
	MIC_WS	P1.14	GPO	I2S word select
SD pins	*nSDon*	P1.08	GPO	SD card power load switch
	SD_CLK	P1.03	Clock	SD Card SPI Clock
	SD_MISO	P1.04	GPI	SD card SPI MISO
	SD_MOSI	P1.09	GPO	SD card SPI MOSI
	SD_CD	P1.05	Input	SD card card detect
	SD_nCS	P1.06	GPO	SD card SPI chip select

Table 25: Signal-to-Pin mapping for SD card and Microphone interfaces

The MIC_CLK pin is shared by the 1-Wire clock and the I2S clock signal. Once the microphone is configured, the clock and threshold selection GPOs are disconnected to not interfere with the I2S communication

8.9.2 recorder_thread_i2s thread

This thread is responsible for acquiring microphone data. It acts as the producer.

8.9.2.1 Recording sequence

The Figure 67 is a simplified representation of a typical saving session.

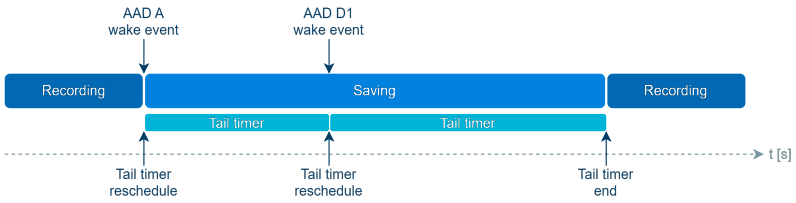


Figure 67: Recording/saving sequence

The sequence diagram below represents the detailed recording process handle in the *recorder_thread_i2s* thread.

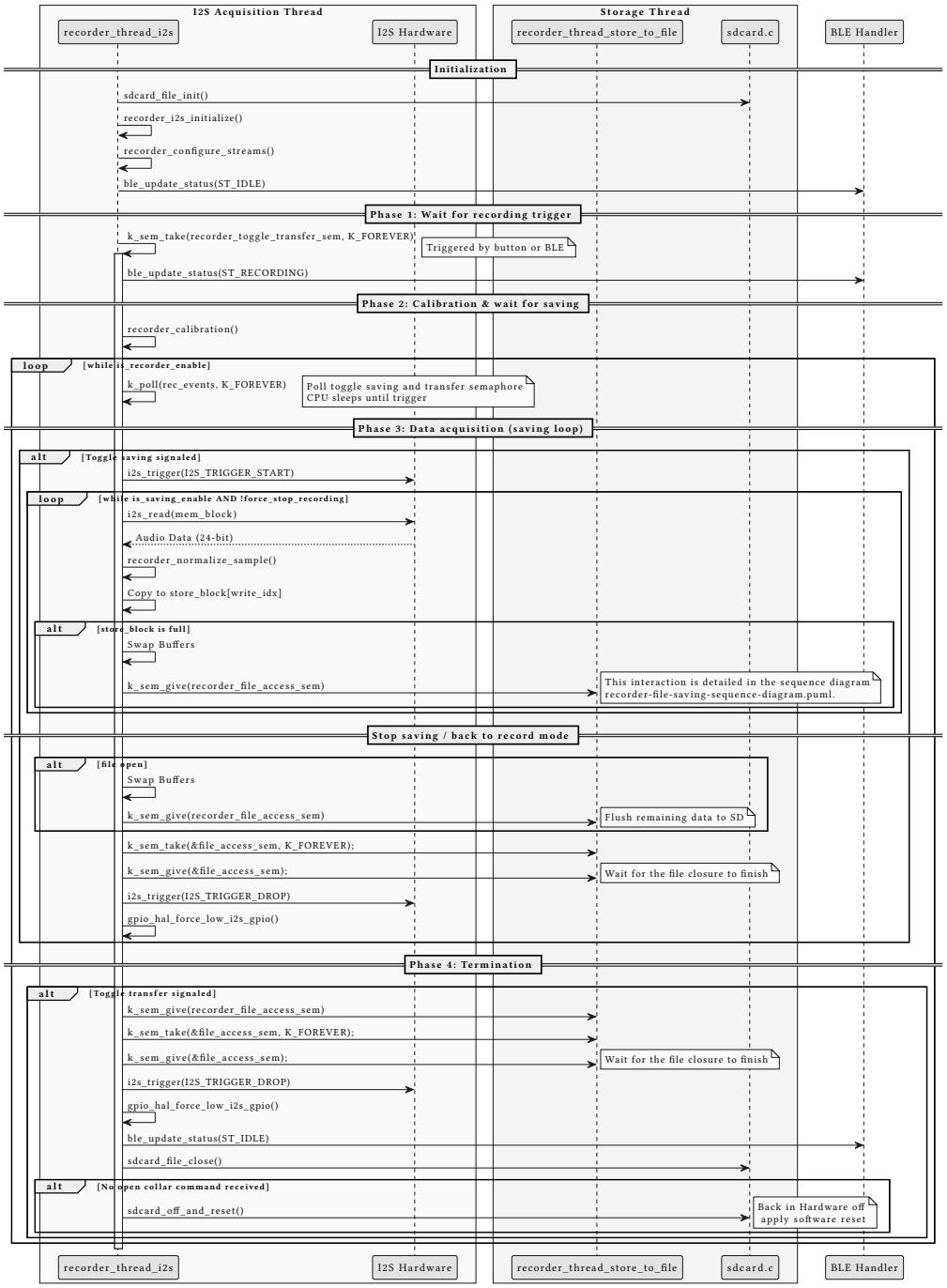


Figure 68: Microphone register write sequence

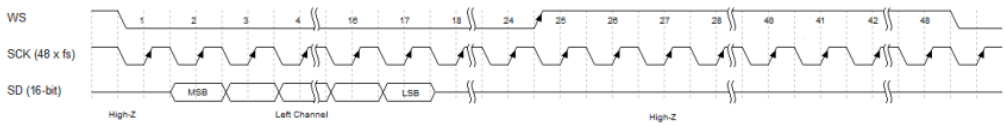
k_poll

Similar to the main thread (see Section 8.5.1.2), the recording sequence utilizes *k_poll* to simultaneously wait for the saving and transfer toggle semaphores. This mechanism enables the thread to efficiently block and react to distinct events: it can either initiate a data saving session or immediately terminate the active recording session.

8.9.2.2 I2S configuration

The I2S interface is configured for unidirectional audio capture from microphone. It as the followings settings :

- **Channel :** 1 channel
  Single microphone, work in mono (left only).
- **Word size :** 24 bits
  In Low Power Mode, it output 24-bit frames with 16 significant bits (lower 8 bits of 24-bit are noise).



Source: https://invensense.tdk.com/download-pdf/t5848-datasheet/

Figure 69: Low power mode output format (stereo left channel)

- **Serial clock :** 768 kHz
  Allows the microphone to operate in low power mode (from 600 to 800 kHz).
- **Word select :** 16 kHz
  The word select frequency is calculated based on the previous configuration :

$$f = f_{\text{ws}} * n_{\text{ch}} * \text{word}_{\text{size}} \rightarrow f_{\text{ws}} = \frac{f}{n_{\text{ch}} * \text{word}_{\text{size}}} = \frac{768000}{2 * 24} = 16k\text{Hz} \quad (9)$$

> As shown in Figure 69, even with 1 channel only, the microphone send the data in stereo mode with the right channel at zero.

- **Format :** Standard I2S data format
  Serial data is transmitted in two's complement with the MSB first. Both word select and serial data signals are sampled on the rising edge of the clock signal. The MSB is always sent one clock period after the word select changes. Left channel data are sent first indicated by word select = 0, followed by right channel data indicated by word select = 1 (see Figure 69).
- **Options :** I2S driver bit clock & frame clock master
  The nRF54 generates both clocks (serial and word select)
- **Memory slab :** *&mem_slab*
  The memory slab to store RX/TX data is *mem_slab*.

```
K_MEM_SLAB_DEFINE_STATIC(mem_slab, I2S_BLOCK_SIZE, I2S_BLOCK_COUNT, 4);
```

It define four memory slab of 6.4 kB

- **Timeout :** $1s$
  Wait 1 second in case TX queue is full or RX queue is empty.

8.9.2.3 Memory architecture

The I2S system uses two memory buffer systems for continuous audio capture. They are optimized for minimal RAM usage. The SoC uses the DMA to save the data received on the I2S in a memory slab. The data is then processed and appended to a store buffer. Once full, the recorder store thread writes it to the SD. The Figure 70 represents this data flow on a timeline.

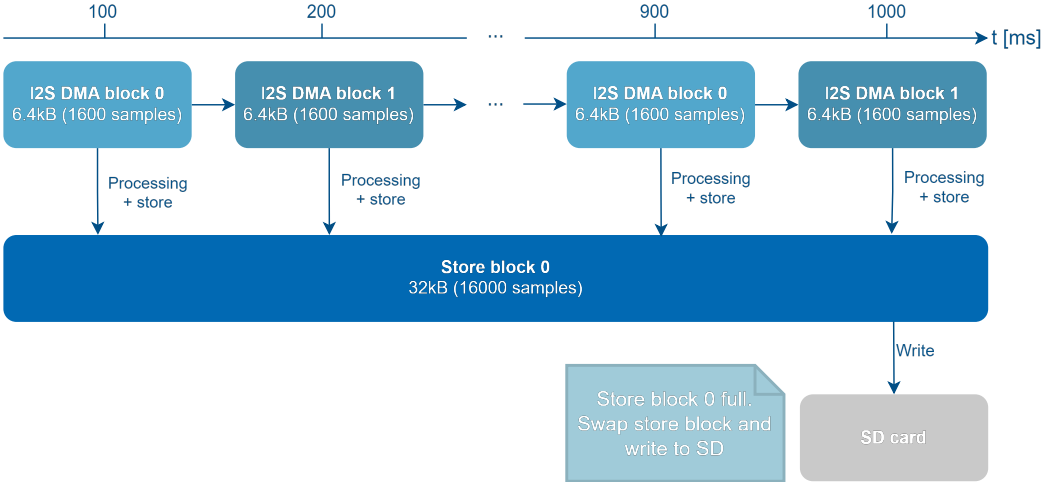


Figure 70: Recorder data flow timeline

I2S DMA Buffers (memory slab)
It is Direct Memory Access buffers for audio capture from *T5848* microphone.

- **Samples per block :** 1600 samples/block
  Each DMA block contains 1,600 samples.

$$\text{samples}_{\text{block}} = \frac{f}{\text{div}_f} * n_{\text{ch}} = \frac{16000}{10} * 1 = 1600 \text{ samples per block} \tag{10}$$

  At 16 kHz sample rate, this represents $100ms$ of : $t = \frac{1600}{16000} = 100ms$ of audio.
- **Bytes per sample :** 4
  Even if the microphone send 24 bits per sample (with only 16 relevant bits), each samples have to be stored in a 32 bits words. The DMA on ARM Cortex requires power of 2 alignment (1, 2, 4, 8 bytes).
- **Block size :** $6.4kB$
  Each block contains $6.4k$ Bytes of data.

$$\text{block}_{\text{size}} = \text{bytes}_{\text{sample}} * \text{samples}_{\text{block}} = 4 * 1600 = 6.4kB \tag{11}$$

- **Block count :** 4
  It has 2 blocks for initial pre-allocation. I2S DMA needs buffers ready before starting. It prevents underrun on first few samples and ensures smooth startup without gaps. The two other blocks are for ping-pong buffering during recording. It use then 25.6 kB of RAM.

- **Sample frequency divider :** 10
  This is not dividing the actual I2S sampling frequency. The I2S peripheral still runs at 16 kHz. Instead, it controls how much audio data is captured per DMA block. It allows to:
  - have a fast response (100 ms)
  - have a low DMA memory usage (25.6 kB)
  - have a manageable processing (1,600 samples every 100 ms)

  The below table compare impact of the divider on the system.

Divider	I2S block	Total I2S mem	Latency	CPU load	SD efficiency
1	64 KB	256 KB	1000 ms	Very low	Excellent
5	12.8 KB	51.2 KB	200 ms	Low	Good
10	**6.4 KB**	**25.6 KB**	**100 ms**	**Medium**	**Good**
20	3.2 KB	12.8 KB	50 ms	High	Poor
50	1.28 KB	5.1 KB	20 ms	Very high	Very poor

Table 26: Impact of I2S block size on system performance and memory

Store block
It is the buffers that accumulate audio data before writing to SD card.

- **Bytes per sample :** 2
  The data stored in the DMA is shifted by 8 and cast into a 16 bits word. Only the significant bits are keep in the store block.
- **Store block size :** 32kB
  A store block contain 32 kB.

$$\begin{aligned} \text{store-block}_{\text{size}} &= \text{div}_f * \text{samples}_{\text{block}} * \text{bytes}_{\text{storage-samples}} * \text{buffer-size\_ratio} \\ &= 10 * 1600 * 2 * 1 * = 32kB \end{aligned} \tag{12}$$

- **Number of store blocks :** 2
  Two store blocks are used to allows producer/consumer pattern (see Section 8.3.8.3>).
- **Buffer size ratio :** 1
  Ratio based on the microcontroller type (RAM capacity not the same).

8.9.2.4 Calibration

The microphone is calibrated when the thread starts. This calibration is performed only once during the thread's lifetime.

Calibration calculates and removes the DC offset from the MEMS microphone signal. To achieve this, the system reads a single I2S block from the microphone. It then calculates the average value of all samples in that block to determine the static bias (DC offset).

```
void recorder_get_dc_offset(void* samples, uint32_t samples_size, uint8_t
rshift)
{
  int64_t dc_sum = 0;
  int32_t *ptr = (int32_t *)samples;
  for (int i = 0; i < samples_size; i++) {
    int16_t sample_16bit = (int16_t)(ptr[i] >> rshift);
    dc_sum += sample_16bit;
  }
  recorder_sample_offset = (int32_t)(dc_sum / samples_size);
  LOG_DBG("DC Offset: %d\n", recorder_sample_offset);
}
```

Listing 20: DC offset calculation function

This calculated offset (*recorder_sample_offset*) is then subtracted from every audio sample (see Section 8.9.2.5). It ensure that the waveform is centered around zero.

If calibration occurs in a noisy environment, the accuracy of the DC offset calculation may be compromised.

8.9.2.5 Data processing

The data is process before append it in the record store using Listing 21.

```
int32_t recorder_normalize_sample(int32_t sample, int gain, int divider,
uint8_t rshift, uint32_t mask)
{
    // Shift right to have the sample in the right bit width
    int64_t s = (int64_t) (sample >> rshift);

    // Apply calibration correction and gain
    if (divider > 1) {
        s = ((s - recorder_sample_offset) * gain) / divider;
    } else if (gain > 1) {
        s = (s - recorder_sample_offset) * gain;
    }

    // Mask return value to keep only significant bit
    return (int32_t) (s & mask) ;
}
```

Listing 21: Sample normalization function

This generic normalization function processes audio samples from any bit-width microphone. It this project, the following settings are used.

- ***rshift* parameter :** 8
  Aligns audio data by removing padding bits, see Figure 69 for received data format.
- ***gain* parameter :** 3
  Virtual gain of the system, combined with the divider.

- ***divider* parameter :** 2
  Divide the gain to have finer choice (overall gain = 1.5 = 3.5dB)
- ***mask* parameter :** 0xFFFF
  Extracts desired output bits. Acts as a cast.

8.9.3 recorder_thread_store_to_file thread

This thread writes the data produced by the microphone to the SD card. It is linked to the *recorder_thread_i2s* and acts as the consumer.

8.9.3.1 Writing sequence

The sequence diagram below represents the recording process handle in the *recorder_thread_store_to_file* thread.

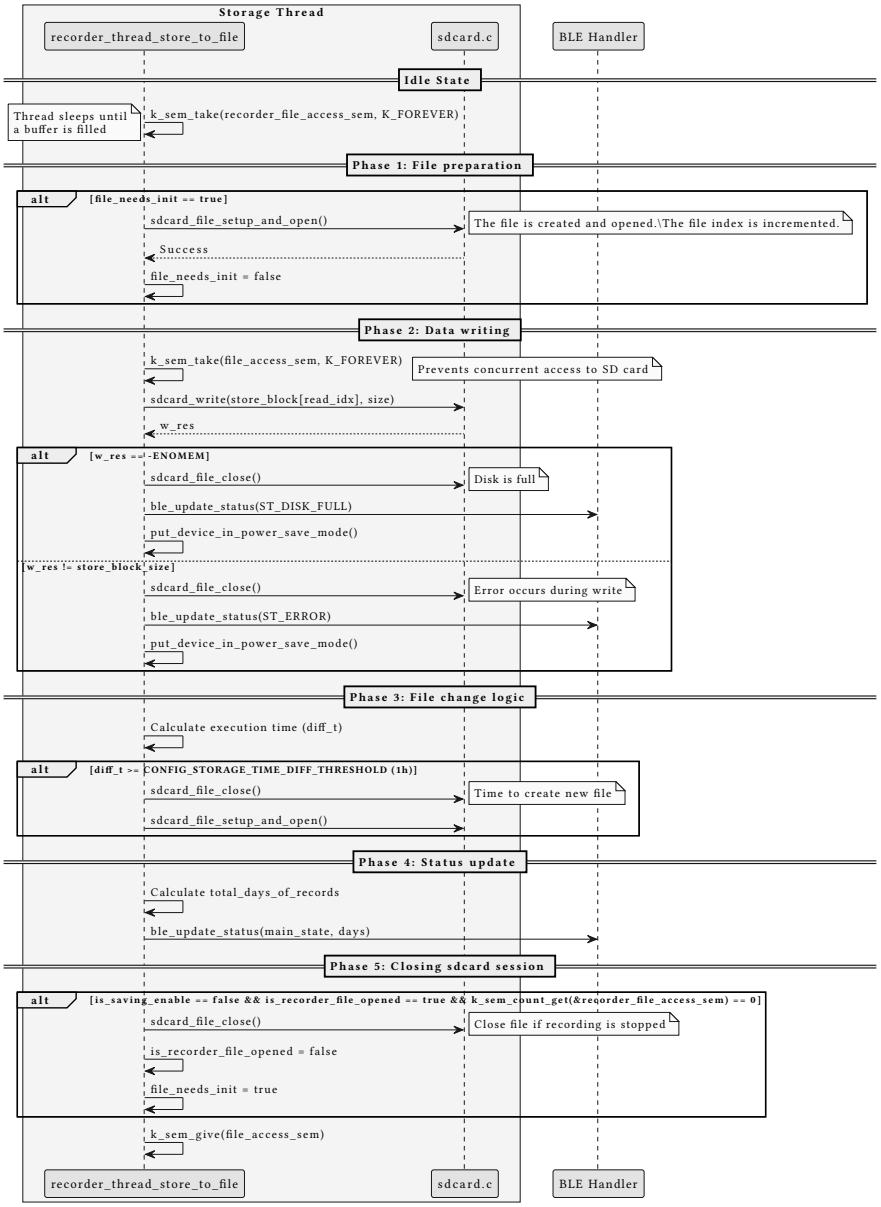


Figure 71: Audio data write to SD sequence

8.9.3.2 Audio file strategy

The audio file strategy for the nRF Monkey project is designed to ensure data integrity, easy retrieval by researchers and long-term reliability in the field.

Audio file format

The audio files are saved in *.DAT*. The *.DAT* format acts as a raw container. This choice reduces overhead and allows the system to write data blocks directly from the RAM buffers to the SD card without formatting during the active session.

It contains raw 16-bit PCM samples. The data is stored linearly, allowing it to be imported into audio software as raw data. The Figure 72 represents the settings used to import the files in *Audacity*

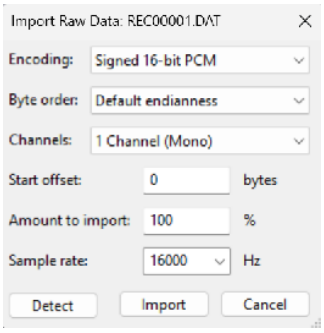


Figure 72: Audacity *raw data import* settings

The files are timestamped with the current time. The current time is updated in the mobile application once it is connected.

Audio file naming convention

Files are named using a prefix and an incrementing index to distinguish between different recording sessions and log entries. The file name can be set using the *CONFIG_RECORDER_STORAGE_FILE_NAME* setting in the prj.conf. The default name is *REC*. The format is :

```
RECxxxxx.DAT
```

It allows the system to record 99'999 audio files. This number is limited by the FATFs file name format : 8.3, 8 characters for the file name and 3 for the extension. The following equation estimates the minimum recording time of the system.

$$t_{\text{rec}} = t_{\text{rec}_{\text{min}}} * n_{\text{file}} = 10s * 99999 = 999990s = 277.78h = 11.57 \text{ days} \quad (13)$$

It gives a maximum recording time of 11.57 days with the file-naming convention used, by saving the minimum number of audio samples per file. This limitation will not cause a problem for this project.

