DTU Autonomous Systems
DTU Department of Electrical and Photonics Engineering

Wi-Fi and Satellite Scanner for Ultra-low Power Geolocation

Master Thesis

Authors:

Magnus Erler - s174820

Christoffer Kolbe Boye-Hansen - s223573

Wi-Fi and Satellite Scanner for Ultra-low Power Geolocation

Master thesis written by:
Magnus Erler, s174820
Christoffer Kolbe Boye-Hansen, s223573

Supervisor:
Martin Nordal Petersen, Associate Professor at the Department of Electrical and Photonics Engineering, Technical University of Denmark - DTU

DTU Department of Electrical and Photonics Engineering
Technical University of Denmark
2800 Kongens Lyngby
Denmark

Thesis period:	22.01.2024 - 22.06.2024
ECTS:	30
Education:	Master of Science
Program:	Autonomous Systems
Notes:	This report is submitted as part of the requirements for the above-mentioned program at the Technical University of Denmark.

Acknowledgements

For both of us, the authors, it has been an enriching time with a very interesting topic. We are grateful for the support that we get and would like to address it.

Thanks to our supervisor, Martin Nordal Petersen, for his guidance, insightful feedback, and continuous support throughout this project. Your expertise, encouragement and availability have been instrumental in completing this thesis.

We also thank Anas Mohamad Al Shalyan for his help with soldering and PCB designing. Your technical assistance was crucial to our research.

Finally, thanks go to all the people behind our interviews, whose insights and perspectives significantly contributed to our understanding of our work's practical applications. We are particularly grateful to Enemærke & Pedersen for inviting us to their site and allowing us to see real-world applications of our research.

Thank you all for your highly valuable contributions.

Abstract

Accurate geolocation of objects, assets, and individuals is increasingly critical across various applications, from logistics and precision agriculture to smart city infrastructure. Traditional geolocation methods often face challenges such as high power consumption, signal interference, and limited coverage. This thesis presents the development, design, implementation, and testing of a geolocation tracker using the LR1110 chip and LoRaWAN technology to address these issues.

We developed a comprehensive ecosystem featuring a custom-made PCB with the LR1110 chip, capable of scanning for Wi-Fi access points and transmitting this data via LoRaWAN to a cloud solver for location calculation. The results are then displayed on a custom-developed web application. Our tracker solution met 13 of 14 essential requirements and 6 of 7 desired features, as outlined in our product requirements. Despite significant achievements, the project did not meet the ultra-low power consumption target, and the GNSS scanning feature was not fully operational. However, these limitations are believed to be hardware-related, and with further development, these challenges can be overcome.

This thesis details the state of the art technologies used, including Wi-Fi and GNSS localisation techniques, communication protocols, and the integration of LoRaWAN with The Things Stack. The design methodologies, including prototyping, empirical data gathering, and iterative PCB design, are discussed. The implementation process, from initial failures to successful integration using Semtech's LoRa Basics Modem software, is also presented. Test and validation results, along with comparisons of LoRaWAN with other communication protocols, are provided.

This work contributes to the understanding and development of ultra-low power, efficient geolocation solutions for IoT applications using the LR1110 chip, and suggests potential improvements and areas for future research to enhance performance and applicability.

Contents

Abbreviations

ABP	**A**ctivation **B**y **P**ersonalization
AP	**A**ccess **P**oint
API	**A**pplication **P**rogramming **I**nterface
AppKey	**App**lication **Key**
AppNonce	**App**lication **Nonce**
AppSKey	**App**lication **S**ession **Key**
BLE	**B**luetooth **L**ow **E**nergy
BOM	**B**ill **O**f **M**aterials
BW	**B**and**w**idth
CR	**C**ode **R**ate
CSS	**C**hirp **S**pread **S**pectrum
DevAddr	**Dev**ice **Addr**ess
DevEUI	**Dev**ice **E**xtended **U**nique **I**dentifier
DevNonce	**Dev**ice **Nonce**
EMC	**E**lectro**m**agnetic **C**ompatibility
FPORT	**F**rame **Port**
FUOTA	**F**irmware **U**pdate **O**ver **T**he **A**ir
GCPW	**G**rounded **Cop**lanar **W**aveguide
GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
GPIO	**G**eneral-**P**urpose **I**nput/**O**utput
GPS	**G**lobal **P**ositioning **S**ystem
GSM	**G**lobal **S**ystem for **M**obile Communication
I^2C	**I**nter-**I**ntegrated **C**ircuit
IoT	**I**nternet **o**f **T**hings
JoinEUI	**Join E**xtended **U**nique **I**dentifier
LED	**L**ight-**E**mitting **D**iode
LNA	**L**ow **N**oise **A**mplifier
LoRa	**Lo**ng **Ra**nge
LoRaWAN	**Lo**ng **Ra**nge **W**ide-**A**rea **N**etwork
LPWAN	**L**ow-**P**ower **W**ide-**A**rea **N**etwork
LTE-M	**L**ong-**T**erm **E**volution **M**achine Type Communication
MAC	**M**edia **A**ccess **C**ontrol
MIC	**M**essage **I**ntegrity **C**heck
MISO	**M**aster **I**n **S**lave **O**ut
MOSI	**M**aster **O**ut **S**lave **I**n
NAV	**Nav**igation Message
NB-IoT	**N**arrow**b**and **I**nternet **o**f **T**hings
NetID	**Net**work **Id**entifier
NwkSKey	**N**et**w**or**k** **S**ession **Key**
ODR	**O**perating **D**ata **R**ate
OTAA	**O**ver-**t**he-**A**ir **A**ctivation

PA	**P**ower **A**mplifier
PCB	**P**rinted **C**ircuit **B**oard
REST	**Re**presentational **S**tate **T**ransfer
RF	**R**adio **F**requency
RSSI	**R**eceived **S**ignal **S**trength **I**ndicator
RX	**R**eceiver
SCK	**S**erial **C**loc**k**
SF	**S**preading **F**actor
SPA	**S**ingle **P**age **A**pplication
SPI	**S**erial **P**eripheral **I**nterface
SS	**S**lave **S**elect
SV	**S**pace **V**ehicle
SWD	**S**erial **W**ire **D**ebug
TCXO	**T**emperature **C**ompensated Crystal **O**scillator
TDOA	**T**ime **D**ifference **O**f **A**rrival
ToA	**T**ime **o**n **A**ir
TTN	**T**he **T**hings **N**etwork
TX	**T**ransmitter
UART	**U**niversal **A**synchronous **R**eceiver / **T**ransmitter
UI	**U**ser **I**nterface
USB	**U**niversal **S**erial **B**us
WSL	**W**indows **S**ubsystem for **L**inux

1 Introduction

Accurate determination of the location of objects, assets, and individuals is crucial across various applications in today's digital and global landscape. The demand for reliable and efficient geolocation solutions continues to rise, from asset tracking in logistics to precision agriculture and smart city infrastructure. Traditional geolocation methods, however, often struggle with challenges such as high power consumption, sensitivity to signal interference, difficulties in diverse environments, and limited coverage.

This thesis aims to explore a geolocation solution designed to address these challenges by leveraging Semtech's capabilities of the LR1110 microchip. It is currently the only product on the market which features **Lo**ng **Ra**nge **W**ide-**A**rea **N**etwork (LoRaWAN) communication, low-power Wi-Fi sniffing, and **G**lobal **N**avigation **S**atellite **S**ystem (GNSS) scanning capabilities all in one chip, reducing the need for additional GNSS and Wi-Fi components. It natively integrates with the LoRa Cloud Modem and Geolocation Services, a simple-to-use and cheap online cloud solution offering **T**ime **D**ifference **O**f **A**rrival (TDOA), GNSS, and Wi-Fi-based location calculations. This dramatically reduces the device's power requirement and enhances asset management while effectively acquiring geolocations for indoor and outdoor settings in urban and open areas.

We were provided with LR1110 development boards for this project, and a significant portion of the testing has been conducted using these boards. The hands-on experience with these development boards has been instrumental in understanding the practical challenges and performance characteristics of the LR1110 chip.

The outline of this report is as follows:

Section 2: State of the art provides an overview of the geolocation and communication technologies used in the project. It starts by exploring the Wi-Fi and GNSS positioning techniques, discusses autonomous and assisted GNSS scanning, and compares Wi-Fi and GNSS positioning effectiveness. Subsequently, it examines the communication protocols employed to interface with the LR1110 chip and the STM32, which manages the overall device operation. This is followed by an in-depth introduction to LoRaWAN, the communication protocol implemented by the LR1110 chip, and an introduction to The Things Stack, the chosen LoRaWAN network for this project. The section concludes with a concise overview of the technologies that integrate The Things Stack with our custom-developed web application, facilitating data forwarding from our geolocation device.

Section 3: Design methods introduces the design methodologies used in developing the geolocation solution. It covers the prototyping process, outlines the setup of our product requirements, and explains the methods used for gathering and organising our empirical data, which were essential for establishing the requirements to develop, implement, test, and evaluate the geolocation solution.

Section 4: Device and application development is the main section of the thesis and presents details on the development of the geolocation device and its applications. This section discusses the formulation of product requirements, which were derived by organising

and analysing empirical data gathered from insights obtained through interviews conducted with users within potential use cases, as well as evaluating current products on the market. The section also presents the iterative process of the **P**rinted **C**ircuit **B**oard (PCB) design, detailing hardware decisions, testing objectives and outcomes from each iteration.
Furthermore, the section explains the development setup, including prerequisites, installation guidelines, and integration with Visual Studio Code, facilitating an efficient development environment. Finally, it describes the implementation of the geolocation solution. It explains the design process of end device application from our initial failed attempt using Semtech's barebone transceiver firmware to our final solution using Semtech's LoRa Basics Modem, which provides high-level **A**pplication **P**rogramming **I**nterface (API) interaction with the LR1110 chip. It finally explains the web application design, detailing the server-side and client-side architecture.

Section 5: Testing and validation of the design describes the testing and validation processes for the geolocation solution. The section outlines the test scenarios and evaluates the system's accuracy and power consumption. It also compares LoRaWAN with other long-range communication protocols to highlight the advantages and disadvantages of the chosen approach.

Section 6: Suggestions for future research discusses potential improvements for the PCB design and web application and identifies areas for further investigation to enhance the performance and applicability of the geolocation solution.

Section 7: Conclusion wraps up the findings of our research, assessing the overall effectiveness of the implemented geolocation solution according to our product requirements. This section critically evaluates the performance of the LR1110 chip, the custom PCB, and the integration with LoRaWAN technology. By reflecting on the challenges encountered and the implications of our results, this section aims to provide a comprehensive overview of the strengths and limitations of our geolocation solution. Additionally, it will discuss the broader applicability of LoRaWAN technology for low-power, efficient geolocation in IoT applications.

We hope this thesis will help the reader understand the complexities and opportunities in developing low-power, efficient geolocation solutions for **I**nternet **o**f **T**hings (IoT) applications using the LR1110 chip.

2 State of the art

This section provides an overview of developing a tracking device with the LR1110 chip. It aims to establish an understanding of the infrastructure and technologies used in our project, including the tools and knowledge to work with the LR1110.

2.1 Geolocating

Geolocating (also known as geopositioning) is the process of determining the geographic position of an object (Secretary, 2018). This can be done using many different methods, each with strengths and weaknesses regarding workload and accuracy. On the LR1110, three methods are implemented: GNSSscanning, Wi-Fi scanning and LoRaWAN network-based localisation. All can be used individually or combined for greater accuracy.

This thesis does not use LoRaWAN network-based localisation. It focuses on Wi-Fi and GNSS geolocation with the use of the LoRa Cloud Modem and Geolocation Services[1] developed by Semtech. A service that provides off-device location solving, reducing the processing power on the tracking device, resulting in lower power consumption.
The following sections introduce the theory behind Wi-Fi and GNSS geolocation.

2.1.1 Wi-Fi geolocating

Wi-Fi is a part of a wireless network protocol based on the technical standard IEEE 802.11[2]. For most people, Wi-Fi is synonymous with wireless internet access. The LR1110 chip does not use Wi-Fi in its conventional role as a data conduit. Instead, it serves as a tool for geolocation through Wi-Fi sniffing, harnessing **M**edia **A**ccess **C**ontrol (MAC) addresses from **A**ccess **P**oint (AP)s to determine the device location.

For this, Wi-Fi APs must broadcast their unique identifier, known as the MAC address. This broadcasting allows the extraction of MAC addresses to determine the location of tracking devices. Optionally also, the **R**eceived **S**ignal **S**trength **I**ndicator (RSSI) of the APs can be used to enhance the accuracy of the geolocation when both the MAC address and RSSI are provided. The level of accuracy using Wi-Fi can range from 5 to 20 m depending upon these factors (Pachuca, 2020).

Fig. 1 shows the principle behind passive Wi-Fi scanning. The LR1110 chip only supports passive Wi-Fi scanning since it is only able to receive but not broadcast Wi-Fi signals. Once MAC addresses are read and identified, the LoRa Cloud's geolocation solver estimates the tracker's location by correlating data indicating the distance between the asset and various Wi-Fi APs, leveraging a global, proprietary database of previously mapped private and public Wi-Fi APs.

[1] LoRa Cloud: https://www.loracloud.com/
[2] Technical standard IEEE 802.11: https://www.ieee802.org/11/

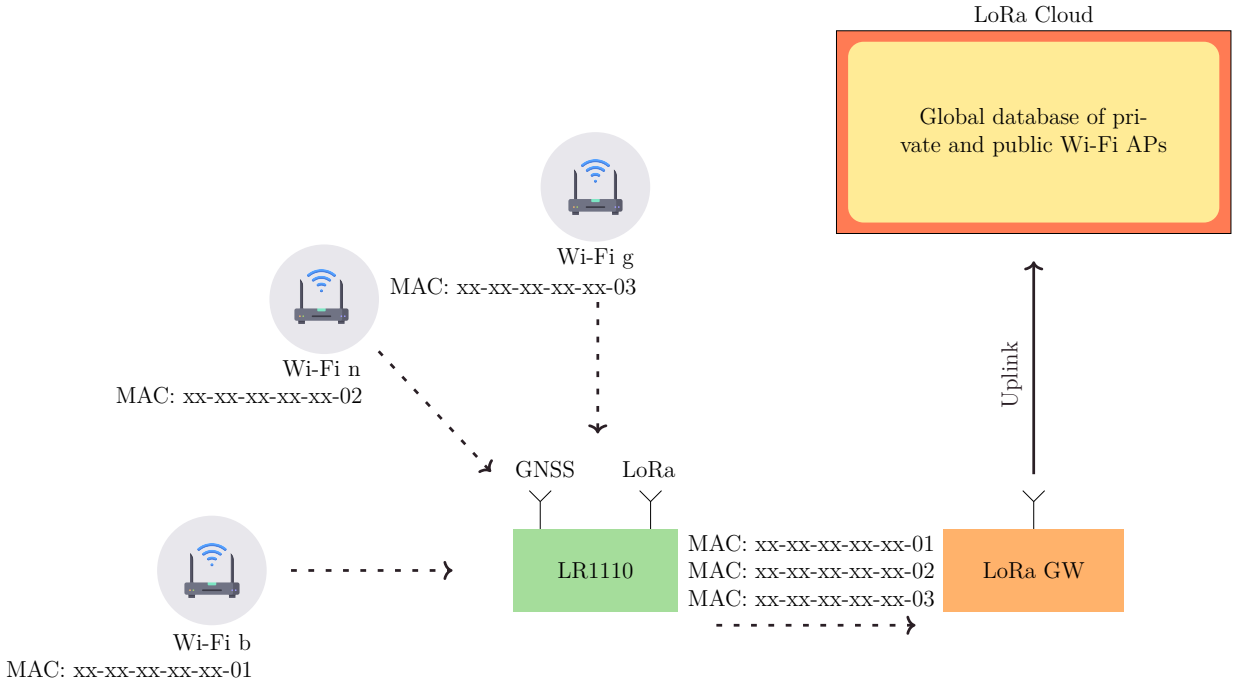


Figure 1: Wi-Fi passive scanning principle. The APs broadcast their MAC address and signal strength to the LR1110 chip. This information is collected and sent wirelessly to a LoRaWAN gateway, which then sends it to LoRa Cloud, where the device's position is estimated.

The accuracy of the location algorithm increases with the number of collected MAC addresses. For example, in a large warehouse or structure with only one or two visible Wi-Fi APs (either inside the building or from nearby private or public networks), the accuracy may be limited to broad geolocating applications that offer general location information.

The payload format is shown in Tab. 1 with and without RSSI.

(a) Wi-Fi results in payload format with MAC addresses only.

0b00	AP1 MAC address	AP2 MAC address	...	APn MAC address
	6 bytes	6 bytes	...	6 bytes

(b) Wi-Fi results in payload format with MAC addresses and RSSI.

0b01	AP1 RSSI	AP1 MAC address	...	APn RSSI	APn MAC address
	1 byte	6 bytes	...	1 byte	6 bytes

Table 1: Payload format of the results of a Wi-Fi scan.

2.1.2 GNSS geolocating

GNSS is a constellation of **S**pace **V**ehicle (SV) orbiting the Earth, providing radio-based positioning, navigation, and timing information to users worldwide. This information enables trilateration, which determines the unknown position coordinates of a point of interest. Its most well-known constellation is the United States-operated **G**lobal **P**ositioning **S**ystem (GPS), while lesser-known constellations are GLONASS from Russia, Galileo from the European Union, BeiDou from China, IRNSS from India and QZSS from Japan. The LR1110 chip can receive signals from both GPS and BeiDou SVs. For a successful 2D[3] location determination, the minimum is 3 SVs for a single constellation scan and 4 SVs for a double constellation scan when using LoRa Cloud's geolocation solver. Because of the low signal strength and disturbing reflections due to, e.g. walls, geolocating using GNSS is predominantly viable in outdoor environments (Novatel; Wikipedia, 2024).

The LR1110 chip includes a fast and low-power GNSS scanner that can be used in two modes:

- **Autonomous mode** operates independently using only the data received from the SVs.
- **Assisted mode** utilises additional information such as time, position, and almanac data to enhance performance.

In addition, the chip allows for indoor/outdoor detection using a quick sample locally to determine if a full GNSS scan would be worthwhile or a Wi-Fi scan would be sufficient, thereby saving power.

2.1.2.1 Autonomous GNSS Scanning

In autonomous scanning mode, the device performs a fast search among all SV's in the selected constellation to find a strong signal. If a strong signal is detected, the device starts searching for weaker signals. If no strong signals are found, the device presumes it is indoors and avoids the more time and energy-consuming search for weak signals. In this case, switching to an alternative positioning mode (e.g., Wi-Fi scanning) is preferable to save power.

2.1.2.2 Assisted GNSS Scanning

Assisted GNSS scanning is employed to enhance the performance of GNSS receivers by providing additional data through external sources. The method is illustrated in Fig. 2. The scan is improved by externally providing three different types of assistance from the network:

1. **Almanac data** informs about the orbital parameters of positioning SVs. This information is valid for up to 120 days but is most accurate approximately 30 days after download. Almanac data can either be demodulated from the SVs or fetched via LoRaWAN from LoRa Cloud Modem and Geolocation Services.

[3] 2D position determination results in latitude and longitude

2. **Precision time:** synchronisation sent from LoRa Cloud Modem and Geolocation Services or directly demodulated from the SV signal.
3. **Position aiding:** is an estimation of the current position of the tracker. Determined by a local 2D solver, given by the user or a former Wi-Fi, LoRa or GNSS scan.

Providing this information, reasonably precise time, location and updated almanac allow for deciding which GNSS SVs are visible and estimating their positions relative to the receiver. Knowing this information accelerates the time-to-first-fix, improves accuracy, and enhances the ability to acquire signals in challenging environments. By eliminating the need to search for non-visible SVs and targeting the receiver towards the available ones, power consumption is decreased and the likelihood of a successful scan is increased; particularly in challenging conditions characterised by weak signals or obstructions. For these reasons, assisted GNSS scanning is the preferred way to carry out a scan. The different assistance ratings are shown in Tab. 2.

Table 2: Rating of assistance for assisted GNSS scanning (Semtech, 2021, p. 8).

	Best assistance	Good assistance	Minimum assistance
Almanac age	< 4 weeks	< 8 weeks	< 15 weeks
Time accuracy	± 10 s	± 30 s	± 120 s
Initial position	50 km	100 km	150 km

In the newest firmware version for the LR1110 chip, the user does not need to configure the GNSS search mode. It is determined by the LR1110 firmware, based on the availability of the assistance information and the actual detection of the visible SVs.

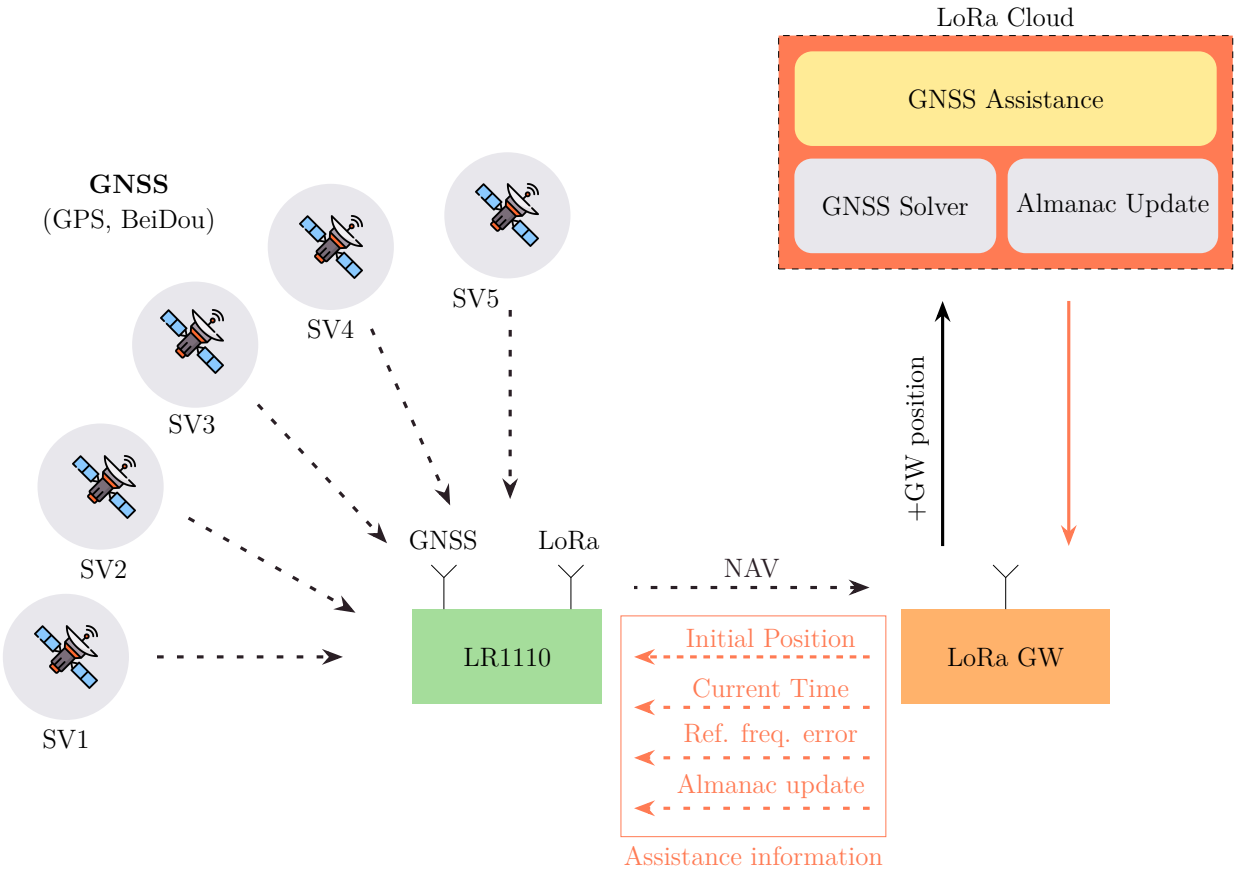


Figure 2: GNSS scanning using the LR1110 chip. SV data are received and formatted into a NAV message forwarded to the GNSS location cloud solver. In assisted mode, the information needed is either provided by the cloud through the LoRaWAN network or directly demodulated from SVs in range.

2.1.2.3 GNSS data format and handling

During the scan, the minimum set of SV information needed for a position calculation is extracted, and a Navigation Message (NAV) is created and forwarded to LoRa Clouds geolocation solver as a GNSS-solve request (see Tab. 3). This can either be for single-frame or multi-frame solving. Single-frame solving uses data from a single captured signal to determine the position, while multi-frame solving combines data from multiple signal captures over time to enhance accuracy.

Table 3: GNSS solve request format.

Data	1 bit	2 bits	5 bits	49 bytes max
	Scan group last NAV	RFU	Scan group token	NAV message

- *Scan group last NAV*; indicates whether this scan is the last of a scan group.
- ***R**eserved for **F**uture **U**se (RFU)* must be set to `0b00`.
- *Scan group token*; a scan group identifier used for multi-frame-solving requests.
- ***Nav**igation Message (NAV)*; the GNSS scan result returned by the LR1110 radio.

The NAV message is an encrypted binary message whose structure is not detailed in the documentation, and its exact contents and design are therefore unknown to us. Looking at the documentation for the LR1110 transceiver firmware (explained in Sec. 4.5) and serial output from a GNSS scan, one can get an idea of the content of the payload. The serial return of GNSS scan return contains SV IDs, their corresponding signal strength, and pseudo ranges; estimated distances between the receiver and SVs, calculated from the time delay of SV signals. Doppler information can also be added to the return.

The API documentation for the *LR1110 Single Frame GNSS Solver* endpoint on LoRa Cloud is as follows in Tab. 4:

Table 4: Endpoint documentation for Single Frame GNSS Solver.

Name	Type	Description
`payload`	HEX	Required. GNSS scan result payload (NAV message)
`gnss_capture_time`	FLOAT	Optional. Capture time estimate, GPST
`gnss_capture_time_accuracy`	FLOAT	Optional. Capture time accuracy, seconds, default: 300
`gnss_assist_position`	FLOAT, FLOAT	Optional. Assistance position WGS84, (latitude, longitude) [deg]
`gnss_assist_altitude`	FLOAT	Optional. Assistance altitude, (above WGS84 reference ellipsoid) [m]
`gnss_use_2D_solver`	BOOL	Optional. Force 2D solve. `gnss_assist_altitude` is mandatory

Thus, the NAV message most likely comprises the SV IDs and their corresponding signal strengths as a minimum. If it is an assisted scan, it also comprises capture time, time accuracy, and assist position. However, the structure remains speculative, as the precise details cannot be confirmed. Semtech states that the information is tailored to an LPWAN network, limiting the data sent.