Audio file rotation
A file is created for each new saving session. This allows to have the precise timestamp for each audio event.

To prevent file corruption, the system implements time-based rotation security. A new file is created whenever the current saving duration exceeds *CONFIG_STORAGE_TIME_DIFF_THRESHOLD* (default: 3600s).

Audio file size
With a minimal samples time of 10 seconds, the minimal file size is 0.32 MB as calculated in the equation below.

$$\text{file}_{\text{size}_{\text{min}}} = \text{data}_{\text{rate}} * t_{\text{rec}_{\text{min}}} = 32000\frac{\text{bytes}}{s} * 10s = 0.32M\text{B} \tag{14}$$

Since the maximum recording time for a single file is 3600 seconds, the maximum file size is 115.2 MB as calculated in the following equation.

$$\text{file}_{\text{size}_{\text{max}}} = \text{data}_{\text{rate}} * t_{\text{rec}_{\text{max}}} = 32000\frac{\text{bytes}}{s} * 3600s = 115.2M\text{B} \tag{15}$$

With a standard 32 GB SD card, the maximum recording time is 11.59 days as demonstrated in the following equation.

$$t_{\text{rec}} = \frac{\text{SD}_{\text{size}}}{\text{file}_{\text{size}_{\text{max}}}} = \frac{32000M\text{B}}{115.2\frac{M\text{B}}{h}} = 278.26h = 11.59 \text{ days} \tag{16}$$

It should be noted that proximity detection log files will also be saved on the same SD card.

8.10 BLE handler module

The Bluetooth Low Energy is manage by the Bluetooth Low Energy handler module. It handles all the Bluetooth initialization.

8.10.1 ble_thread_init thread

The *ble_thread_init* thread initialize the stack and manage the advertising/scanning logic for the proximity detection.

8.10.1.1 Initialization sequence

The sequence diagram below represents the initializing process handle in the *ble_thread_init* thread.

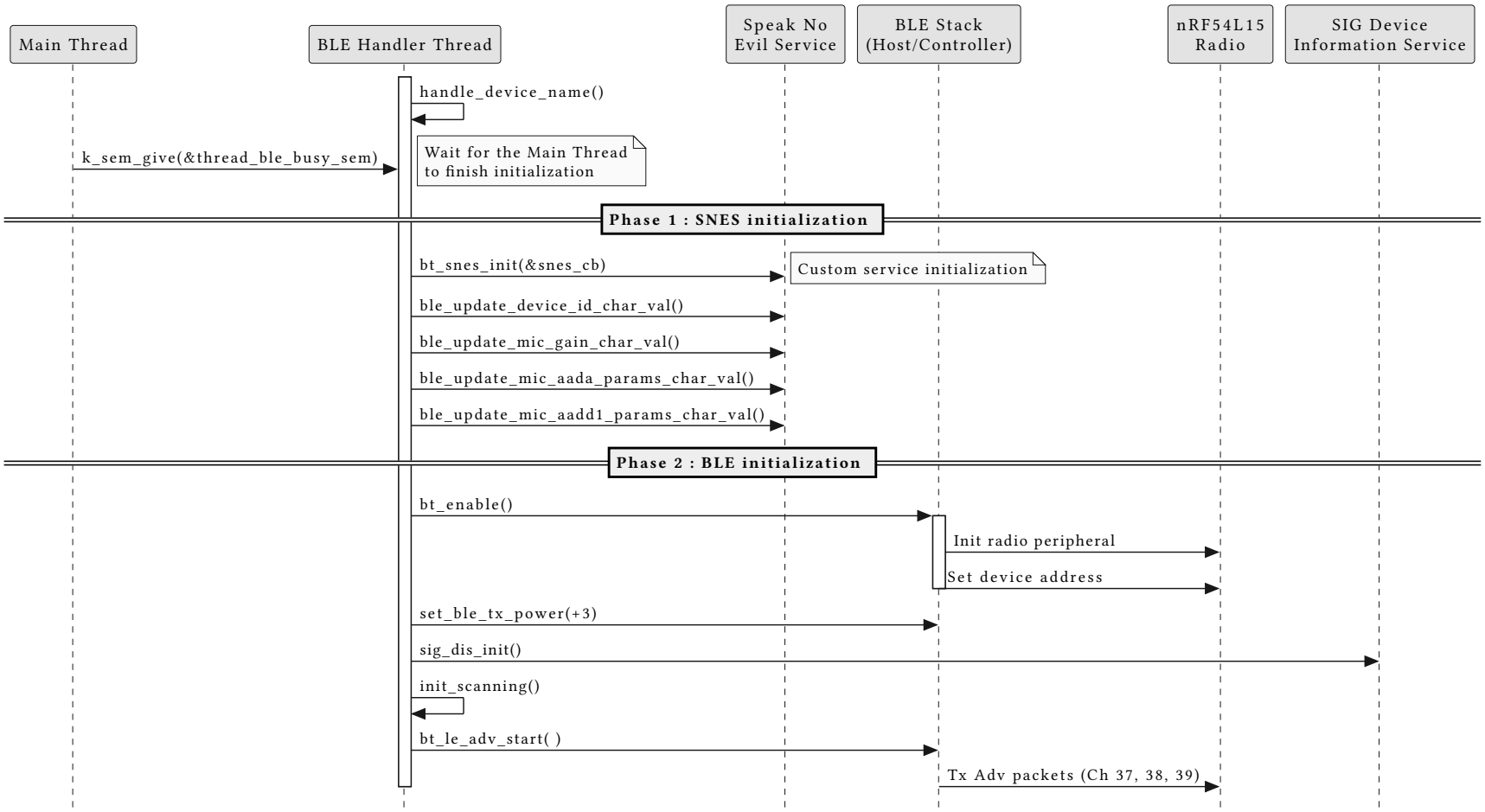


Figure 73: BLE initialization sequence

8.10.1.2 Operational sequence

The Figure 74 represent the operational sequence. When proximity detection is enabled, the system alternates between scanning and advertising phases. These phases must be managed by the program because the maximum scanning window allowed by the BLE specification is 10.24 seconds. Therefore, the driver cannot handle longer intervals in the background.

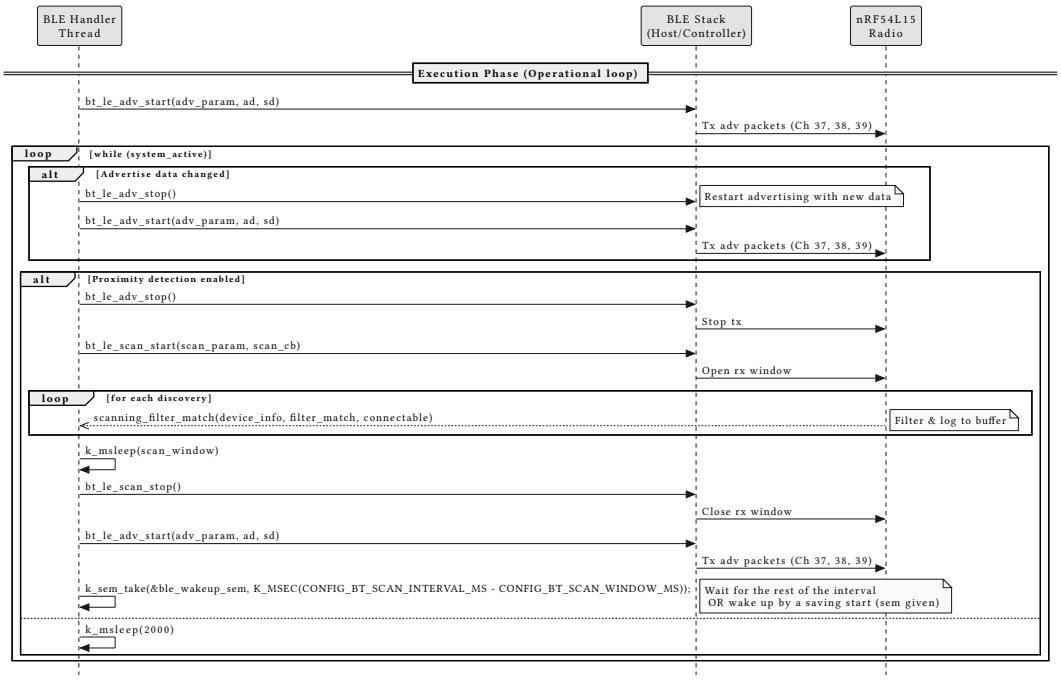


Figure 74: BLE operational sequence

A proximity detection is run :

1. Periodically every 60 seconds
2. When a wake event is received

8.10.2 BLE configuration

Advertising configuration

- **Advertising interval :** 300 ms
  Delay between advertising to remain discoverable while saving power.
- **TX power :** +3 dBm
  Radio transmission strength configured via HCI commands in the *set_ble_tx_power*. The function utilizes vendor-specific HCI commands to manually adjust the radio's transmission signal strength on the *nRF54L15*.
- **Device name :** Speak No Evil xxx
  The name of the collar is composed of *Speak No Evil* follow by the device identifier (1 to 255). The name is handle in the *handle_device_name* function. It dynamically generates

the device name by concatenated the generic device name with a unique identifier retrieved from flash memory.

- **Manufacturer data :** 4 bytes
  Payload containing HEI ID, days of recording and system status.

```
static uint8_t manufacturer_data[] = {
  0x5A, 0x02, // HEI Company Id
  0x00, // Number of days of recording
  0x00  // Status : 0x001 -> waiting for SD Card, 0x02 -> IDLE, 0x03 ->
Recording, 0x04 -> Saving, 0x05 -> Disk full,  0x06 -> Low batt, 0x06 ->
Power saving, 0xff -> Error
};
```

Listing 22: Manufacturer data

Connection configuration

- **Options :** connectable
  Accept connection from a central device.
- **Min connection interval :** 618.75 ms
  Minimum time between connection events ($495 * 1.25ms$).
- **Max connection interval :** 937.5 ms
  Maximum time between connection events ($750 * 1.25ms$).
- **Supervision Timeout :** 400 ms
  Time before the link is considered lost ($4 * 100ms$).
- **Max connection :** 1
  Maximum number of connected central at the same time.
- **Max paired :** 1
  Maximum number of paired central at the same time.

Scanning configuration

- **Options :** passive
  Scan without requesting additional information from advertisers to avoid power loss.
- **Scan Window :** 1.4 s
  Active duration for listening to other collars.
- **Scan Interval :** 60 s
  The firmware manually coordinates scanning periods to bypass the 10.24-second limitation of the BLE controller. The driver is initialized with the maximum theoretical window, but the thread logic actively terminates the scan immediately after a single scanning event. It then enters a timed sleep for the remainder of the interval before triggering the next cycle.
- **Scanning callback** *scanning_filter_match()*
  The callback call by the BLE stack when a device is scanned. The *scanning_filter_match* function is detailed in the Section 8.11.3.

8.11 BLE proximity module

Proximity detection is handled in this module according to the principle explained in Figure 55.

A proximity detection can be started by two events :
1. Standard scanning windows
2. Microphone wake event

It periodically scans each 60 seconds for nearby device and also performed a detection when the audio saving is started. This is done with the aim of getting to know the monkeys in the vicinity during a vocalization.

8.11.1 Proximity detection sequence

The Figure 75 represents the proximity detection sequence. This sequence occurs when the system is currently scanning, as represented in the Figure 74.

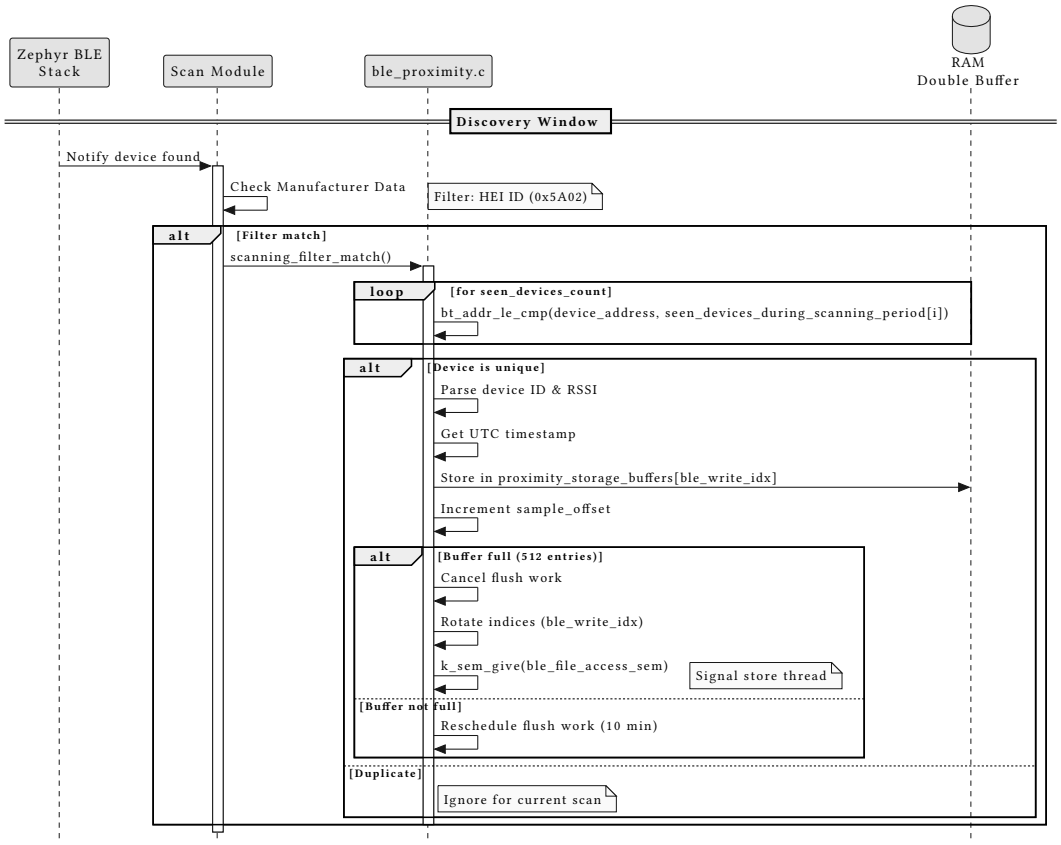


Figure 75: BLE proximity detection sequence

8.11.2 Scanning filter

A callback is invoked whenever a device is scanned. To reduce processing workload and thus save energy, a scanning filter has been added. This filter ensures that the callback (*scanning_filter_match*) is only called when the conditions are met.

Different types of filters exist: name, address, UUID, appearance or manufacturer data. The manufacturer data filter will be used in this project. Manufacturer data is included directly in the advertising data. It doesn't request sending a scan request or establishing a GATT connection. Furthermore, the HEI of Sion has a BLE unique company identifier: *0x025A, University of Applied Sciences Valais/Haute École Valaisanne.* [35]

The Listing 23 represents the helper structure to set the manufacturer data filter. The data length is set to two bytes. With this, the filter will only compare the company ID and not the following bytes, which are subject to change.

```
const struct bt_scan_manufacturer_data scan_manufacturer_data = {
  .data = manufacturer_data,
  .data_len = 2
};
```

Listing 23: BLE manufacturer data

The filter is applied the first time to the scan module. The Listing 24 is a filer related extract of the *start_scanning()* function.

```
// Initialize filter only the first time
if (!filters_initialized) {
    uint8_t filter_mode = 0;

    // Add manufacturer data filter. Filter with HEI BLE identifier (0x025A)
    err = bt_scan_filter_add(BT_SCAN_FILTER_TYPE_MANUFACTURER_DATA,
                             (const void *) &scan_manufacturer_data);
    if (err) {
        LOG_ERR("Manufacturer data filter cannot be added (err %d)\n", err);
        return err;
    }
    filter_mode |= BT_SCAN_MANUFACTURER_DATA_FILTER;  // Select filter mode

    err = bt_scan_filter_enable(filter_mode, false);
    if (err) {
        LOG_ERR("Filters cannot be turned on (err %d)\n", err);
        return err;
    }

    filters_initialized = true;
}
```

Listing 24: BLE manufacturer data

8.11.3 scanning_filter_match

This *scanning_filter_match* is the scan callback function triggered only if the filter is matched.

8.11.3.1 Multiple detection

Because the advertising interval is shorter than the scanning interval, the same device may be detected multiple times within a single scanning period. To avoid logging the same information multiple times, a single-scan deduplication strategy is implemented.

A temporary seen device list is used to track already seen devices as represented in the Listing 25.

```
// --- Single-scan de-duplication ---
#define MAX_DEVICES_PER_SCAN 64
static bt_addr_le_t seen_devices_during_scanning_period[MAX_DEVICES_PER_SCAN];
static uint8_t seen_devices_count = 0;
```

Listing 25: BLE scan de-duplication list

The Listing 26 shows the logic implementation in the *scanning_filter_match* callback. Each device is compared with the already seen device. If a device is already in this list, the scan device is discarded. The list is reset at the start of every scan.

```
// Check if we have already seen this device during this scanning period
for (int i = 0; i < seen_devices_count; i++) {
    if (bt_addr_le_cmp(device_info->recv_info->addr,
&seen_devices_during_scanning_period[i]) == 0) {
        return;
    }
}

if(seen_devices_count < MAX_DEVICES_PER_SCAN) {
  bt_addr_le_copy(&seen_devices_during_scanning_period[seen_devices_count++],
device_info->recv_info->addr);
}
```

Listing 26: BLE scan de-duplication logic

8.11.3.2 Data processing

The detected device informations are extract, formated and save in the proximity storage buffer. The scanned device information is encapsulated in a packed structure to ensure memory efficiency and a 15-byte record size when stored on the SD card.

```
struct proximity_device_info {
    uint32_t timestamp;         // Time of detection (Unix Epoch)
    uint8_t  addr[6];           // Unique 48-bit Bluetooth MAC address
    int16_t  device_number;     // Numeric ID parsed from peer name
    int8_t   rssi;              // Received Signal Strength Indicator in dBm
    uint8_t  days_of_recording; // Field duration logged by the peer
    uint8_t  system_status;     // Peer state (Idle, Recording, Error, ...)
} __packed;
```

Listing 27: Proximity detection data structure

The estimated distance based on the RSSI value is not computed on edge. It can be done during data processing, after the data collection campaign, using the Equation 4.

8.11.3.3 Audio saving enable by proximity detection

By enabling *CONFIG_BT_PROXIMITY_ENABLE_SOUND_SAVING*, it is possible to start audio saving when a device is detected if the device RSSI is greater than *CONFIG_BT_PROXIMITY_ENABLE_AUDIO_RSSI_MIN*.

```
// Enable sound saving when a device is detected near by with a RSSI superior
to CONFIG_BT_PROXIMITY_START_SOUND_RSSI_MIN
// And only once per scanning session
#ifdef CONFIG_BT_PROXIMITY_ENABLE_AUDIO_SAVING
  if(device->rssi >= CONFIG_BT_PROXIMITY_ENABLE_AUDIO_RSSI_MIN
      && seen_devices_count == 1) {
    recorder_enable_record_saving();
  }
#endif //#ifdef CONFIG_BT_PROXIMITY_ENABLE_AUDIO_SAVING
```

Listing 28: Start audio saving when proximity detection

This feature should be used with caution, as it may lead to a large number of recording and therefore increased consumption.

8.11.3.4 Store thread notification

The *scanning_filter_match* also handles the store thread notification. The proximity buffer contains a fixed number of devices (*CONFIG_BT_PROXIMITY_STORAGE_BUFFER_SIZE*, set to 512). Once the buffer is full, it is swapped and the write-to-file thread is notified.

8.11.3.5 Flush logic

The flush logic acts as a safeguard to prevent data loss when sparse social interactions fail to fill the RAM buffer for extended periods. To ensure persistence against system resets, the system utilizes a delayable work item (*proximity_flush_work*) that periodically forces the writing of pending data to the SD card. The logic operates as in the Figure 76.

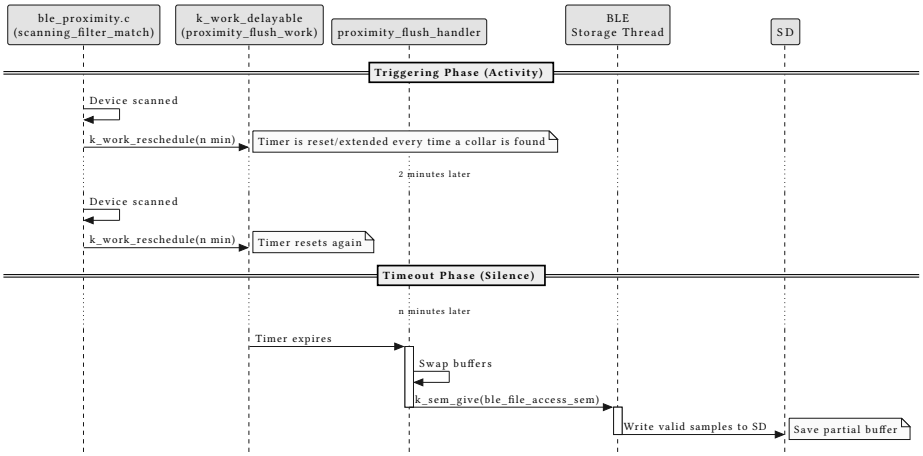


Figure 76: BLE flush procedure

8.11.4 ble_proximity_thread_store_to_file thread

This thread writes the proximity detection data. It is linked to the *ble_thread_init* and acts as the consumer.