2.1.3 Comparing Wi-Fi and GNSS geolocating

The primary difference is that Wi-Fi scanning functions well indoors and in urban environments where Wi-Fi APs are commonly found, while GNSS scanning functions well in open terrain where it can receive clear signals from SVs orbiting the Earth.

The differences between the two methods are summarised in Tab. 5.

Table 5: Comparing Wi-Fi and GNSS scanning for positioning (Charlie, 2023).

Parameter	Wi-Fi	GNSS
Accuracy (optimal)	5-15 m	5-10 m
Latency	Seconds	Seconds
Power consumption	Moderate	Very high
Read distance	150 m	Outdoor
Additional device	Wi-Fi AP	-
Use-case	Room approximation	Accurate outdoor

GNSS is precise and relies on a trusted network of SVs; however, it consumes substantially more power compared to Wi-Fi scanning, as indicated in Sec. 5.2, making continuous use in battery-powered geolocation trackers impractical. In contrast, Wi-Fi AP-based geolocation lacks regulatory oversight, and there is no way to ensure that APs remain in their original positions. Relocating a router can significantly impair location accuracy, particularly when only a few APs are accessible. This means the accuracy of Wi-Fi geolocation depends heavily on up-to-date information in the database containing the locations of APs. Additionally, it is important to note that the specific database utilised by LoRa Cloud for Wi-Fi scanning is unknown, as is the frequency of its updates.

2.2 Communication protocols

Communication protocols serve as the backbone for data exchange in electronic systems, offering a range of methods tailored to specific applications. This section explores the critical communication protocols for developing a tracking device with the LR1110 chip.

2.2.1 Serial communication

A broad term, serial communication, refers to data transmission in which bits are sent sequentially, one after the other, over a single communication channel.
Examples of serial communication methods are **I**nter-**I**ntegrated **C**ircuit (I^2C), **U**niversal **A**synchronous **R**eceiver / **T**ransmitter (UART), and **S**erial **P**eripheral **I**nterface (SPI). These methods may differ in protocol, hardware requirements, speed, and application.

2.2.1.1 Universal Asynchronous Receiver/Transmitter (UART)

UART is a full-duplex, asynchronous interface used in microcontrollers, embedded systems, and other electronic devices. It enables transmission via dedicated **T**ransmitter (TX) and **R**eceiver (RX) lines and a shared ground but without a shared clock signal. This allows for greater flexibility in communication between devices operating at different speeds and with varying clock characteristics.
Moreover, its asynchronous operation simplifies hardware implementation, promotes cross-platform compatibility, and renders it cost-effective and straightforward to deploy (Rohde and Schwarz; Siebeneicher, 2024).

2.2.1.2 Serial Peripheral Interface (SPI)

The full-duplex, synchronous SPI is one of the most widely used interfaces between microcontroller and peripheral ICs such as sensors, ADCs, DACs, shift registers, and more. It operates on a master-slave model, using a four-wire configuration, encompassing shared **M**aster **O**ut **S**lave **I**n (MOSI), **M**aster **I**n **S**lave **O**ut (MISO), and **S**erial **C**lo**c**k (SCK) lines, complemented by individual **S**lave **S**elect (SS) lines for each distinct device in a multi-slave environment.

SPI's synchronous operation is characterised by dedicated data lines alongside a clock signal to maintain synchronisation between transmitting and receiving entities. This synchronisation ensures precise timing for data exchange, enhancing communication reliability and efficiency while eliminating the overhead caused by including start and stop bits with each transmitted byte and the risk of data corruption due to mismatched baud rates between communicating devices.

Compared to UART, SPI provides superior speed and supports multiple peripherals. However, it still requires more signal lines (wires) than other communication methods, as each peripheral requires a SS line, and communication remains centralised under the master device, precluding direct communication between peripheral devices (Dhaker, 2018; Grusin; Wikipedia, 2024).

2.2.1.3 Inter-Integrated Circuit (I^2C)

I^2C is a synchronous, multi-master, multi-slave, serial communication bus widely used in embedded systems for connecting low-speed peripherals to microcontrollers. It only uses two bidirectional open-drain lines: a data line and a clock line, simplifying wiring and reducing the number of pins required for communication.

One of the key advantages of I^2C is its ability to support multiple masters and slaves on the same bus, enabling complex communication networks with hierarchical control schemes. Each device on the bus has a unique address, which the master uses to target specific slaves during communication. This addressing mechanism, combined with the simplicity of the two-wire interface, makes I^2C highly versatile and efficient for short-distance communication within a circuit board.

2.2.2 Long Range/Wide-Area-Network (LoRa/LoRaWAN)

Long **Ra**nge (LoRa) is an open standard networking layer that operates on license-free sub-gigahertz radio frequency bands and is particularly well-suited for the IoT and other applications where devices need to transmit small amounts of data over large distances while operating on low power (LoRa Alliance, 2015).
The current world range record stands at 1336 km (Team, 2023). However, the practical range of LoRa technology is subject to significant attenuation due to environmental obstructions, resulting in substantially diminished operational ranges. Since the range of LoRa transmissions is heavily dependent on the receiver's position, indoor deployments typically yield ranges of approximately 500 m. In contrast, outdoor placements atop buildings ex-

tend to around 2 km, and further elevation atop tall antennas may facilitate transmission distances exceeding 10 km (Smartmakers; Guide).

As shown in Fig. 3, other low-power communication technologies have different advantages and disadvantages. LoRa is the only communication protocol implemented in the LR1110 chip and, therefore, the chosen protocol for this project. Considerations on the communication protocol and LoRa versus protocols with similar purposes are presented in Sec. 5.4.

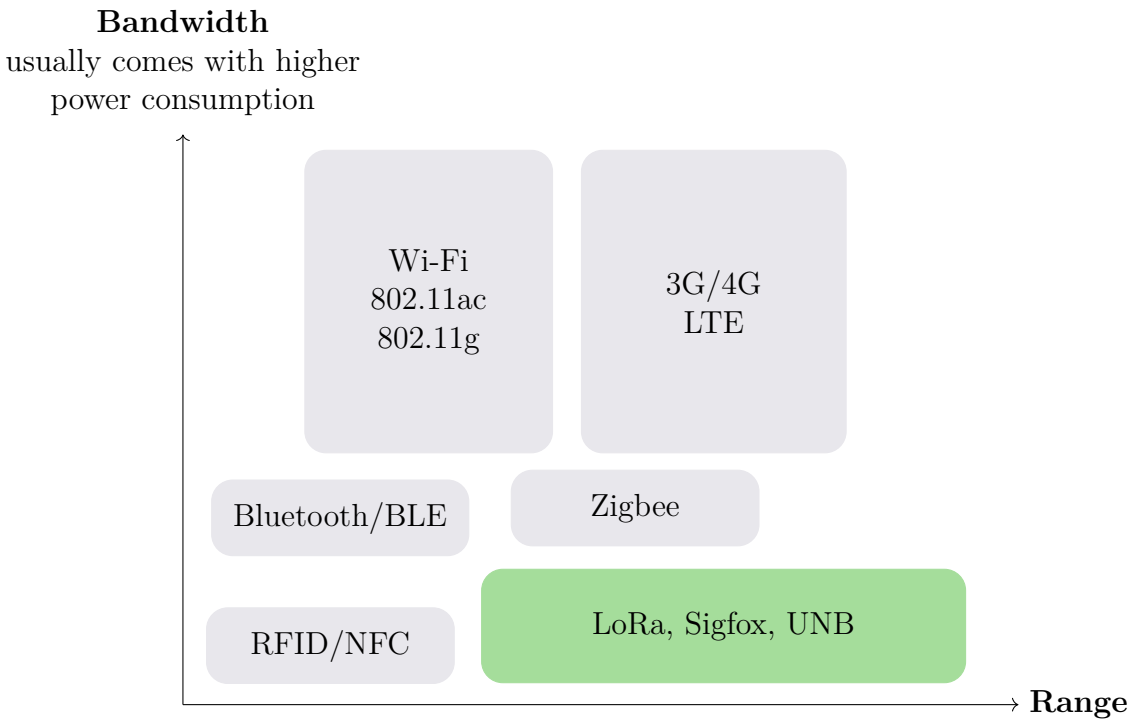


Figure 3: Bandwidth versus the range of different wireless communication technologies (Dasari and Das, 2020, slide: 7).

LoRa refers to the physical layer and Semtech's proprietary spread spectrum modulation technique. In contrast, LoRaWAN refers to the network layer that lies on top and forwards the data to the application while defining the communication protocol and system architecture, enabling devices to communicate with LoRa Gateways and Network Servers.
This hierarchy is shown in Fig. 4.

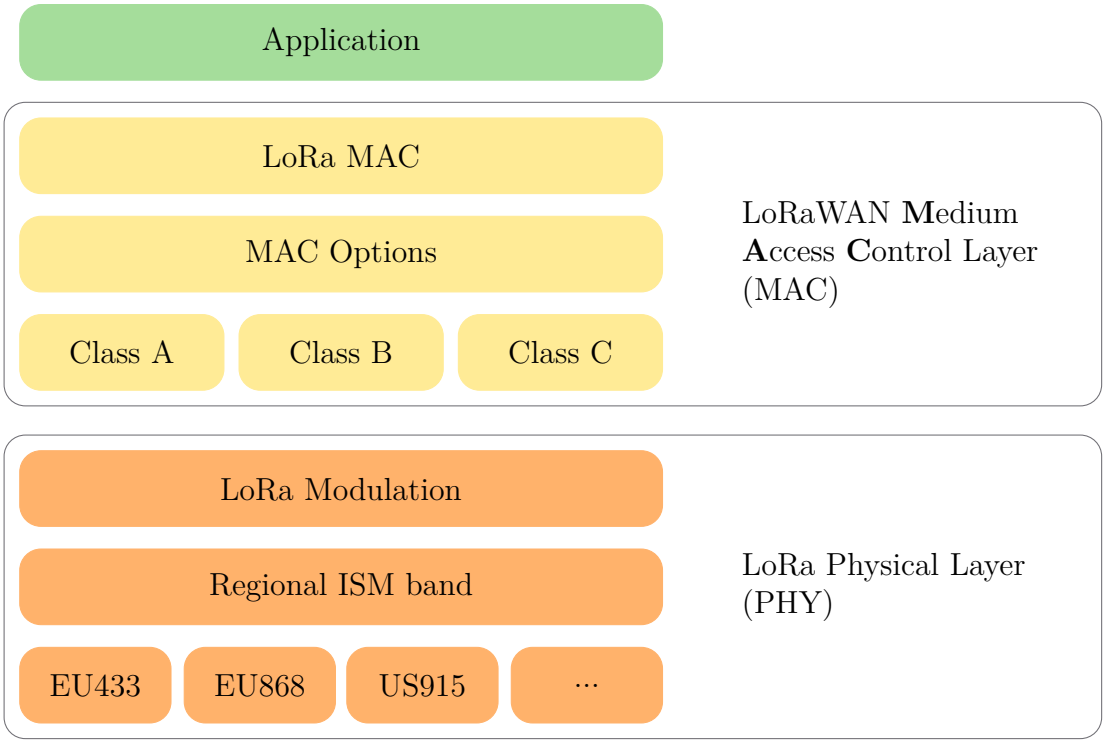
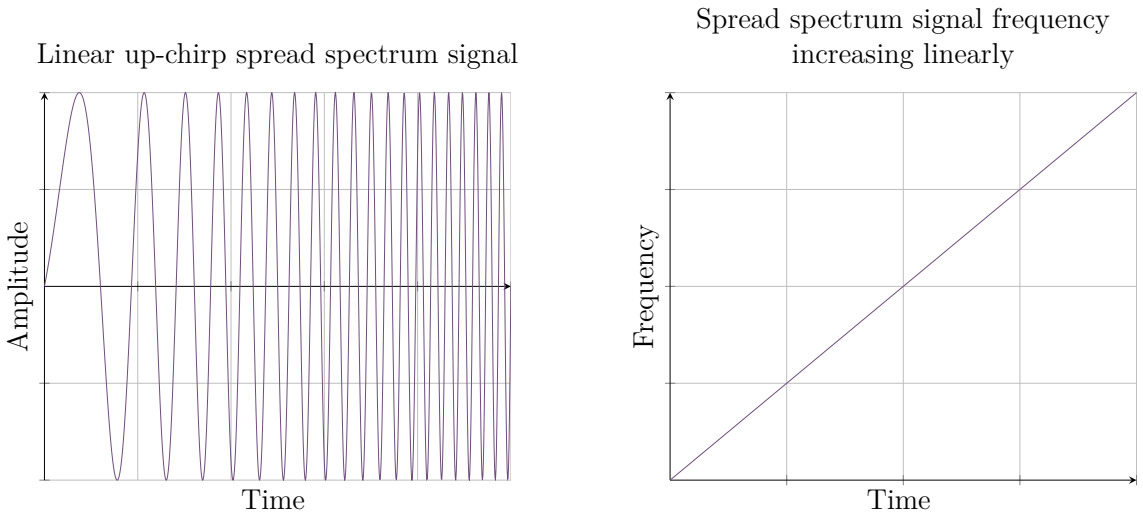


Figure 4: LoRa protocol stack (Haidarzhy, 2023). The MAC layer is the media access control layer that defines the communication protocol and network architecture for managing data transmission between LoRa devices and gateways, while the PHY layer describes the physical layer that handles the modulation and transmission of data using LoRa technology.

2.2.2.1 LoRa technology

LoRa is derived from **C**hirp **S**pread **S**pectrum (CSS) technology and encodes information on radio waves using frequency-modulated chirp pulses. A chirp is a sinusoidal signal whose frequencies vary linearly over time. Fig. 5 sketches a chirp in the time- and frequency domain (Network).



(a) Linear up-chirp signal in time domain representation.

(b) The signal's frequency increases linearly as time passes.

Figure 5: Linear up-chirp signal represented in different domains.

LoRa is the physical layer of the protocol (see Fig. 4) and is responsible for transmitting the raw bit stream over the physical medium. The main variables that influence power usage are data rate and range. These are affected by **B**and**w**idth (BW), **S**preading **F**actor (SF) and the **C**ode **R**ate (CR).

BW is the frequency range the signal sweeps over. Increasing the BW increases the data rate and decreases **T**ime **o**n **A**ir (ToA) but reduces the resistance to noise and interference and, therefore, the range.
Doubling the BW (e.g. from BW125 to BW250) allows to send twice the amount of bytes in a fixed time span. In Europe, the BW is restricted to 125 kHz and 250 kHz.

SF describes how fast the sweep changes over the chosen BW.
The quicker the sweep rate, the more information can be passed in a given time, decreasing the ToA but making the signal more sensitive to noise, which reduces the range. Reducing the SF by one step (e.g. from SF10 to SF9) allows sending twice as many bytes in a fixed time span (see Fig. 6).

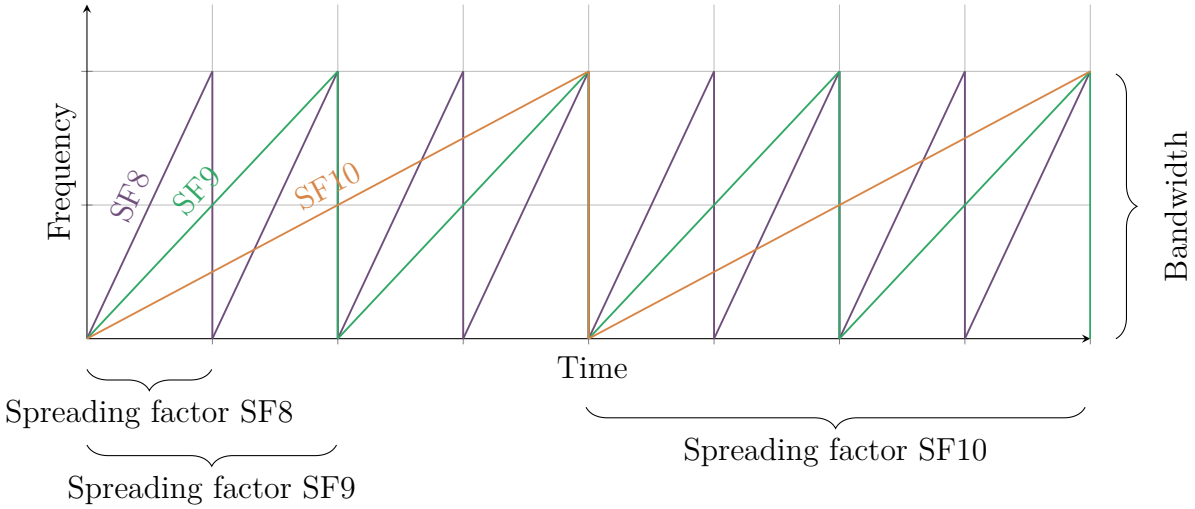


Figure 6: Different spread spectrum signal frequency increasing linearly.

CR determines the number of additional bits added to each payload, which are utilised for error correction. LoRaWAN employs four different code rates: 4/5, 4/6, 5/7 and 4/8. As an example, for a code rate of 5/7, two extra bits are introduced for every 5 bits of information. A higher CR increases the ToA but makes communication more robust and, therefore, also increases the range.

ToA measures how long it takes to transmit a message. This is the primary factor for battery usage of the end device, as a higher ToA results in longer active times for the radio transceivers.

All these parameters can be configured in software, determining the real power of LoRa. Developers can tailor the communication parameters to suit specific requirements, such as prioritising extended transmission distances or minimising power consumption (Bäumker, Garcia, and Woias, 2019).

2.2.2.2 LoRaWAN protocol

LoRaWAN is the MAC protocol sublayer on top of the LoRa physical (PHY) layer (see Fig. 4). It is a set of guidelines for communication between devices and gateways in a LoRa network. It specifies;

- how data should be transmitted. The MAC layer structures data into frames for transmission, interprets received frames, selects frequency channels for communication and manages the duty cycle, ensuring compliance with regional regulations.
- its security. How the network handles authentication, encryption and decryption, ensuring secure communication between devices and the network.
- rules for adaptive data rates. Describing how the data rate should dynamically be adjusted, based on network conditions, to optimise power consumption and communication range.

- rules for scheduling. How transmissions should be coordinated, and communication between end devices and gateways should be managed to avoid collisions and optimise network performance.

2.2.2.3 LoRaWAN Regional parameters and duty cycle

LoRaWAN operates within unlicensed frequency bands, where allocations and regulatory rules vary significantly between countries. Regional parameters describe a region's allowed frequency bands, power limits, and duty cycles. For instance, in Europe (EU433/EU868), the bands span from 433 MHz and 863–870 MHz, whereas South America (AU915/AS923) uses 915–928 MHz, and North America (US915) operates within 865–867 MHz. Similar variations are found in India (IN865) and Asia (AS923).

The duty cycle in LoRaWAN defines the maximum percentage of time a LoRaWAN device can transmit over a given period. This is essential to prevent network congestion and interference among multiple devices operating in the same frequency band, particularly in unlicensed radio bands. An example could be based on the implemented 2-byte long *keep-alive* message, which is described in Tab. 11. Using **T**he **T**hings **N**etwork (TTN)'s airtime calculator[4] with parameters SF9 and BW250, a ToA of 82.4 ms is calculated. If a duty cycle limit of 1% per hour is imposed the device can transmit for $60\,\text{s} \cdot 1\,\text{h} \cdot 1\% = 36\,\text{s}$ each hour. With an ToA of 82.4 ms, it would take $\frac{36\,\text{s}}{82.4\,\text{ms}} = 436$ transmission to reach the hourly limit.

In Europe, the frequency band is divided into sub-bands which each has a duty cycle:

- Sub-band K (863 - 865 MHz): 0.1%
- Sub-band L (865 - 868 MHz): 1%
- Sub-band M (868 - 868.6 MHz): 1%
- Sub-band N (868.7 - 869.2 MHz): 0.1%
- Sub-band P (869.4 - 869.65 MHz): 10% (usually reserved for gateways)
- Sub-band Q (869.7 - 870 MHz): 1%

Unlike Europe, there are no duty cycle limitations imposed in the USA. However, a 400 ms max dwell time per channel exists. This means a device can transmit on a channel for a maximum of 400 ms before it has to switch to another.

Additionally, the LoRaWAN specifies duty cycles for the join frequencies, which are used for **O**ver-**t**he-**A**ir **A**ctivation (OTAA). In most regions, the duty cycle for these frequencies is set to 1%.

Adherence to these regulatory frameworks significantly influences the design and operational strategies of LoRaWAN applications. Developers must account for varying frequency bands during the hardware design and duty cycle limits when planning message frequencies and payload sizes. This contrasts sharply with technologies like NB-IoT, which operate under

[4]LoRaWAN airtime calculator: https://www.thethingsnetwork.org/airtime-calculator

regulated radio bands without duty cycle restrictions, leaving network providers responsible for managing network efficiency and interference.

2.2.2.4 LoRaWAN device classes

LoRaWAN supports different classes of devices. These classes define when an end device can receive a downlink (end devices can always send uplinks at will), allowing for intelligent battery and network usage.

Class A communication is characterised by initiating communication solely by the end device, with downlink messages queued until an uplink is established.
When a connection is established, a Class A end device waits a predetermined amount of time (usually 1 s) and then opens a receive window (RX1). If no downlink is received within a given timeframe, it waits again and opens a secondary receive window (RX2).
Fig. 7 illustrates Class A's different receive window state possibilities. It is the most energy-efficient and must be supported by an end device (Semtech, 2019).

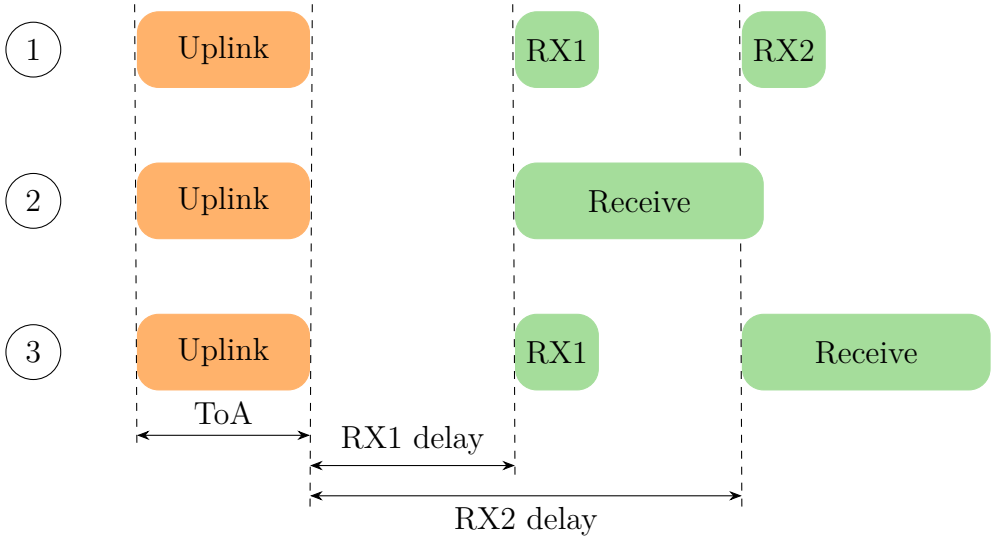


Figure 7: Class A transmissions with different received downlink scenarios.
1. No received downlink, 2. Received downlink in RX1, 3. Received downlink in RX2

Class B (beaconing) builds on top of Class A by, in addition to having two receive windows after an uplink, periodically opening receive windows, known as ping slots, to receive downlink messages. During these ping slots, the end device wakes up after a fixed interval and is ready to receive downlinks. If no downlink is detected, it goes back to sleep. For the end device and the gateway to be time-aligned, a time-synchronised beacon is broadcast periodically by the network via the gateways to align the end device's internal clock with the network. This is illustrated in Fig. 8. Class B has lower latency than Class A but a higher power consumption (Semtech, 2019).

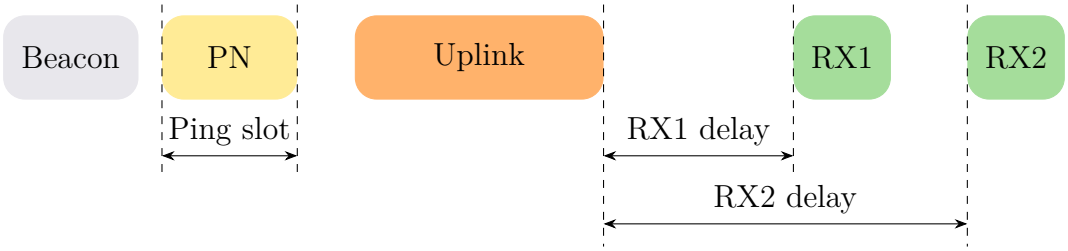


Figure 8: Class B transmission, showing beacon sent out by the gateways, the ping slots, and the two receives also present in Class A.

Class C (continuous) implements the same schedule as Class A with two receive windows but keeps the RX2 window open until the next uplink. It is the class with the lowest latency but highest power consumption, and it is therefore mainly used for mains-powered devices or short periods (Semtech, 2019).

2.2.2.5 LoRaWAN architecture - The Things Stack

To run a LoRaWAN-enabled device, it must be integrated with a network stack, such as Cibicom[5], Helium[6], or TTN[7]. These network stacks are the backbone of LoRaWAN networks, facilitating critical functions such as device management, data routing, and network security. TTN was chosen for several reasons. Firstly, it is a large provider with extensive, well-maintained documentation and a user-friendly setup process. Additionally, there were nearby gateways, making connectivity straightforward. Helium was experimented with, but we could not establish a connection with a LoRaWAN gateway. Moreover, TTN offers native integration with LoRa Cloud, eliminating the need for additional setup and significantly reducing development time.
The following section explains the core components of LoRaWAN architecture, with The Things Stack as the reference point, and its architecture can be seen in Fig. 9.