8.11.4.1 Writing sequence

The sequence diagram below represents the recording process handled in the *ble_proximity_thread_store_to_file* thread.

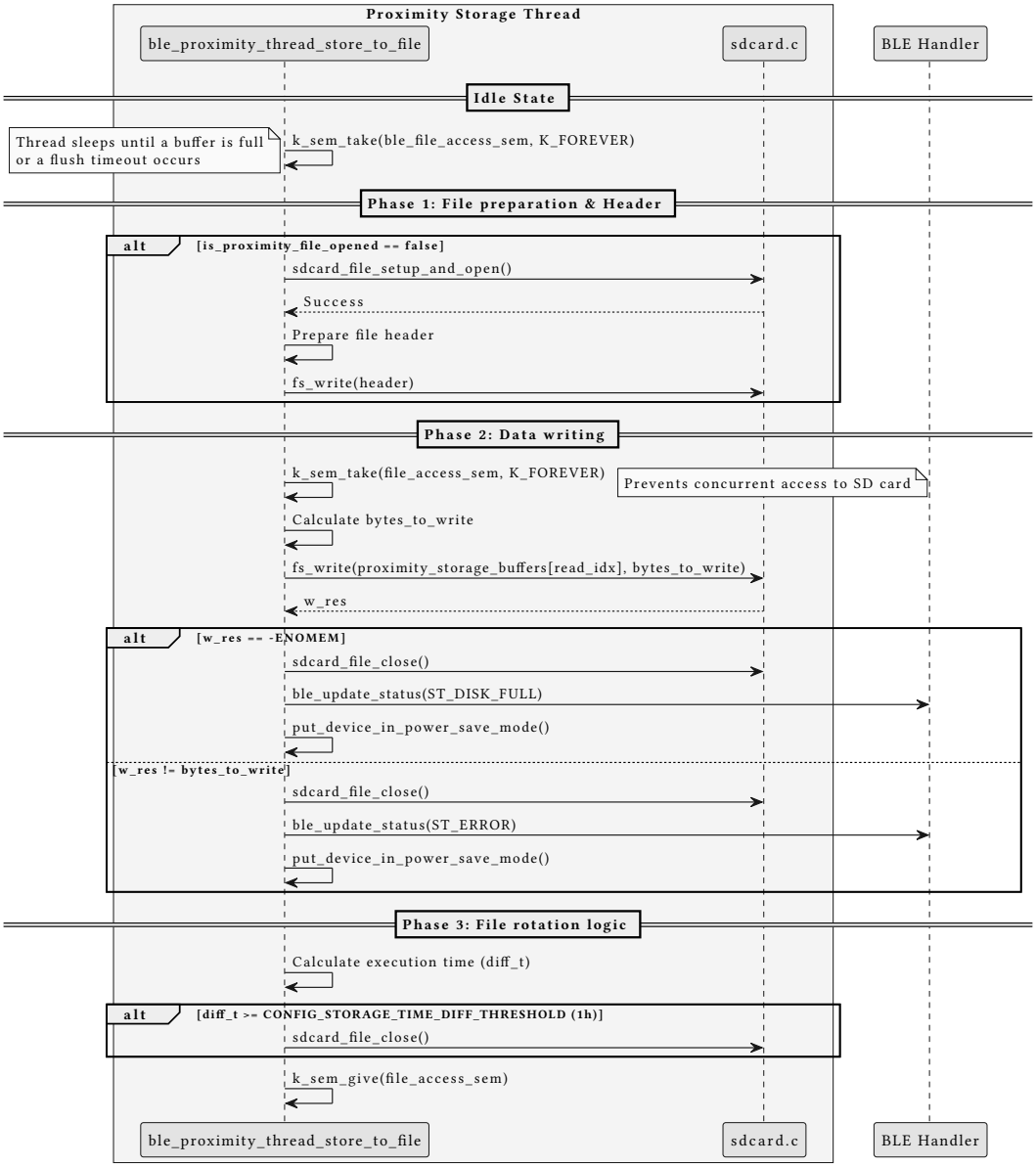


Figure 77: BLE proximity write sequence

8.11.4.2 BLE proximity file strategy

The BLE proximity file strategy for the nRF Monkey is inspired by the audio file strategy.

BLE proximity file format

The proximity files are saved in *.DAT*. The *.DAT* format acts as a raw container. This choice reduces overhead.

The files are timestamped with the current time. The current time is updated in the mobile application once it is connected.

File layout

A BLE proximity file is composed of two main elements : a header (written at the opening) and device payloads. The Figure 78 represents the file layout.

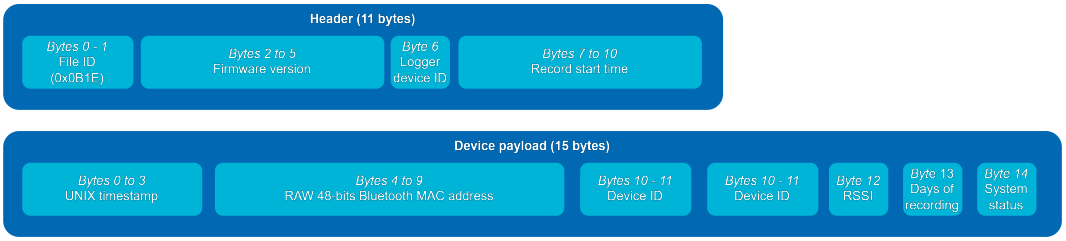


Figure 78: BLE proximity file layout

BLE proximity file naming convention

Files are named using a prefix and an incrementing index. The file name can be set using the *CONFIG_PROXIMITY_STORAGE_FILE_NAME* setting in the prj.conf. The default name is PROX. The format is :

```
PROXxxxx.DAT
```

It allows the system to record 9'999 audio files. This number is limited by the FATFs file name format : 8.3, 8 characters for the file name and 3 for the extension. It is impossible to calculate the number of possible detections with this number of files because the detection rate is not regular and known.

BLE proximity file rotation

Since each detection is timestamped, they can be appended to the same file. However, to prevent file corruption, the system implements time-based rotation security. A new file is created whenever the current recording duration exceeds *CONFIG_STORAGE_TIME_DIFF_THRESHOLD* (default: 3600s).

BLE proximity file size

It is difficult to calculate the exact size of a file because the number of detections is not known in advance. However, it is possible to estimate its size.

With a *MAX_DEVICES_PER_SCAN* of 64 and a scanning interval of 60 secondes, it give the following calculations:

$$\text{scan}_{\text{rate}} = \frac{\text{file}_{\text{time-diff}}}{\text{scan}_{\text{inter}}} = \frac{3600s}{60s} = 60\frac{\text{scan}}{s} \tag{17}$$

$$\begin{aligned} \text{file}_{\text{size}_{\text{max}}} &= \text{header} + \text{scan}_{\text{rate}} * \text{device}_{\text{scan}_{\text{max}}} * \text{payload} \\ 11\ \text{bytes} + 60\frac{\text{scan}}{s} &* 64 * 15\ \text{bytes} = 57.6k\text{B} \end{aligned} \tag{18}$$

The files are very small and negligible compared to the audio files ($\text{prox\_file}_{\text{size}_{\text{max}}} < \text{audio\_file}_{\text{size}_{\text{min}}}$).

A python script is available in the project repository to convert the *.DAT* into *.CSV* file.

8.12 Speak No Evil Service module

The production collar version will not have a physical user interface. To monitor and control the system, a wireless user interface via a custom Bluetooth Low Energy service was already implemented in the V1: Speak No Evil Service (SNES). This allows a mobile application or another device to act as a remote console for monitoring system status and triggering the collar release. It has been updated to allow adjustment of recording parameters.

The communication is based on the Generic Attribute Profile. In this architecture, the collar acts as a GATT server, hosting a database, while the mobile app acts as the GATT client. GATT service is explained deeply in the Section 7.1.

The Figure 79 represents the service hierarchy.

Figure 79: SNES hierarchy

Command

The system implements a control point characteristic. These controls allow monitoring of the collar's condition. While a Bluetooth Low Energy radio packet contains preamble, addresses and CRCs, the application firmware only interacts with the ATT payload. A valid ATT payload is built as represented in the Figure 80.

Figure 80: BLE payload

The header is a fixed value set to 0xA5. The system accepts the commands listed in the Table 27. Finally, the data completes the payload. It can range from 1 to 10 bytes (see Figure 79).

Command	Value	Action / Description
OPEN_COLLAR	0x01	Triggers the release mechanism (BURN).
RESET_START	0x02	Syncs time and starts hardware drivers.
TOGGLE_REC	0x03	Disables recording and proximity monitoring.
SHUTDOWN_HW	0x04	Powers down hardware to low-power state.
STARTUP_HW	0x05	Reactivates hardware drivers.
SET_TIME	0x06	Sets system RTC with 10-byte timestamp.
SET_DEVICE_ID	0x07	Updates device identifier in flash.
SET_MIC_GAIN	0x08	Updates the microphone input gain.
SET_AAD_A	0x09	Sets LPF and Threshold for analog wake.
SET_AAD_D	0x0F	Sets digital algorithm and pulse limits.
UNPAIR	0xFF	Unpairs devices and disconnects peer.

Table 27: SNES command protocol definitions with opcode mapping

The service allows different actions :

1. **System control :** The system can be controlled using the commands. The recording can be started or toggled. The hardware switch is ON or OFF. Finally, the time can be set and the collar opened.
2. **System monitoring :** The system updates the battery state, recording status and recording time via notifications.
3. **Microphone configuration :** It allows adjusting microphone sensitivity and both analog and digital wake-up thresholds without needing a physical debug connection. Those settings are stored in flash memory and trigger a software reset to apply the modifications.
4. **Identity Management :** Allows dynamic device naming by modifying the unique identifier stored in flash memory.

8.13 Memory footprint

The footprints of both RAM and ROM flash are presented in this chapter. The complete memory reports are available in the project repository (see Section AA)..

8.13.1 RAM

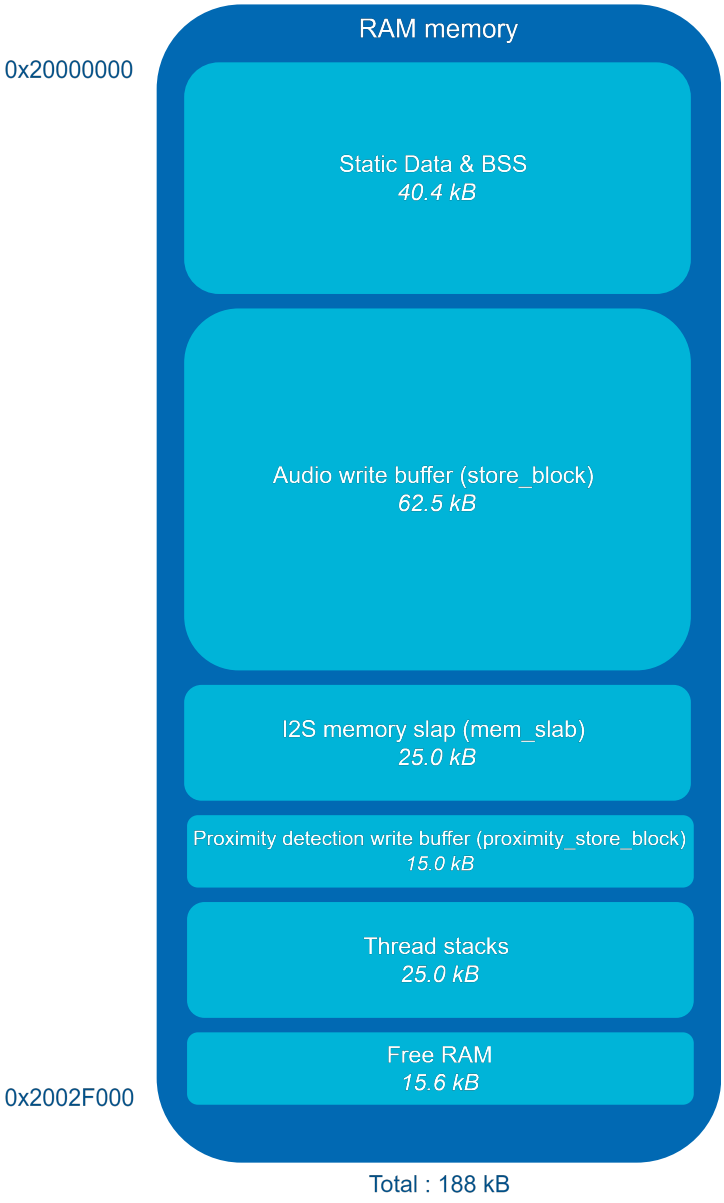


Figure 81: RAM memory footprint

The visible memory footprint represents only the partition assigned to the application core. The remaining memory is reserved for the network core, security and inter-processor communication.
Total : 256 kB

8.13.2 Flash

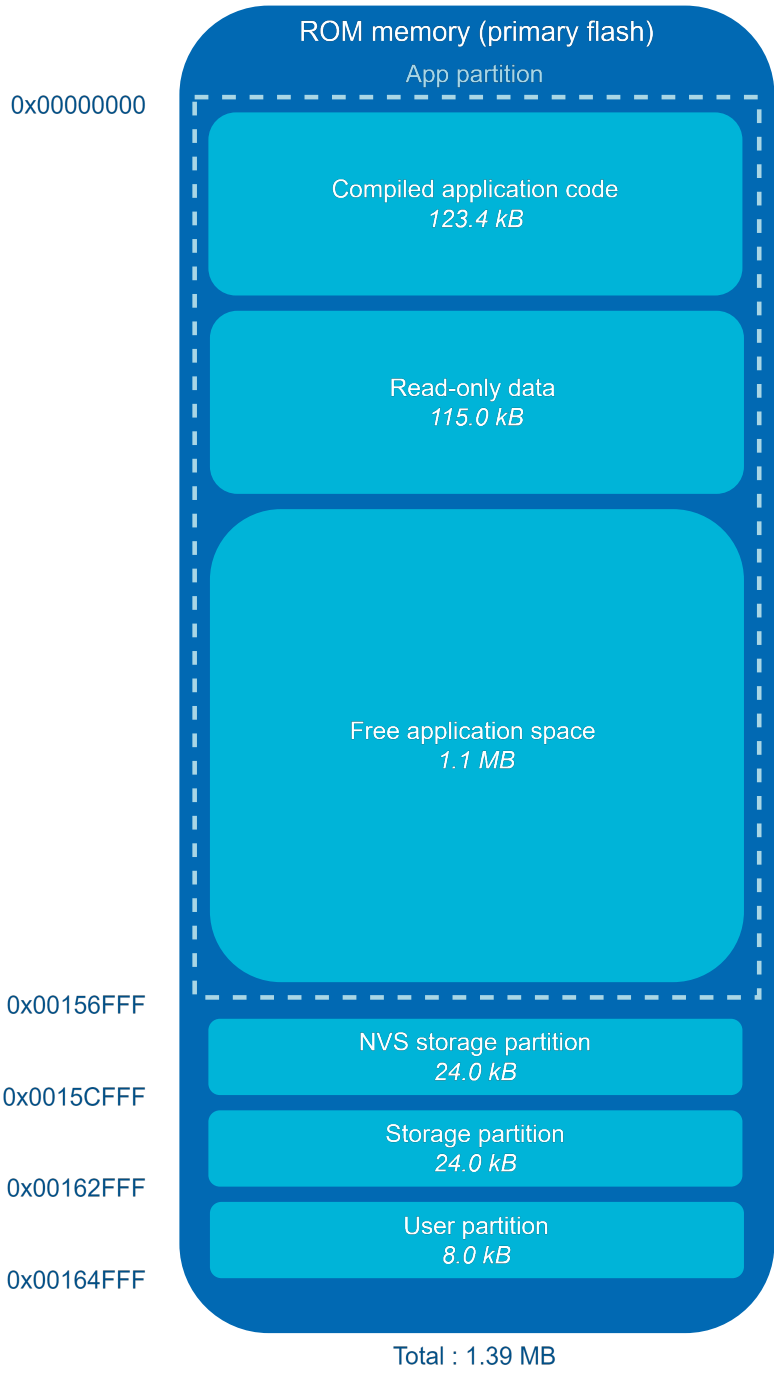


Figure 82: Rom memory footprint

The footprint indicates the size of the compiled application binary, not the total flash capacity. The remaining storage is partitioned for the bootloader, one-time programmable and SRAM primary.
Total : 1.5 MB

9 Consumption optimization

System consumption is a critical aspect of this project. To maximize the system's life, the consumption needs to be optimized. In this chapter, all the actions taken to reach this are detailed.

9.1 Buffer size optimization

Buffer sizes influence global consumption. This will dictate the write frequency of the SD card. A balance must be found between write frequency and the size of the data to be written. It's important to keep in mind that buffer size is limited by the available RAM space.

9.1.1 Metrics

To allow for comparison, two distinct metrics have been evaluated:

1. **Write efficiency [$\frac{mC}{kB}$]**

   The first metric, energy cost per unit of data, measures the energy required by the SD card protocol to write a kilo-bytes of data. It highlights the extra energy used for file system management relative to the payload size.

$$\text{efficiency} = \frac{\text{charge}_{\text{write}}}{\text{buffer}_{\text{size}}} = \frac{mC}{kB} \tag{19}$$

2. **Global average current [mA]**

   The second metric, average system current (I_{avg}), estimates the impact of write events on battery life. It is determined by integrating the write charge over the entire recording cycle, including both active write and sleep intervals.

$$I_{\text{avg}} = \frac{Q_{\text{write}} + Q_{\text{sleep}}}{t_{\text{total}}} \tag{20}$$

9.1.2 Test protocol

Objective:
Determine the optimal buffer size that minimizes energy consumption while ensuring data integrity and system stability.

Setup and equipment:
- **Hardware :** Embedded system with nRF54L15 and Class 10 SD Card.
- **Measurement :** *Nordic Power Profiler Kit II (PPK2)* : Source meter mode @4.2V, sampling rate 100 kHz.
- **Software :** Firmware builds with various buffer definitions.

Procedure

ID	Test	Stimulus	Pass criteria
1	*Energy Profiling*	Cycle through all buffer sizes (8.2 to 64 kB).	Identify configuration with lowest I_{avg} and efficiency.
2	*Timing Safety*	Measure t_{write} during active recording.	Write time must be significantly less than fill time.

Table 28: Summary of buffer optimization tests

i The power measurement should focus on the write sequence which includes the active SPI transfer time and the post-write SD card busy wait time.

9.1.3 Audio buffer size optimization

The audio samples write to SD is the greatest energy expenditure of the system. The system has then been tested with different buffer sizes from 8.2 to 64 kB. For each test, the average, the write time, the charge and the write interval have been measured and compile in Section AD.

9.1.3.1 Results

The write efficiency is ploted in Figure 83.

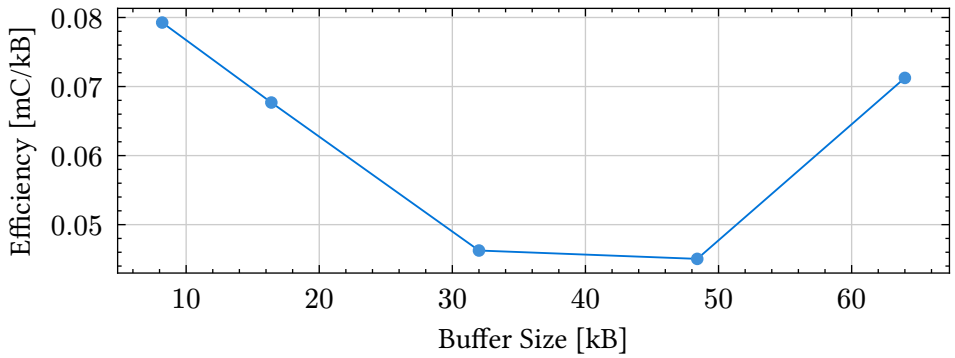


Figure 83: Energy efficiency per buffer size (lower is better)

The Figure 85 displays the ratio of write time to cycle period.