[5]Cibicom: https://cibicom.dk/
[6]Helium: https://www.helium.com/lorawan
[7]The Things Network: https://www.thethingsnetwork.org/

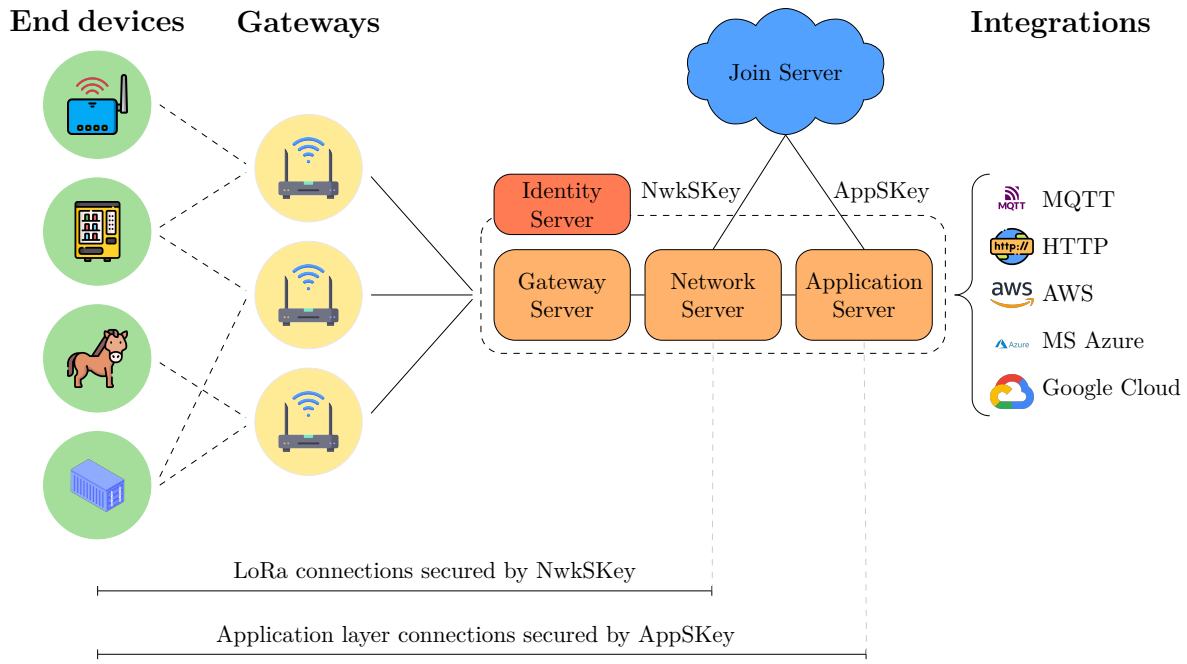


Figure 9: The structure of The Things Stack architecture.
End devices connect wirelessly via LoRa to LoRaWAN gateways, which the Gateway Server manages. The Network Server, central to the network, decrypts and verifies the LoRa header and MIC field using the NwkSKey and controls the data flow between the Gateway and Application Servers. The Application Server decrypts the payload with AppSKey and forwards it to the end application using the chosen integration. The Join Server manages secure device keys and authentication, while the Identity Server tracks entity information.

The **Network Server** is the central component responsible for managing the network's connectivity and data flow and the heart of every LoRaWAN setup. It handles the routing of messages between end devices and the Application Server, ensures data security through encryption and decryption processes, and manages device registrations and network sessions. The Network Server coordinates with other essential components, such as the Gateway and Join Servers, to provide seamless communication and maintain the integrity and efficiency of the LoRaWAN network.

The **Application Server** ensures secure end-to-end data encryption, managing the safe transmission of information between devices and the application layer. It processes the payload from the Network Server by decoding and converting raw data into usable information. The Application Server can integrate with external services through webhooks, MQTT, or other protocols, facilitating seamless data flow between IoT devices and external applications, databases, or cloud services.

The **Gateway Server** manages the gateways in the network and the connections with them. This means coordinating network traffic, optimising routing, providing deduplication, securely forwarding uplink traffic to the Network Server, and scheduling downlink traffic on gateways.

A **Join Server** manages the authentication and activation of devices joining the network. It stores the root keys of the devices and generates session keys for the Network and Application Server.

The **Identity Server** provides entity registries that store standard information, including names, descriptions, and attributes of all significant entities such as applications, end devices, gateways, users, and more. It also manages access control through memberships and API keys.

2.2.2.6 LoRaWAN security

LoRaWAN is designed with security in mind, adhering to state-of-the-art principles, such as using standard, well-tested algorithms and ensuring end-to-end security. Both authentication and encryption are mandatory, with the protocol providing encryption at both the network and application levels, as illustrated in Fig. 9.

This dual-layer encryption allows for implementing so-called *multitenant* shared networks. In such networks, multiple tenants (users and organisations) can share the same network infrastructure without compromising the privacy and security of their data. The network operator handles the data transmission but cannot decrypt the payload.

To ensure that only legitimate devices can join and communicate within the network, devices need to be registered and activated on the LoRaWAN network. Upon activation, a **Net**work **S**ession **Key** (NwkSKey) and **App**lication **S**ession **Key** (AppSKey) are generated, and each LoRa-frame is encrypted with these keys.
The NwkSKey is kept by the Network Server and is used to secure communication between the device and the Network Server, providing message integrity and authenticity. In contrast, the AppSKey is distributed to the Application Server and is used to encrypt and decrypt the payload data between the device and the Application Server, ensuring data confidentiality. This encryption setup is illustrated in Fig. 10.

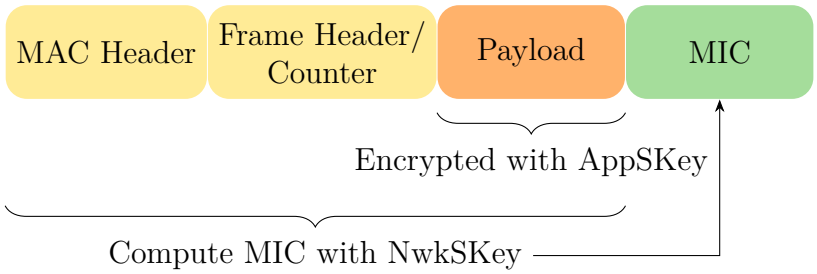


Figure 10: The structure of a standard LoRaWAN frame.

There exist two primary LoRaWAN device activation methods: OTAA and **A**ctivation **B**y **P**ersonalization (ABP). OTAA is the preferred method for device activation because it is more secure than ABP. During OTAA, an end device sends a Join-request (see Fig. 11) to the network in the form of a special non-encrypted message which includes:

- a MAC; a header describing the message type and used LoRaWAN version.

- a **Join** **E**xtended **U**nique **I**dentifier (JoinEUI) (also known as AppEUI); an EUI-64-based number used to identify the Join Server employed in the activation process.
- a **Dev**ice **E**xtended **U**nique **I**dentifier (DevEUI) (also known as ChipEUI); an EUI-64-based number used as the device's unique identifier assigned by the manufacturer.
- a **Dev**ice **Nonce** (DevNonce); an unique, random, 2-byte value generated by the end device, used by the Network Server to prevent replay attacks.
- a **M**essage **I**ntegrity **C**heck (MIC); a value (similar to a checksum) calculated over all the fields in the Join-request message using the **App**lication **Key** (AppKey) to prevent intentional tampering with messages.

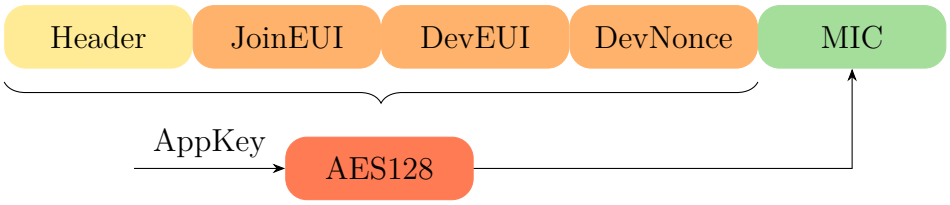


Figure 11: The structure of the LoRaWAN Join-request frame.

If the Join-request gets handled successfully, a NwkSKey and AppSKey are generated. The Network Server responds with an encrypted Join-accept frame (see Fig. 12) consisting once again of a MAC Header, JoinNonce and a MIC calculated over all the fields.
In addition to these, the following fields are added:

- a **Net**work **Id**entifier (NetID); a network identifier of the Network Server issuing the Join-accept.
- a **Dev**ice **Addr**ess (DevAddr); a device address assigned by the Network Server to identify the end device within the current network.
- radio settings; describing downlinks settings and delay between transmission and receive windows.

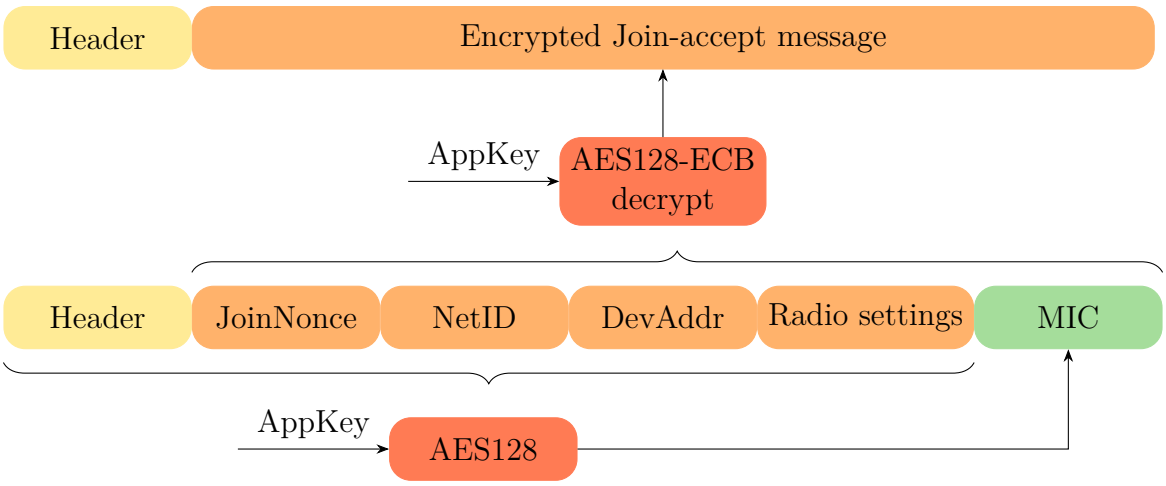


Figure 12: The structure of the LoRaWAN Join-accept frame.
Only the end device, which knows the AppKey, can decrypt the Join-accept frame and verify integrity. NwkSKey and AppSKey are derived using AES128 encryption of AppKey, NetID, JoinNonce, and DevNonce. The first uplink that follows the Join-accept frame confirms the new session keys.

Upon receiving the Join-accept frame, the end device decrypts the Join-accept message and uses the AppKey and **App**lication **Nonce** (AppNonce) to derive the two session keys, NwkSKey and AppSKey.

In ABP, devices are pre-configured with network and application session keys (NwkSKey and AppSKey) before deployment, and these values are also stored on the Network and Application Server, which ties the end device to a determined Network Server.
While ABP simplifies the deployment process, it is less secure than OTAA because the keys are static and do not change. This makes them more vulnerable to interception and replay attacks, and ABP is therefore reserved for devices for which this security level is still acceptable[8].

2.2.3 Integration with The Things Stack

Representational **S**tate **T**ransfer (REST) APIs and webhooks are fundamental concepts in modern web development that facilitate communication between systems and applications.

2.2.3.1 REST API

Software architecture like REST imposes conditions on how an API should work. APIs that align with the REST architectural constraint are called REST APIs, and they follow a request-response model where clients initiate requests to interact with server resources using standard HTTP methods (GET, POST, PUT, DELETE).

[8]Note that this describes the security authentication protocol for LoRaWAN version 1.0.x. The fields and methods are a little different for LoRaWAN version 1.1.x

The server responds to these requests with the requested data or performs the requested actions. The interaction is driven by client requests, and the server does not actively push updates or events to clients without a new request.

2.2.3.2 Webhooks

Webhooks are needed to provide a mechanism for real-time communication and event-driven architecture. Unlike REST APIs, where the client must poll the server for updates, webhooks allow the server to push data to the client when an event occurs.

2.3 The LoRa Edge ecosystem

The LoRa Edge ecosystems shown in Fig. 13 integrate into TTN and extend the capabilities. The LoRaWAN infrastructure and application is implemented by TTN as shown in Fig. 9, and the Application Server integrates into LoRa Cloud Modem Geolocation Services, providing seamless Wi-Fi and GNSS geolocation lookups.

The Join Server is also implemented by Semtech, and their LR1110 chips are pre-provisioned with a hardware root of trust, meaning they have built-in security credentials added during manufacturing. This ensures secure device identity and communication without requiring OEMs and customers to add keys later, enhancing security and simplifying device lifecycle management across diverse applications and networks.
The last part of the platform is Semtech's LoRa Basics Modem[9], an easy-to-use software library that provides a full LoRaWAN protocol standard implementation that natively supports LoRa Cloud services while offering a high level of abstraction without the need to delve into the details of the LoRaWAN protocol.

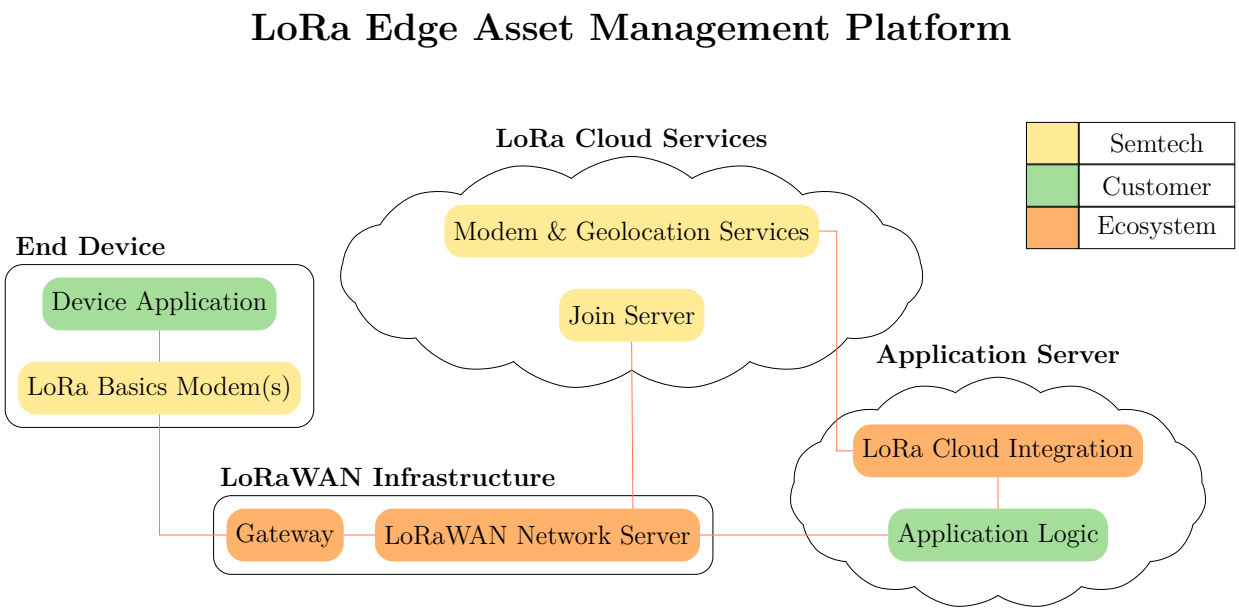


Figure 13: LoRa Edge Asset Management Platform.

[9]Semtech's LoRa Basics Modem: https://github.com/Lora-net/SWL2001

3 Design methods

This section summarises the design methods used in developing our project. It introduces the methods used to derive the product requirements and the prototyping methodology used to explore, create, and evaluate the solution.

3.1 Product requirements

When designing a product, many ways exist to define the product requirements. Sometimes these are given beforehand as specific requirements or constraints set by stakeholders, product owners, regulatory bodies, and or industry standards. Other times, the project is to identify these requirements through iterative processes such as market research, user feedback, co-designing, and prototyping.
For this project, the (only) constraint given beforehand was the development of an ultra-low power geolocation device using the LR1110 chip without quantified requirements. Therefore, a part of this project has been to investigate potential use cases and, based on these, to define a list of measurable requirements for the final product.
The product requirements are divided into requirements and wishes, with parameters the product must meet and desirable but non-essential enhancements. The main methods for investigating potential use cases have been market research of existing products and user interviews.

3.1.1 Interview method

The interviews used to gather information before defining the product requirements have been set up as structured interviews ("kvalitative forskningsinterview") (Thagaard, 2004, p. 87). The interview guide contains the questions to be answered but not the order in which the questions should be asked, and the interviewer has the flexibility to deviate from the questionnaire and ask follow-up questions. The interview guide ensures that the conversation addresses the central topics while allowing for flexibility to explore unexpected but relevant themes that may arise during the interview.

3.1.2 Affinity diagram using the KJ-method

The collected data was arranged into an affinity diagram using the KJ method to organise and derive insights. This method helps organise and visually interpret complex, non-repetitive data so that themes, trends and needs emerge, enabling knowledge to be extracted. The process is sketched in Fig. 14.

The first step of the method is to transfer all the gathered empirical data, including notes, images, and other relevant materials, onto note cards. Ensuring that each note card contains only one thought or concept is crucial.

The second step is shuffling the cards and rearranging them into groups. Some suggest that this step should be done in silence; however, we find it more beneficial to do it verbally. Start by reading each note aloud, discussing the meaning of the data, and placing it on a wall,

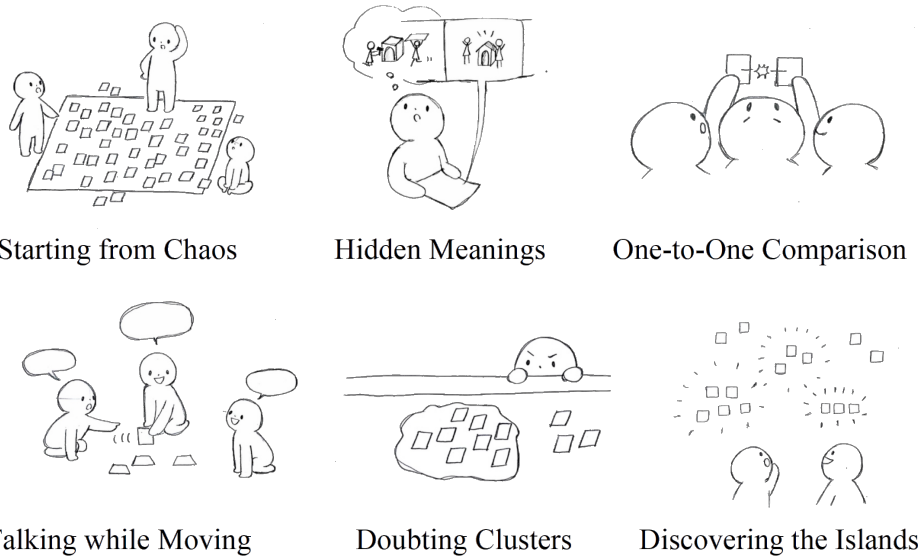


Figure 14: Representation of the workflow when creating an affinity diagram using the KJ-method (Iba, Yoshikawa, and Munakata, 2017).

table or flat surface, slowly grouping the notes into categories. This is a non-linear-non-logical method, meaning that it is an iterative process where preconceived biases must not influence the grouping of notes; instead, intuitive feelings should guide the process over logical analysis. By repeatedly reading and organising the labels, distinct groups will gradually emerge.

In the third step, the categories receive descriptive labels. A well-crafted label articulates the issue that binds the notes together, which is not necessarily the obvious headline.

The last step is reviewing notes and groups. Where appropriate, related groups can be combined into larger supergroups and assigned a label. This creates a hierarchical structure that further organises the data and emphasises overarching themes.

This method of collaboratively organising empirical data without imposing predefined categories and allowing diverse perspectives encourages team collaboration and creative thinking. This fosters a deeper understanding of the data and its implications and helps to uncover hidden connections and insights within the data while achieving team consensus (Holtzblatt and Beyer, 2016; Scupin, 1997).

We have used this approach for our interviews presented in Sec. 4.1.5.

3.2 Prototyping

A method for examining design problems and evaluating solutions is prototyping. This method might revolve around the prototype's inherent attributes, such as how they are utilised in their creation and how they resemble a final product in appearance or functionality (low- or high-fidelity). However, this approach can be misleading as it often overlooks the purpose of the prototype.
The critical part is usually not the specific media or tools employed for their creation but

what designers try to investigate in future artefacts. When trying to represent the weight and scale of a future artefact, a brick might be a valid prototype, but if it is trying to evaluate the form, cardboard might be a better material choice. As Houde and Hill write in their paper *What do Prototypes Prototype?*, they try to "*establish a model that describes any prototype in terms of the artefact being designed, rather than the prototype's incidental attributes.*". They conclude: "*By focusing on the purpose of the prototype - that is, on what it prototypes - we can make better decisions about the kinds of prototypes to build.*" (Houde and Hill, 1997, p. 367).

In their prototype model, they divide prototypes into four categories: role, look and feel, implementation, and integration (see Fig. 15).

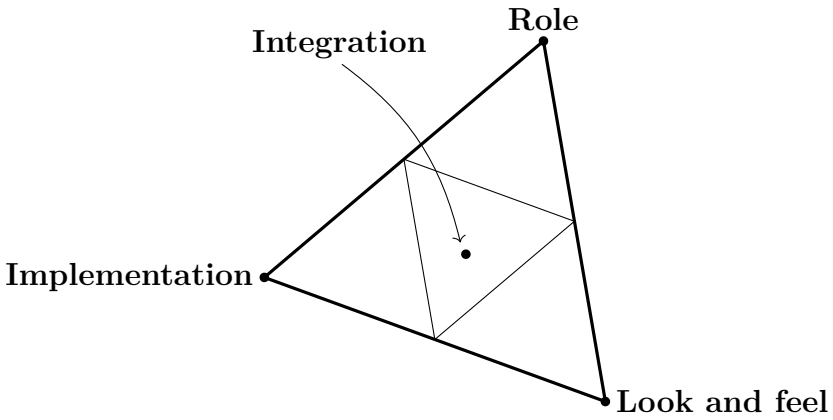


Figure 15: Four principal categories of prototypes upon the model by Houde and Hill (Houde and Hill, 1997, p. 371).

It is important to note that a prototype does not have to be in one category. It can be anywhere on the prototype triangle model and test multiple things simultaneously. The critical part is reflecting on what you want the prototype to test before creating the prototype.

Role prototypes are primarily constructed to explore the benefits an artefact could offer. They focus on describing the functionality users might derive value from without delving much into the artefact's aesthetics or the technical intricacies of its implementation.

Look-and-feel prototypes relate to investigating and showcasing potential experiences with an artefact. They simulate the visual and interactive aspects of interacting with the artefact without necessarily delving into its functional role in the user's life or the technical details of its operation.

Implementation prototypes focus on technical details about how a prospective artefact could function. They uncover methodologies for attaining sufficient specifications for the final product without specifying its aesthetics or intended role for the user.
These prototypes are usually only built as experimental models for the designers and their teams to demonstrate to the organisation the technical feasibility of the artefact and get feedback on performance issues.

Integration prototypes are usually regarded as the last step in the design process. They showcase and test an artefact's complete user experience. Combining the artefact's envisioned design, encompassing its intended role, look and feel, and implementation.

The model has been used as a discussion tool to determine the purpose of our prototypes before designing them. More on this is provided in Sec. 4.

4 Device and application development

The following section describes the design of the geolocation tracker, starting with defining the product requirements. These will serve as the foundation and guidelines for the design of the PCB and the accompanying web application and test scenarios.

After describing the product requirements, we discuss the different versions of the PCB and the specific aspects we aimed to test with each iteration. This includes the initial prototype and the subsequent refinements based on performance evaluations and test results.

Following the PCB development, the progress of programming the end device application is described. This section covers the challenges encountered and the evolution of our approach, particularly the transition from the LR1110's low-level transceiver firmware to LoRa Basics Modem software, which provides high-level API communication with the LR1110 chip.

Lastly, an overview of the web application is provided. This section focuses on the reasons for our web application, the chosen technologies and the structure of the frontend, the backend and the API.

By outlining these elements, we aim to give a comprehensive view of the project's development process, from initial requirements to final implementation.

4.1 Development of product requirements

Developing a product and obtaining a design assignment can occur through various pathways. As described in Sec. 3.1, one may receive an assignment where the requirements have been clearly defined in advance. Alternatively, they may be derived as part of the project. The following sections explain how product requirements were derived.

4.1.1 Potential use cases

The development of product requirements started with considering use cases and researching competitors on the market. Comparing their features allows for determining the requirements for the product to compete effectively and to assess market demand. The main competitors for which product specifications were available are gathered in Tab. 6. Through brainstorming sessions and internet research, the following potential use cases for the product were identified:

Vehicle tracking devices can be installed in cars, trucks, or fleet vehicles to monitor their real-time location, speed, and route. This is useful for fleet management, theft prevention, and optimising logistics.

Asset tracking devices can be attached to valuable assets such as equipment, machinery, or containers to monitor their location and movement. This helps businesses track inventory, prevent loss or theft, and improve asset utilisation.

Personal safety can be carried by individuals, such as elderly people, children, or outdoor enthusiasts, to provide their location in emergencies. This allows caregivers or authorities

to locate them quickly and provide assistance if needed.

Pet tracking devices attached to pet collars enable owners to monitor their pet's location and receive alerts if they move outside a designated area. This helps prevent lost pets and facilitates quick recovery if necessary.

Wildlife geolocation trackers can be attached to wildlife or environmental sensors to track the movement and behaviour of animals, monitor habitat use, and collect data on environmental conditions such as temperature, humidity, or pollution levels.

Package delivery trackers will service companies to follow the location of packages in transit, provide customers with real-time delivery updates, and ensure timely and accurate delivery.

Having identified these potential use cases, we investigate them further by analysing technological feasibility and market demand to pinpoint the most promising applications for our geolocation tracker. This involves engaging with potential users and evaluating the technical requirements of each use case to determine which application best aligns with the LR1110 chip's capabilities. By examining these factors, we can ensure that our product development efforts are focused on creating a solution that effectively meets real-world demands and has strong market potential.

4.1.2 Current products

Two primary technologies dominate the consumer market for object tracking today: GNSS trackers and Bluetooth trackers. GNSS trackers work as standalone devices, utilising SVs to pinpoint their location and use a network like **G**lobal **S**ystem for **M**obile Communication (GSM) or a **L**ow-**P**ower **W**ide-**A**rea **N**etwork (LPWAN) like LoRaWAN or Sigfox to broadcast this information. Bluetooth trackers, on the other hand, like Apple Airtag[10], connect to nearby phones and use the phone's location and data connection to broadcast its location. We focus on exploring use cases that need an independent operating device and benefit from the advanced capabilities and broader reach of GNSS/Wi-Fi tracking. Bluetooth trackers will, therefore, not be considered.

Table 6: List of popular geolocation GNSS trackers on the current market, with their price, size, battery life, and monthly cost.