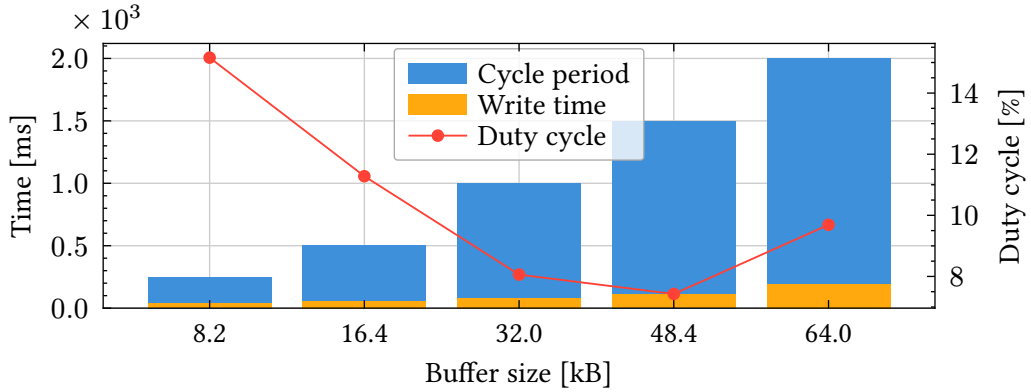


Figure 84: Write duration vs. Cycle period

Finally, the Figure 85 translates these factors into the final battery drain using the global current metric.

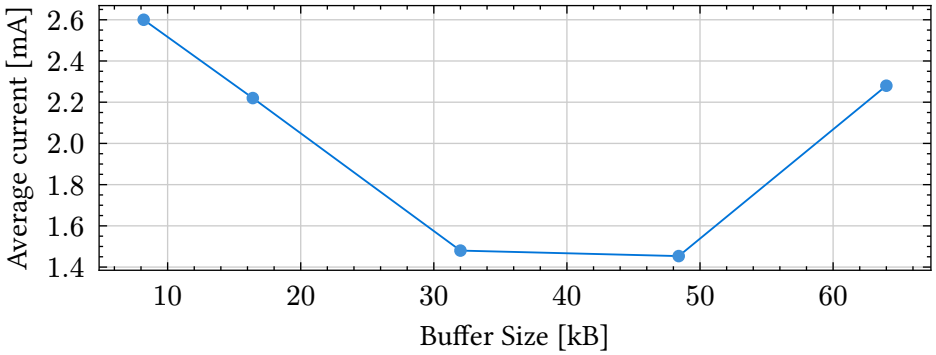


Figure 85: Global average current consumption

9.1.3.2 Conclusion

By comparing the three curves, it appears that efficiency and global current are correlated to the percentage of time spent writing outside of a cycle. The 48.4 and 32 kB configurations provide the lowest average current draw. While smaller buffers suffer from high overhead, the 64 kB buffer exceeds the available memory.

While the 48.4 kB buffer offers a marginal gain, the 32 kB buffer aligns perfectly with standard 512-byte SD card sectors. The final choice therefore leans towards buffers of $\mathbf{32kB}$.

9.1.4 BLE proximity buffer size optimization

Optimizing the size of proximity buffers is difficult. The buffer fill rate is not as regular as that of audio. Furthermore, writes are much less frequent, as proximity detection occurs only once every minute.

Therefore, optimization can only be done with estimates. The worst-case energy-consuming scenario will be used for this estimation (scanning interval = 60 s and discover devices per scan = 64). The same procedure as in the previous chapter will be followed with size size from 128 to 2048 devices per buffer. The results are available in the Section AD.

9.1.4.1 Results

The write efficiency is ploted in Figure 86.

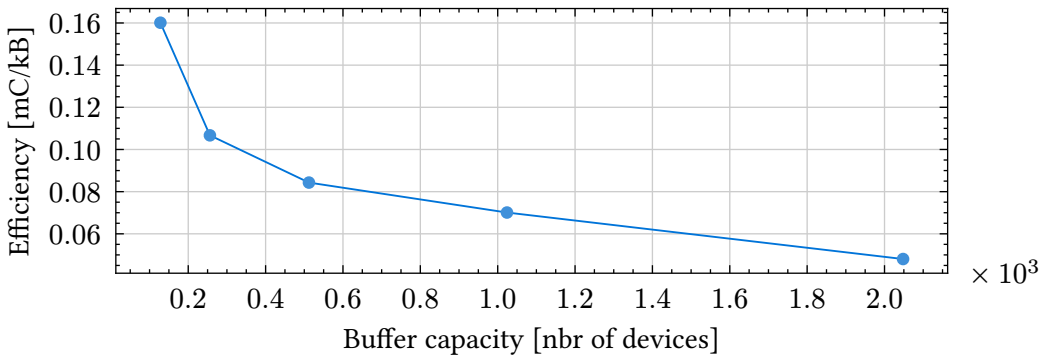


Figure 86: Energy efficiency per buffer capacity plot

The Figure 87 displays the ratio of write time to cycle period.

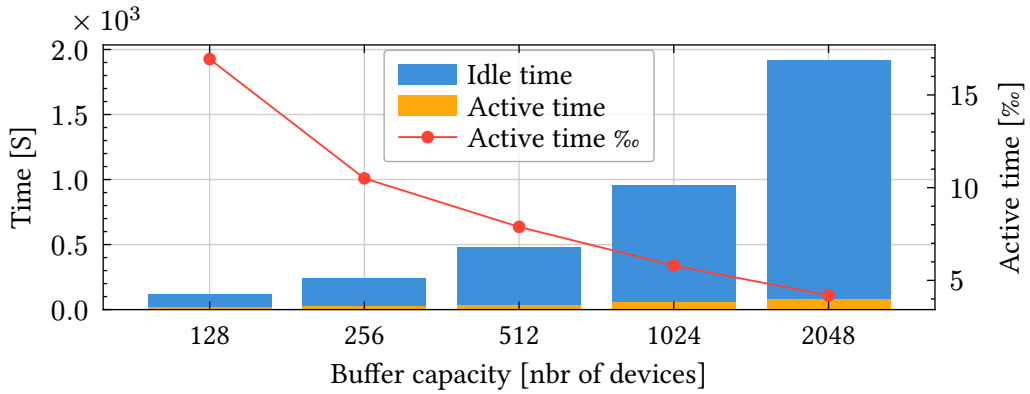


Figure 87: Active and idle time per buffer capacity plot

Finally, the Figure 88 translates these factors into the final battery drain using the global current metric.

The Figure 88 show the calculation results.

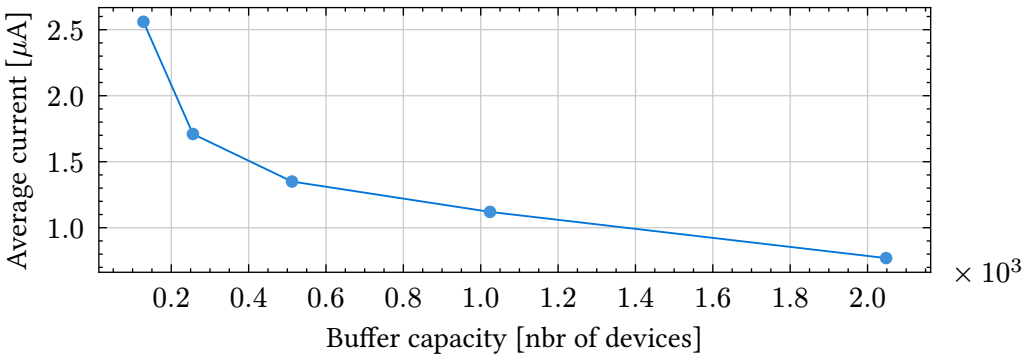


Figure 88: Average current per buffer capacity plot

9.1.4.2 Conclusion

The analysis demonstrates that increasing buffer capacity significantly reduces average current consumption. The 2048 devices configuration offering the lowest draw (0.77 μA). However, this theoretical optimum is strictly constrained by the *nRF54L15*'s available RAM.

As shown in Figure 81, the system operates with only 15.6 kB of free space when configured with 512-device buffers. The total memory pool available for proximity buffering is the sum of the currently allocated space and the free margin:

$$\mathrm{RAM}_{\mathrm{pool}} = \mathrm{RAM}_{\mathrm{used\ (512)}} + \mathrm{RAM}_{\mathrm{free}} = 15.36kB + 15.6kB = 30.96kB \quad (21)$$

Comparing this limit against the requirements for larger buffers:

$$\mathrm{RAM}_{\mathrm{req\ (2048)}} = 2 * 2048 * 15B = 61.44kB \quad (22)$$

$$\mathrm{RAM}_{\mathrm{req\ (1024)}} = 2 * 1024 * 15B = 30.72kB \quad (23)$$

This two options saturates the entire available pool, leaving no room left.

Consequently, the **512-device configuration ($2 * 7.68kB$)** is selected as the maximum viable capacity. It optimizes power consumption while maintaining a critical safety margin for system stability.

Future field analysis is recommended to further refine these parameters based on real-world detection density.

9.1.5 Comparison of theoretical vs. measured write times

The following table compares the theoretical write times calculated in the design phase with the actual values measured during hardware profiling.

Data Type	Buffer Size	Theoretical Time	Measured Time
Audio	32.0 kB	3.2 ms	80.6 ms
BLE proximity	7.7 kB	0.8 ms	37.8 ms

Table 29: Comparison of theoretical and measured SD card write times

The significant disparity between the theoretical and measured times is due to several factors not accounted for in basic transfer rate formulas:

- **File system overhead :** The FATFS library requires additional SPI transactions to update the File Allocation Table (FAT).
- **SD card latency :** SD cards have busy wait times after receiving data blocks.
- **Thread architecture :** The RTOS context switching adds latency to the raw data transfer.

Despite these discrepancies, the safety ratio remains robust, 50:1 for audio.

9.2 Microphone parameters optimization

The system records only when the sound level exceeds a predefined threshold. As described in Section 5.6.3.4.1, the microphone alternates between two modes. One is a sleep mode with analog Acoustic Activity Detection (awaiting). The other is a low-power mode with digital 1 Acoustic Activity Detection (recording). This approach is expected to substantially reduce energy consumption. Detection thresholds must be optimized. Activation should occur exclusively in response to the monkey's vocalizations while it is wearing the collar.

This process involves balancing sensitivity and power consumption. Excessive microphone sensitivity may capture extraneous sounds. This results in unnecessary energy expenditure. If the threshold is set too high, the system may miss some vocalizations. This can cause data loss.

Before testing new parameter sets, data from version 1 (V1) of the collar will be analyzed. Insights from this analysis will inform parameter choice and improve the optimization process. Laboratory optimization may not completely replicate field conditions. However, it provides a basis for later in field adjustments.

9.2.1 Record data analysis

Initial field testing of the collar produced substantial data. Because the microphone recorded continuously, most of the audio consists of ambient sound. For analysis, only samples containing vocalizations will be selected, based on criteria including clarity and representation of various vocalization types. The objective is to establish a baseline for subsequent parameter tuning.

Three characteristics were analyzed for each sample. Each characteristic helps to define one or more microphone parameters:

1. **Frequency spectrum**

   This characteristic determines the low-pass filter cutoff for the AAD A mode. Spectral analysis identifies the highest frequency present in the monkey's vocalizations.

   Parameter: AAD A low-pass frequency

2. **Sample level**

   This characteristic determines the absolute threshold levels for both analog and digital modes. Measuring the dB SPL of the vocalizations at the collar's distance enables the establishment of a trigger point to detect only monkey vocalizations.

   Parameter: AAD A absolute threshold and AAD D absolute threshold

3. **Difference between vocalization and ambient level**

   This characteristic determines the relative threshold settings. The ambient level establishes the floor, which is the level below which the threshold remains static to prevent false detections in quiet environments. The difference between vocalization and ambient levels defines the number of decibels by which the vocalization must exceed the background noise floor to be recognized as activity.

   Parameter: AAD D relative threshold and AAD D floor.

9.2.1.1 FS to SPL decibel conversion

Audacity was employed to analyze the recordings, using decibel values in Full Scale. Microphone settings are specified in Sound Pressure Level decibels. To enable adjustments based on the analyses, conversion from FS to SPL is required.

SPL vs FS decibel
Decibel Sound Pressure Level (SPL) quantifies the physical acoustic pressure in air relative to the threshold of human hearing.

Decibel Full Scale (FS) is a digital measurement in which 0 dB denotes the maximum signal level before clipping. The 0 dB reference is determined by hardware configuration.

Conversion from FS to SPL is possible when a fixed manufacturer reference is provided, enabling calculation of an offset. According to the *SPH0645LM4H-B* microphone (V1)

datasheet, a typical full-scale level is −26 dB FS for an input of 94 dB SPL at 1 kHz. The SPL for any value recorded with this microphone can be determined as follows:

$$\text{offset} = 94 + (-26) = 120 \tag{24}$$

$$\text{SPL} = \text{FS} + \text{offset} = \text{FS} + 120 \tag{25}$$

In the V1 firmware, a gain was applied to the recording signal path. To ensure accurate conversion, the digital gain must be accounted for. The resulting decibel change is calculated as follows:

$$\text{gain} = 20 * \log_{10}\left(\frac{\text{gain}}{\text{divider}}\right) = 20 * \log_{10}\left(\frac{3}{2}\right) = +3.52 \text{ dB} \tag{26}$$

Because the digital signal is artificially amplified in the firmware, the formula for determining the actual SPL is as follows:

$$\text{SPL} = \text{FS} + \text{offset} = \text{FS} + 120 - \text{gain} \tag{27}$$

i All values in the following sections have been converted from FS to SPL using the Equation 27.

9.2.1.2 Vocalization analysis

Different vocalizations were analyzed to allow for a representative view of their characteristics. This report provides a detailed analysis of only one of the vocalizations. The same procedure has been applied to all the other samples. All the analyzed files are available in the project repository (see Section AA).

Analysis example :
The following example shows the analysis procedure for a monkey scream (27_ZEU/REC_002 -> 1h52min43sec).

1. **Frequency of the spectrum**
   The peaks between 150Hz and 500Hz likely represent the core pitch of the screams. The dense area between 1kHz and 3kHz is where perceived harshness or shrillness is most noticeable. The high energy at the far left (below 50Hz) is likely ambient rumble or microphone handling noise.

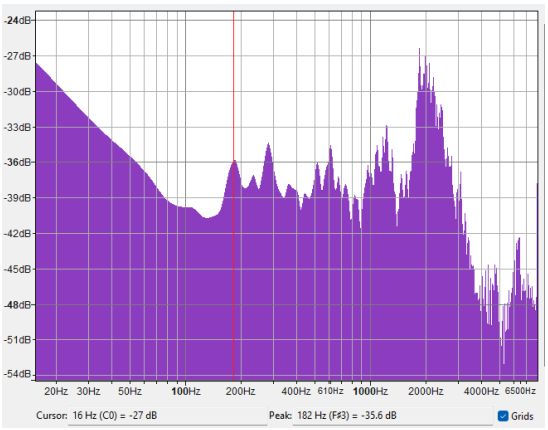


Figure 89: Vocalization frequency analysis

2. **Level of the samples**
   The monkey scream starts around 90.48dB.

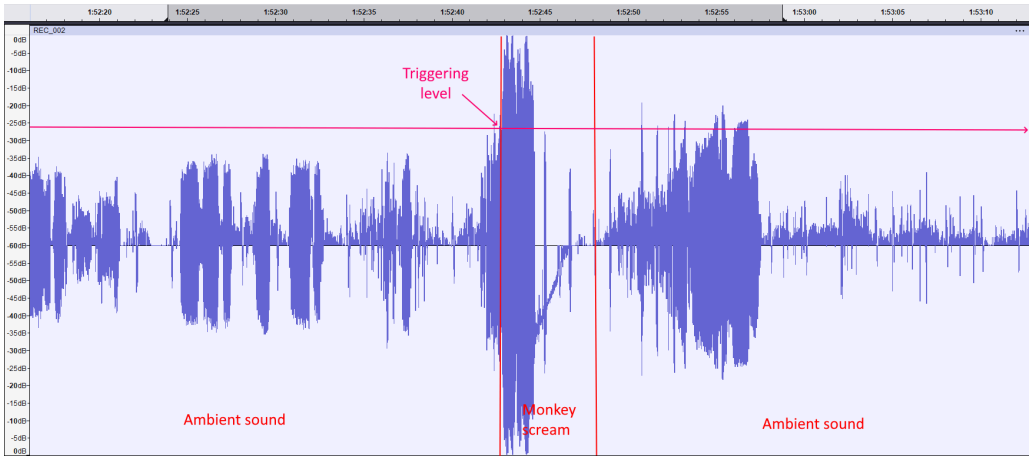


Figure 90: Vocalization level analysis

3. **Difference between vocalization and ambient level**

The foreground vs background RMS loudness (average) is measured to determine the contrast.

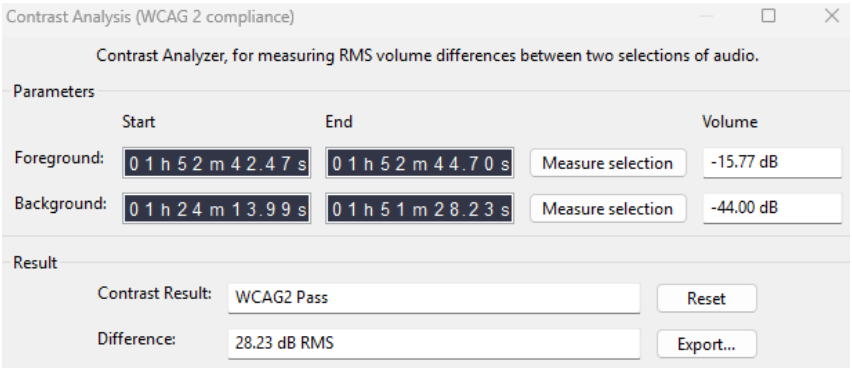


Figure 91: Contrast between vocalization and ambient level

The RMS value of the scream is 101.71dB. The volume difference with the background is 28.23dB RMS.

9.2.1.3 Ambient sound analysis

To facilitate improved categorization of the acoustic environment and precise calibration of detection settings, the continuous background recording (27_ZEU/REC_024) was analyzed.

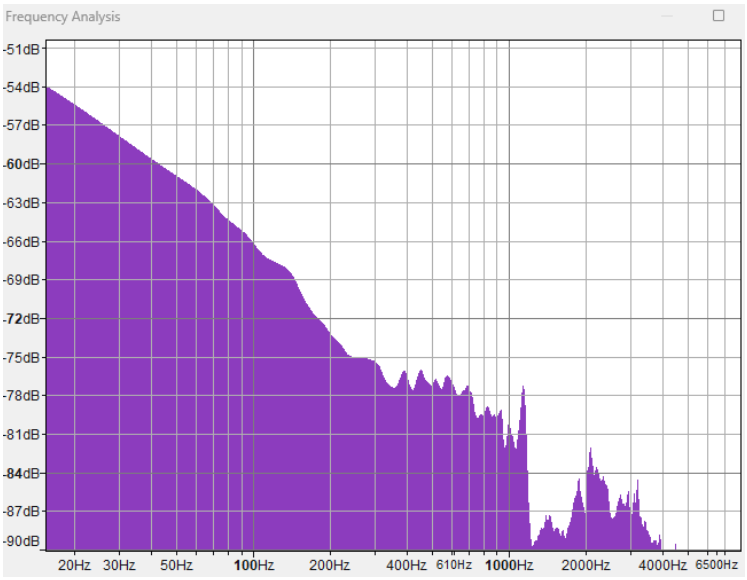


Figure 92: Ambient sound frequency analysis

The ambient sound samples show substantially lower overall sound levels, with a mean value of 63.15 dB. The observed spectral slope is characteristic of natural environments as represented in Figure 92.

The spectrum also shows a valley between 1 and 2 kHz. As vocalizations do not compete with background noise within this specific range, they remain distinct and audible.

9.2.1.4 Acoustic distractors analysis

In addition to screams, the monkey produces other sounds that are recorded by the microphone. Four specific actions were identified and analyzed using the same criteria as the vocalizations to evaluate the risk of false triggers:

1. Monkey eating
2. Monkey moving
3. Monkey hitting the collar housing
4. Monkey scratching/cleaning itself

The detailed results for these distractors are compiled in the summary table below.

9.2.1.5 Results summary

The following samples have been analyzed to allow for the selection of appropriate parameters.

File ID	Timestamp	Event type	Description
27_ZEU/REC_002	1h52m43s	Vocalization	Simple monkey scream
27_ZEU/REC_004	46m54s	Vocalization	Simple, isolated monkey scream
27_ZEU/REC_024	19m06s	Vocalization	Series of short screams spaced 1-5s apart
27_ZEU/REC_024	30m - 1h	Ambient	Continuous background environment noise
27_ZEU/REC_001	21m31s	Distractor	Eating (chewing and grunting sounds)
27_ZEU/REC_003	40m50s	Distractor	Moving (rhythmic stepping sounds)
27_ZEU/REC_006	18m49s	Distractor	Shock (sharp impact on collar housing)
27_ZEU/REC_007	25m40s	Distractor	Scratching

Table 30: List of field recordings selected for analysis

The Table 31 summarizes the results of all the analyses. The analysis of field recordings identified the following characteristics.