Product name	Price [DKK]	Size [mm]	Battery	Monthly cost [DKK]
MiniFinder Pico	2950	61 × 41 × 16	40 h (5 min interv.), 20 d (standby)	159
MiniFinder nano	3000	47 × 41 × 16	36 h active, 120 h stand-by	159
Minifinder Extreme	3600	88 × 62 × 34	10 yr (standby)	159
Cobblestone Pro	1500	64 × 64 × 23	4 yr (1 update/day)	Free. 399 for battery replacement
Zmartgear Sigfox	1300	42 × 24 × 17	3 mth (standby)	Free for 3 yr
Odin tracker	2299	58 × 41 × 19	10 yr or 3000 tracks	Free for 10 yr

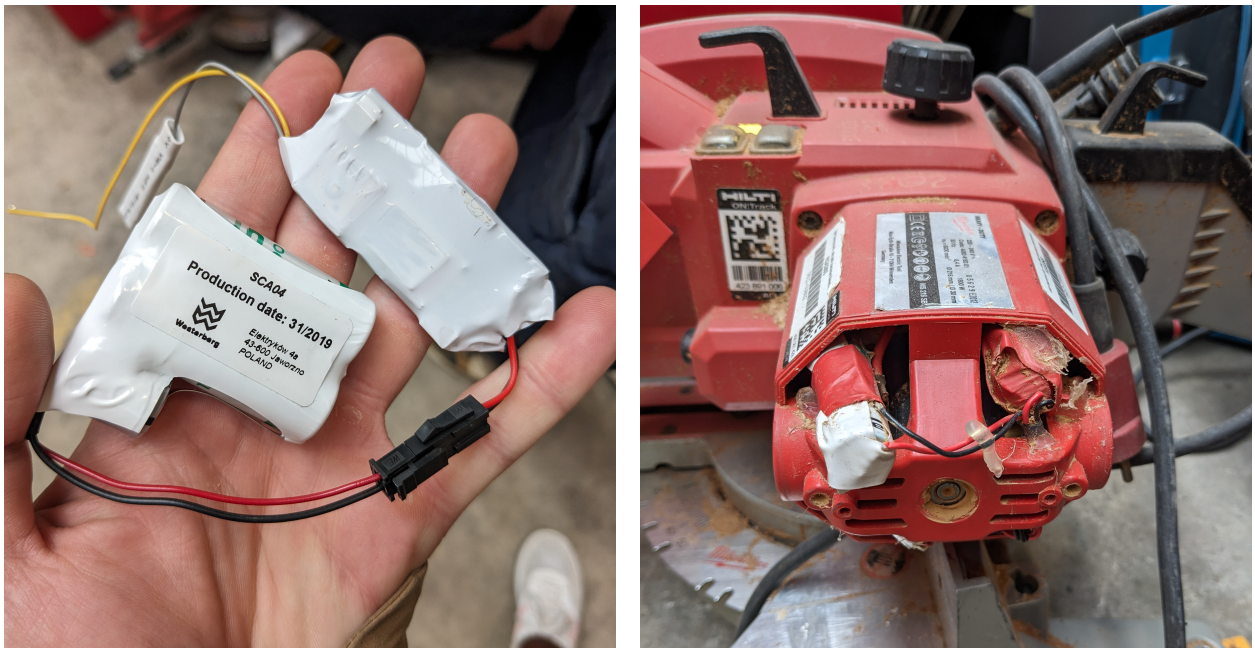
[10]Apple Airtag: https://www.apple.com/airtag/

4.1.3 User interviews

To further investigate the use cases, we sent questionnaires to stakeholders within sectors of all our identified use cases and received feedback from:

- 3 stakeholders with knowledge or experience in vehicle tracking,
- 9 stakeholders with a potential need for asset tracking,
- 41 stakeholders in personal safety; 33 of which being sheltered residences for the mentally handicapped and 8 being care homes for people with dementia,
- 2 people with a potential need for pet tracking,
- 6 stakeholders in wildlife monitoring monitoring, and
- 9 stakeholders in package delivery and logistics.

Further, we conducted 6 telephone interviews with stakeholders within asset tracking, personal safety, pet tracking, and wildlife monitoring, as well as one physical interview regarding asset tracking. Photos from the physical interview are shown in Fig. 16.



(a) Tracker from Unisecure in its whole form. The battery (left) and tracker (right) have antennas sticking out of the top.

(b) The tracker from Unisecure placed inside a mitre saw, a medium-sized power tool.

Figure 16: Pictures during the interview at Enemærke & Petersen showcasing their current geolocation trackers from Unisecure[11] they use for theft protection of their power tools.

[11] Unisecure: https://unisecure.dk/

4.1.4 Data organising

The desk research, the questionnaires and the interviews accumulated a substantial amount of empirical data. The next critical step involved organising these data to uncover underlying themes and derive meaningful insights. We organised the data into an affinity diagram to achieve this, following the KJ-method described in Sec. 3.1.2. Through this process, we aim to condense the perhaps divergent viewpoints into themes that reveal patterns, similarities, and differences among the perspectives of the various stakeholders. This helped us to turn the data into actionable insights that can be used to pinpoint product requirements and steer the direction of the product development.

Splitting the gathered data into individual statements (around 300 labels), printing them, discussing their meaning, sorting them, and organising them into groups took two days. The initial sorting and final grouping with labels is shown in Fig. 17

(a) Initial chaos of over 300 uncategorised notes.

(b) More than 20 categorised groups of notes on the floor because of lack of space on the table.

Figure 17: The making of our affinity diagram, organising our empirical data into sensible knowledge. Red notes denoting statements related to people with dementia, orange: sheltered residences for mentally handicapped, yellow: animal tracking, light blue: asset tracking in construction and green: product specifications for current trackers on the market.
White labels are used for groups and supergroups shown in

4.1.5 Defining product requirements

Based on the data organising session findings, we determined that focusing on personal safety devices was not viable. Although there is a notable need for tracking devices for dementia patients, the market is highly regulated, and numerous effective products already exist. Additionally, connectivity has to be guaranteed for safety, which cannot be done with the current coverage of the LoRaWAN network. The next generation of dementia patients will also use smartphones as they are familiar with them, reducing the need for an external

geolocation tracker. No significant demand was identified among individuals with mental disabilities.

We also found no big demand for pet tracking, except among hunters. However, their requirements are too advanced for the LR1110 chip. Hunters need topographic maps, rapid update rates, and precise location tracking. Battery life, on the other hand, is not a major concern since recharging after each hunt is acceptable, and there are already well-functioning tracking devices available for this audience. Thus, this market is unsuitable for our application.

Vehicle tracking also did not present a viable use case. Since vehicles provide an "unlimited" power supply, ultra-low power consumption is not a significant advantage. Additionally, most users prefer tracking devices that integrate with the vehicle's computer system to monitor aspects like speed and warning messages, which is beyond this project's scope.

The same goes for packing delivery. The companies we contacted indicated that live tracking is unnecessary and not cost-effective compared to, e.g. sufficiently accurate barcode systems.

We identified substantial market potential in asset tracking and wildlife monitoring. In the construction industry, theft of materials is a significant issue. Power tools are expensive and typically stolen in large quantities (thieves often take an entire box of tools rather than just one power drill), so contractors have a significant financial incentive to install trackers in their equipment. However, current solutions are quite large and only suitable for bigger power tools, as shown in Fig. 16. The current approach to integrating tracking devices into smaller power tools involves removing and replacing their internal components with a tracking device. This process degrades the tool to a non-functional decoy, making it obvious that it is being used as a tracker. Consequently, a considerable need exists for smaller trackers with lower power consumption, allowing for reduced battery size. This enables more discreet and effective integration into a broader range of tools, enhancing theft prevention and asset recovery efforts in the construction industry.

In wildlife monitoring, we found that Naturstyrelsen (the Danish Nature Agency) is currently exploring the market for geolocation trackers to track animals as part of their rewilding projects. Legal requirements mandate physical monitoring of these animals to prevent neglect, and locating animals in national parks can be time-consuming without geolocation trackers. Also, since these animals are wild, they have an aversion towards human interaction, and long battery life is advantageous. The current products are expensive and have short battery life. Additionally, the agency has the technical and financial resources to set up its own network and gateways, ensuring LoRaWAN coverage in the designated area. This makes them an ideal candidate for our future solution.

Based on these insights from our empirical data, we chose asset tracking for the construction industry and wildlife monitoring as the primary use cases to develop our product requirements. All the empirical data relevant to these use cases resulted in the product requirements below in Tab. 7.

Table 7: Product requirements deducted from empirical data obtained from interviews and desk research. *R* denoting requirements and *W* denoting wishes.

Category	Requirements	Wishes
Functional requirements	*R1:* Location accessible by phone and tablet	Data accessible via API
	R2: Low battery alarm	*W2:* View battery charge
	R3: Configurable ping interval	*W3:* View temperature
	R4: Remote configuration of ping interval	*W4:* User adjustable LoRaWAN class
	R5: Location triggered by movement	*W5:* Geofence
	R6: Accuracy of 20 m	
Performance requirements	*R7:* LoRa range of 5 km in open terrain	
	R8: 2 years battery life and 1000 positions on one battery charge[12]	
Physical requirements	*R9:* Shape that enables easy case design	
	R10: Size of max 50 × 40 × 10 mm	
	R11: External antennas	
	R12: External battery	
	R13: Weight of 20 g without battery and antennas	
Sustainability requirements	*R14:* Unit price < 1000 DKK	*W6:* Rechargeable battery
		W7: Open-source

4.1.5.1 Functional requirements

The functional requirements describe the functional features, functions, tasks or activities that the tracker must be able to perform.

Location accessible by phone/tablet
A requirement that was mentioned repeatedly by our interviewed users was the ability to access the location data from a mobile device since they do not necessarily have a laptop in many of their usage scenarios.

Low battery alarm
Users highlighted the importance of receiving notifications when the tracker's battery approaches a critical level. This functionality ensures that users can maintain continuous tracking capability without unexpected interruptions or the risk of losing assets due to a depleted battery.

Configurable ping interval

[12] LiPo battery: 1200 mAh, 3.7 V, 6 × 34 × 50 mm

Another prevalent requirement is the ability for a user to change the time interval between the location pings. User preferences vary significantly based on the nature of the tracked object or animal's activity level. For instance, frequent pings are necessary during periods of high movement, whereas less frequent updates suffice during inactivity.

Remote configuration of ping interval
Given that the tracker is likely to be deployed in environments where little interaction with the device is preferable — such as on an animal or concealed within a power tool — remote configuration of the ping interval is essential.

Location triggered by movement
The empirical findings underscored the demand for tracking devices capable of triggering location updates upon detecting movement. This functionality conserves power by limiting unnecessary transmissions and enhances security by providing real-time updates when movement occurs unexpectedly.

Accuracy of 20 m
Our interviews revealed a consensus among users that an approximately 20 m accuracy is sufficient for practical tracking purposes. Better accuracy is preferred, but 20 m will accommodate the requirements articulated.

Data accessible via API
The users also wish to integrate the location data into their own applications. This necessitates the provision of a well-documented API, which would facilitate integration with existing systems or custom applications, empowering users to leverage tracker data according to their specific operational needs.

View battery charge
In addition to a low battery alarm, the users wish to be able to monitor the tracker's battery status.

Temperature monitoring
Temperature monitoring emerged as a desirable feature, particularly in anti-theft scenarios. Viewing the tracker's temperature helps users determine the whereabouts of stolen items based on environmental conditions, aiding recovery efforts.

User-adjustable LoRaWAN class
Users wish to adjust the tracker's LoRaWAN class. This flexibility allows for prioritising either power consumption or response time.

Geofencing
Although not essential, interviewees regard geofencing as a beneficial feature. This capability allows to define virtual boundaries and receive alerts when the tracker enters or exits designated areas, enhancing security and operational efficiency.

4.1.5.2 Performance requirements

The performance requirements specify the necessary parameters that the device must meet to ensure it is both competitive and practical for users.

LoRa range of 5 km in open terrain
Our data indicated that a minimum tracking distance of 5 km in open terrain is essential for the tracker to be useful across various applications. This range ensures the device can reliably transmit location data over sufficient distances, making it practical for use in diverse environments.

2 Years battery life and 1000 positions on one battery charge
To significantly outperform existing products and serve as a compelling alternative, the device equipped with a 1200 mAh battery must operate for 2 years and transmit 1000 positions on a single charge.

4.1.5.3 Physical requirements

The physical requirements specify the physical characteristics and attributes the tracker must adhere to.

Shape that enables easy case design
Users have diverse and specific needs regarding the tracker's size and form factor. Some users prioritise minimal size for discrete placement, while others require robustness and waterproof capabilities for harsh environments. Consequently, we have chosen not to focus on a complete product with casing but only on making a PCB with a shape that facilitates easy adaptation to different case designs. This flexibility allows the device to be housed in collars, anklets, or other form factors as required by the application.

Size constraints
Another critical consideration is the tracker's physical dimensions. To meet the various potential usage scenarios discussed with interviewees and outperform existing market products, the device shall have a maximum dimension of $50 \times 40 \times 10$ mm. This compact size ensures that the tracker can be discreetly integrated into smaller power tools without breaking them and used on animals without causing discomfort or hindrance.

External antennas
Users prefer an external antenna, as separating the PCB and antenna allows for easier customisation to fit different applications.

External battery
The same applies to the battery. Separating the PCB and battery allows for easier customisation to fit different applications. It also allows the users to select battery sizes according to their specific application needs. For instance, larger batteries are acceptable for use with large animals, where additional weight is not a significant concern, whereas smaller batteries are preferable for small animals to minimise the burden.

Weight of 20 g without battery and antennas
To ensure the tracker is suitable for various applications, a device weighing up to 20 g, excluding battery and antennas, ensures that the tracker can be used with smaller animals and integrated into smaller power tools without adding suspicious weight.

4.1.5.4 Sustainability requirements

The sustainability requirements focus on environmental and economic considerations that aim to enhance the tracker's viability and success as a product.

Unit price
Feedback from Enemærke & Petersen indicates that while the absolute price is less critical, the cost of the tracker should be proportionate to the value of the tool it accompanies. The average price of the current trackers we examined is 2475 DKK. However, this figure represents the cost of trackers aimed at regular consumers. Professional-grade trackers, such as those used by Enemærke & Petersen and those investigated by the Danish Nature Agency, generally have a higher price tag. A unit price under 1000 DKK, including components and assembly, makes our product competitive and attractive. This allows the device to be priced lower than current products on the market and makes it economically feasible to track smaller power tools.

Rechargeable battery
A rechargeable battery was not required but regarded as a desirable feature. Fulfilling the requirement for an external battery inherently addresses the desire for a rechargeable battery, allowing users to select battery types.

Open-source design
Another significant sustainability consideration is an open-source design. By making the product open-source, we enhance its repairability and adaptability for our customers. Users can modify and customise the product to meet their specific needs, ensuring long-term viability and reducing the risk of obsolescence. This approach lets customers modify the design, offering greater flexibility and control over their investment.

4.2 Hardware design decisions

Building a geolocation tracker with the LR1110 chip requires, besides the chip, transmitters, receivers, and, most importantly, a device which controls the whole system. This section aims to provide an overview of the hardware architecture and rationale behind the hardware design decisions made for the development of the tracker.

4.2.1 Microcontroller - STM32

For the primary communication of the tracker, the project uses an ultra-low-power Arm® Cortex®-M4 32-bit STM32L476 chip by STMicroelectronics[13]. It has 1 MB flash and a built-in internal voltage regulator and voltage scaling that makes the chip particularly suited for portable battery-supplied products, down to 1.71 V (ST, 2019). It also possesses SPI and I^2C for communication with different chips on the PCB and UART for debugging sent to the user's machine. Since the size of the end device application firmware is around 250 kB, the chip does not need more than 1 MB in flash memory.

[13]STMicroelectronics: https://www.st.com/content/st_com/en.html

The wiring schematic of the STM32L476 and nearby components is shown in Fig. 18. Note that the reference prefixes (e.g. *R7*) on all the following wiring schematics are not in a particular order as multiple values are from different datasheets, and, to minimise confusion, we have left the prefixes the same.

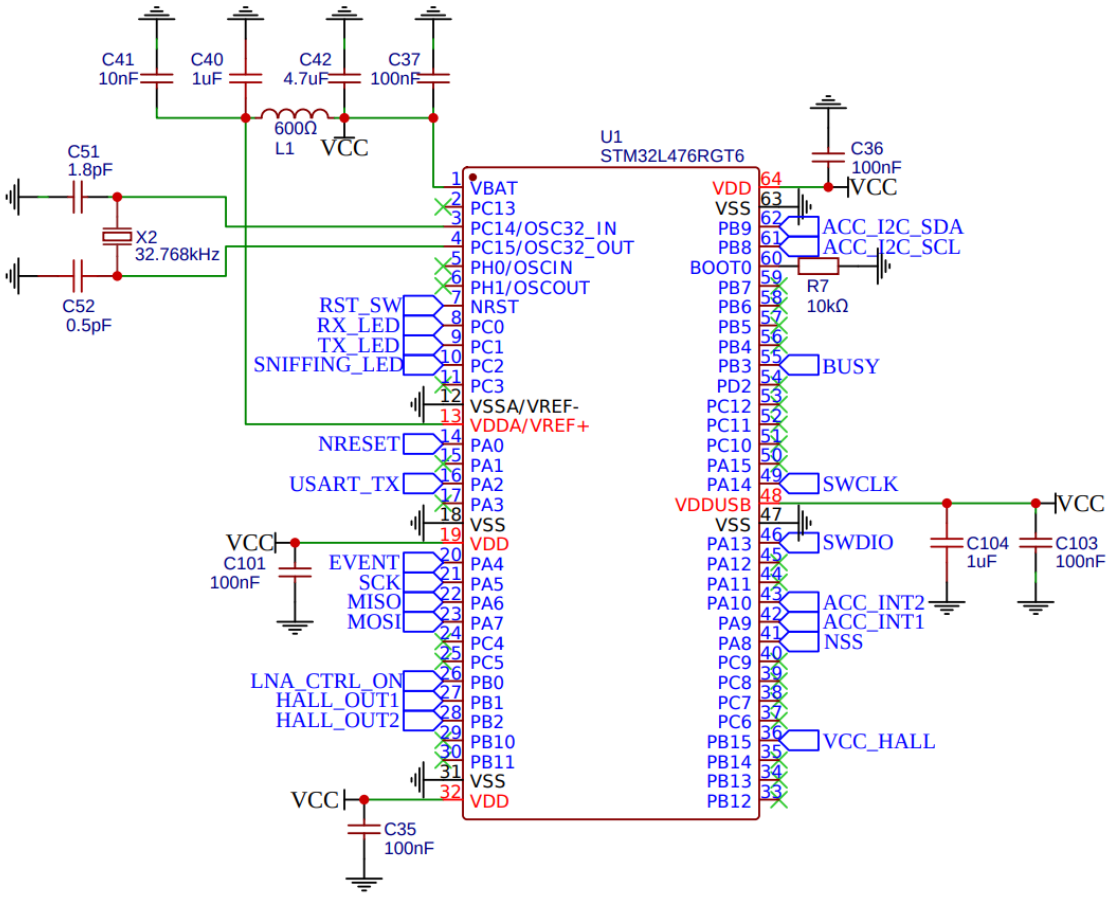


Figure 18: Wiring schematic of microcontroller STM32L476.

The decoupling capacitors *C35*, *C36*, *C37*, *C40*, *C41* ,*C42*, *C101*, *C103*, and *C104*, are defined in the datasheet of STM32L476 ([2], p. 109). Setting *BOOT0* (with pull-down resistor *R7*) to low boots from the main flash memory, which is the normal operating mode when the application code runs. A low pass filter was added to suppress high-frequency noise, using a ferrite bead (*L1*). The placing between *VDDA/VREF+* and *VBAT* helps to isolate the analogue power supply from digital noise generated by the microcontroller's core and other digital circuits.

The 32.768 kHz crystal resonator *X2* and its associated capacitors *C51* and *C52* are defined in the STM32L476 datasheet ([2], p. 151).
The following sections describe the rest of the NET labels in the wiring diagrams.

4.2.2 RF transceiver and receiver - LR1110

The LR1110 chip is a LoRa **R**adio **F**requency (RF) receiver and transceiver by Semtech[14] with Wi-Fi and GNSS scanning capabilities. Wi-Fi includes scanning for 802.11b/g/n Wi-Fi access point MAC addresses on the 2.4 GHz frequency band, while the GNSS scanning searches for GPS and BeiDou SV signals.
GNSS and Wi-Fi scan data is collected on the device and sent to the cloud solver. Fig. 19 shows the simplified wiring schematic of the LR1110 required to perform geolocation.

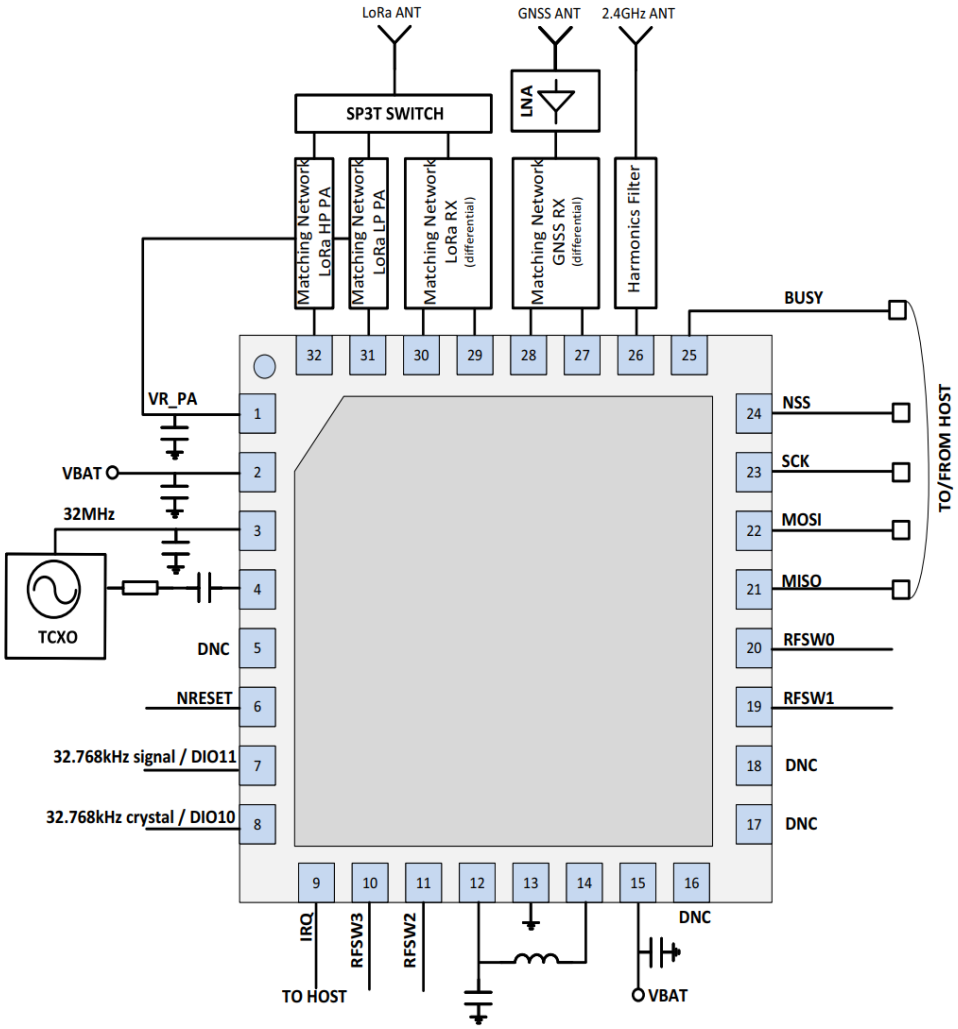


Figure 19: Simplified diagram of LR1110 with its essential components to perform geolocation. ([1], p. 35).

The wiring schematic of the LR1110 and nearby components is shown in Fig. 20.

[14]Semtech: https://www.semtech.com/

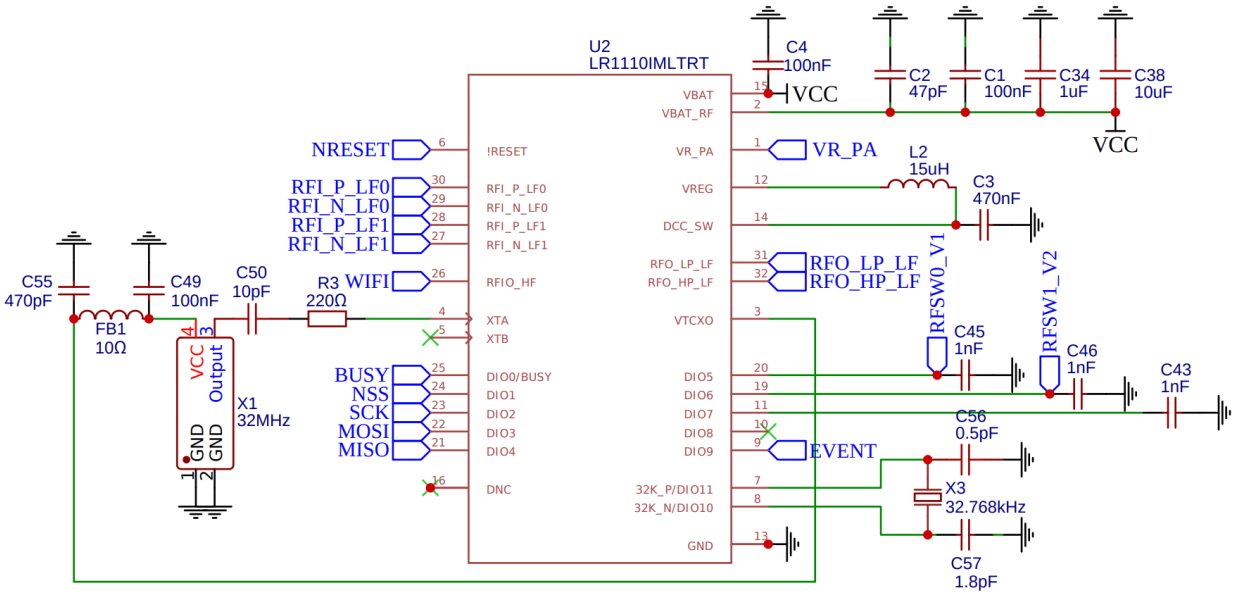


Figure 20: Wiring schematic of RF transceiver and receiver, LR1110.

L2 and *C3*, connecting the regulated output voltage from the internal regulator DC-DC (pin 12) to the DC-DC switcher output (pin 14), are specified in the PCB Design Guideline for LR1110 (Semtech, 2022, p. 7). Details of other components in the schematic are provided in subsequent sections covering Oscillators, and under the LoRa-, Wi-Fi-, and GNSS antenna designs.

4.2.3 Oscillators

A 32 MHz crystal oscillator is the cheapest and lowest power-consuming approach to provide the 32 MHz clock reference to the LR1110 (Semtech, 2023, p. 50). A 32 MHz **T**emperature **C**ompensated Crystal **O**scillator (TCXO)[3] (see Fig. 20 *X1*) and a 32.768 kHz clock source (see *X3*) are mandatory by the LR1110 for any GNSS operation. Furthermore, the TCXO helps minimise power consumption of outdoor geolocating (Semtech, 2023, p. 51, and p. 132). Since the crystal oscillators are affected by surrounding temperature, no copper planes have been added on the PCB where crystal oscillators are placed.

A ferrite bead is needed on the supply to the TCXO (Semtech, 2022, p. 27), and following [1, p. 25], a serial capacitor (10 pF) and serial resistor (220 Ω) are mandatory for the oscillator output to lower the amplitude (Semtech, 2022, p. 27). *C49* and *C55* are defined in (Semtech, 2023, p. 51).

4.2.4 Antenna designs

Antenna design is challenging as it involves complex considerations such as frequency tuning, impedance matching, and physical constraints, which demand specialised knowledge and extensive testing. Given these considerations, it was decided to use pre-tuned antennas, one for each wireless protocol implemented on the board: Wi-Fi, GNSS and LoRa.

The following sections describe the design of the filter and power lines for the antennas. Proper filtering is necessary to remove unwanted frequencies and noise from the signal, ensuring that only the desired frequency bands are transmitted and received. This improves signal clarity and reduces interference, leading to more accurate and reliable communication.

4.2.4.1 LoRa antenna

The PCB Design Guideline for LR1110 (Semtech, 2022, p. 14) has been used as inspiration for the LoRa RF trace shown in Fig. 21.

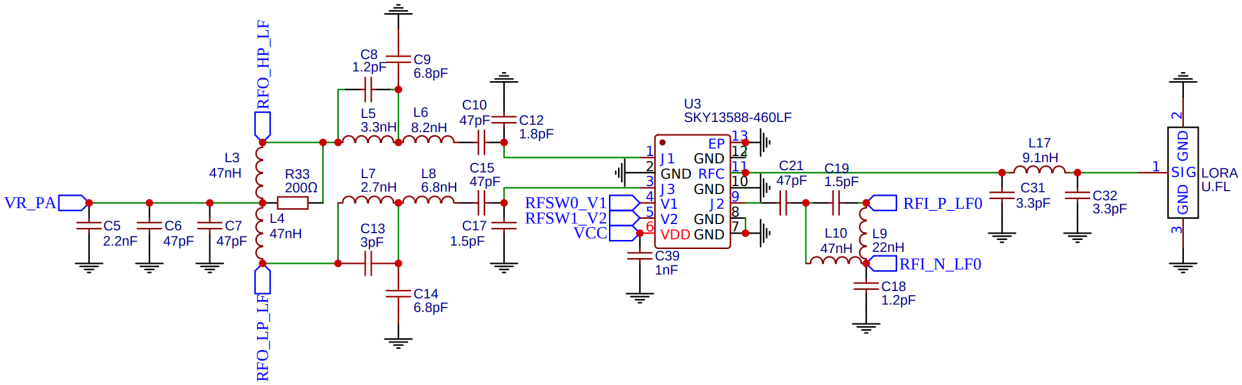


Figure 21: Wiring schematic of the LoRa RF trace.

C5 (2.2 nF), *C6* (47 nF), *C7* (47 nF), and *R33* (200 Ω) are defined (Semtech, 2022, p. 36) to match the **P**ower **A**mplifier (PA). We are using the RF splitter SKY13588-460LF which can choose RF traces between 0.1 and 6.0 GHz. The values for the harmonics filter between the PA matching and the RF switch are taken from (Semtech, 2023). The same goes with the RX matching filter at input *J2*.

After the RF switch, we added another harmonic RX filter as a π low pass filter[15], which filters out all the frequencies over cutoff frequency f_c. The impedance magnitude of the capacitor is set to $Z_C = 50\,\Omega$ (matches the impedance of the antenna) to calculate the capacitance, *C32* (which also is the same as *C31*):

$$Z_C = \frac{1}{\omega \cdot C32}$$

where $\omega = 2\pi f_c$. LoRa signals in Europe can be send using the 433 MHz or 868-870 MHz spectrum. The 868-870 MHz spectrum was chosen as it is close to the USA (902-915 MHz) frequency span for LoRa signals. This minimises the design of the filter as the same components support both regions. The cut-off frequency is therefore set to 920 MHz:

[15] π low pass filter named π because the wiring schematic looks like the π symbol

$$50\,\Omega = \frac{1}{2\pi \cdot 920\,\text{MHz} \cdot C32}$$
$$\downarrow$$
$$C32 = 3.45\,\text{pF}$$

In practice, the closest commercial capacitor is $C32 = 3.3\,\text{pF}$.

The formula for a low pass filter is given by:

$$f_c = \frac{1}{2\pi\sqrt{L17 \cdot C32}} \tag{4.1}$$

where $L17$ is the inductor. Adding f_c and $C32$ to calculate $L17$:

$$920\,\text{MHz} = \frac{1}{2\pi\sqrt{L17 \cdot 3.3\,\text{pF}}}$$
$$\downarrow$$
$$L17 = 9.06\,\text{nH}$$

In practice, the closest commercial inductor is chosen to be $L17 = 9.1\,\text{nH}$. The edge values of the filter according to the tolerance of inductor ($\pm 5\%$) and capacitor ($\pm 0.25\,\text{pF}$) will set the cut-off frequency from 864 to 980 MHz. Although the lower cut-off frequency is close to the EU LoRaWAN frequency span, this uncertainty does not affect the signal.

For the LoRa antenna we use a Molex 211140 50 Ω 868/915 MHz which is small and can be bend around corners. Also, it comes with a U.FL connector, is self-adhesive, and can be mounted within the tracker's enclosure.

4.2.4.2 Wi-Fi antenna

For the Wi-Fi RF trace, we apply a double π filter, which means we use Eq. 4.1 twice, one for each link. The values of the components are shown in Fig. 22.

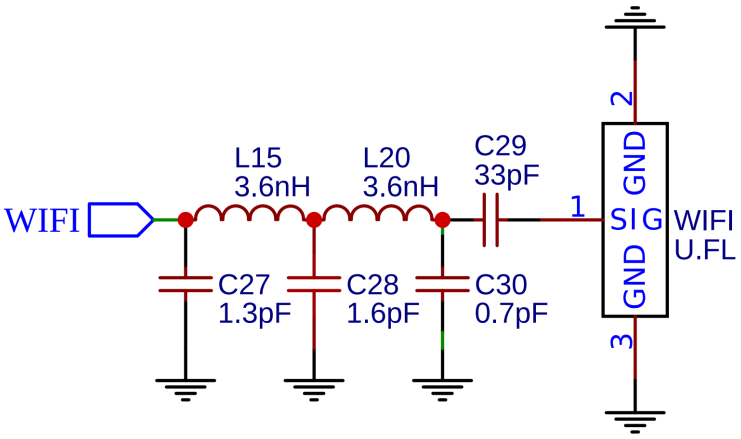


Figure 22: Wiring schematic of the Wi-Fi RF trace.

Here *C29* blocks any DC component in the signal path reaching the U.FL antenna. This is crucial in RF applications where only AC signals are desired.

For the Wi-Fi antenna we use a Molex 146153 50 Ω 2.4/5 GHz antenna.

4.2.4.3 GNSS antenna

For this project, a passive GNSS antenna was chosen due to the assumption that it would consume less power than an active antenna since power efficiency is crucial in our design. As the LR1110 development board is equipped with active and passive antennas, testing this hypothesis must be done before the final PCB design.

The BGA524N6E6327 was selected for its compact size and low-voltage operation, making it an ideal GNSS **L**ow **N**oise **A**mplifier (LNA). It is depicted in Fig. 23 along with the GNSS RF trace.

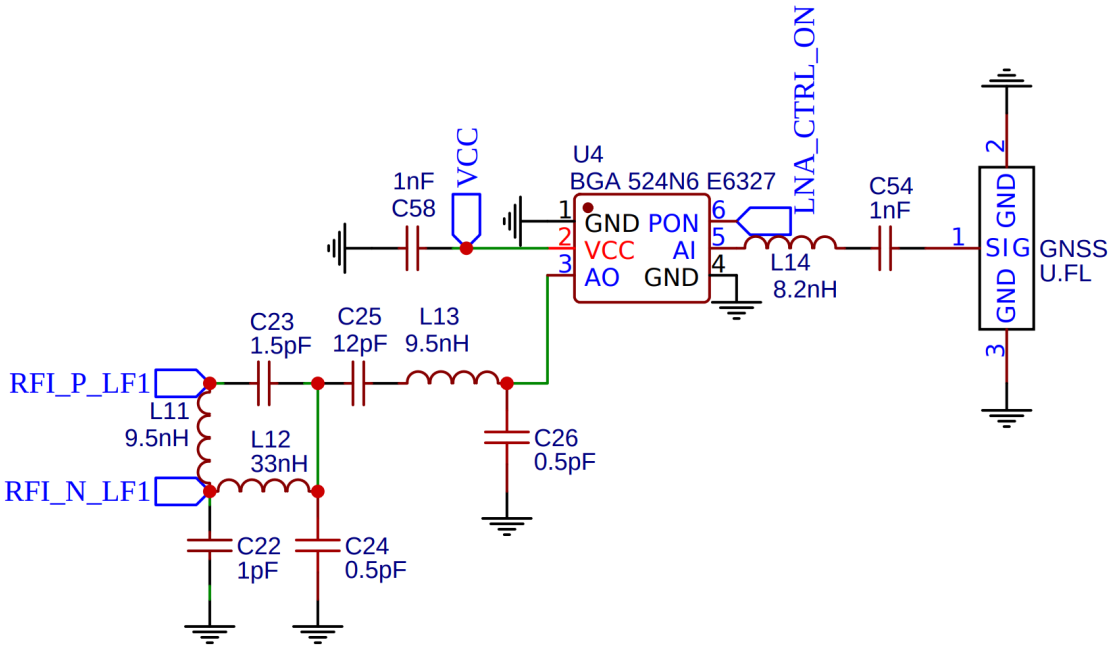


Figure 23: Wiring schematic of the GNSS RF trace.

C54, *C58* and *L14* are determined in the datasheet of BGA524N6E6327 ([7], p. 12), where the circuit left of BGA524N6E6327 is inspired by the PCB Design Guideline for LR1110 (Semtech, 2022, p. 18). The RF output is internally matched to 50 Ω, so no extra external components are needed to match the antenna's impedance.

For the GNSS antenna we use a Molex 206560 50 Ω 1558−1610 MHz antenna.

4.2.5 Accelerometer

To determine if the tracker has been at rest for a long time or has suddenly moved, an accelerometer was implemented. LIS2DE12, high-performance 3-axis accelerometer with

builtin SPI and I^{2}C was chosen for its ultra-low-power capabilities. To further enhance its low power capabilities, I^{2}C was selected as the communication protocol, as it has lower power requirements due to its reduced speed, as high-speed data transfer is unnecessary for this specific use case. The wiring schematic of the accelerometer is shown in Fig. 24.

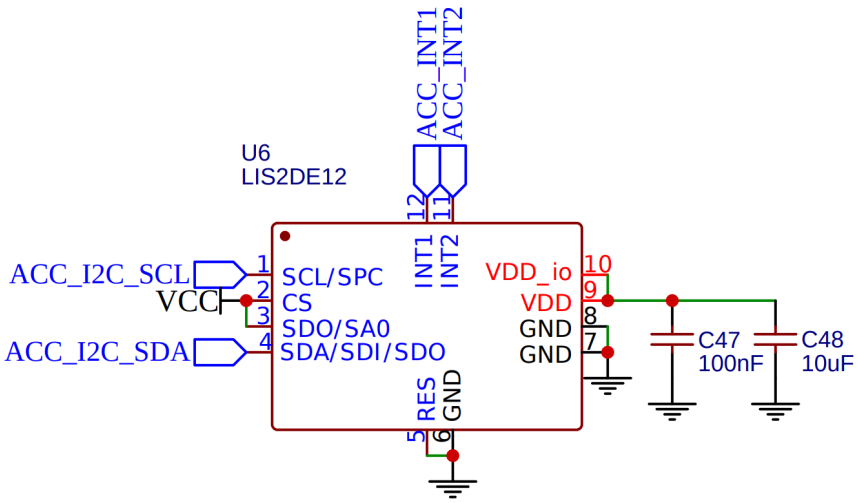


Figure 24: Wiring schematic of the accelerometer.

C47 and *C48* are determined in the datasheet of LIS2DE12 ([4], p. 19).
The accelerometer has two independent programmable interrupt generators (*ACC_INT1* and *ACC_INT2*) for free-fall and motion detection. We use both to determine if the tracker has moved. Its implementation is described further in Fig. 48.

The accelerometer's current consumption depends on the **O**perating **D**ata **R**ate (ODR) shown in Tab. 8. It has been set to 10 Hz as this is fast enough to update its measurement data and is, therefore, the best compromise between speed and power consumption.

Table 8: LIS2DE12 current consumption in normal mode ([4], p. 16).

Operating Data Rate (ODR) [Hz]	Normal mode [µA]
1	2
10	3
25	4
50	6
100	10
200	18
400	36
1620	100
5376	185

Fig. 25 shows the interface between the accelerometer and the STM32 microcontroller.

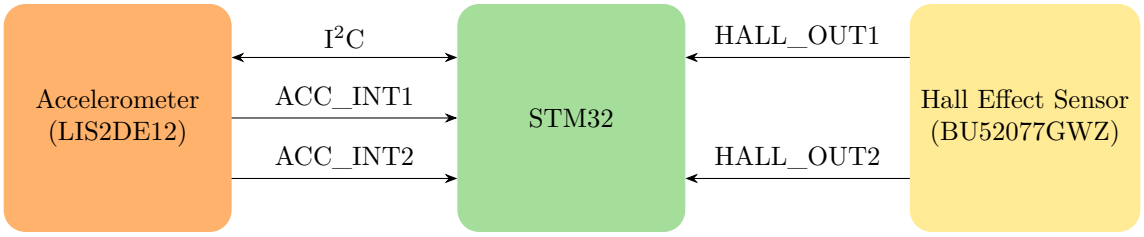


Figure 25: Interface between accelerometer, STM32 and Hall effect sensor.

4.2.6 Hall effect sensor

A local way of configuring the tracker, such as device resetting, was implemented to further enhance the device's capabilities. A Hall effect sensor was chosen to facilitate ease of case design and waterproofing. BU52077GWZ was chosen because of its typical low operational current of 5 µA and low minimum operation voltage of 1.65 V. It has a standard digital output interface and is, therefore, easy to implement with a simple interrupt on the STM32, which is ideal for the application.
The Hall effect sensor wiring diagram is shown in Fig. 26 and as a simple block diagram in Fig. 25.

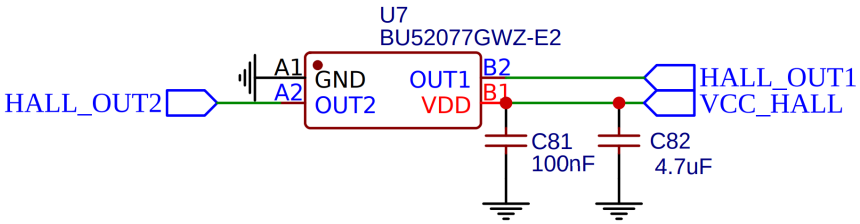


Figure 26: Wiring schematic of the Hall effect sensor.

C81 and *C82* are defined in the datasheet of BU52077GWZ ([5], p. 1). *HALL_OUT1* is triggered when reacting to the south pole of a magnet, and *HALL_OUT2* is triggered when responding to the magnet's north pole. This capability allows the chip two send two distinct interrupts to the tracker, that can be used to control different actions, such as toggling between configuration modes or initiating a reset operation.
VCC_HALL is used to turn on/off the chip if not in use to save power.

4.2.7 Power sourcing

The terms *low power* and *ultra-low power* refer to electronic components and devices designed to consume minimal amounts of energy, making them ideal for battery-operated and energy-efficient applications. There is no official standard describing what documents an ultra low power component or device. However as the IC's on the PCB are described as *ultra-low power* components, we will describe our product as an ultra-low power device.

For a stable power supply, we use the voltage regulator XC6220B331MR. The chip is a 3.3 V voltage regulator with an input voltage ranging from 1.6 V to 6.0 V and a maximum output

current of 1000 mA. The quiescent current is 8 µA and the dropout voltage is around 20 mV. The voltage regulator wiring diagram is shown in Fig. 27.

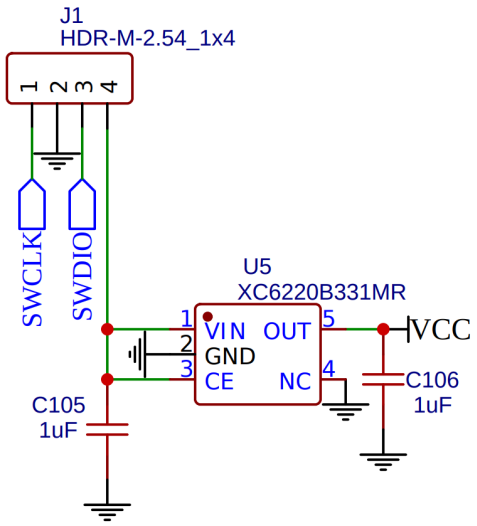


Figure 27: Wiring diagram of the voltage regulator.

Crucial for any battery-powered device is the voltage regulator. The chosen one has high and low power modes, which means the quiescent current and dropout voltage can be reduced if the device is in low power mode. When the output current is less than 1 mA (minimum), the quiescent current is reduced from 50 µA to 8 µA; if the output current becomes 10 mA (maximum) or more, the mode changes automatically to the high power mode, and the chip returns to high-speed operation.

4.3 PCB Design

This section describes the iterative process and considerations involved in the design of the PCB for the tracker. The section explores RF trace impedance considerations, highlighting their critical role in ensuring efficient signal transmission and reception. The following sections detail the evolution through different PCB versions, each iteration aimed at testing specific hypotheses and refining the design to meet the product requirements. We refer to Houde and Hill's perspective on *What do prototypes prototype* (Houde and Hill, 1997) presented in Sec. 3.2, to explain the strategic decisions made at each stage of PCB development before designing the PCB in EasyEDA[16] and ordering from JLCPCB[17].

By detailing the journey from initial concepts to refined prototypes, this section aims to provide a comprehensive understanding of the PCB design process for the tracker, highlighting the iterative nature of development and the strategic insights gained through prototyping.

[16]EasyEDA: https://easyeda.com/
[17]JLCPCB: https://jlcpcb.com/

4.3.1 Impedance of RF traces

Designing a PCB with RF traces starts by placing the antennas(Semtech, 2022, p. 12); GNSS antenna is the priority for the LR1110. The LR1110 and all necessary decoupling capacitors are placed close to the chip.
All RF traces are routed from the chip to the antenna while keeping the ground plane under the RF complete and without openings. All other critical lines are routed away from the RF, e.g. high-speed digital lines and reference high-frequency clocks like the 32 MHz TCXO. Finally, all DC supply lines on a specific power plane and all other analogue and digital lines on the PCB are routed. For determining the impedance and maintaining it throughout the PCB, the layout of the components and stackup of the PCB is critical. We chose a 4-layer PCB as this is preferable to optimise RF PCB layout, especially if there is dense routing, as in our case.

Any RF trace can potentially radiate or receive interfering signals, which is why the RF trace between the antenna and the matching network is structured using a **G**rounded **Co**planar **W**aveguide (GCPW) configuration, illustrated in Fig. 28. This design incorporates a solid ground plane beneath the RF trace on the second layer to effectively minimise **E**lectro**m**agnetic **C**ompatibility (EMC) issues. The GCPW is dimensioned to have a characteristic impedance equal to the antenna impedance, which we set to 50 Ω.

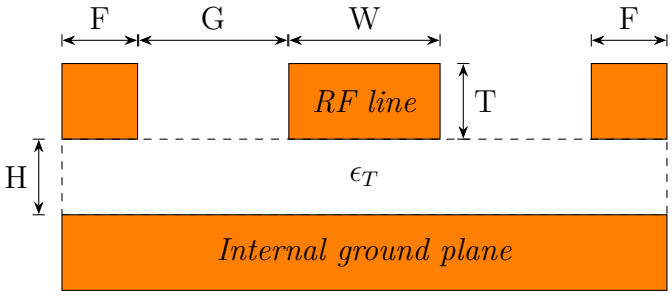


Figure 28: Cross section of RF line layout. The layer closest to the RF line is a ground layer, which will minimise EMC issues.

Choosing JLCPSB's JLC04101H-7628 Stackup has a dielectric constant of $\kappa = 4.4$, which will suit this project. The stackup is described in Tab. 9.

Table 9: Stackup of JLC04101H-7628 with PCB thickness of 1.0 mm[18].

Layer	Material Type	Thickness [mm]
Top Layer	Copper	0.035
Prepreg	7628	0.2104
Internal Ground Plane	Copper	0.0152
Core	Core	0.45
Internal Power Plane	Copper	0.0152
Prepreg	7628	0.2104
Bottom Layer	Copper	0.035

To calculate the transmission line characteristic impedance [Ω], we will use Coplanar Waveguide by AppCAD [7]. Fig. 29 shows the top layer, prepreg, and the internal ground plane.

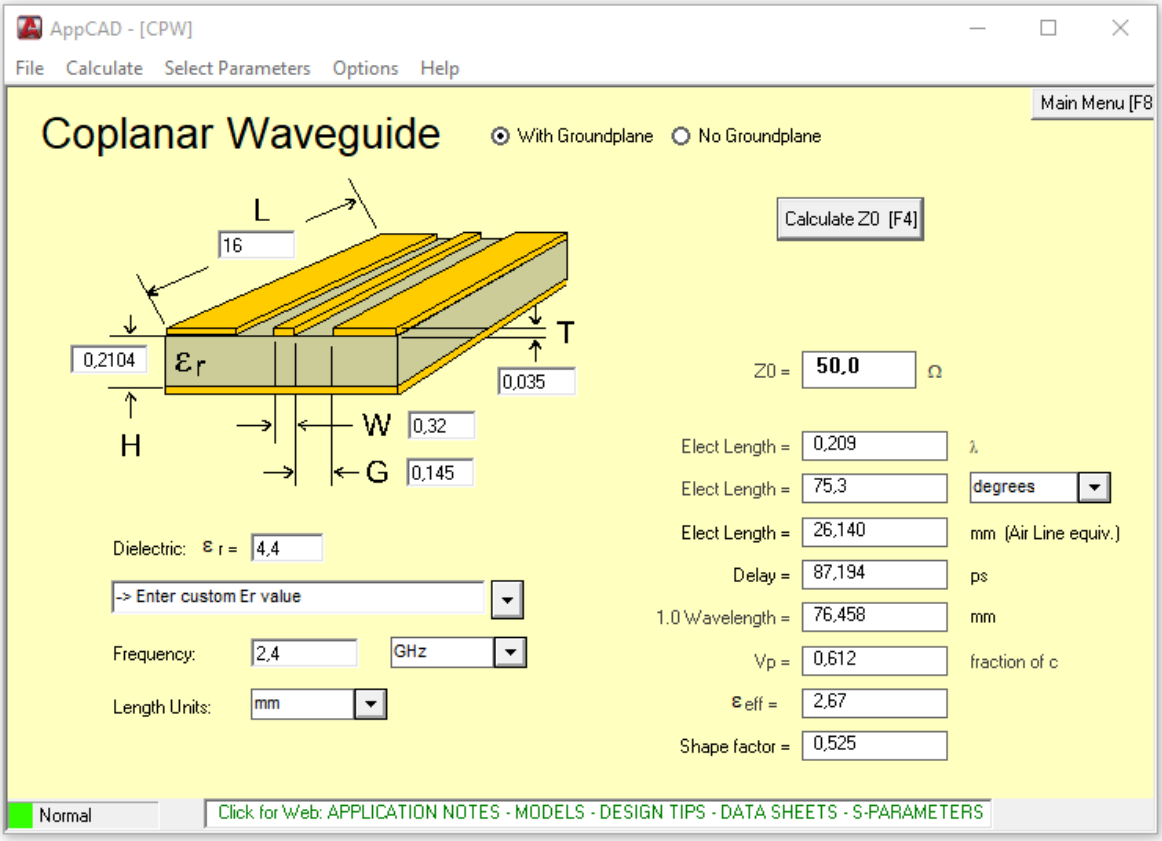


Figure 29: AppCAD Coplanar Waveguide: Calculating the impedance of the Wi-Fi RF line.

According to the PCB Design Guideline for LR1110, the value of G, W, T, H, and ϵ_r (see diagram in Fig. 29) shall be determined to ensure 50 Ω with coplanar waveguide with ground to RF trace. W must be constant all along the RF path, and if possible, the same width as the components' pads. Furthermore, the PCB vias must be stitched at a distance of a least $D = \lambda/20 = 26.150\,\text{mm}/20 = 2.645\,\text{mm}$, where λ is the wavelength of the signal passing through the trace (Semtech, 2022, p. 9-11).
The width of the ground on each side of the RF trace F (see Fig. 28) must be larger than the width of the RF trace W + the gap G: $F > W + G$. Even if two RF traces are close to each other (e.g. the RF traces for LoRa), a ground area should separate them to avoid coupling (Semtech, 2022, p. 20-21).

[18] JLCPCB impedance control: https://jlcpcb.com/impedance

Most components on the RF line have a width of approximately 0.32 mm; thus, W was also set to 0.32 mm. The PCB thickness was decided to be 1 mm (as shown in Tab. 9). According to JLC04101H-7628, the height T is 0.035 , resulting in H being 0.2104 mm. The value G was selected as 0.145 mm to achieve an impedance of the RF line at 50 Ω.

Finally, we chose the PCB design rules for the RF traces to be as track width $W = 0.32$ mm, clearance $G = 0.145$ mm, PCB via diameter of 0.25 mm, and PCB via drill diameter of 0.15 mm

4.3.2 PCB version 1

The primary purpose of the PCB prototypes is to test the product's technical feasibility. The initial PCB design was a pure implementation prototype (see Fig. 30). Given our lack of prior experience in programming, flashing, and debugging on barebone STM32 chips, this prototype was explicitly designed for performing experiments with this. It features a simple 2-layer layout with minimal components and oscillators. Debugging tools such as **L**ight-**E**mitting **D**iode (LED)s and UART were included, along with the footprint for the LR1110 chip. A rendering can be seen in Fig. 31). This setup allowed us to test the LR1110 chip as a hat/shield configuration, facilitating easier removal for troubleshooting purposes.