Sound source	Frequency range	Trigger level	RMS level	Contrast
Monkey vocalization	150-500hz & 2-3khz	90 to 100dB	100 to 110dB	28 to 45dB
Ambient noise	< 300hz	n/a	63.15dB	0 (baseline)
Eating	20-200hz & 500-1000hz	84.48dB	65.71dB	2.56dB
Movement	20-100hz	81.48dB	91.86dB	28.71dB
Shock	210hz & 2-5khz	109.48dB	113.11dB	49.96dB
Scratching	20-200hz	86.48dB	100.71dB	37.56dB

Table 31: Records analysis summary

9.2.1.6 Conclusion

Monkey vocalizations have a distinct frequency signature: 150–500Hz (fundamental) and 2–3kHz (harmonics). Trigger levels are 90–100dB and RMS levels are 100–110dB. Crucially, vocalizations are more than 28dB above the ambient noise baseline of 63.15dB. This contrast helps prevent startups from being drowned out by background noise.

There is potential for false triggers due to acoustic distractors with sound characteristics similar to monkey vocalizations, making differentiation challenging.

9.2.2 Primary microphone configuration

Based on these results, the following baseline configuration is proposed for the microphone.

Mode	Parameter	Value	Hex	Description
AAD A	*Threshold*	80 dB SPL	0x08	Set below 90dB vocalization level for early detection; avoids 63dB ambient trigger.
	Low-pass filter	2.0 kHz	0x02	Captures fundamental and harmonic frequencies. Can be increased to 4kHz if needed.
AAD D1	*Algorithm*	Both	0x03	Combines absolute (robustness) and relative (adaptability) to minimize false positives from wind/rain.
	Abs. Threshold	80 dB SPL	0x370	Fixed trigger point set 10dB below the expected vocalization level.
	Abs. Min Pulse	29 ms	0x12C	Filters out short mechanical shocks (<10ms) to prevent false triggers.
	Rel. Threshold	15 dB	0x8F	Ensures detection even if the noise floor rises.
	Rel. Min Pulse	29 ms	0x12C	Same as absolute; avoids short duration artifacts.
	Floor	65 dB SPL	0xA0	Set slightly above the 63.15dB average background level to define the baseline.

Table 32: Analysis of microphone configuration parameters

The dual-algorithm approach requires signals to exceed both the absolute threshold and be 12dB above the background level, reducing false triggers.

The microphone has limitations that must be considered:

1. The AAD modes rely only on amplitude detection. It cannot perform frequency discrimination. This means any sound with similar amplitude characteristics may trigger recording.
2. The built-in voice band filter (100-500Hz) provides some protection against low-frequency movement noise. It cannot distinguish between the target monkey's vocalizations and those of nearby conspecifics. Environmental sounds such as rain, wind or other animals within the 150-3000Hz range may also trigger false recordings.

3. The 4kHz low-pass filter may attenuate harmonic content above this frequency. The primary vocalization energy falls within the passband.

These settings prioritize capturing all vocalizations over minimizing false positives. Laboratory tests and field adjustments will be necessary. The settings will be evaluated based on actual habitat conditions and collar positioning effects. The conservative thresholds ensure that target sounds are not missed during deployment.

9.2.3 Test protocol

Objective:
Tuning the AAD settings of the *T5848* microphone for triggering only monkey screams while not reacting to ambient noise or distractors.

Setup and equipment:
- **Hardware :** Embedded system with *T5848* microphone in 3D printed collar housing.
- **Audio source :** Stereo speaker setup (*Maxxtro USB min speaker*). Sound level control with a *Norsonic Nor150* sound level meter.
- **Test files :** Stereo .WAV (Left: ambient sound / Right: monkey vocalization).
- **Measurement :** *Digilent Analog Discovery 2* connected to the microphone WAKE pin and I2S to monitor the signal. *Audacity* to analyze the audio files produced during the test

Procedure

ID	Test	Stimulus	Pass criteria
1	*Indoor sensitivity*	Right channel (monkey vocalization)	WAKE pin transitions HIGH. I2S streaming starts. Recording persists for tail duration.
2	*Ambient rejection*	Left channel (ambient, eating, shocks)	WAKE pin remains LOW. System stays in sleep mode regardless of volume variation.
3	*Dual-channel*	Stereo mix (vocalization + ambient)	System triggers only when scream peaks exceed threshold. Recording persists for tail duration.

Table 33: Summary of AAD Tuning Validation Tests

i The tests are designed to compare and calibrate the absolute, relative and combined algorithms for AAD D1. First, test with the absolute algorithm only, then with the relative algorithm only and finally with both combined to measure individual and combined effects.

i Reproduce these tests with different parameter sets to determine which set optimizes the algorithms' performance.

i Set the speaker levels to replicate as closely as possible the natural sound levels of a monkey's vocalizations and its environment, aiming to simulate realistic test conditions.

i Use the analyzed sound for setup validation: right channel will play monkey vocalizations and left channel will play ambient sounds. This checks the correct differentiation and playback of sources.

9.2.4 Microphone parameters tuning

The test results are provided in the project repository (see Section AA).

9.2.4.1 Results

Testing identified the following configuration for field deployment:

Mode	Parameter	Value	Hex Code
AAD A	*Threshold*	85 dB SPL	0x0A
	Low-pass filter	2.0 kHz	0x02
AAD D1	*Algorithm*	Both	0x03
	Absolute threshold	75 dB SPL	0x1E0
	Absolute min pulse	188 ms	0x7D0
	Relative threshold	15 dB	0x8F
	Relative min pulse	19 ms	0xC8
	Floor	65 dB SPL	0xA0

Table 34: Microphone optimal configuration

9.2.4.2 Analysis

The system detects all target vocalizations at 90dB SPL and reliably rejects ambient noises up to 75dB SPL.

However, a trade-off exists between power efficiency and selectivity:

- **Wake-up (AAD A):** To minimize consumption, the primary trigger uses simple analog thresholding. It is susceptible to false positives from mechanical noises such as scratching, movement or collar impacts.
- **Active Filtering (AAD D1):** Once awake, the digital algorithm analyzes the signal. It effectively distinguishes vocalizations from these mechanical artifacts.

This two-stage approach ensures valid data capture while tolerating the limitations of low-power logging.

9.2.4.3 Observations

The tests carried out allowed for the following observations:

1. **Threshold strategy :** AAD A operates at higher threshold to minimize unwanted records. AAD D1 maintains a lower threshold level to ensure continuous recording once triggered.
2. **Relative algorithm essential for field robustness :** The adaptive threshold mechanism allows rejection of false positives in saturated background environments.

3. **Minimum pulse duration effectiveness :** By extending the pulse length, it offers a viable strategy for rejecting digital detection of scratching and shock events.
4. **Unwanted triggering event :** Due to the simple primary analogical detection, the system will trigger unintentionally. But the digital detection will be able to identify the majority of them once the recording process has started and will not prolong the recording.

9.2.5 Field tuning guide

Based on the tests performed and the observation, a guide is proposed in this section to help tune the system in the field.

Problem	AAD adjustment	Effect
Too many wind /rain recordings	A thr ↗, D1 rel thr ↗	Fewer wake events, stronger contrast required
Constant triggering during movement	A thr ↗	Reduces sensitivity
Missing vocalization	A thr ↘	More sensitive
Missing vocalization starts	A thr ↘	Earlier detection, more sensitive
Long impacts / scratching recordings	D1 abs min pulse ↗ D1 rel min pulse ↗	Filters shocks/impacts
Long empty recordings	D1 abs thr ↗, D1 rel thr ↗	Reduces sensitivity
Long background recordings	D1 rel thr ↗	Stronger contrast required

Table 35: AAD configuration adjustment strategies

Changing one parameter to fix a problem often creates another. The system must be tuned as a whole.

9.3 Recording tail timer optimization

Another factor influencing the system's overall power consumption is the recording tail timer. This allows recording to continue for a certain period after the last detected acoustic event. The goal of this timer is to avoid file fragmentation by recording interactions in a single block.

Setting this value too short may prevent the end of an acoustic event from being recorded. Conversely, if this value is too long, it could cause the system to record for too long after the actual activity has ended.

To allow for the correct selection of this value, the average length of an audio event or interaction must be known. To do this, the recordings produced during the first campaign will be analyzed. The goal is to determine :

1. **Event duration :** The duration of a vocalization event.
2. **Event interval duration :** The interval between two successive vocalization events.

To measure the duration, the *Label Sounds* analyses tool from *Audacity* to find all the sounds intervals exceeding 90dB in a 24-hour sample.

9.3.1 Record data analysis

A typical monkey vocalization has a rapid attack, followed by sustained high amplitude for a few seconds, as shown in the figure Figure 93.

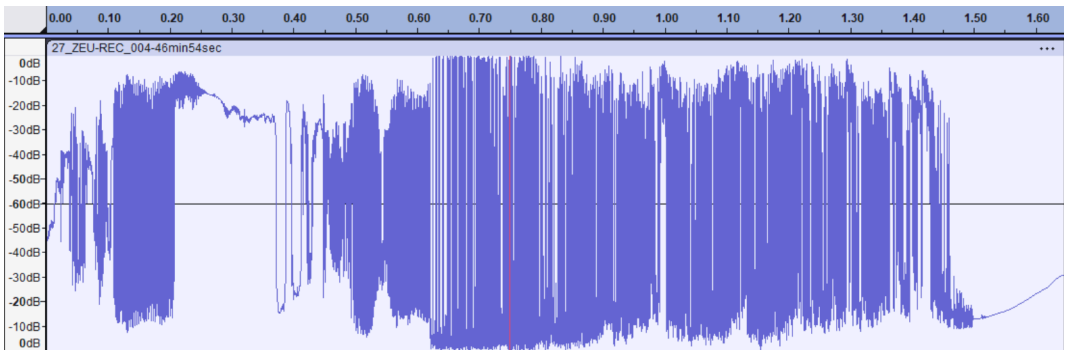


Figure 93: Typical monkey vocalization

The analyzed vocalizations generally do not last long. The mean measure duration is 0.753 seconds. However, some screams last up to 2 seconds. That is the maximal event duration measured.

However, during certain interactions, several vocalizations follow one another with short intervals (see Figure 94). The measured mean interval is 1.677 seconds. The maximal measured interval is 4,258 seconds.

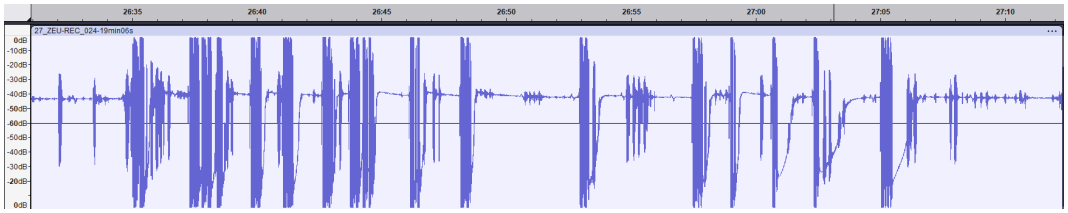


Figure 94: Complex interactions

The system will be able to record this series of vocalizations without interruption, provided that the timer does not expire during the interval. Therefore, a value must be chosen that provides a safety margin to avoid creating two files for the same interaction.

The maximal time to cover is 4,258 seconds. To be as accurate as possible, a value of 5 seconds would be needed. A value of 10 seconds would provide a sufficient safety margin. Higher values are also possible, but they risk strongly influencing the system's behavior.

All the measure event and event interval durations are available in the Section AE.

9.3.2 Tail timer setting system impact

To evaluate the impact of this tail timer setting on the system, a simulation is performed. A real situation is simulated for 1 hour using the following parameters.

9.3.2.1 Simulation parameters

The parameters chosen are selected based on previous observations.

- **Target vocalizations :** 20 events per hour (duration is randomly distributed in min/max range).
- **Complex interactions :** 2 sequences per hour with the maximum measured interval of 4.258s and 12 vocalizations (duration is randomly distributed in min/max range).
- **Recording specs :** 16kHz, 16-bit, mono.
- **Microphone current :** 389.03 μA for recording mode and 2.13 mA for saving mode.

9.3.2.2 Results

The recording time over an hour, the generated data size and the global current are calculated for different tail times. The calculation sheet is available in the project repository (see Section AA). The results are compiled in the Table 36.

Tail value [s]	Rec/hour [s]	Data [MB]	I_{global} [mA]
0	47	1.50	0.41
5	131	4.19	0.45
10	**236**	**7.55**	**0.50**
30	656	20.99	0.71
45	971	31.07	0.86
60	1286	41.15	1.01
Always recording	3600	115.20	2.13

Table 36: Tail timer impact simulation results

9.3.2.3 Conclusion

The selection of the tail timer is a trade-off between the risk of file fragmentation or data loss and the preservation of battery and storage autonomy. **A value of 10 seconds is recommended.** It insure all complex interactions recording with a safety margin (maximal gap of 4.258s). Moreover, the current consumption difference with the 5 seconds tail timer is negligible.

9.4 BLE parameters optimization

This chapter covers proximity detection optimization, specifically the optimization of scanning and advertising parameters.

9.4.1 Probability equations

This section outlines the mathematical model used to determine the probability that a scanning device will detect an advertising device.

9.4.1.1 Parameters

The following parameters are used in the detection equations:

- w_s: The **scanning window**, representing the duration the scanner is actively listening for advertisements.
- T_s: The **scanning interval**, representing the total period of one scan cycle.
- T_a: The **advertising interval**, representing the time between consecutive advertisement packets from the target device.
- t: The **observation time**, representing the time observation time for the trial.
- N: The average number of advertising events that occur during the observation time t.
- P: The desired cumulative probability of detection.

9.4.1.2 Detection probability

The probability of detecting a single advertising event (P_{det}) is determined by the scanner's duty cycle. It corresponds to the ratio between the active scan window (w_s) and the total scan interval (T_s):

$$P_{\text{det}} = \frac{w_s}{T_s} \tag{28}$$

Consequently, the probability of missing a single advertising event is

$$P_{\text{miss}} = (1 - P_{\text{det}}) = \left(1 - \frac{w_s}{T_s}\right) \tag{29}$$

The detection process can be modeled as a series of Bernoulli trials, where each advertising event is a trial.

A Bernoulli trial is a random experiment with exactly two possible outcomes: success and failure. The probability of success is constant every time the experiment is conducted and each trial is independent.

The average number of advertising events N that occur during a given observation time t is:

$$N = \frac{t}{T_a} \tag{30}$$

The probability of missing all N of these independent events is P_{miss}^N. Therefore, the cumulative probability of detecting at least one of these N events is:

$$P_{\text{detect}(N)} = 1 - P_{\text{miss}}^N = 1 - \left(1 - \frac{w_s}{T_s}\right)^N \tag{31}$$

By substituting Equation 30 into Equation 31, we derive the complete formula for detection probability as a function of observation time t:

$$P_{\text{detect}(t)} = 1 - \left(1 - \frac{w_s}{T_s}\right)^{\frac{t}{T_a}} \tag{32}$$

This equation is the key to the model, linking the physical parameters of both the scanner and the advertiser to the probability of a successful detection at time t.

This model can be used in two critical ways for system design:
1. To predict the performance of a given set of parameters.
2. To determine the parameters required to achieve a target performance.

9.4.1.3 Single scan detection probability

A key performance metric is the probability of detecting a device within a single scan interval. This is particularly relevant for systems with long scan intervals, as it defines the per-cycle detection chance. It can be calculated by replacing the observation time t by the scan interval T_s in Equation 32:

$$P_{\text{1scan}(T_s)} = 1 - \left(1 - \frac{w_s}{T_s}\right)^{\frac{T_s}{T_a}} \tag{33}$$

This value provides a clear benchmark for the effectiveness of a single scan procedure.

9.4.1.4 Minimum detection time

Conversely, the model can be inverted to solve for the time t required to achieve a specific cumulative detection probability P. This is essential for guaranteeing detection within a known timeframe. By rearranging Equation 32 to solve for t:

$$P = 1 - \left(1 - \frac{w_s}{T_s}\right)^{\frac{t}{T_a}} \tag{34}$$

$$1 - P = \left(1 - \frac{w_s}{T_s}\right)^{\frac{t}{T_a}} \tag{35}$$

$$\ln(1 - P) = \left(\frac{t}{T_a}\right) * \ln\left(1 - \frac{w_s}{T_s}\right) \tag{36}$$

This gives the final equation for the minimum required time to achieve the probability P:

$$t = T_a * \frac{\ln(1 - P)}{\ln\left(1 - \frac{w_s}{T_s}\right)} \tag{37}$$

9.4.2 Parameters range definition

The parameters range is defined in this chapter.

9.4.2.1 Advertising

The legacy advertisement interval goes from 20ms to 10.24s. [36] The interval time will influence the consumption and the reactivity of the system.

Online power profiler tool

All the current values used in this chapter were taken from the *Nordic Semiconductor's Online Power Profiler for Bluetooth LE* tool [37]. It allows simulating current consumption based on the chosen parameters. The Figure 95 shows the parameters used for these estimations.

Source: https://devzone.nordicsemi.com/power/w/opp/2/online-power-profiler-for-bluetooth-le

Figure 95: Online Power Profiler for Bluetooth LE parameters

The tool is based on a model of measured values and does not show the actual measurement. The results are therefore estimates of the expected value. It is intended for evaluation purposes only and will not yield exact numbers in every use case. Testing shows that the estimated average current is typically within 5% of the actual value for the reference parts.

Advertising current vs. interval

To begin the selection process, the advertisement current was plot compared to the advertisement interval in the Figure 96.

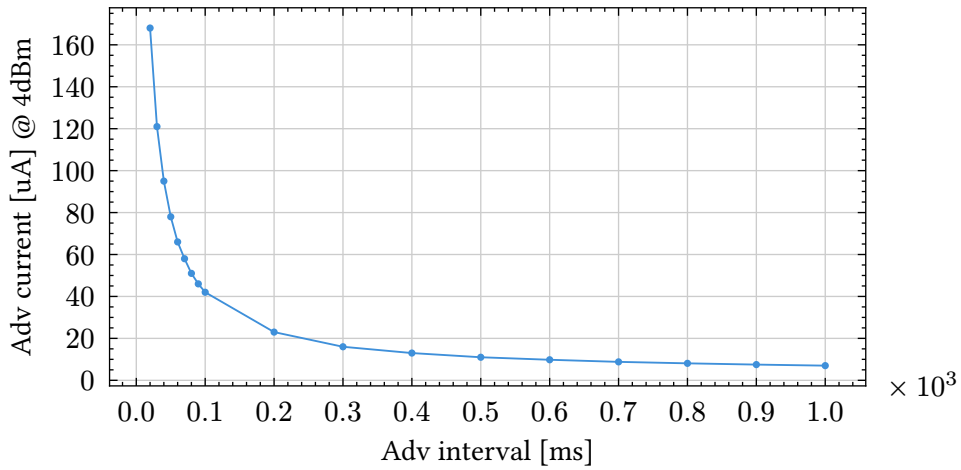


Figure 96: BLE Advertisement Current vs Interval

It appears that above an advertising interval of 200 ms, the differences in power consumption are very low. This allows to exclude values less than 200 ms for this parameter. **The test will therefore focus on an advertising interval between 200ms and 1 s.**

9.4.2.2 Scanning

The scanning windows and interval will also influence the system's consumption and reactivity.

To maintain proper system availability, the scanning interval range will be limited to a maximum of 5 minutes. The legacy maximum scanning window and interval are also 10.24 seconds. [38] However, this value can be circumvented by manually managing the scan cycles.

The scanning procedure is more energy-consuming than the advertising ($3mA$ VS less than 200 μA). By limiting the scanning windows to 5 seconds, the consumption is controlled.

The test will use :

- **A scanning windows between 1 and 5 seconds.**
- **A scanning interval between 10 seconds and 5 minutes.**

9.4.3 Model validation

In the exploration phase, the tests carried out aim to validate the developed model. The following sets of parameters were tested arbitrarily.

	Adv interval [s]	Scan window [s]	Scan interval [s]	Detection prob	Real detection prob	Current [uA]
1	0.2	0.4	5.2	0.875	0.524	167.63
2	0.2	0.4	10.24	0.869	0.476	90
3	1	2	10.24	0.892	0.333	434.31

Table 37: Model validation test

The results in Table 37 confirm three critical system behaviors:

- **Interval dominance:** Doubling the scan interval reduces the average current consumption by nearly 50%.
- **Window sensitivity:** Current draw scales linearly with the scan window. A fivefold increase in window size results in a fivefold increase in current, identifying it as the primary driver of energy consumption.
- **Model accuracy:** Empirical detection probabilities were approximately 50% lower than theoretical estimates.