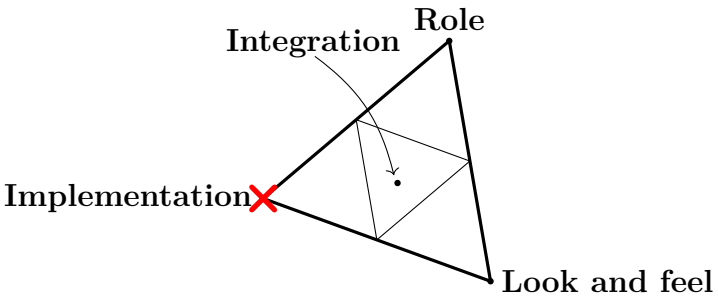


Figure 30: Regarding the four principal categories of prototypes on the model from Houde and Hill (Houde and Hill, 1997, p. 369), our first PCB prototype lies in the implementation part of the triangle, as it was built as an experimental model to test the technical feasibility.

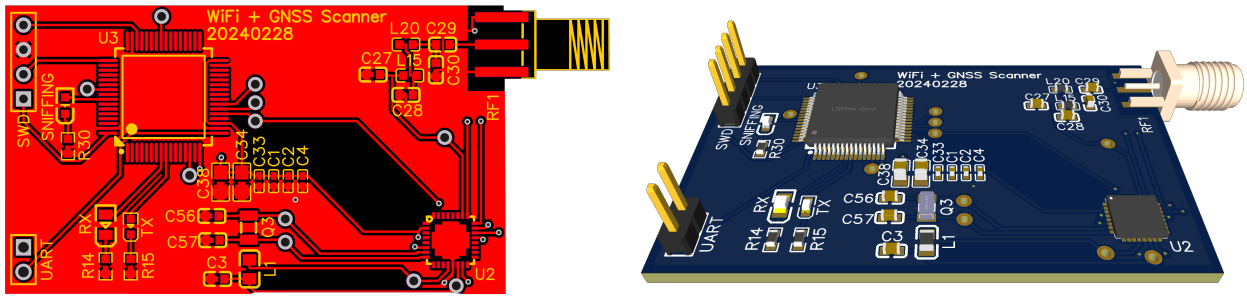


Figure 31: First version of our PCB on 54 × 32 mm.

The tests were conducted successfully, as communication was established with the STM32 and LR1110 chip, and the STM32 could be flashed and debugged. An unexpected result of

the prototype test was the discovery that the placement of the RF traces was suboptimal. The non-linear placement of the filter components led to heightened interference and signal loss, thereby hindering the device's wireless capabilities, making it unable to detect Wi-Fi APs.

4.3.3 PCB version 2

After successfully being able to flash and debug the system, we designed the second version (see Fig. 32). This time, we designed a 4-layer PCB for better routing and with a whole plane just for the ground (as suggested in Sec. 4.3.1). We added the RF traces for Wi-Fi, GNSS and LoRa, a 3.3 V voltage regulator, a reset switch, and an accelerometer to determine if the tracker has moved.

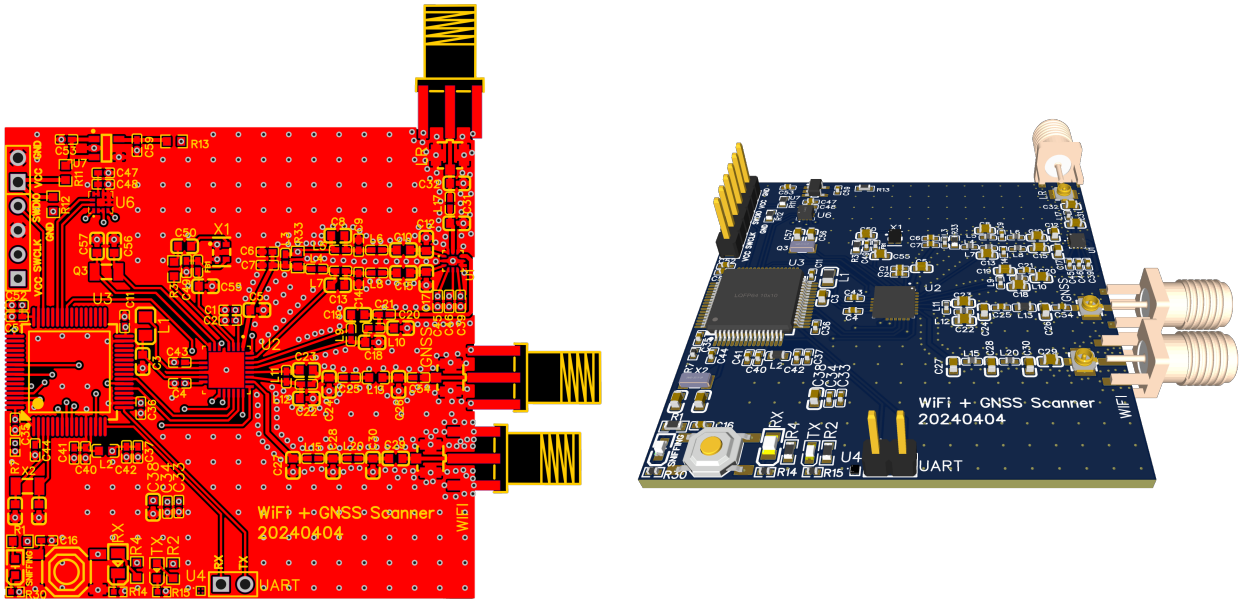


Figure 32: Second version of our PCB on 50 × 50 mm.

While our second prototype is still mostly a pure implementation prototype, it is a bit further towards the look-and-feel side since we have also started experimenting with how small we can make the PCB to get closer to a final form of the future device (see Fig. 33).

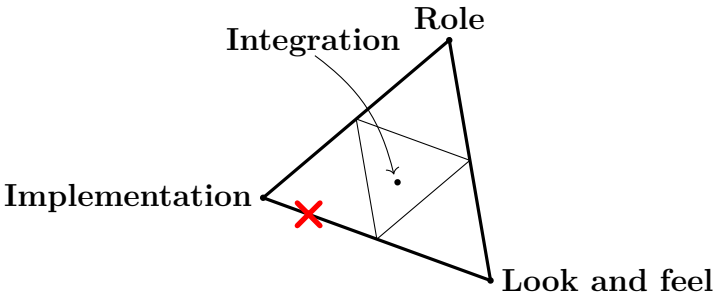


Figure 33: The placement of our second PCB prototype on the four principal categories of prototypes model from Houde and Hill (Houde and Hill, 1997, p. 369).

Besides successfully testing using 4-layer PCB and integrating a voltage regulator, a reset switch, TCXO, and an accelerometer, not many other results came out of the test. A crucial trace between the LR1110 chip and the STM32 was missing (and impossible to add later with the physical board), making communication with the LR1110 chip impossible.

4.3.4 PCB version 3

On the third and current version of our PCB (see Fig. 34), we removed the LEDs and the reset button to make the board even smaller; the pins were still there, which means they further could be used with the combination of a hat/shield for debugging.
The size of the PCB is only 49 × 33 mm. We added the Hall effect sensor and removed the big SMA antenna connectors while keeping the smaller U.FL connectors.

Like the other prototypes, this is still mostly a pure implementation prototype. We did, however - as with the second prototype - make this one even smaller to get closer to the possible final form of the device. It is, therefore, a bit further towards the look-and-feel side (see Fig. 35).

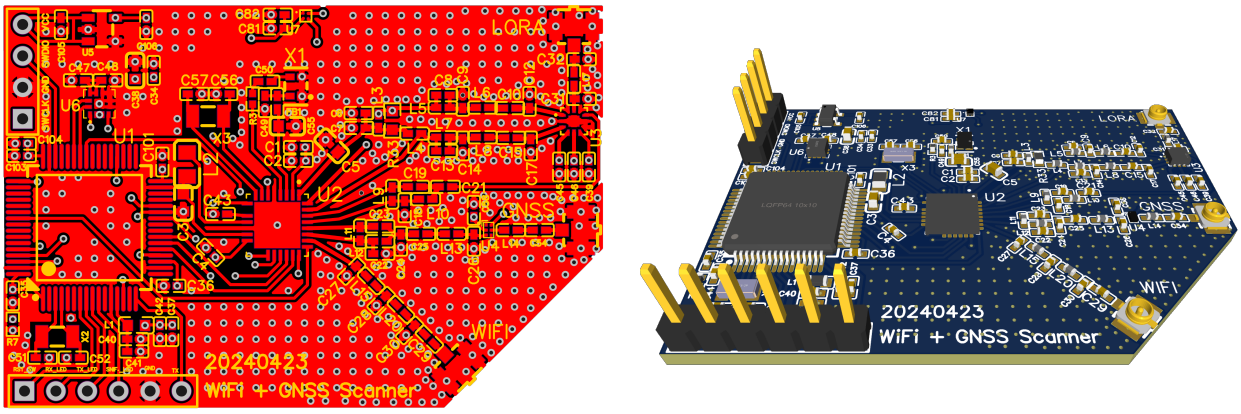


Figure 34: Third and current version of our PCB on 49 × 33 mm. A larger picture can be found in App. E.C and App. E.D.

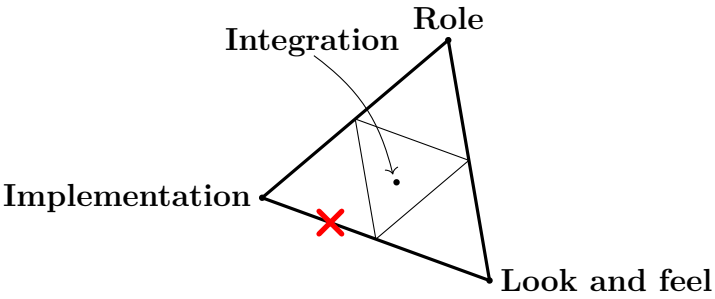


Figure 35: Placement of our third PCB prototype on the four principal categories of prototypes model from Houde and Hill (Houde and Hill, 1997, p. 369).

As most components are tiny and sometimes will shorten when soldering on, X-ray images (see Fig. 36) were taken to verify that the components will not shorten.

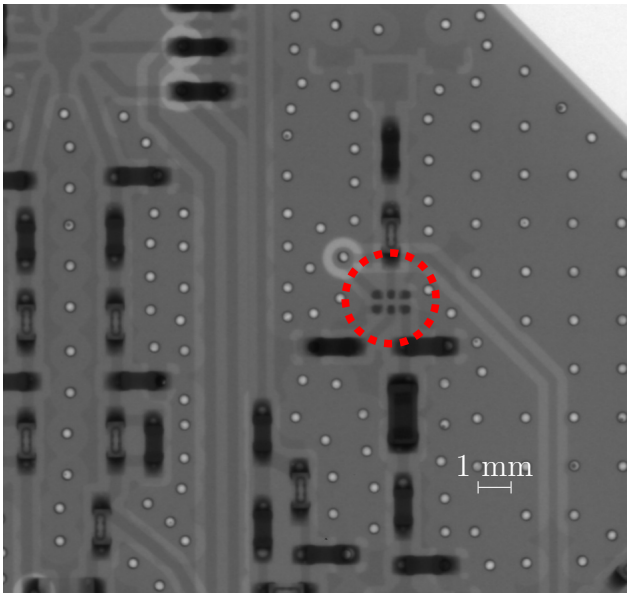


Figure 36: X-ray (90 kV, 80 µV) image of the smallest components. The component U4 (BGA524N6E6327) with its six pins has a size of 1.1 × 0.7 mm.

Fig. 37 shows the wiring schematic of our third and current end device geolocation tracker version.

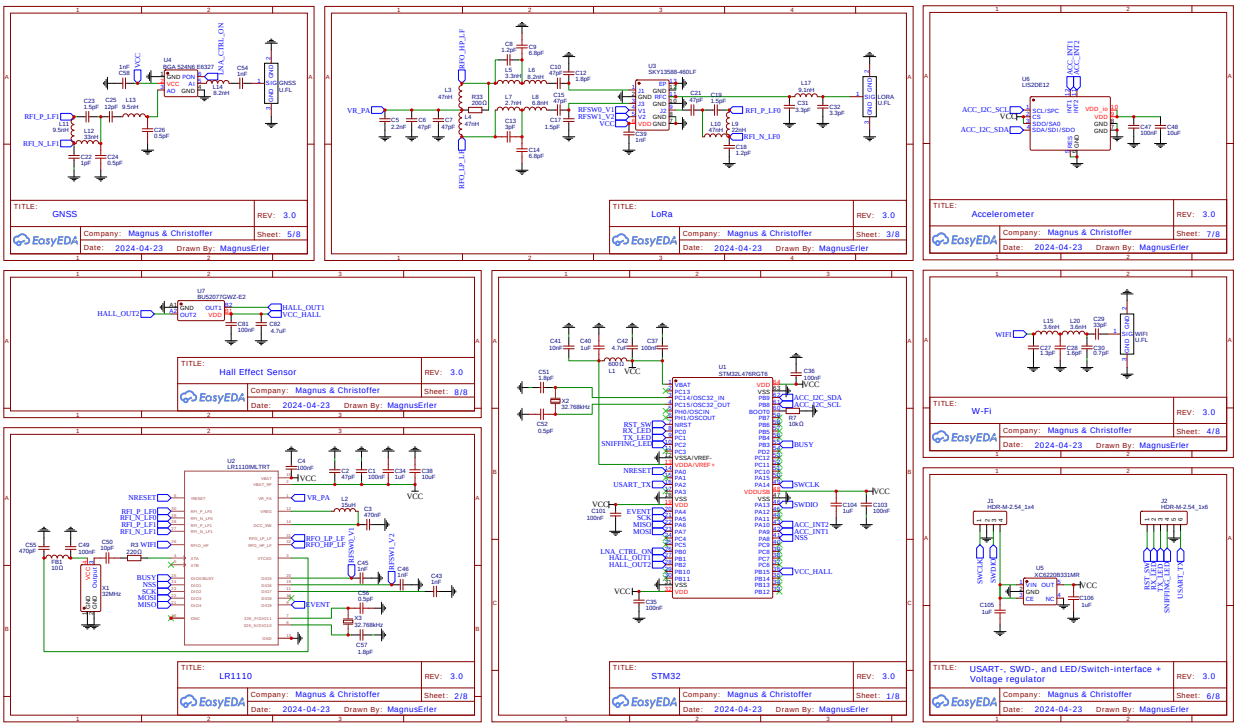


Figure 37: Overview of the wiring schematic of the third and current version. A larger schematic can be found in App. E.B.

4.3.5 Expenses

The total cost of the end device (PCB and antennas) come out to a total cost of around $38 or 265 DKK in local currency. The LR1110 chip was ordered from DigiKey Denmark[19] where it costs 72.23 DKK per chip. Ordering most of the other components from JLCPCB costs $2 for the PCB and an extra $33.35 for the assembly of two PCBs. Combining these, including the antennas, the U.FL connectors, TCXO, and the voltage regulator, gives us the total cost. A detailed list of the prices of the components is given in the **B**ill **O**f **M**aterials (BOM) in App. E.A.

4.3.6 Case design

As explained in Sec. 4.1.5, potential users have diverse and specific needs regarding the tracker's size and form. Some users prioritise minimal size for discrete placement, while others require robustness and waterproof capabilities for harsh environments. Consequently, we have chosen not to focus on a complete product with casing but only on making a PCB with a shape that facilitates easy adaptation to different case designs. This flexibility allows the device to be housed in collars, anklets, or other form factors as required by the application.

However, we chose to add a 50 mm diameter PVC heat-shrinkable tube around it to prevent

[19] DigiKey Denmark: https://www.digikey.dk/

electrical shock or shortcuts. Fig. 38 shows the tracker wrapped in the heat-shrinkable tube and its three antennas.

Figure 38: Picture of the tracker with the blue heat-shrinkable tube and the three antennas.

Regarding the battery, any battery with an output between 1.7 and 6 V can be used, as this is defined by the voltage regulator. The battery is connected to the tracker via a simple two-wire connector (e.g. JST PH2 connector) and can be extended with a longer cable if needed in case of a tight fit.

4.4 Development setup

Before describing the firmware design for the tracker, we summarise the guidelines for developing an efficient environment for coding, flashing, and debugging the LR1110 with an STM32L4xx chip using Visual Studio Code in both Linux and Windows environments.

4.4.1 Development prerequisites

Windows: If working in Windows[20], ensure the following programs are installed:

- Git
- Visual Studio Code
- Arm GNU Toolchain (download as zip)
- OpenOCD

[20] **W**indows **S**ubsystem for **L**inux (WSL) can not be used due to lack of peripheral access (**U**niversal **S**erial **B**us (USB)) in WSL2

- Node.js and NPM

Additionally, the following binaries are used:

- Make for Windows (both binaries and dependencies)

Linux: Software installations will be managed through Linux's built-in package manager; no separate program downloads are required.

For both Windows and Linux, download the LR1110 firmware `0x0401` from GitHub.

4.4.2 Installation guide

Install Git, then clone[21] the repository from GitHub. Next, update the LR1110 chip firmware to version `0x0401`, and plug in the STM32 chip connected to the LR1110 chip (as depicted in Fig. 39 and Fig. 40). Transfer the `.bin` file with the `0x0401` firmware version onto the device (e.g. via drag-and-drop).

Windows:

1. Install Visual Studio Code and unpack the Arm GNU Toolchain, OpenOCD, Git, and Make compressed files to a folder.
2. Add paths for Arm GNU Toolchain, Make, OpenOCD, and Git to system environment variables[22].
   (a) Add new system variable with name `ARMGCC_DIR` with value
   `<your path>\arm-gnu-toolchain-<version>.Rel1-mingw-w64-i686-arm-none-eabi\bin`
   (b) Edit path variable and add the following
      i. `\%ARMGCC_DIR\%`
      ii. `<your path>\make-<version>-bin\bin`
      iii. `<your path>\make-<version>-dep\bin`
      iv. `<your path>\OpenOCD-<version>\bin`
      v. `<your path>\git\bin`
      vi. `<your path>\git\usr\bin`
3. Open a terminal and test that the programs are installed correctly by running
   ```
   make --version
   openocd --version
   git --version
   arm-none-eabi-gcc --version
   ```

[21] Cloning a repository: https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository
[22] Set Windows environment variables: https://www.computerhope.com/issues/ch000549.htm

If the commands return a version number, the prerequisites have been installed correctly.

4. Go to the `stm32/geolocation`-directory inside the repository and compile the project with `make full_lr1110` with the Git Bash terminal.
   If no error appears, a `.bin`-file is created inside the newly created folder: `.../build_stm32_l4_lr11xx_crypto_with_cred`.
5. Flash the board by plug in the STM32 via USB (as seen in Fig. 39 and Fig. 40) and drag the `.bin`-file to the device.

Linux:

1. Install Visual Studio Code.
2. Open a terminal and run the following commands:
   (a) `sudo apt-get update && sudo apt upgrade`
   (b) `sudo apt-get install git make gcc-arm-none-eabi openocd`
3. Follow from step 3 in the above installation guide for Windows.

4.4.3 Debugging with Visual Studio Code

The compiling of the project is based on a simple makefile and can be compiled by running `make full_lr1110`[23] in the terminal. However, developing and compiling with Visual Studio Code is preferred.
To integrate the project, open the newly cloned repository in Visual Studio Code.

1. Install Cortex-Debug from the Visual Studio Code Extension Marketplace.
2. Create a new folder called `.vscode` in the STM32 folder and add the two files `c_cpp_properties.json` and `launch.json`.
3. Debug the program by going into "Run and Debug", choosing Cortex Debug as the debugger and pressing "Start Debugging".

For setting up the web application, follow App. D.C.

4.5 End device application design

This section explores the development process of our end device application (see Fig. 13). It starts by providing a brief overview of the procedures for flashing and debugging when working with an STM32 chip. Subsequently, it discusses our initial, naive attempt to create an application utilising the barebone transceiver firmware of the LR1110 chip and the reasons for abandoning this approach. Finally, we present our implemented application design, incorporating the LoRa Basics Modem software.

[23] Run the make command without arguments to see all the compile options

4.5.1 Flash and debug with ST-LINK

Flashing and debugging any STM32 chip is straightforward. All that is needed is an ST-LINK/V2-1[24] or any STM32 Nucleo board (which already has an ST-LINK/V2-1 built-in). The ST-LINK/V2-1 is connected to the STM32 chip through the **S**erial **W**ire **D**ebug (SWD) interface by a two-wire interface consisting of a clock (*SWDCLK*) and a data line (*SWDIO*).

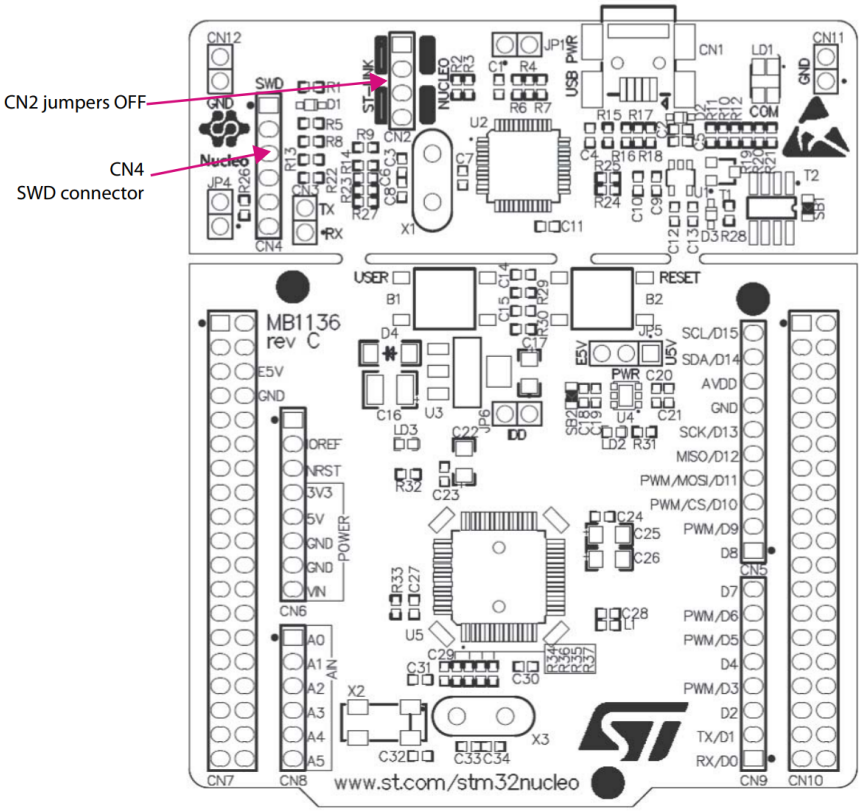


Figure 39: Using ST-LINK/V2-1 to program the STM32 chip on an external application (ST, 2020, p. 19).

Per default, the ST-LINK/V2-1 is connected to the in-built STM32 chip on the STM32 Nucleo board. For flashing and debugging the STM32 chip on an external application, the two jumpers from *CN2* are removed as illustrated in Fig. 39. Finally, connect the application to the *CN4* debug connector according to Fig. 40.

[24]ST- LINK/V2-1: https://www.st.com/en/development-tools/st-link-v2.html

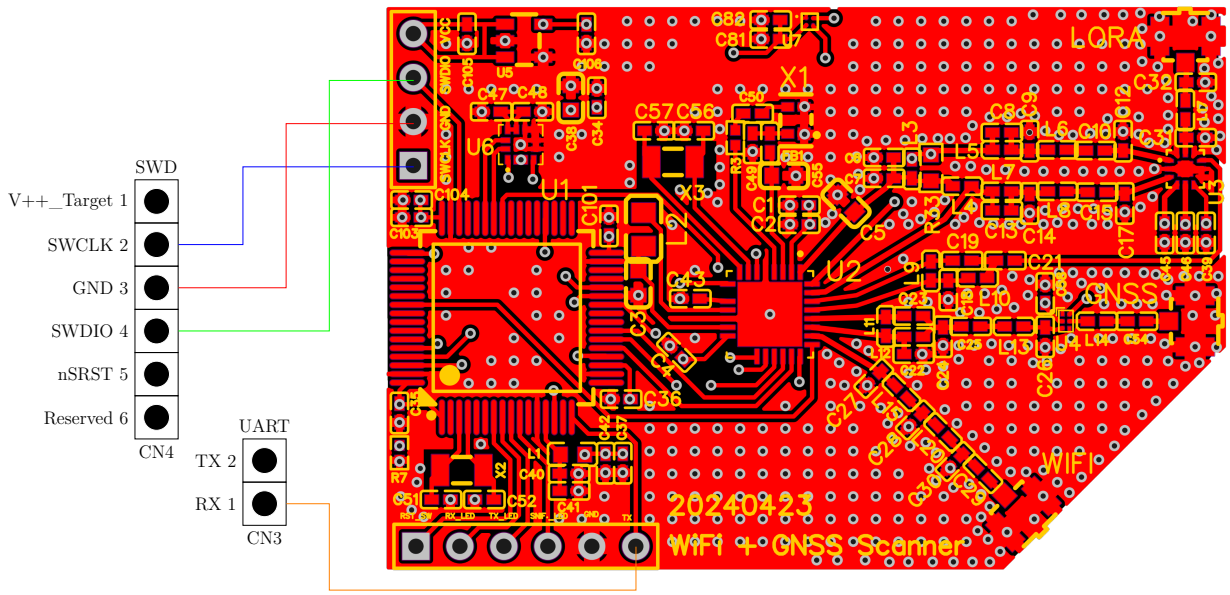


Figure 40: Diagram showing the connection between ST-LINK/V2-1 and a STM32 chip.

For communicating with the STM32 chip over UART, we use the *UART2* interface available on *PA2* and *PA3* of the STM32 chip. To enable the UART communication between the ST-LINK/V2-1 and an external application, the related solder bridges *SB13* and *SB14* must be OFF. *SB62* and *SB63* must be ON if using an ST-LINK that is part of a Nucleo board (ST, 2020, p. 25).

The ST-LINK/V2-1 is connected to the computer through USB, and the STM32 chip is powered through the ST-LINK/V2-1. The STM32 is flashed using Visual Studio Code or the STM32CubeProgrammer[25], which is a software tool to flash and debug STM32 microcontrollers. Here, the STM32 can be debugged, which is done by setting breakpoints in the code and running it step by step.