  The equations can be updated with a correction factor :

$$P_{\text{detect}(t)} = 1 - \left(1 - 0.5 \cdot \frac{w_s}{T_s}\right)^{t/T_a} \tag{38}$$

$$t = T_a \cdot \frac{\ln(1-P)}{\ln\left(1 - 0.5 \cdot \frac{w_s}{T_s}\right)} \tag{39}$$

This indicates the direction in which optimization efforts should be focused.

9.4.4 Availability and window scaling

Reducing the scanning window decreases the detection probability. Increasing the window relative to the advertising interval serves as a lever to ensure detection, as shown in Figure 97.

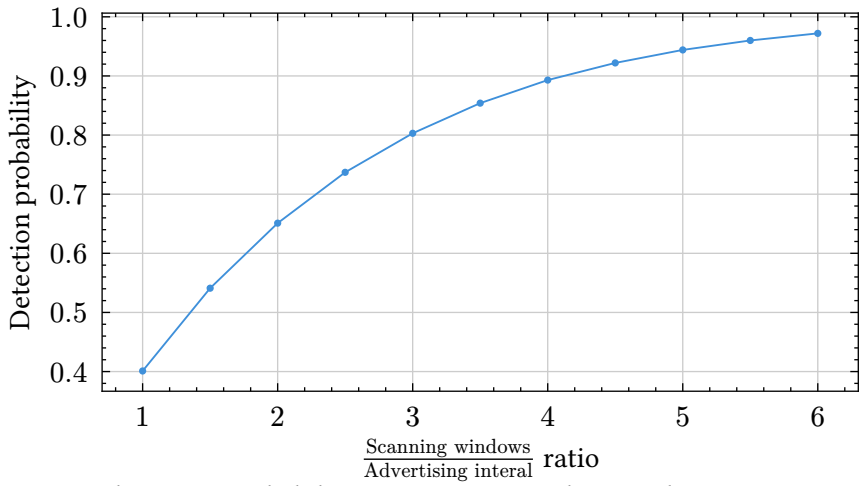


Figure 97: BLE detection probability vs scanning windows / advertising interval ratio

Data reveal that the single-scan detection probability is primarily driven by the ratio of the scanning window to the advertising interval. This probability is also influence by the scanning interval but the proportional relationship between the window and advertising interval facilitates a more predictable optimization process. By maintaining this specific ratio, a simulation can achieve with a consistent target probability across a wide range of duty cycles.

9.4.5 Target probability selection

To guide parameter optimization, a target probability for a single scan must be defined. This should serve as a basis for selecting parameter sets. To balance responsiveness with battery life, a target probability of $P = 0.9$ for a single scan was established. It is a compromise between two important aspects: power efficiency and system dependability. It is helpful to consider the impractical extremes in order to see why 0.9 is a good target:

- **Target $P = 0.99$**:
  ✅ Extremely reliable, near-instant detection.
  ❌ This requires very aggressive parameters. The power cost is often high, leading to poor battery life.
- **Target $P = 0.50$**:
  ✅ Excellent power savings.
  ❌ The system would need to wait multiple scan intervals before detecting a device.

Therefore, targeting $P = 0.9$ enables the system to be power-efficient while still providing robust detection. The Figure 97 also demonstrated that to achieve higher probabilities the scanning windows/advertising interval ratio must be very large, which implies high consumption.

9.4.6 Parameter optimization by simulation

A Python simulation was used to handle the many possible combinations of advertising and scanning parameters. The script tested specific ranges for advertising intervals (100–1000 ms), scan windows (1000–5000 ms) and scan intervals (10–300 s). It filtered configurations to meet a target detection probability of $P = 0.9 \pm 0.01$ (see the project repository, Section AA).

For each valid combination, the theoretical current consumption was calculated using the following formula:

$$I = I_{\text{adv}} + I_{\text{scan}} * \frac{w_s}{T_s} \tag{40}$$

Advertising current I_{adv} was derived from the Nordic Online Power Profiler, while scanning current I_{scan} was measured at 3 mA.

From 322 possible matches, the list was reduced to the 30 most efficient options by choosing the configuration with the lowest current for each scanning interval.

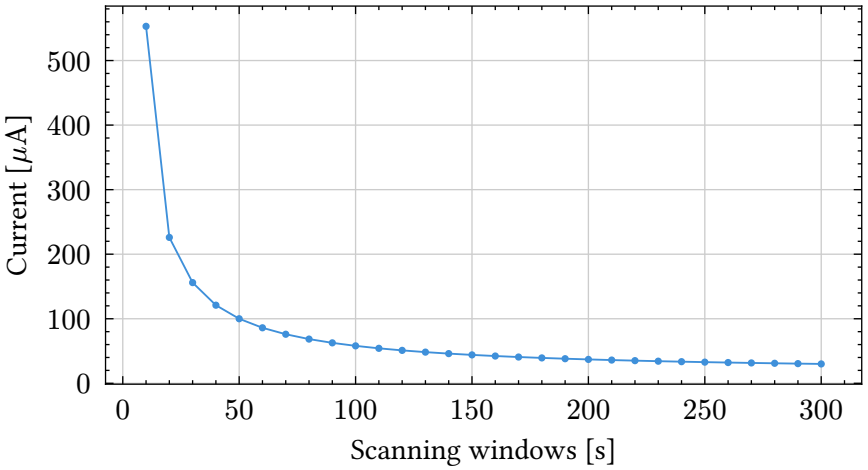


Figure 98: BLE detection probability vs scanning windows time

The resulting dataset, illustrated in Figure 98, provides a standardized baseline for physical testing, allowing for parameter selection based on specific requirements for either power consumption or system reactivity. The detailed list is available in Section AF.

The most interesting range is between scanning interval from 60 to 180 seconds. They allow for a balance between consumption and availability

9.4.7 Conclusion

Adding proximity detection is possible because intelligent saving consumes less energy than continuous recording. The savings, therefore, set our maximal energy budget for proximity detection. The average current consumption over 10 seconds during the recording phase is considered.

$$I_{\text{prox}_{\text{max}}} = I_{\text{rec}_{\text{V1}}} - I_{\text{rec}_{\text{V2}}} = 4.02mA - 2.19mA = 1.83mA \tag{41}$$

This value is the maximum allowed. Lower power use enables a smaller battery. A smaller battery reduces the collar's size and weight. It is essential to choose a value that saves power and keeps the collar compact.

The chosen set of parameters is :

- **Scanning interval :** **60 seconds**
- **Scanning window :** **1.4 seconds**
- **Advertising interval :** **0.3 seconds**
- **Average current :** **86.0 μA**

This parameter ensures reliable proximity detection. It uses only a small part of the power savings. It balances animal-tracking availability with hardware weight and size constraints.

10 Validation

This chapter discusses the evaluation of boards and requirements. The battery life is estimated. Finally, the global system consumption is addressed.

10.1 nRF54L15 development board

Before diving into the *nRF54L15* development board modification and testing, it is important to know that the card was developed by a member of the ECS research group of the HES-SO Valais/Wallis, printed by *EuroCircuits* and assembled by the electronics workshop at the HES-SO Valais/Wallis.

This chapter presents the board tests.

10.1.1 Development board power-on results

The result of the development board power on process is described in Table 38.

Block	TP	Expected	Measured	Unit	OK/NOK
Battery	V_BATT	3.00 to 3.60	3.60	V	OK
+3V3 step-down	VDD_nRF	3.30	3.28	V	OK

Table 38: Development board power-on results

10.1.2 Development board functionality tests

Once the development board has been successfully powered on, the features are validated based on the requirements specification.

Block	Specification	OK/NOK
Computing unit	SoC executes basic code	OK
HF clock	32MHz Crystal oscillates stably	OK
LF clock	32.768kHz Crystal oscillates stably	OK
Programming	SWD interface detects target ID	OK
RF hardware	Radio emits signal at 2.4GHz	OK
User interface	User LED toggles	OK
Reset circuit	Physical Reset button reboots the SoC	OK
Power switch	Source switch correctly selects battery or regulator	OK

Table 39: Development board hardware specifications validation

10.2 MIC/SD board

This board was also design by the same person, printed by *EuroCircuits* and assembled by the electronics workshop of the HES-SO Valais/Wallis.

In this chapter, the board tests and the changes made are presented.

10.2.1 MIC/SD board modifications

The defined pinout for SD card communication has caused problems. This chapter explains these issues and provides solutions.

SD card detect interrupt
With the initial pin assignment, the SD card detect interrupt callback was never called. The SD_DC was assigned to pin 1 of the port (P2.01). But port 2 of the *nRF54L15* doesn't support GPIOTE (GPIO tasks and events), so it can't support interrupts. [39], [40]

The SD_DC signal has been moved to a free pin on port 1: P1.05 (see Table 41).

SD card SPI signal
The SPI signal was also mapped to port 2. This posed two problems:

1. The pins of port 2 cannot be assigned to any signal (clock, SPI SDI, SPI SDO, ...), refer to the documentation. [41] An online pin planner tool is available to help map the *nRF54L15*. [42]

   The pinout has been modified to correspond to the pin limitations. The Table 40 shows the modifications and the reasons.

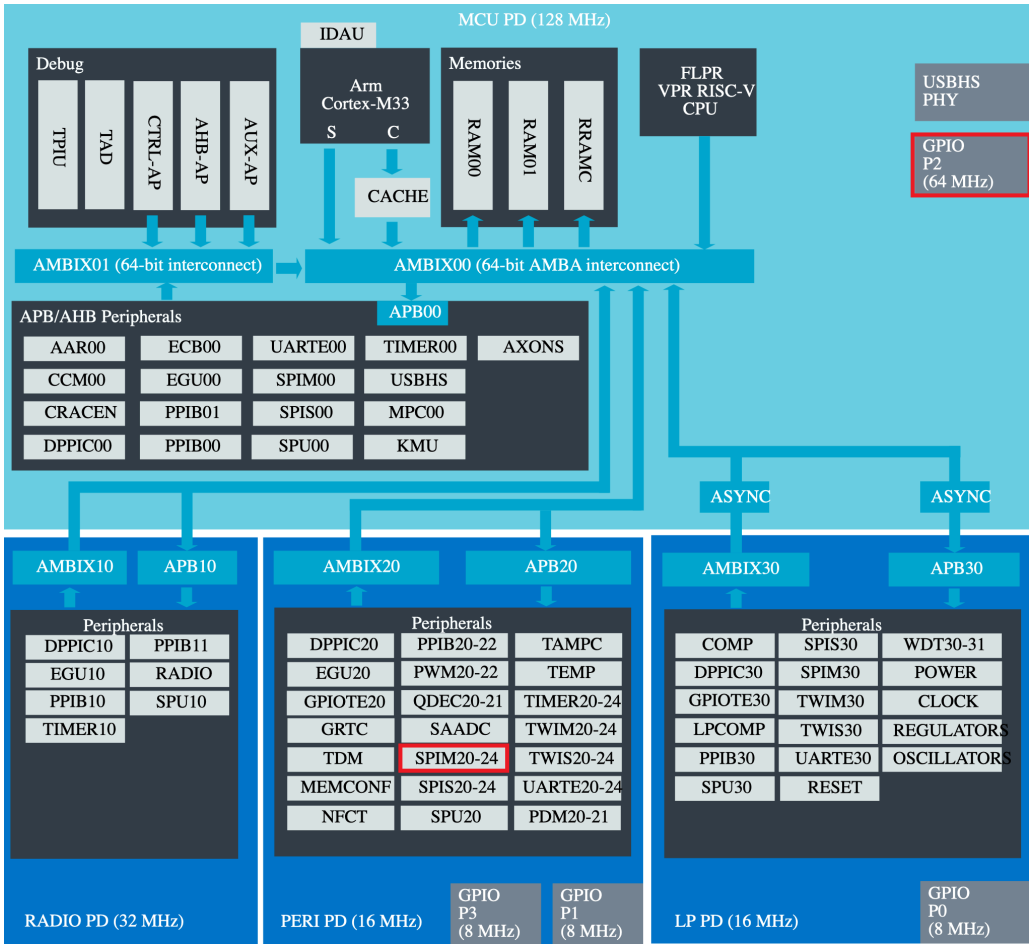
Name	Old pin	New pin	Explanation
SD_CLK	P2.04	P2.01	P2.04 doesn't support clock signal
SD_MISO	P2.03	P2.04	P2.03 was not support for SPI20
SD_MOSI	P2.05	P2.02	P2.05 was not support for SPI20
SD_CS	P2.06	-	P2.06 support GPIO pin
Burn	P2.01	P2.03	Leave P2.01 for SPI CLK, P2.03 support GPIO pin

Table 40: SoC pin correction for port 2 pin

   The SPI is working correctly with this reorganization.

   This is not the final board pinout. See the following point.

2. A power domain is a large chip section that can be powered on or off independently. This helps optimize energy consumption. With SPI20 GPIO on port 2, a power domain conflict occurs. SPI20 is in the peripheral domain (PERI), while port 2's GPIO pins are in the microcontroller domain (MCU), as shown in Figure 99. When the chip enters low-power states, each domain shuts down independently. As a result, the SPI peripheral loses connection to the GPIO pins. This break prevents successful communication.



Source: *https://academy.nordicsemi.com/courses/nrf54l-series-express-course/lessons/lesson-2-power-domains-event-system-and-gpio/topic/power-domains/*

Figure 99: nRF54L15 power domain

Different solutions are possible to fix this problem:

1. Disable power management

   ❌ High power consumption (~500 μA)
2. Handle power management manually

   ❌ Complex and high power consumption (~300 μA)
3. Change SPI peripheral

   ✅ Software modification. SPIM00 resides in the MCU Domain, matching port 2 power domain.
4. Change SPI pin to port 1

   ✅ Hardware modification. Port 1 belongs to the peripheral domain, matching the SPIM20 power domain.

   As the board is a prototype with plated holes, modifications are possible. The last solution was then chosen. The SPI GPIOs were moved to port 1 pin. The modifications are listed in the Table 41.

Name	Old pin	New pin	Function
Burn	P2.01	-	Release
SD_MOSI	P2.02	P1.09	SPI MOSI
SD_CLK	P2.03	P1.03	SPI clock
SD_MISO	P2.04	P1.04	SPI MISO
SD_DC	P2.05	P1.05	SPI card detect
SD_nCS	P2.06	P1.06	SPI chip select
MIC_THSEL	P1.03	P2.00	Mic threshold select
MIC_THSEL_OE	P1.04	P2.02	Mic threshold OE

Table 41: SoC pin reassignment for power domain alignment

10.2.2 MIC/SD board power-on results

The result of the MIC/SD board power-on process is described in Table 42.

Block	TP	Expected	Measured	Unit	OK/NOK
Input header	VDD_nRF	3.30	3.28	V	OK
Mic LDO input	VDD_MIC	3.30	3.27	V	OK
Mic LDO output	VDD_1v8	1.80	1.82	V	OK
SD	VDD_SD	3.30	3.27	V	OK
Mic level shifter	MIC_...	3.30/1.8V	3.27/1.78	V	OK
Mic THSEL level shifter	MIC_THSEL	3.30/1.8V	-	V	-
Burn capacitor	Vcap	3.30	-	V	-

Table 42: MIC/SD Board Power-On Process Result Table

The level shifter for the microphone THSEL is not mounted on the used prototype board. The THSEL level is shifted in the main microphone level shifter. The release related hardware is also not mounted.

10.2.3 MIC/SD board functionality tests

Once the MIC/SD board has been successfully powered on, the specific hardware peripherals are electrically validated.

Block	Specification	OK/NOK
Audio interface	I2S clock and word select signals are active	OK
Audio data	Microphone outputs digital data on SD line	OK
Wake interrupt	MIC_WAKE pin toggles on acoustic trigger	OK
SD interface	SPI clocks and data lines are active	OK
Card detect	Card detect switch toggles on insertion/removal	OK
Release circuit	MOSFET switches ON when gate is driven high	-
Power management	Microphone and SD power rails switch OFF via GPIO	OK
Mic level shifter	Microphone level shifter switch ON/OFF via GPIO	OK
Mic THESEL level shifter	MIC_THSEL level shifter switch ON/OFF via GPIO	-
Battery monitor	Comparator output toggles at threshold voltage	OK

Table 43: MIC/SD Board Hardware Specifications Validation

The requirements are satisfied after the pin modification is explained in Section 10.2.1.

10.3 Requirements validation

A list of requirements was defined in Section 5.1. Each requirement was tested to verify compliance. The results are compiled in this chapter.

10.3.1 Functional requirements

1. **Acoustic recording :** OK, the system capture audio using T5848 microphone.
2. **Intelligent triggering :** OK, the system utilize Acoustic Activity Detection to wake the processor and initiate saving sound samples.
3. **Continuous recording :** OK the system records until the sound levels fall below the trigger levels.
4. **BLE proximity detection :** OK, The device passively scan for nearby collars and log their informations to log files filtering it using manufacturer data.
5. **Start recording when device nearby :** OK, a proximity detection can start the recording of the audio stream.
6. **Persistent storage :** OK, all audio captures and system logs are stored on a SD card.
7. **Remote release mechanism :** NOK, the system include the GPIO-controlled actuator logic but was not tested. The related hardware is not mounted on the used prototype board .
8. **Wireless monitoring :** OK, the system provides a custom GATT service to monitor the device.

10.3.2 Performance requirements

1. **Audio quality :** OK, the audio is sampled at a frequency of 16 kHz in a 16 bits resolution.
2. **Field autonomy :** NOK, the consumption has been optimize, but the autonomy could not be tested (battery not yet defined). For more information on consumption, please refer to Section 10.4.
3. **Power efficiency :** OK, the system maintain the microphone in a sleep mode and the MCU in deep sleep during periods of silence.
4. **Tail recording :** OK, the system shall continue recording for a defined tail period after the last detected sound.

10.3.3 Interface requirements

1. **BLE GATT service :** OK, the system implements the proprietary Speak No Evil Service.
2. **Debugging interface :** OK, the firmware supports debugging and logging via the SEGGER Real Time Transfer protocol.
3. **Hardware configuration :** OK, the system supports run-time microphone configuration via BLE.

10.3.4 Operational requirements

1. **Automatic fail-safe :** OK, the system automatically halt recording and enter a *Power Saving* mode if the battery is low or th SD card is full.
2. **State persistence :** OK, important data are stored in protected RAM sections.
3. **Error handling :** OK, the system detects and reports hardware issues.
4. **Recoverability :** OK, the collar release command remains functional even after the system has reached a *Disk Full* or *Low Battery* state.

10.4 Global consumption

The Table 44 summarizes the current consumption of the nRF Monkey system across its different operating modes. The values are derived from power profiler measurements on the PCB. The screenshots of the measure are available in the project repository.

Operational State	Average current	Peak current	Description
Idle	56.89 μA	6.50 mA	System sleeping, BLE Advertising periodic spikes.
Idle (BLE connected)	84.33 μA	4.84 mA	BLE connected to mobile app
Recording toggle	1.34 mA	62.73 mA	Transition from idle to recording
Recording	389.03 μA	7.74 mA	Mic in sleep mode, BLE advertising/ scanning in background.
Saving	2.13 mA	40.51 mA	Save audio samples, flushing it to SD card
Recording V1	4.02 mA	77.92 mA	Save audio samples, flushing it to SD card for V1

Table 44: System power consumption by operational state

10.4.1 Comparison with V1

The V1 system consumed 4.02 mA continuously because it was constantly recording and writing to the SD card (see Figure 100). The new system consumes only 2.13 μA when actively saving data. Otherwise, it consumes only 389.03 μA.

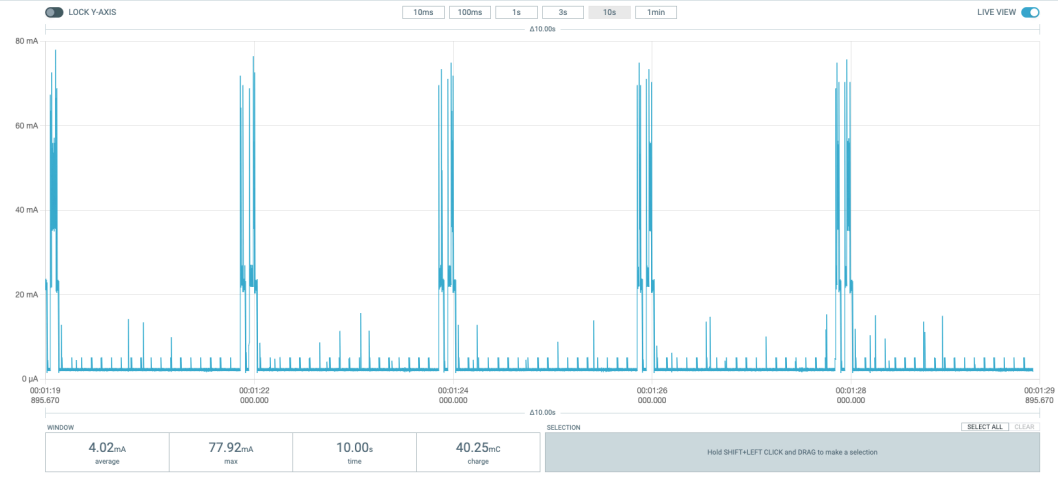


Figure 100: nRF Monkey V1 saving consumption

Figure 101: nRF Monkey V2 saving consumption

Summary of gains

- **During silence :** The system is 90.3% more efficient. As wildlife monitoring mostly involves long, silent periods, this is the main metric for autonomy.
- **During activity :** even when the monkey is screaming and the device is fully active, the system is approximately 50% more efficient than the V1 prototype.