Fig. 41 shows the interface between the STM32 microcontroller and the user's computer if the tracker needs to be debugged.

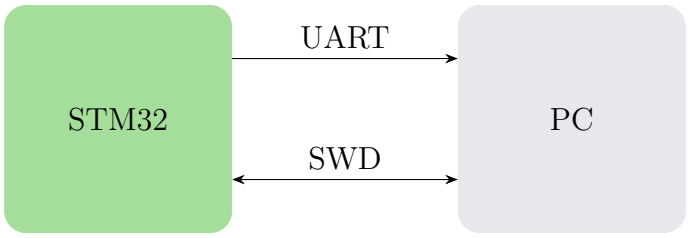


Figure 41: Overview of the connection between STM32 and the host computer.

[25]STM32CubeProgrammer: https://www.st.com/en/development-tools/stm32cubeprog.html

4.5.2 Code development with barebone transceiver firmware

This section describes our initial attempt at developing a device application for the LR1110 chip. When developing an application for the LR1110 chip, one has to decide between two main approaches. The first option is to go with the barebone transceiver firmware. This firmware offers complete control over the chip's operations but at the cost of significantly increased programming effort and expertise to leverage its full potential. This control extends to setting power output levels, accessing interrupt registers for error tracking (such as TCXO calibration status), and complete control over the code execution sequence. Once installed, communication with the LR1110 chip from the STM32 is done with opcodes through SPI. These commands are documented in the user manual for the LR1110 transceiver firmware (Semtech, 2023).

Conversely, the alternative approach involves installing the LoRa Basics Modem software package atop the transceiver firmware. This software layer operates on the LR1110 hardware, implementing and exposing a high-level API that incorporates pre-built support for LoRaWAN protocols and device management features. While this option simplifies interactions with the transceiver and accelerates development by abstracting low-level details, it sacrifices the control provided by the barebone firmware.

Given our project's focus on minimising power consumption, it is desirable to control every aspect of the chip's behaviour.

4.5.2.1 Establishing communication with LR1110

The STM32 and LR1110 communication is through SPI (see Fig. 42). The LR1110 exposes a low-level API, which allows the STM32 to communicate with the LR1110 through a set of SPI commands and responses.

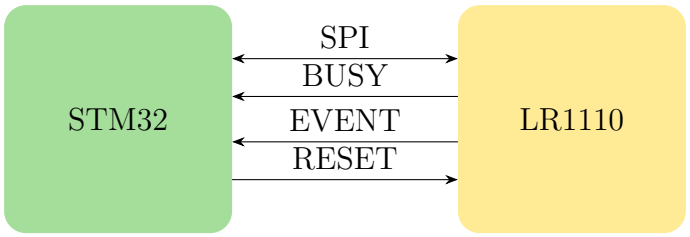


Figure 42: Interface between STM32 and LR1110.

The *BUSY* signal is used as a handshake to indicate if the LR1110 is ready to accept a command. Therefore, it is necessary to check the status of *BUSY* before sending a command.

The *EVENT* signal serves as an output from the LR1110, indicating to the STM32 microcontroller that the device has pending asynchronous event data. These events can encompass various types, such as LoRaWAN stack activities, GNSS updates, or Wi-Fi events. The signal is crucial in notifying the STM32 about new data arrivals or significant updates from the LR1110 module, enabling timely processing and response within the embedded system. The

STM32 must use the `GetEventSize(...)` command to determine the event size and `GetEvent(...)` command to retrieve such data.
The *EVENT* signal stays high until all events are cleared. The STM32 must clear all the events before returning to sleep mode. This prevents the hit from missing the rising edge of the *EVENT* signal when there is a new event. Furthermore, it saves several µA of current consumption when the *EVENT* signal is kept low by the LR1110.

Fig. 43 shows an example of a Wi-Fi passive scan transaction. The STM32 sends a command to the LR1110 to scan for Wi-Fi APs. The LR1110 scans the Wi-Fi APs and stores the data in its internal memory. When the scan is complete, the LR1110 sets the *EVENT* signal high to indicate data to be read. The STM32 then reads the data from the LR1110 with `GetEventsize(...)` and `GetEvent(...)`, which then clears the *EVENT* signal.

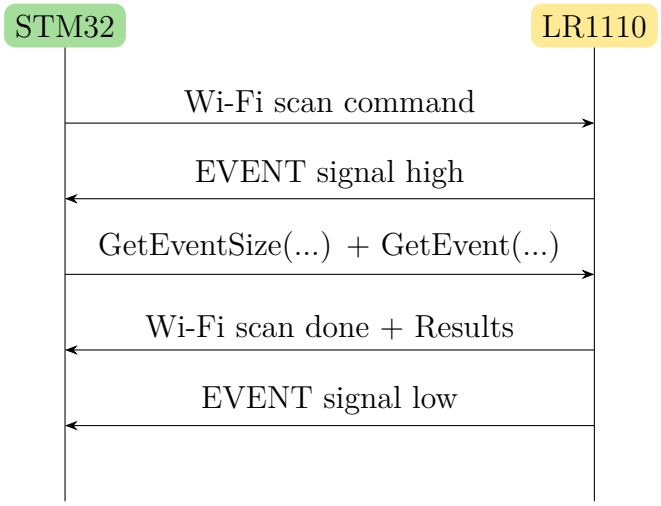


Figure 43: Example of Wi-Fi passive scan transaction. The transactions are read from top to bottom.

Some SPI commands generate an immediate response by the LR1110 and, therefore, do not generate any *EVENT* signal, such as the command `RequestTX(...)` as shown in Fig. 44.

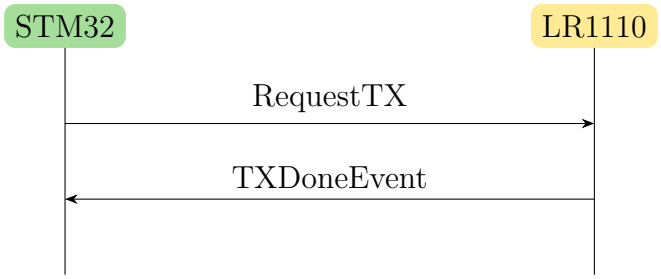


Figure 44: Example of a transaction which does not generate a *EVENT* signal. The transactions are read from top to bottom.

4.5.2.2 Write commands to LR1110

The STM32 initiates write commands (see Fig. 45) by pulling the *SS* line low. Activation of the *BUSY* pin occurs automatically upon the falling edge of the *SS*, signalling that the LR1110 chip is ready to receive a command. A 16-bit opcode (*Op0* and *Op1*) is dispatched through the *MOSI* pin, followed by the necessary arguments (*Arg0*, *Arg1*, etc.). When sending the *Op0* and *Op1* commands, the LR1110 chip replies through the *MISO* pin with the two status registers *Stat1* and *Stat2*, which contains the status of the previous write command. If arguments are sent, the LR1110 chip will reply with interrupt registers (*IrqStat*), which can be used for further debugging. Once the LR1110 completes processing the command, the *BUSY* signal is disengaged, signalling readiness to accept another command.

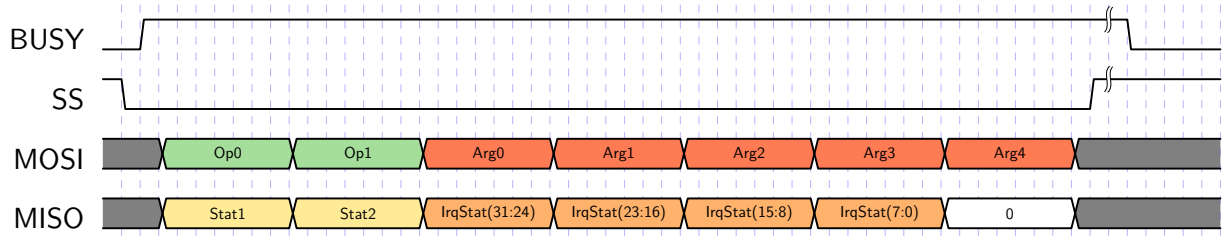


Figure 45: Timing diagram of a SPI write command between STM32 and LR1110.

4.5.2.3 Read commands from LR1110

Specific read commands, such as Wi-Fi scans, extract processed data from the LR1110 chip. Fig. 46 shows an example of reading data from the LR1110. The STM32 initiates read commands the same way as write commands. However, after completion, the STM32 waits for a falling edge on the *BUSY* signal, after which it pulls the *SS* signal low. When the *BUSY* goes high, the host can retrieve the data by transmitting NOPs (`0x00` bytes) to extract the data sequentially. Here, *Stat1* refers to the current command, and *Rsp0* and *Rsp1* demanded data.

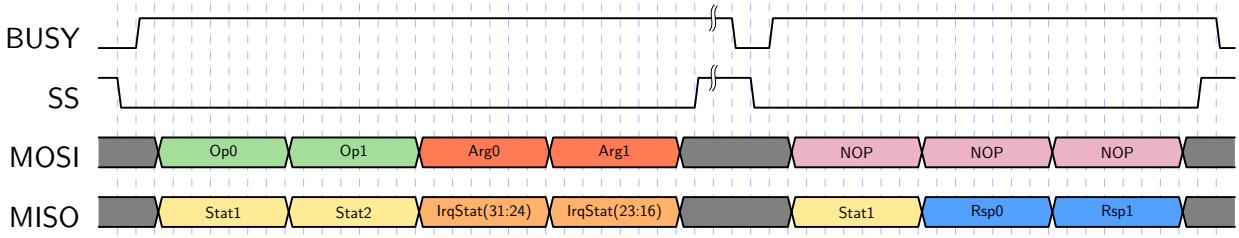


Figure 46: Timing diagram of a SPI read command between STM32 and LR1110.

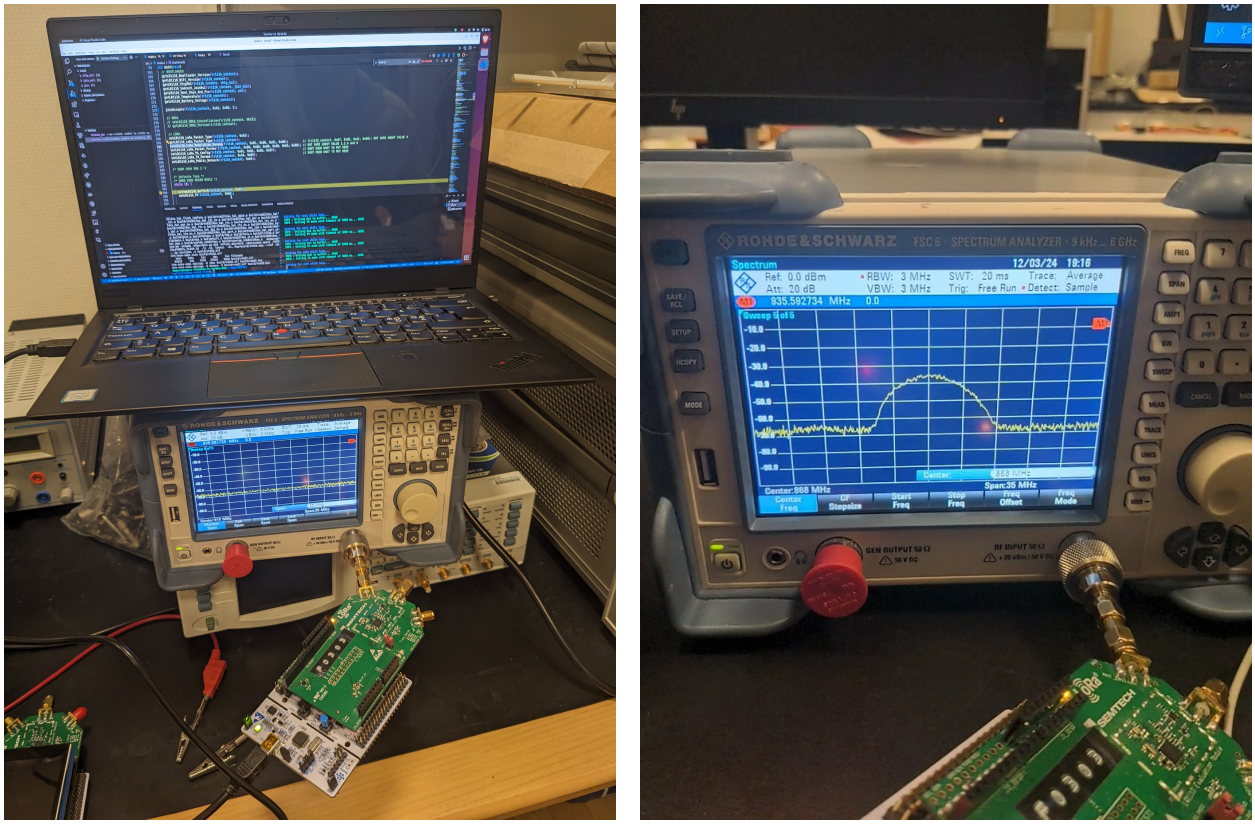
4.5.2.4 Results and challenges with the transceiver firmware

Over the first two months of the project period, using the transceiver firmware, we successfully implemented communication and error handling with the LR1110 chip. This allowed us to extract the security keys from the device used for authentication on the LoRaWAN

network, enabling us to add the device on TTN and Semtech's Join Server and perform a successful calibration procedure; An procedure essential for the chip to establish wireless communication with a stable frequency.

Furthermore, we successfully configured the development board to scan for Wi-Fi access points and perform unassisted GNSS scans. We then created correct geolocation payloads with either AP MAC addresses or NAV messages. The power amplifiers were correctly initiated and powered up, and we could set the necessary LoRa parameters for sending a LoRa package.

However, problems arose when attempting to join the LoRaWAN network. Initially, we believed our primary issue lay somewhere in the code, preventing us from sending a LoRa packet. After many days of debugging, repeatedly reviewing the manual, and testing the antenna output with a spectrometer, we detected a signal at the correct frequency, giving us hope after much frustration (see Fig. 47). Additionally, we observed some activity from the device on TTN. However, this led us to discover a more significant problem: we received MIC errors when the LoRa packet was received on TTN. After extensive investigation, we realised that the transceiver firmware's lack of a LoRa stack made it unable to encrypt and construct a correct LoRa packet.



(a) Picture showing the test setup of determining if any LoRa signal is being broadcast by the development board.

(b) A spectrum analyser showing a signal being sent with a peak frequency at around 868 MHz, the frequency LoRaWAN should communicate on.

Figure 47: A spectrum analyser test showing a signal with the correct peak frequency, thereby confirming the ability of our code and development board to send out a LoRa signal.

Semtech previously offered a pure LoRa stack that could be implemented on top of the transceiver firmware, but this has been discontinued in favour of the LoRa Basics Modem software. According to an infographic by Semtech, implementing a LoRaWAN stack that can correctly construct LoRaWAN packets that meet the requirements of the selected LoRaWAN version would take an estimated 3-6 months (Semtech, 2020).

Therefore, we felt compelled to change our approach, discarding most of the code base we had spent two months developing and transitioning to Semtech's LoRa Basics Modem software. Although this decision meant giving some control over how the chip utilises its resources, it was necessary to proceed with our project.

4.5.3 LoRa Basics Modem implementation

The LoRa Basics Modem library is still being actively developed, and more features will therefore arise in the future.

Because most of the basic functions, e.g. scanning for Wi-Fi APs, are already written, a simple system of scanning and sending data over LoRaWAN can be set up. The downside, as mentioned before, is that the control of this integration is much less controllable - maybe some code is unnecessary for our application and, therefore, wastes precious battery power.

The following describes the implementation of geolocating. It includes the implementation of Wi-Fi scanning, SV scanning, LoRa communication, and other notable features, as well as power-saving features.

4.5.3.1 End device setup

Before any geolocation processes can commence, the STM32 chip must be properly configured. This setup begins by disabling any interrupts. After this, the system clock is initialised, alongside **G**eneral-**P**urpose **I**nput/**O**utput (GPIO) pins, timer, SPI, and I^2C protocols, which are essential for communication with other components on the board. Following this, both the LR1110 chip and the accelerometer are configured according to their specific requirements. For the accelerometer, this includes setting its ODR, high-pass filter, and the interrupt for detecting the movement of the tracker.

After configuring these components, the scan mode and LoRaWAN communication need to be set up. The tracker sends a Join-request message to the LoRaWAN network and waits for a Join-accept response. To conserve power, the tracker enters sleep mode (Stop 2 mode) whenever it is idle, waking up only to process new information, such as downlink messages from the network or different interrupts.

4.5.3.2 Wi-Fi scanning

Scanning of Wi-Fi APs is done by setting the Wi-Fi channels and the time to scan each channel. The LR1110 will scan for Wi-Fi AP's and store the data in the internal memory until the STM32 chip is ready to receive the data. We are only sending the Wi-Fi AP and not the RSSI value (view Tab. 1 for details about the payload format). Sec. 5.1.1 shows that the accuracy is sufficient without RSSI values. Sending fewer bytes will improve the battery lifetime.

The LR1110 is configured to conduct Wi-Fi scans, stopping after detecting 10 Wi-Fi AP or hitting a time-out. Upon scan completion, the device transmits the MAC addresses of the 5 AP with the highest RSSI values via LoRaWAN. To optimise performance, we must test and determine the relationship between the number of APs, accuracy, and power consumption. This will enable selecting the optimal number of APs for efficient operation.

4.5.3.3 GNSS scanning

The LR1110 is configured to scan for GPS and BeiDou SVs. The scanning process involves setting the scan time and initiating the scan. During this period, the LR1110 chip actively searches for SV signals within the specified duration. Once the scan is completed, the LR1110 processes the received signals to extract relevant SV information. After the LR1110 finishes processing the scan data, it is sent to the cloud solver.

4.5.3.4 LoRaWAN communication

Sending data from the Wi-Fi and GNSS scans to the cloud is possible through LoRaWAN. The data is sent to the LoRaWAN Application Server, which then decrypts the payload and forwards it to the cloud solver for processing. The Application Server receives the response in JSON format, which is left to the user to decide how to display it.

As transmitting over LoRaWAN consumes significant power, a geolocation scan is only transmitted when the device has been moved recently. This logic of exclusively transmitting location data upon detecting movement is illustrated in Fig. 48.

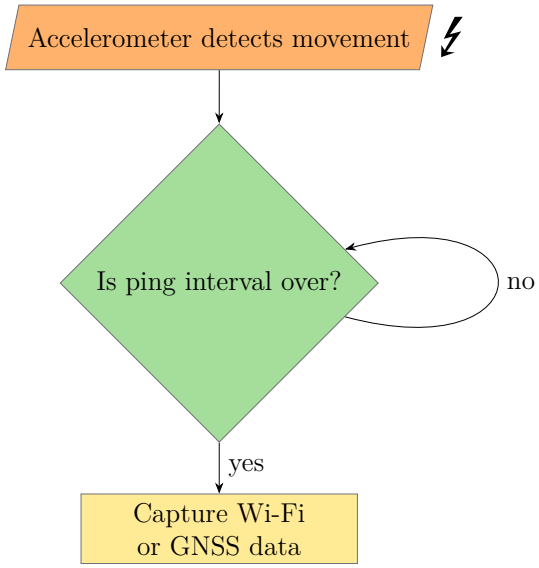


Figure 48: Flowchart diagram of the accelerometer in action. When the accelerometer detects a movement, it sends an interrupt to the STM32 chip, which then scans for nearby Wi-Fi APs or GNSS SVs.

In case the accelerometer has not detected any movement in the period specified in the Wi-Fi or GNSS ping interval, the device does not scan for Wi-Fi APs and GNSS SVs. This would mean the device has not moved, and an additional scan for location is unnecessary. Instead, it goes back to sleep and checks again after the Wi-Fi and GNSS ping interval has ended.

4.5.3.5 Uplink LoRaWAN communication

Tab. 10 presents the uplink commands that the tracker can transmit, detailing the **F**rame **Port** (FPORT) it sends to and the corresponding payload sent with each command.

Table 10: Uplink commands send from the tracker.

FPORT	Usage	Payload
2	Keep alive information	Temperature and battery voltage (see Tab. 11)
3	Low battery voltage warning	Battery voltage
10	Initial tracker configuration	Scan interval, device class, Wi-Fi and GNSS status, temperature, and battery voltage (see Tab. 12)
192	GNSS scan results to be forwarded to LoRa Cloud	Multiple NAV messages (see Tab. 3)
197	Wi-Fi scan results to be forwarded to LoRa Cloud	See Tab. 1
198	GNSS scan results to be forwarded to LoRa Cloud	See Tab. 3

FPORT 2 is used as a *keep alive* port for the tracker to periodically send a small portion of info to the LoRaWAN gateway to determine that it is still "alive" and to inform the web application about the tracker's current temperature and battery voltage (used to calculate battery percentage) (see Tab. 11).

Table 11: Keep alive information. In total, 2 bytes are sent via FPORT 2.

Bits	(15:8)	(7:0)
	Temp	BatVol

- *Temp* defines the temperature of the tracker [°C]
- *BatVol* defines the battery voltage of tracker [V]

Further information about the current state of the tracker can be sent if necessary.

The tracker uses FPORT 3 to inform the user that it is running low on battery (this is currently set to alert at a voltage below 3 V).

On the first Join-accept request after powering on or reset, information about the tracker's configuration and state is sent uplink to the Application Server and forwarded via a webhook to the web application. This information is described in Tab. 12.

Table 12: Upon recieving a Join-accept for the first time after a reset, the configuration is sent to the web application to show the state of the tracker. In total, 6 bytes are sent via FPORT 10.

Bits	(47:40)	(39:32)	(31:24)	(23:16)	(15:8)	(7:0)
	ScanPer1	ScanPer2	DevClass	ScanOpt	Temp	BatVol

- *ScanPer* defines the scanning period of both Wi-Fi and GNSS [min]
- *DevClass* defines the LoRaWAN device class (see Sec. 2.2.2.4)
  - `0`: Class A
  - `1`: Class B
  - `2`: Class C
- *ScanOpt* defines the scanning option
  - `0`: No scanning
  - `1`: Scanning for Wi-Fi APs
  - `2`: Scanning for GNSS SVs
  - `3`: Scanning for both Wi-Fi APs and GNSS SVs
- *Temp* defines the temperature of the tracker [°C]
- *BatVol* defines the battery voltage of tracker [V]

The maximum ping interval is 65 535 min (45.6 d) as this is chosen to be long enough and fit inside 2 bytes.

If using Wi-Fi, each found MAC address is appended and sent as one long payload (see Tab. 1) on FPORT 197.
If using GNSS, the device sends the NAV message (see Tab. 3) on FPORT 198 or 192 depending on if one or multiple frames have been captured.

4.5.3.6 Downlink LoRaWAN communication

Tab. 13 shows the downlink commands that the user can send to the end device. The user can configure the tracker from abroad by sending a string of bytes on FPORT 1

Table 13: Downlink commands the user can send to the tracker from the web application.

FPORT	Usage	Payload
1	Tracker configuration	See Tab. 14

Here, the ping interval of both Wi-Fi and GNSS is set, along with the LoRaWAN device class and the scan option.

Table 14: Configuration of the end device chosen by the user. In total, 3 bytes are sent via FPORT 1.

Bits	(23:16)	(15:8)	(7:4)	(3:2)	(1:0)
	ScanPer1	ScanPer2	RFU	DevClass	ScanOpt

4.5.4 Local tracker configuration

The end device can also be configured by the Hall effect sensor. Whenever the accelerometer detects a movement, the Hall effect sensor will be powered on and scan for nearby magnets for a short period. The tracker is set into configuration mode if a magnet is detected within the period. As of this time of writing, this mode has not been completed in the code. The principle is shown in Fig. 49.

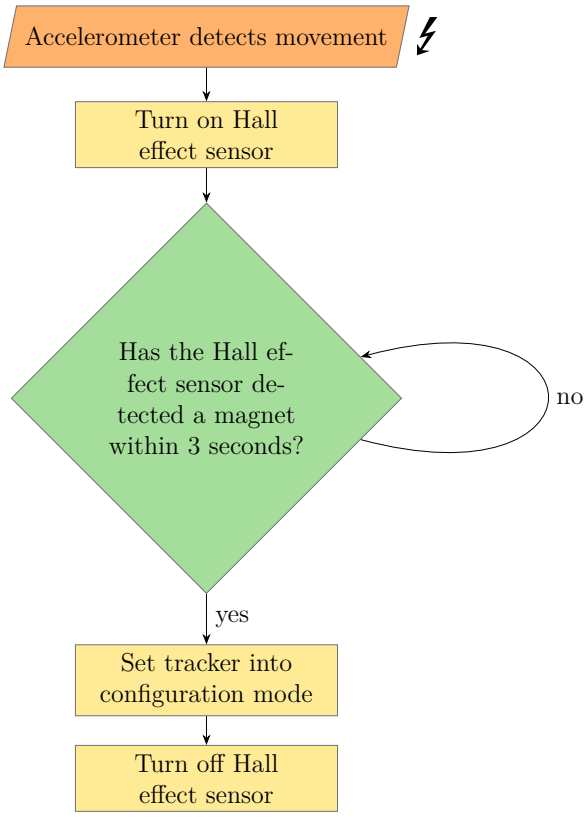


Figure 49: Flowchart diagram of the Hall effect sensor in action.

4.5.5 End device memory usage

Tab. 15 shows the memory usage of the tracker's firmware. As the STM32 microcontroller has an internal flash of 1 MB, there is enough space for further features if needed. The amount of used RAM is also minimal and does not need to be upgraded.

Table 15: Memory used by the firmware.

Memory region	Used size	Total size	%age used
RAM	9.8 kB	96 kB	8.98%
Flash	226.5 kB	1 MB	21.60%

4.6 Application design

One predominant requirement from our interviews was a simple device setup process and an easy-to-use application for presenting the data. Despite the growing adoption of LoRaWAN-enabled devices, network setup remains considerably more intricate than devices that utilise Wi-Fi or cellular networks for data transmission, which primarily work out of the box. Wi-Fi devices require access to a Wi-Fi AP and a network password. Cellular devices only need a SIM card, and the technical part of joining the network is left to the network provider. LoRaWAN-enabled device, on the other hand, entails several additional considerations. One must consider provisioning, obtaining the security keys authenticating the network, and determining the right network protocol version and regional requirements.

Let us break down the process of adding an LR1110 end device to TTN, setting up the integration with LoRa Cloud, and displaying the geolocation data.