With an active time of approximately 1% (measured over 5 hours of recording -> 20 events of 10 seconds each), it gives a global current :

$$I_{\text{V1}} = 4.02mA \tag{42}$$

$$I_{\text{V2}} = (2.13mA * 0.01) + (0.38903mA * 0.99) = 406.44\mu A \tag{43}$$

In summary, under realistic conditions, V2 consumes approximately 10 times less power than V1.

10.4.2 Battery life

The final battery choice for the V2 is not yet defined, but a battery life estimation was performed based on the V1 battery: the *Tadiran SL-760* (Size AA).

Battery characteristics :

- **Battery voltage :** 3.6 V
- **Battery capacity :** 2200 mAh
- **Low battery detection :** 3.27 V

As plot in the Figure 102, the battery discharge curve is extremely flat, maintaining 3.6 V for most of its life before dropping vertically.

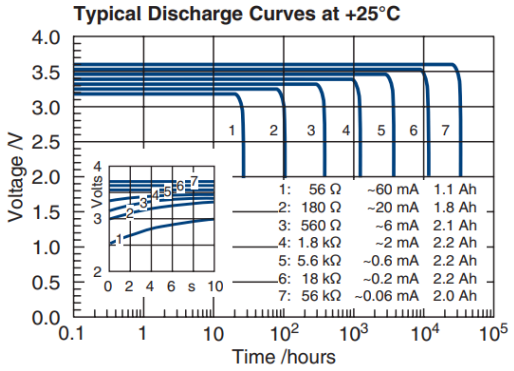


Figure 102: *Tadiran SL-760* discharges curves

Because of this discharge profile, a cutoff at 3.27 V corresponds to the very end of the battery's life. Cutting at 3.27 V allows to use nearly 100% of the nominal capacity.

nRF Monkey discharge characteristics :

- **Recording current (awaiting) :** 389.03 μA
- **Saving current (saving) :** 2.13 mA

Battery life estimation

The Table 45 estimates the autonomy for various recording schedules using the nominal capacity of the SL-760. (2.2 Ah).

Recording Schedule	Average Current	Autonomy (Hours)	Autonomy (Days)
0 min (Standby)	0.39 mA	~5′641 h	~235 days
1 min / h	0.42 mA	~5′238 h	~218 days
5 min / h	0.54 mA	~4′074 h	~170 days
10 min / h	0.68 mA	~3′235 h	~135 days
30 min / h	1.26 mA	~1′746 h	~73 days
60 min (Continuous)	2.13 mA	~1′032 h	~43 days
V1 (Reference)	4.02 mA	~547 h	~22 days

Table 45: Battery life estimation for Tadiran SL-760 (2.2Ah).

The results demonstrate the optimized V2 system a consequent battery life improvement. The V2 continuous recording mode lasts nearly twice as long as the V1 system in continuous mode and more that 200 days with a realistic recording scenario. This efficiency gain suggests that future iterations could transition to significantly smaller batteries to reduce housing size and weight.

10.5 Global system test

A 1 hour field representative test was performed to evaluate the performance of the *T5848* microphone's Acoustic Activity Detection selected settings (see Table 34) and its power consumption. During the test, a track was played that included both specific sounds (vocalizations) and distraction sounds (movement, scratching, eating or collar shock).

10.5.1 Test protocol

Objective:
Validate the real-world performance of the *T5848* AAD settings and measure the global power consumption in a representative acoustic environment.

Setup and equipment:
- **Hardware:** Final embedded system with *T5848* microphone.
- **Measurement:** *Nordic Power Profiler Kit II* (source meter mode: 3.6V) to record consumption over the full hour.
- **Stimulus:** 60 minutes audio track (50 min normal background, 10 min saturated) containing 9 vocalizations and 8 acoustic distractors.

Procedure

ID	Test	Stimulus	Pass criteria
1	*Detection Sensitivity*	9 monkey vocalizations (*ge* 90dB SPL).	100% detection rate. System generates exactly 9 files for these specific events.
2	*Distractor Filtering*	3 movement, 2 shock, 2 scratching, 1 eating sounds.	Analog trigger doesn't detect these distractors. If so, digital filtering successfully rejects mechanical artifacts.
3	*Energy Autonomy*	1-hour continuous monitoring with background noise.	Average current I_{avg} aligns with the theoretical model.

Table 46: Global System Performance Validation

i The 10 minutes saturated background section is critical to verify that the system does not enter a continuous trigger loop.

i The resulting audio files must be manually inspected in *Audacity* to ensure that the tail duration correctly captured the full vocalization without truncation.

10.5.2 Results

10.5.2.1 Audio

The test resulted in 37 recorded files, significantly exceeding the expected count. The summary of the recording files contents is presented in Table 47.

Category	Expected	Actual	Observation / Root Cause
Vocalizations	9	11	100% detection rate. 2 extra files due to fragmentation
Acoustic distractors	0	6	Consistent with the conclusion of the previous test. Shock not detected once
False Positives	0	6	Environmental sounds contain in background track.
Speaker errors	0	13	System glitch causing false triggers.
System overhead	1	1	Last file empty. Normal behavior (file created direct after previous is closed)

Total recording time : 4min54s (8.2%)

Table 47: Analysis of the 1 hour acoustic test results

A glitch was detected in the speaker used for this test. The audio signal remained unchanged for all records (see Figure 103). These 13 files are excluded from the remaining analysis.

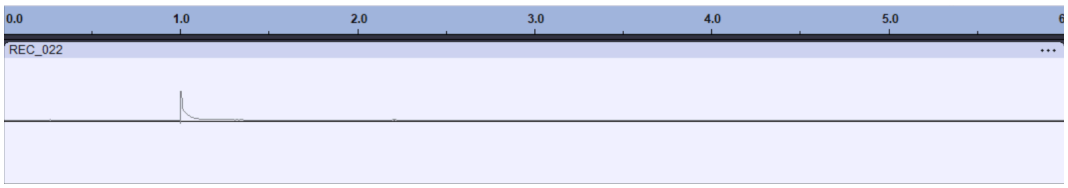


Figure 103: Speaker glitch

10.5.2.2 Current

The global current observed over the hour is 536.85 μA. But the records of the speaker errors are take in count in this value. The following equation approximate the real consumption ignoring these errors.

$$\begin{aligned} I_{\text{real}} &= I_{\text{measured}} - \frac{T_{\text{error}}}{T_{\text{total}}} \cdot (I_{\text{sav}} - I_{\text{rec}}) \\ &= 536.85\mu\text{A} - \frac{130\text{s}}{3600\text{s}} \cdot (2130\mu\text{A} - 380\mu\text{A}) = 481.96\mu\text{A} \end{aligned} \tag{44}$$

The corrected current is 481.96 μA for the system over a hour with a total of 4 minutes and 54 seconds of recording.

10.5.3 Conclusion

Without taking into account the errors induced by the loudspeaker, we obtain the following results:

- **Target accuracy :** The system has detected all the vocalizations, the detection rate is 100%. Due to complex interactions, file fragmentation has occurred during a low-sound-pressure-level vocalization.
- **False positive :** As in the parameter-optimization phase, the system detected false positives. It corresponds to half of the total records. The files have always the minimal size. The recording is started by the AAD A detection. AAD D1 is not triggered by this false positive and the recording is then stopped by the tail time.
- **Audio quality :** The sound quality was not high. This was due to the enclosure, which acted as a resonant chamber. Particular attention should be paid to the design of future enclosures to avoid this issue.
- **Energy efficiency :** After correcting for test-bench artifacts , the estimated real-world consumption is approximately 480 μA. This value aligns with the theoretical model and the battery life estimation. It confirms that the system maintains a viable power profile.

All vocalizations were recorded correctly. The system fulfilled its role. Since acoustic distractors cannot be avoided, this inevitably leads to unexpected detections.

11 Conclusion

11.1 Project summary

This project builds upon the existing data collar, *nRF Monkey*, which continuously records monkey vocalizations for approximately 12 days. A second version is required to address the limitations of the first version, incorporate interaction logging and optimize power consumption.

Integration of the *nRF54L15* SoC and the *T5848* MEMS microphone enables the firmware to utilize an intelligent audio trigger that filters out silence and prioritizes primate vocalizations. Furthermore, a BLE proximity monitoring module has been implemented to map social interactions among monkeys.

11.2 Comparison with initial objectives

The primary objectives for developing an updated version of the *nRF Monkey* have been achieved. The prototype successfully records monkey vocalizations and logs their interactions.

The new hardware has been integrated into the firmware. The firmware now utilizes an Acoustic Activity Detection to record only the targeted sounds.

With BLE, nearby devices can be detected through scanning and subsequently saved.

The system's power consumption has been significantly reduced. Continuous consumption decreased from 4.01 mA to 0.389 mA in recording mode and 2.13 mA in saving mode, resulting in a tenfold reduction in overall power usage under realistic conditions.

11.3 Encountered difficulties

Early hardware challenges, such as defects in soldered microphone and microcontroller boards, resulted in significant project delays. Resolving these issues was necessary to establish functional hardware prior to firmware development.

Development with the *nRF54L15* required addressing the complexities of a new System-on-Chip architecture and its requirements within the *Zephyr* ecosystem.

Zephyr RTOS presented a significant learning curve, especially in understanding *Device-Tree* and *Kconfig* structures necessary for the *nRF54L15* hardware.

Attaining the target low current consumption required a detailed microampere hunt to identify and eliminate current leaks throughout the system.

11.4 Future perspectives

- **Mobile application update :** The iOS application requires an update to incorporate the new V2 features (see Section AG).
- **Field testing and optimization :** Field deployments are necessary to further optimize AAD trigger thresholds and BLE scanning cycles in response to real world environmental noise and primate behavior.
- **Battery downsizing :** Improved autonomy allows replacement of the *Tadiran SL-760* with a lower-capacity battery, significantly reducing the collar's weight and volume. The use of a rechargeable battery should also be evaluated.
- **Final PCB production :** The final version of the electronic board must be manufactured and tested to incorporate hardware modifications identified during prototyping.
- **Housing design :** The mechanical design should prioritize a more compact and waterproof enclosure to minimize impact on natural primate behavior.
- **V2 hardware :** Future iterations should transition to a permanently sealed housing with fixed internal memory and external charging and data ports.
- **Intelligent Base Station:** Develop a central node with proximity detection capabilities to identify nearby monkeys and therefore knowing its geographical position. This station will be able to broadcast grouped commands, allowing for mass parameter updates (e.g., synchronization, recording mode changes) without requiring physical handling or individual connection to each animal.

11.5 Self-assessment

I am satisfied with the project outcomes. Through this work, I have established a robust technical foundation in several essential engineering domains:

- **Zephyr RTOS expertise :** I mastered the complexities of a modern RTOS, including advanced DeviceTree configurations and thread management for the *nRF54L15* architecture.
- **Low-power system design :** I developed specialized expertise in designing low-power systems. Through optimization, I gained familiarity with a variety of energy-saving strategies.
- **BLE and wireless knowledge :** I gained substantial practical experience with Bluetooth Low Energy protocols, particularly regarding the Channel Sounding feature.
- **Project management :** I successfully managed a long-term project, overcoming initial hardware failures and technical challenges to deliver a functional scientific tool.

I found this project rewarding and would have appreciated additional time to further develop the collar and to see the final system.

Glossary

The descriptions in the glossary were written by a generative AI.

AAD – **Acoustic Activity Detection**: A feature in MEMS microphones that identifies the presence of sound activity by monitoring audio energy levels or specific acoustic patterns. Acoustic Activity Detection enables low-power operation by allowing devices to remain in sleep mode until sound is detected, making it useful for voice-triggered systems, environmental monitoring, and smart IoT applications.

Argos – **Advanced Research and Global Observation Satellite**: A satellite-based system used for collecting, transmitting, and processing environmental data from fixed and mobile platforms worldwide. It supports applications such as wildlife tracking, oceanography, and climate research.

ASIC – **Application-Specific Integrated Circuit**: A custom-designed integrated circuit optimized to perform a specific task or set of tasks with high efficiency. ASICs offer superior performance, lower power consumption, and reduced size compared to general-purpose processors, and are widely used in sensors, communication devices, and embedded systems.

ATT – **Attribute Protocol**: A low-level Bluetooth protocol used to exchange data between devices in the form of attributes. ATT underpins the Generic Attribute Profile (GATT) and defines how data is read, written, or notified between Bluetooth Low Energy (BLE) devices.

BLE – **Bluetooth Low Energy**: A wireless communication technology designed for short-range data exchange with minimal power consumption. BLE is widely used in wearable devices, sensors, and IoT applications for transmitting data such as location, temperature, or activity levels.

CS – **Channel Sounding**: A technique used to measure and characterize the properties of a wireless communication channel by transmitting known signals and analyzing the received responses. Channel sounding helps optimize signal processing, improve localization accuracy, and enhance communication reliability in systems such as Wi-Fi, UWB, and 5G.

DMA – **Direct Memory Access**: A feature that allows hardware subsystems to access main system memory independently of the central processing unit (CPU). This speeds up memory operations and reduces CPU overhead during data transfers.

ECS – **Embedded Communicating Systems**

FS – **Full Scale**: The maximum amplitude or value that a system can represent or process. In digital audio, 0 dBFS refers to the maximum level before clipping occurs.

GAP – **Generic Access Profile**: A Bluetooth specification that defines how Bluetooth devices discover each other, establish connections, and manage roles such as broadcaster, observer, central, and peripheral. GAP ensures interoperability and standard behavior across Bluetooth Low Energy (BLE) devices.

***GATT* – Generic Attribute Profile**: A Bluetooth Low Energy (BLE) protocol that defines how data is organized, stored, and transferred between devices. GATT uses a hierarchical structure of services and characteristics to enable efficient communication between BLE peripherals and central devices.

***GLS* – Global Location Sensor**: A lightweight data-logging device that estimates an animal's geographic position using ambient light levels over time. Commonly used in ecological and migratory studies to track animals such as seabirds and fish when GPS tracking is not feasible.

***GNSS* – Global Navigation Satellite System**: A system that provides geolocation and time information to receivers anywhere on Earth using satellite signals. GNSS includes systems like GPS (USA), Galileo (EU), GLONASS (Russia), and BeiDou (China).

***GPS* – Global Positioning System**: A satellite-based navigation system developed and maintained by the United States that provides accurate location, navigation, and timing information to users worldwide. It is one of the main GNSS constellations.

***HEI* – Haute École d'Ingénierie**

***I2S* – Inter-IC Sound**: An electrical serial bus interface standard used for connecting digital audio devices together. It separates clock and data signals, resulting in lower jitter than typical composite digital audio systems.

***IoT* – Internet of Things**: A network of interconnected physical devices embedded with sensors, software, and communication technologies that collect and exchange data over the internet. IoT enables real-time monitoring and automation across various fields, including environmental sensing, agriculture, and wildlife tracking.

***IPS* – Indoor Positioning System**: A technology used to locate and track objects or people inside buildings where GPS signals are weak or unavailable. IPS solutions use methods such as Wi-Fi, Bluetooth, UWB, RFID, or magnetic field mapping to provide accurate indoor location information for navigation, asset tracking, and analytics.

***ISI* – Industrial Systems Institute**

***LoRaWAN* – Long Range Wide Area Network**: A low-power, wide-area networking protocol designed for wireless communication between battery-operated devices over long distances. LoRaWAN is widely used in Internet of Things (IoT) applications, including environmental monitoring and wildlife tracking.

***LPS* – Local Positioning System**: A positioning technology that provides location information within a limited or predefined area, such as a building, factory, or campus. LPS solutions commonly use technologies like UWB, RFID, Bluetooth, or infrared to achieve high-accuracy tracking where GPS is ineffective.

***MAC* – Media Access Control**: A unique identifier assigned to network interfaces for communication on a physical network segment. The MAC address helps distinguish individual devices within networks such as Wi-Fi, Bluetooth, or Ethernet, and is used in data routing and device management.

***MEMS* – Micro-Electro-Mechanical Systems**: Miniaturized mechanical and electromechanical components integrated onto semiconductor chips. MEMS devices include sensors such as accelerometers, gyroscopes, and pressure sensors, and are widely used

in IoT devices, wearables, and navigation systems due to their small size and low power consumption.

MEMS microphone – **Micro-Electro-Mechanical Systems microphone**: A tiny microphone built using MEMS fabrication techniques, consisting of a miniature diaphragm and sensing structure on a silicon chip. MEMS microphones offer low power consumption, high sensitivity, and small size, making them ideal for smartphones, IoT devices, wearables, and acoustic sensing applications.

MITM – **Man-in-the-Middle**: A type of cybersecurity attack where an unauthorized party intercepts and possibly alters communication between two entities without their knowledge. In Bluetooth and network security, MITM protection mechanisms are implemented to ensure secure data exchange and authentication.

PBR – **Phase-Based Ranging**: A distance measurement technique that calculates the range between devices by comparing the phase difference of continuous radio signals. PBR provides high accuracy and is commonly used in Ultra-Wideband (UWB) and other radio localization systems for precise positioning.

PCM – **Pulse Code Modulation**: A digital representation of an analog signal where the magnitude of the signal is sampled regularly at uniform intervals, then quantized to a series of symbols in a digital code.

PTT – **Platform Transmitter Terminal**: A small transmitter device used in the Argos satellite system to send data such as location, temperature, or movement from remote sensors or wildlife tags to satellites for environmental monitoring and research.

RFID – **Radio-Frequency Identification**: A technology that uses electromagnetic fields to automatically identify and track tags attached to objects or animals. RFID systems consist of a tag, reader, and antenna, and are commonly used for inventory management, access control, and wildlife monitoring.

RSSI – **Received Signal Strength Indicator**: A measurement of the power level received by a radio antenna from a transmitting device. RSSI is commonly used in wireless communication systems, such as Wi-Fi, LoRaWAN, and Bluetooth, to estimate signal quality and connection reliability.

RTT – **Round-Trip Time**: A measurement of the time it takes for a signal to travel from a transmitter to a receiver and back again. RTT is commonly used in communication and localization systems, such as Wi-Fi and UWB, to estimate distance and evaluate network performance.

SNES – **Speak No Evil Service**: A custom primary BLE service using a 128-bit UUID, acting as the main command-and-control interface for the collar project.

SoC – **System-on-Chip**: An integrated circuit that integrates all or most components of a computer or other electronic system on a single chip, including a central processing unit, memory, and input/output ports.

SPI – **Serial Peripheral Interface**: A synchronous serial communication interface used for short-distance communication, primarily in embedded systems. It uses a master-slave architecture with four wires: SCLK, MOSI, MISO, and SS.

***SPL* – Sound Pressure Level**: A logarithmic measure of the effective pressure of a sound relative to a reference value, typically expressed in decibels (dB) to quantify loudness.

***SWD* – Serial Wire Debug**: A 2-pin (SWDIO/SWCLK) electrical alternative to JTAG for programming and debugging ARM-based microcontrollers. It offers the same debug functionality as JTAG, such as memory access and breakpoints, but uses fewer pins to save PCB space.

***TDoA* – Time Difference of Arrival**: A technique used to determine the position of a signal-emitting device by measuring the difference in arrival times of the signal at multiple receivers. TDoA is widely used in GPS, LoRaWAN, and other radio-based localization systems for accurate positioning.

***ToF* – Time of Flight**: A measurement technique that determines distance by calculating the time it takes for a signal, such as a radio wave or light pulse, to travel from a transmitter to a receiver. ToF is used in positioning systems, LiDAR, and proximity sensors for accurate distance estimation.

***UUID* – Universally Unique Identifier**: A 128-bit identifier used to uniquely distinguish information or resources across systems. In Bluetooth Low Energy (BLE), UUIDs are used to identify services and characteristics within the GATT architecture, ensuring interoperability between devices and applications.

***UWB* – Ultra-Wideband**: A short-range radio communication technology that uses very low energy pulses across a wide frequency spectrum to provide highly accurate distance and location measurements. UWB is commonly used in real-time location systems (RTLS), indoor positioning, and proximity sensing applications.

***VHF* – Very High Frequency**: A radio frequency range from 30 to 300 MHz used for communication and tracking applications. In wildlife research, VHF transmitters are commonly used in telemetry to locate and monitor animals using handheld or automated radio receivers.

***Wi-Fi* – Wireless Fidelity**: A wireless networking technology that enables devices to connect to the internet or communicate with each other using radio waves. Wi-Fi is commonly used for data transmission in IoT systems, remote sensing, and location-based tracking applications.

Bibliography

[1] arey, "DrinkDark/Master_PI_MobilSens." Accessed: Jan. 26, 2026. [Online]. Available: https://github.com/DrinkDark/Master_PI_MobilSens

[2] arey, "DrinkDark/Master_PA_Data-collar." Accessed: Jan. 26, 2026. [Online]. Available: https://github.com/DrinkDark/Master_PA_Data-collar

[3] "The Art of Designing Embedded Systems." Accessed: Jan. 22, 2026. [Online]. Available: http://www.sciencedirect.com:5070/book/monograph/9780750686440/the-art-of-designing-embedded-systems

[4] E. White, *Making Embedded Systems: Design Patterns for Great Software*. "O'Reilly Media, Inc.", 2024.

[5] "TS-Market: Miniature digital audio recorders Edic-mini Tiny." Accessed: Jan. 22, 2026. [Online]. Available: https://ts-market.com/products/series/234/

[6] C. Couchoux, M. Aubert, D. Garant, and D. Réale, "Spying on small wildlife sounds using affordable collar-mounted miniature microphones: An innovative method to record individual daylong vocalisations in chipmunks," *Scientific reports*, vol. 5, p. 10118, May 2015, doi: 10.1038/srep10118.

[7] "Datasheets/MicroMoth_Datasheet/MicroMoth_Datasheet.pdf at main · OpenAcousticDevices/Datasheets." Accessed: Jan. 22, 2026. [Online]. Available: https://github.com/OpenAcousticDevices/Datasheets/blob/main/MicroMoth_Datasheet/MicroMoth_Datasheet.pdf

[8] A. P. Hill, P. Prince, J. L. Snaddon, C. P. Doncaster, and A. Rogers, "AudioMoth: A low-cost acoustic device for monitoring biodiversity and the environment," *HardwareX*, vol. 6, p. e73, Oct. 2019, doi: 10.1016/j.ohx.2019.e00073.

[9] F. Fahy and D. Thompson, *Fundamentals of Sound and Vibration*. CRC Press, 2015.

[10] "Handbook of Silicon Based MEMS Materials and Technologies." Accessed: Nov. 19, 2025. [Online]. Available: https://www.sciencedirect.com/book/edited-volume/9780128177860/handbook-of-silicon-based-mems-materials-and-technologies

[11] R. Zekavat and R. M. Buehrer, *Handbook of Position Location: Theory, Practice, and Advances*. John Wiley & Sons, 2019.

[12] "(PDF) A REVIEW OF GLOBAL NAVIGATION SATELLITE SYSTEMS (GNSS) AND ITS APPLICATIONS," *ResearchGate*, Accessed: May 22, 2025. [Online]. Available: https://www.researchgate.net/publication/357579523_A_REVIEW_OF_GLOBAL_NAVIGATION_SATELLITE_SYSTEMS_GNSS_AND_ITS_APPLICATIONS

[13] CLS, "Argos User's Manual." [Online]. Available: https://www.argos-system.org/wp-content/uploads/2023/01/CLS-Argos-System-User-Manual.pdf

[14] V. Afanasyev, "A miniature daylight level and activity data recorder for tracking animals over long periods."

[15] "Elephant Seals." Accessed: Oct. 24, 2025. [Online]. Available: https://publishing.cdlib.org/ucpressebooks/view?docId=ft7b69p131&chunk.id=d0e20466&toc.depth=1&toc.id=d0e20466&brand=eschol

[16] "An Overview of LoRa Localization Technologies," *Computers, Materials and Continua*, vol. 82, no. 2, pp. 1645–1680, Feb. 2025, doi: 10.32604/cmc.2024.059746.

[17] G. Y. Ha, S. B. Seo, H. S. Oh, and W. S. Jeon, "LoRa ToA-Based Localization Using Fingerprint Method," in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, Oct. 2019, pp. 349–353. doi: 10.1109/ICTC46691.2019.8939702.

[18] S. Larivière, "Millspaugh, J. J., and J. M. Marzluff (eds.). 2001. Radio Tracking and Animal Populations. Academic Press, San Diego, California, 474 pp. ISBN 0-12-497781-2, price (hardcover), $69.95," *Journal of Mammalogy*, vol. 84, no. 1, pp. 326–327, Feb. 2003, doi: 10.1644/1545-1542(2003)084<0326:R>2.0.CO;2.

[19] "Tracking Animal Location and Activity with an Automated Radio Telemetry System in a Tropical Rainforest | OUP Journals & Magazine | IEEE Xplore." Accessed: Oct. 24, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/8130806

[20] S.-H. Kim, D.-H. Kim, and H.-D. Park, "Animal Situation Tracking Service Using RFID, GPS, and Sensors," in *2010 Second International Conference on Computer and Network Technology*, Apr. 2010, pp. 153–156. doi: 10.1109/ICCNT.2010.40.

[21] P. Gogendeau *et al.*, "Dead-Reckoning Configurations Analysis for Marine Turtle Context in a Controlled Environment," *IEEE Sensors Journal*, vol. 22, no. 12, pp. 12298–12306, June 2022, doi: 10.1109/JSEN.2022.3170414.

[22] Y. Wang, X. Yang, Y. Zhao, Y. Liu, and L. Cuthbert, "Bluetooth positioning using RSSI and triangulation methods," in *2013 IEEE 10th Consumer Communications and Networking Conference (CCNC)*, Jan. 2013, pp. 837–842. doi: 10.1109/CCNC.2013.6488558.

[23] R. Joseph and S. B. Sasi, "Indoor Positioning Using WiFi Fingerprint," in *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET)*, Dec. 2018, pp. 1–3. doi: 10.1109/ICCSDET.2018.8821184.

[24] Y. Liu and Y. Bao, "Real-time remote measurement of distance using ultra-wideband (UWB) sensors," *Automation in Construction*, vol. 150, p. 104849, June 2023, doi: 10.1016/j.autcon.2023.104849.

[25] "Reference circuitry." Accessed: Jan. 20, 2026. [Online]. Available: https://docs.nordicsemi.com/bundle/ps_nrf54L15/page/chapters/ref_circuitry.html

[26] M. U. b. Aftab, *Building Bluetooth Low Energy Systems*. Packt Publishing Ltd, 2017.

[27] G. Koulouras, S. Katsoulis, and F. Zantalis, "Evolution of Bluetooth Technology: BLE in the IoT Ecosystem," *Sensors (Basel, Switzerland)*, vol. 25, no. 4, p. 996, Feb. 2025, doi: 10.3390/s25040996.

[28] A. Thaljaoui, T. Val, N. Nasri, and D. Brulin, "BLE localization using RSSI measurements and iRingLA," in *2015 IEEE International Conference on Industrial Technology (ICIT)*, Mar. 2015, pp. 2178–2183. doi: 10.1109/ICIT.2015.7125418.

[29] R. Ramirez, C.-Y. Huang, C.-A. Liao, P.-T. Lin, H.-W. Lin, and S.-H. Liang, "A Practice of BLE RSSI Measurement for Indoor Positioning," *Sensors*, vol. 21, no. 15, p. 5181, Jan. 2021, doi: 10.3390/s21155181.

[30] J. Wieme *et al.*, "Performance Evaluation of Bluetooth Channel Sounding on Commercial Hardware," *IEEE Access*, vol. 13, pp. 180862–180876, 2025, doi: 10.1109/ACCESS.2025.3623242.

[31] "Devicetree." Accessed: Jan. 11, 2026. [Online]. Available: https://academy.nordicsemi.com/courses/nrf-connect-sdk-fundamentals/lessons/lesson-2-reading-buttons-and-controlling-leds/topic/devicetree/

[32] "Partition Manager." Accessed: Jan. 11, 2026. [Online]. Available: https://docs.nordicsemi.com/bundle/ncs-latest/page/nrf/scripts/partition_manager/partition_manager.html

[33] "SPI clock-frequency not allowed to go to 125kHz in - Nordic Q&A - Nordic DevZone - Nordic DevZone." Accessed: Jan. 29, 2026. [Online]. Available: https://devzone.nordicsemi.com/f/nordic-q-a/118129/spi-clock-frequency-not-allowed-to-go-to-125khz-in

[34] "IRNAS/irnas-t5838-driver." Accessed: Jan. 16, 2026. [Online]. Available: https://github.com/IRNAS/irnas-t5838-driver

[35] "Assigned Numbers."

[36] "Bluetooth Core Specification Advertise." [Online]. Available: https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-62/out/en/host-controller-interface/host-controller-interface-functional-specification.html#UUID-d33351b8-bd92-76f5-47b7-2051fc6e7379

[37] "Online Power Profiler for Bluetooth LE - opp - Online Power Profiler - Nordic DevZone." Accessed: Nov. 13, 2025. [Online]. Available: https://devzone.nordicsemi.com/power/w/opp/2/online-power-profiler-for-bluetooth-le

[38] "Bluetooth Core Specification Scan." [Online]. Available: https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-62/out/en/host-controller-interface/host-controller-interface-functional-specification.html#UUID-28d28698-0fd7-d273-2e31-7a6731617c77

[39] "GPIO — General purpose input/output." Accessed: Jan. 25, 2026. [Online]. Available: https://docs.nordicsemi.com/bundle/ps_nrf54L15/page/gpio.html

[40] "nRF54L15 SPI corruption on GPIO2/P2 when RTT is disconnected - Nordic Q&A - Nordic DevZone - Nordic DevZone." Accessed: Jan. 15, 2026. [Online]. Available: https://devzone.nordicsemi.com/f/nordic-q-a/126270/nrf54l15-spi-corruption-on-gpio2-p2-when-rtt-is-disconnected

[41] A. Aljaani, “Essential pin planning guidelines for the nRF54L Series - Blogs - Nordic Blog - Nordic DevZone.” Accessed: Jan. 25, 2026. [Online]. Available: https://devzone.nordicsemi.com/nordic/nordic-blog/b/blog/posts/essential-pin-planning-guidelines-for-the-nrf54l-series

[42] “nRF54L Pin Planner | by Helmut Lord.” Accessed: Jan. 31, 2026. [Online]. Available: https://pinplanner.app/

A Appendix

AA Project repository

All the appendices and source code are available in the git repository at the following link:

https://github.com/DrinkDark/Master_TM_DataCollar

AB SoC pin assignment

The Table 48 is a comprehensive table for all microcontroller pins and power nets as identified in the schematics. The unused pins are not listed in this table.

Old SoC pin	New SoC pin	Net	Description
ANT	ANT	ANT	2.4 GHz antenna signal
DCC	DCC	DCC	DC/DC converter output to L4
DECA	DECA	DECA	Decoupling for analog circuits
DECD	DECD	DECD	Decoupling for digital core
DECRF	DECRF	DECRF	Decoupling for RF radio circuits
P1.00 / XL1	P1.00 / XL1	P1.00/XL1	32.768 kHz crystal input
P1.01 / XL2	P1.01 / XL2	P1.01/XL2	32.768 kHz crystal output
P1.02	P1.02	MIC_CLK	I2S serial clock SCK
P1.03	P2.00	MIC_THSEL	Microphone threshold selection
P1.04	P2.02	MIC_THSEL_OE	Threshold select output enable
P1.05	-	SD_CD	SD card detect (Not used in V2)
P1.07	P1.07	BATT_LOW	Low battery alarm (comparator U4B)
P1.08	P1.08	nSDon	SD card power load switch (Q1)
P1.10	P1.10	nMicOn	Microphone power control (SB17)
P1.11	P1.11	MIC_OE	Audio level shifter enable (U1/U2)
P1.12	P1.12	MIC_WAKE	Acoustic wake interrupt (AAD)
P1.13	P1.13	MIC_DATA	I2S serial data input
P1.14	P1.14	MIC_WS	I2S word select
P2.01	P1.03	SD_CLK/LD1	SD Card SPI Clock/LED1
P2.02	P1.09	SD_MOSI	SD card SPI MOSI
P2.03	P2.01	BURN	Release mechanism trigger pulse
P2.04	P1.04	SD_MISO	SD card SPI MISO

P2.06	P1.06	SD_nCS	SD card SPI chip select
RESET	RESET	nRF_RESET	Hardware reset pin
SWDCLK	SWDCLK	SWDCLK	Serial Wire Debug clock
SWDIO	SWDIO	SWDIO	Serial Wire Debug data
VDD	VDD	VDD	System power supply
VDDM	VDDM	VDDM	Main digital/memory power supply
VSS	VSS	VSS	Common ground
VSS_PA	VSS_PA	VSS_PA	Power amplifier ground
XC1	XC1	XC1	32 MHz crystal input
XC2	XC2	XC2	32 MHz crystal output

Table 48: Complete SoC pin assignment with changes in red

AC Work packages

The project is structured into four primary work packages.

WP0: Project management & reporting Focuses on the administrative oversight of the project, including timeline management and the final documentation of all technical findings.

- **Task 0.1 - Management :** Project management, milestone tracking, and coordination.
- **Task 0.2 - Technical report :** Writing the technical thesis report and final documentation.

WP1: Microcontroller & microphone upgrade

- **Task 1.1 - Hardware review :** Analyze nRF54L15 and T5848 datasheets.
- **Task 1.2 - Low-level driver development :** Implement the 1-Wire bit-banging protocol for T5848 configuration and the Zephyr I2S driver for PCM audio capture.
- **Task 1.3 - Triggered recording logic :** Develop the Acoustic Activity Detection interrupt handler to wake the SoC upon sound detection.
- **Task 1.4 - SD card optimization :** Implement the power-gating logic for the SD card load switch and optimize FATFS mount/unmount cycles.
- **Task 1.5 - Peripheral power gating :** Configure Kconfig to disable unused peripherals and optimize GPIO states for sleep mode.
- **Task 1.6 - Baseline power profiling :** Use *nRF Power Profiler II* to measure sleep current and I2S acquisition current independently.

WP2: BLE proximity detection

- **Task 2.1 - Protocol selection :** Evaluate passive scanning versus active connection-based ranging.
- **Task 2.2 - Multi-role logic :** Implement concurrent scanning and advertising threads using the Zephyr Bluetooth stack.
- **Task 2.3 - Filter optimization :** Implement manufacturer data filters to ignore non-HEI BLE devices at the controller level.
- **Task 2.4 - Data logging strategy :** Design a circular buffer system to store peer IDs and RSSI values before flushing to the SD card.
- **Task 2.5 - Cycle testing :** Conduct Range vs. Interval tests to determine the minimum scan window required for reliable detection.
- **Task 2.6 - SNES protocol upgrade :** Upgrade the custom GATT service with control point characteristics for microphone settings update.

WP3: Global power analysis

- **Task 3.1 - Consumption profile :** Determine the power profile of the different modes.
- **Task 3.2 - Continuous current logging :** Run long-term power profiling on the integrated prototype to capture transient peaks during SD card writes.
- **Task 3.3 - Battery life projection :** Calculate the total charge consumption (mC) to determine autonomy with the selected battery capacity.
- **Task 3.4 - Efficiency Validation :** Compare final results against the initial project requirements to verify the low-power targets are met.

AD Buffer size optimization

ADA Audio

Action	I_{avg} [mA]	Time [ms]	Charge [mC]	Interval [ms]	Eff. [mC/kB]	Mean current [mA]
File opening	23.29	136.30	3.17	-	-	-
Write 64kB	23.53	193.8	4.56	2000	0.0712	2.28
Write 48.4kB	19.57	111.4	2.18	1500	0.045	1.45
Write 32kB	18.38	80.6	1.48	1000	0.0463	1.48
Write 16.4kB	19.64	56.4	1.11	500	0.0677	2.22
Write 8.2kB	17.27	37.9	0.65	250	0.0793	2.6

Table 49: Audio buffer size test values

ADB BLE

Buffer capacity [devices]	Buffer size [kB]	Time to write [s]	I_{avg} [mA]	Time [ms]	Charge [mC]	Efficiency [mC/kB]	Mean current [uA]
128	1.92	120	2.56	20.32	0.31	2.4019	2.56
256	3.84	240	1.71	25.2	0.41	1.6006	1.71
512	7.68	480	1.35	37.84	0.65	1.2645	1.35
1024	15.36	960	1.12	55.67	1.08	1.0514	1.12
2048	30.72	1920	0.77	80.43	1.48	0.7206	0.77

Max devices per scan: 64 | Scan interval: 60 s

Table 50: BLE buffer size test values

AE Tail timer optimization

Nº	Start [s]	End [s]	Diff [s]
1	0.020	0.464	0.444
2	0.030	1.855	1.825
3	0.098	1.664	1.566
4	4.080	4.695	0.615
5	6.361	7.744	1.383
6	8.830	9.307	0.477
7	10.128	10.649	0.521
8	11.720	12.202	0.482
9	12.797	13.832	1.035
10	15.220	15.768	0.548
11	17.240	17.772	0.532
12	22.030	22.679	0.649
13	26.560	27.175	0.615
14	28.060	28.415	0.355
15	29.750	30.052	0.302
16	31.410	32.121	0.711
17	34.110	34.849	0.739
		Min	**0.302**
		Max	**1.825**
		Mean	**0.753**

Table 51: Vocalization duration analysis (maximal value in bold)

Nº	Start [s]	End [s]	Diff [s]
1	4.695	6.361	1.666
2	7.744	8.830	1.086
3	9.307	10.128	0.821
4	10.649	11.720	1.071
5	12.202	12.797	0.595
6	13.832	15.220	1.388
7	15.768	17.240	1.472
8	17.772	22.030	4.258
9	22.679	26.560	3.881
10	27.175	28.060	0.885
11	28.415	29.750	1.335
12	30.052	31.410	1.358
13	32.121	34.110	1.989
		Min	**0.821**
		Max	**4.258**
		Mean	**1.677**

Table 52: Interval between vocalizations (maximal value in bold)

AF Scanning and advertising parameters optimization

This table contains the best parameter sets for each scanning interval tested with a probability of $P = 0.9$.

Scan interval [s]	Adv interval [s]	Scan window [s]	Current [µA]
10	0.4	1.8	553
20	0.3	1.4	226
30	0.3	1.4	156
40	0.3	1.4	121
50	0.3	1.4	100
60	0.3	1.4	86
70	0.3	1.4	76
80	0.3	1.4	68.5
90	0.3	1.4	62.66
100	0.3	1.4	58
110	0.3	1.4	54.18
120	0.3	1.4	51
130	0.3	1.4	48.3
140	0.3	1.4	46
150	0.3	1.4	44
160	0.3	1.4	42.25
170	0.3	1.4	40.7
180	0.3	1.4	39.33
190	0.3	1.4	38.1
200	0.3	1.4	37
210	0.3	1.4	36
220	0.3	1.4	35.09
230	0.3	1.4	34.26
240	0.3	1.4	33.5
250	0.3	1.4	32.79
260	0.3	1.4	32.15
270	0.3	1.4	31.55
280	0.3	1.4	31
290	0.3	1.4	30.48
300	0.3	1.4	30

Table 53: Best parameter for each scanning interval (with $P = 0.9$)

AG Mobile app modification

Below is a list of the changes that need to be made to the mobile application to meet the new requirements:

1. Add the *SNES_MIC_AAD_A_PARAM* characteristic to allow user to udpate the AAD A settings.

   UUID : 00000207-4865-7673-025A-4845532D534F
   Properties : Notify and read
   Data strucuture :
   - Byte 0 (uint8_t) : Low Pass Filter (range 0x1-0x7)
   - Byte 1 (uint8_t) : Input Threshold (range 0x0-0xF)

2. Add the SNES_MIC_AAD_D1_PARAM characteristic to allow user to update the AAD D1 settings.

   UUID : 00000208-4865-7673-025A-4845532D534F
   Properties : Notify and read
   Data structure :
   - Byte 0 (uint8_t) : Algorithm selection (0:None, 1:Rel, 2:Abs, 3:Both)
   - Byte 1-2 (uint16_t) : Relative floor (range 0x00F-0x7BC)
   - Byte 3-4 (uint16_t) : Relative pulse minimum (range 0x000-0x12C)
   - Byte 5-6 (uint16_t) : Absolute pulse minimum (range 0x000-0xDAC)
   - Byte 7 (uint8_t) : Relative threshold (range 0x24-0xFF)
   - Byte 8-9 (uint16_t) : Absolute threshold (range 0x00F-0x7BC)

The value are send in little endian.

3. Update the main states to add the new *ST_SAVING* :

```
enum main_state {
  ST_INIT         = 0x00,
  ST_WAIT_SD_CARD = 0x01,
  ST_IDLE         = 0x02,
  ST_RECORDING    = 0x03,
  ST_SAVING       = 0x04,
  ST_DISK_FULL    = 0x05,
  ST_LOW_BATT     = 0x06,
  ST_POWER_SAVING = 0x07,
  ST_ERROR        = 0xff
};
```