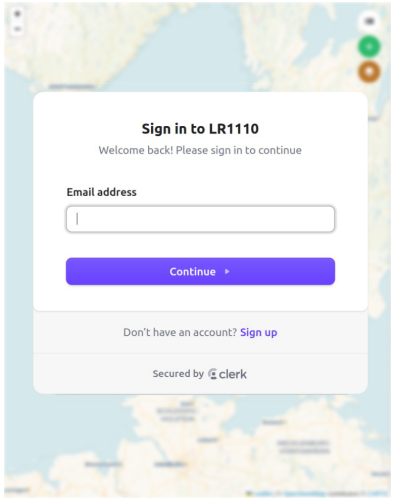
1. Extract end device security keys from the chip (DevEUI, PIN and JoinEUI).
2. Sign up as a user and create an application on LoRa Cloud[26].
3. Sign up as a user and create an application on TTN.
4. Change bindings for LoRa Cloud application to allow Join-requests from TTN.
5. Claim device on the Join Server on LoRa Cloud with DevEUI and pin.
6. Claim device on the Modem & Geolocation Services on LoRa Cloud.
7. Register end device on TTN with correct frequency plan, LoRaWAN version, regional parameters, JoinEUI and DevEUI.
8. Create an API token on LoRa Cloud Modem & Geolocation Services.
9. Enter the LoRa Cloud API key on TTN to enable the integration with LoRa Cloud Modem & Geolocation Services.
10. Choose an application to display geolocation data.

[26]LoRa Cloud: https://www.loracloud.com/

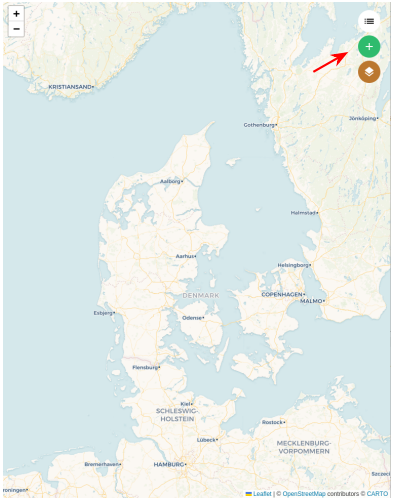
11. Setup integration between the Application Server and choose geolocation display application.

Looking at the above list, one can see that setting up an LR1110 device is complex and technical. By simplifying this process through our proposed application, we aim to provide a proof-of-concept of a more streamlined and user-friendly user experience, reducing technical complexity and enhancing overall usability. Let us break down the process of adding an LR1110 end device to TTN, setting up the integration with LoRa Cloud, and displaying the geolocation data with our application (see Fig. 50).

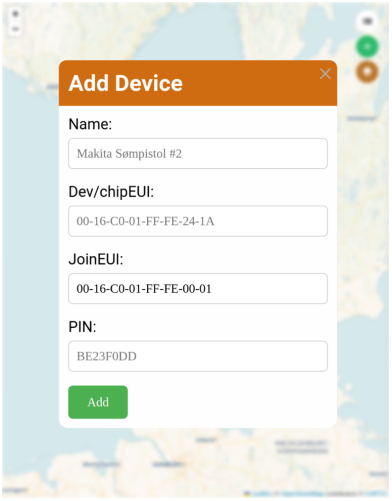
1. Navigate to our application and sign up for an account (see Fig. 50a)
2. Add the device by pressing the add button (see Fig. 50b, choose a name, and type in the DevEUI, JoinEUI and PIN found in the packaging (see Fig. 50c).



(a) Sign-in dialogue window when opening the web application. Either log in or sign up as a user.



(b) Main screen showing map application, with *add device* button pointed out.



(c) Dialog window when adding a new device. Choose a name and type in DevEUI, JoinEUI, and the PIN found on the packaging.

Figure 50: Illustration of adding an end device on the web application.

4.6.1 Server-side architecture

Our application's server-side architecture is built using Express.js[27], a minimalist web framework for Node.js, in conjunction with SQLite[28], a lightweight database library. The server, written in JavaScript, runs on Node.js[29], a JavaScript runtime environment that enables developers to execute JavaScript on the server side. This allows for a cohesive development environment where JavaScript is used across both client and server sides.

[27]Express.js: https://expressjs.com/
[28]SQLite: https://sqlite.org/
[29]Node.js: https://nodejs.org/en

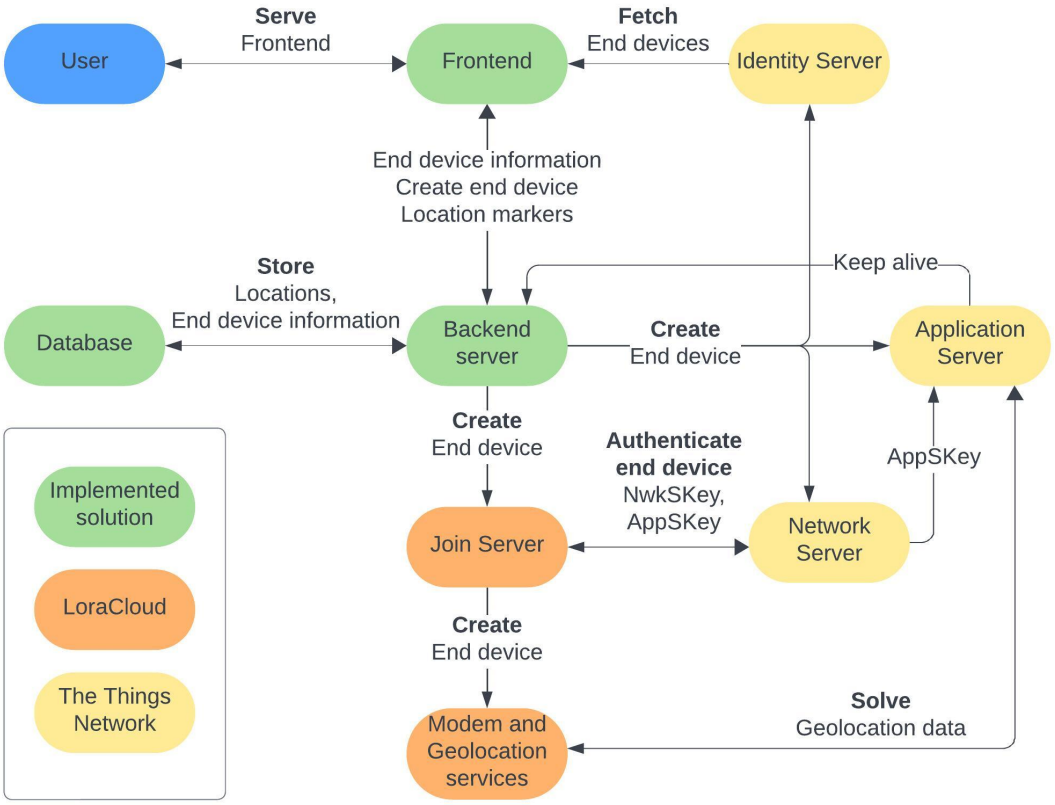


Figure 51: Diagram illustrating the primary dataflow to, from and within the web application.

Express.js is a Node.js web application framework that simplifies the development of robust and scaleable server-side applications handling HTTP requests and responses. It is used in the application to set up a web server that exposes an API, parsing incoming request bodies in JSON format and facilitating seamless communication with the database.

SQLite serves as our database management system. It is a disk-based relational database that is self-contained and serverless, storing its entire database in a single file on disk and integrating directly into the application without needing a separate server process. This design simplifies the architecture and reduces overhead, making SQLite an ideal choice for small — to medium-sized applications like ours.

The dataflow between the different components of our system is depicted in Fig. 51.

The database consists of three key tables, each storing specific types of information. The structure can be seen in Tab. 16

Table 16: The data structure of the database for the web application.

(a) The important columns of the **GeolocationSolve** table.

Name	Type	Description
deviceID	STRING	DevEUI of end device
geocode	JSON	Latitude and longitude of location marker
source	STRING	Wi-Fi or GNSS solve
accuracy	FLOAT	Accuracy of location (calculated by Semtech)
createdAt	DATETIME	Location marker created at
updatedAt	DATETIME	Location marker updated at

(b) The important columns of the **TrackerInformation** table.

Name	Type	Description
deviceID	STRING	DevEUI of end device
temperature	INTEGER	Temperature of end device
updateInterval	INTEGER	Interval between geolocation scans
batteryStatus	INTEGER	Battery status of end device
wifiStatus	BOOLEAN	Is Wi-Fi scan turned on
gnssStatus	BOOLEAN	Is GNSS scan turned on
loraWANClass	INTEGER	LoRaWAN class of device

(c) The important columns of the **KeepAliveGatewayLocation** table.

Name	Type	Description
deviceID	STRING	DevEUI of end device
gatewayID	STRING	Gateway ID
geocode	JSON	Latitude and longitude of gateway

The *TrackerInformation* and *GeolocationSolve* tables provide the user with the end device's status and location markers. The *KeepAliveGatewayLocation* table is used for testing to determine the range of LoRa signal reception by gateways from our devices.

Besides storing tracker information and location markers, the backend communicates with LoRa Cloud and TTN, claiming and unclaiming the device on the appropriate servers. The documentation for the endpoints exposed by the API can be found in the ReadMe for our web application.

4.6.2 Client-side architecture

Many different frameworks or libraries could have been used for the client-side of our application. React[30] was chosen because of its prominence as one of the most used web technologies (Overflow, 2023) offered extensive availability of libraries, online resources and community support.

[30]React: React: https://react.dev

In web application development, it is generally unnecessary to reinvent the wheel, as doing so often increases development times, leading to poorer performance and security vulnerabilities. For displaying the map, we selected Leaflet[31], the leading open-source JavaScript library for mobile-friendly interactive maps. While we considered using Google's map service for its familiarity (a preference noted in our interviews), we ultimately chose Leaflet, as it is the free option. The empirical data showed that price was a significant factor for potential users. The similarity of user experience between Leaflet and Google Maps played a crucial role in prioritising cost savings over familiarity.

As our development application is deployed on the web, authentication was implemented for security reasons. This will most likely also be a feature in a final commercial product. Again, to not reinvent the wheel and to make sure that authentication was implemented securely, it was decided to use Clerk[32], a user management platform that integrates smoothly with React, providing authentication and user management.

The application is a component-based **S**ingle **P**age **A**pplication (SPA). The structure can be seen in Fig. 52. The major components will be described below.

Figure 52: Tree structure of the web application components.

`Index.js` serves as the main entry point for the application. It renders the Clerk sign-in page if the user has not signed in and the root component when the user has signed in.

`App.js` is the root component. It defines the overall layout and structure of the application, including which components get rendered, where they are placed and how they interact. It is also in charge of top-level state management, keeping track of the state of the different

[31] Leaflet: https://leafletjs.com/
[32] Clerk: https://clerk.com

global variables in the application and passing functions down to child components. The layout can be seen in Fig. 50a.

`MapComponent.js` serves as the entry point for users. This component is responsible for fetching the map data, rendering the chosen map tiles, calculating lines between location markers, and showing these in conjunction with the markers.

`AddDeviceMenu.js` is in charge of rendering the *add device* menu, which can be seen in Fig. 50c. Besides rendering the **U**ser **I**nterface (UI), the component is also responsible for formatting the strings, preventing input errors, importing API keys, and posting *create device* requests to the server.

`SettingsMenu.js` is in charge of rendering the settings menu, which can be seen in Fig. 53. In this menu, the user can see all the essential information about each tracker (battery charge, temperature and interval between each ping). They can also choose to rename the device, show the current and former location, click the gear icon on the right side of the tracker to change the settings of a specific tracker or delete the tracker, subsequently deleting it on TTN and LoRa Cloud.

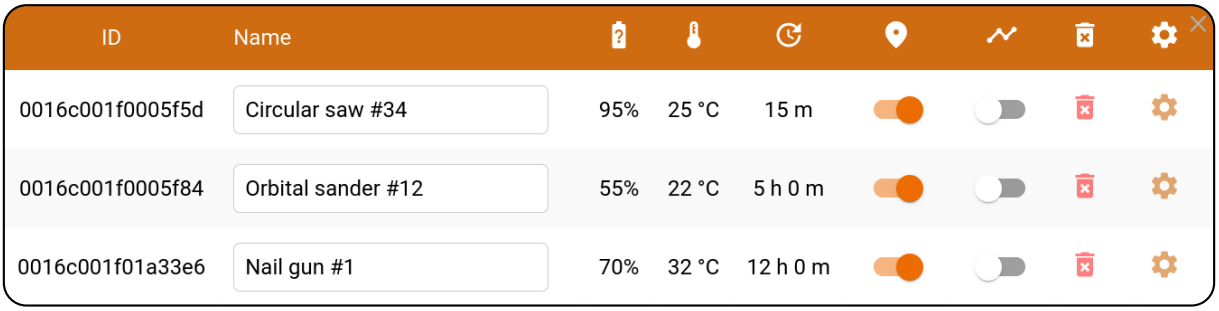


Figure 53: Dialog box of all added trackers; the user can observe the battery percentage and tracker temperature here. The user may disable the tracker on the map and choose to view the history of the tracker's location. Renaming and removing it from the application is also possible.

`ConfigeDeviceDialog.js` is responsible for rendering the menu that opens when the user clicks the gear icon. It can be seen in Fig. 54. The menu allows the user to change the ping interval for location scanning, the LoRaWAN class, and which scan function should be enabled. Besides rendering the tracker information, it is also responsible for notifying `App.js` to schedule a downlink on the Application Server if the device setting changes.

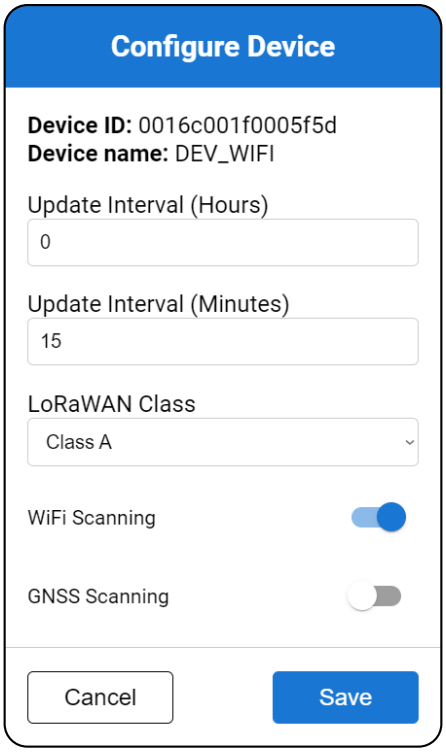


Figure 54: Dialog box where the user can configure the tracker. The user can choose between setting the ping interval and the LoRaWAN classes and choosing which method to use when scanning.

In conclusion, the web application successfully demonstrates a simplified process of adding LR1110-enabled devices to TTN, streamlining device management tasks effectively, and an enhanced user interface to display location markers and device information. Its React-based SPA architecture ensures navigation without interruptions, while its server design and API documentation make it easy to integrate with other applications.

However, there are notable areas for improvement before it can be turned into a commercial product. The reliance on fixed API endpoints tied to TTN limits flexibility. Moreover, the current necessity for users to manually input API keys and security certificates poses a significant technical barrier, defeating the purpose of creating an easier and less technical device management flow. A future iteration would enhance usability by automating these setup processes upon account creation.

The application is available for two weeks after the handin of this project on https://lr1110.hollandsvej.click/ (email: lr1110.frontend@gmail.com, password: LR1110frontend). It is running in a Ubuntu Desktop 22.04[33] virtual machine on a Proxmox Virtual Environment[34] server connected to the internet via a Cloudflare tunnel[35].

[33]Ubuntu Desktop 22.04: https://ubuntu.com/download/desktop
[34]Proxmox Virtual Environment: https://www.proxmox.com/en/proxmox-virtual-environment/overview
[35]Cloudflare Tunnel: https://www.cloudflare.com/en-gb/products/tunnel/

5 Test and validation of the design

This section presents the evaluation and validation of our solution through tests that we could perform within the limited time frame of this project. Detailed test scenarios and their results demonstrate how effectively our design meets the specified product requirements and pinpoint potential areas of improvement.

However, some unforeseen setbacks during the project, such as transitioning from transceiver firmware to LoRa Basics Modem library and the absence of a wire in PCB version 2, have impacted the depth of our testing. Additional tests would have enhanced our understanding and insights and likely improved our geolocation tracker's quality. These additional tests are discussed in Sec. 6.

Tab. 17 outlines the list of product requirements alongside corresponding test scenarios. Each test scenario aims to validate the specific requirement.

Table 17: List of product requirements and their corresponding test scenarios.

Category	Requirements	Test
Functional requirements	Accuracy of 20 m	1, 2
Performance requirements	LoRa range of 5 km in open terrain	3
	2 years battery life and 1000 geolocation positions on a single battery charge[36]	4, 5, 6, 7

5.1 Location accuracy

Accuracy, measured in meters, quantifies the disparity between a device's physical and reported locations. It serves as a critical metric in evaluating the precision of geolocation technologies and their practical application scenarios.

5.1.1 Test 1: Wi-Fi scanning

The accuracy of Wi-Fi scanning is assessed by positioning the tracker at various locations and obtaining location data from it. Since nearby Wi-Fi APs typically remain stationary, conducting multiple measurements at a single location does not yield meaningful variations. Instead, we deploy the tracker across multiple locations and record the resulting geolocation data. We derive statistical metrics from these datasets, such as average, minimum, and maximum values, to determine the accuracy of Wi-Fi AP-based geolocation.

20 results of collocations found by scanning for Wi-Fi APs has resulted in an average accuracy of 12.6 m (max: 16 m, min: 9 m), which is lower than the 20 m stated in requirement R6.

[36] LiPo battery: 1200 mAh, 3.7 V, 6 × 34 × 50 mm

5.1.2 Test 2: GNSS scanning

As of this writing, the tracker's GNSS accuracy is determined by the LR1110 Development Kit (Semtech, 2023) because the GNSS module on the custom board is not implemented correctly. When scanning for GNSS SVs on our custom PCB, we are unable to detect any SVs. The system returns a timeout, and not an error code, indicating that no SVs were identified during the scan period. Conversely, the same code operates correctly on our LR1110 development board, suggesting that the issue resides in the hardware of our custom PCB.

Scanning for GNSS SVs using the development board we achieved an accuracy averaging of around 15.5 m (max: 26.6 m, min: 5.9 m) among 22 readings in outdoor environment (see Fig. 55).

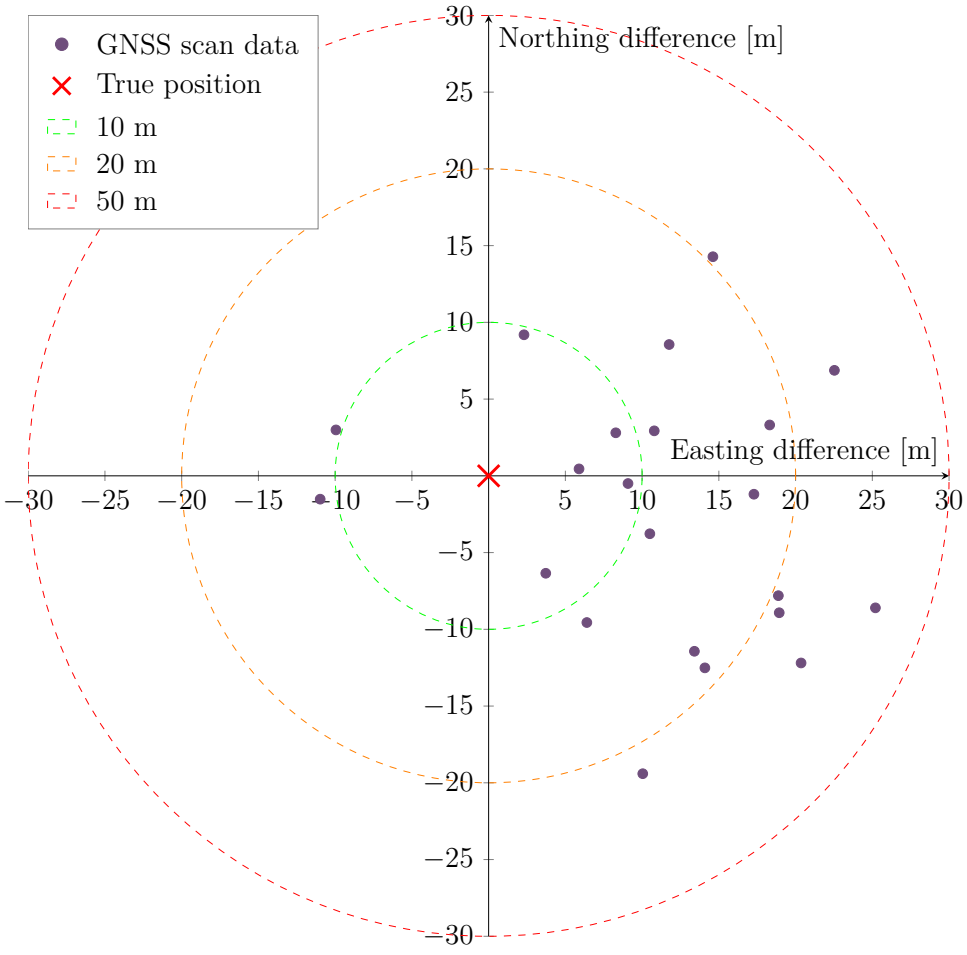


Figure 55: Scatter plot with GNSS scan data and the true position, along with circles representing 10 m, 20 m, and 30 m radii. Average accuracy: 15.5 m, max: 26.6 m, min: 5.9 m.

According to the results, the LR1110 chip can get a GNSS location with accuracy lower than 20 m, and therefore, in conjunction with test 1, fulfils requirement *R6*.

5.1.3 Test 3: LoRaWAN range and coverage

Testing the LoRaWAN network and coverage with the LR1110 tracker is essential to ensure reliable long-range communication. Our approach involves deploying the tracker in diverse environments and at varying distances from gateways to assess range and data transmission reliability. These tests enable us to evaluate the performance of our tracker and the LoRaWAN network, thereby determining whether the tracker meets requirement *R7*.

We were able to send LoRaWAN messages on TTN through city-environment of 2 km (Lyngby station to DTU Skylab) and 3.7 km (Drosselvej to Emdrup station). A test of open terrain range was carried out near Roskilde, where a range of 6.9 km was obtained (see Fig. 56), thereby confirming that our tracker meets requirement *R7*. Attempts were also made to establish a connection to the Helium network, but they proved unsuccessful.

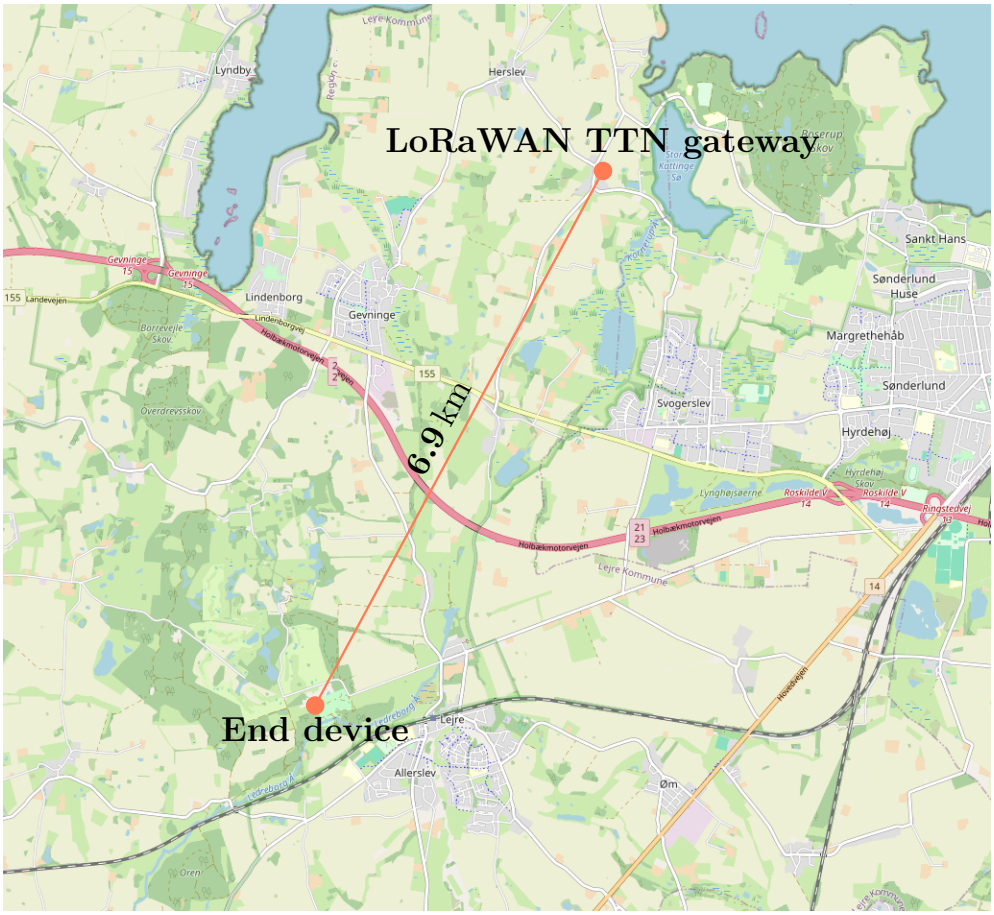
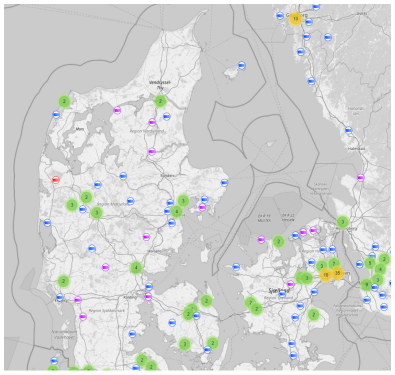


Figure 56: Map showing the longest LoRaWAN transmission with the custom PCB. The transmission was 6.9 km in the countryside of Roskilde.

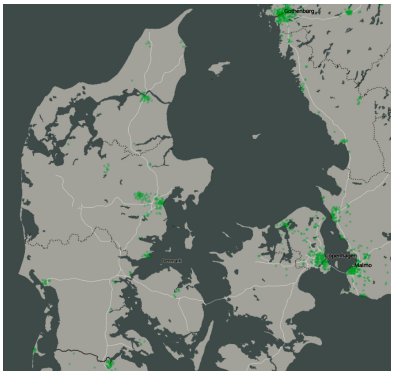
Field tests were conducted by travelling with the tracker around Lyngby, Copenhagen and Roskilde to assess the reliability of obtaining and maintaining a connection. The results revealed that achieving a stable connection is generally challenging due to sparse network coverage. The test also showed that establishing the initial connection was the most difficult

part, especially when travelling at higher speeds. Attempts to join a LoRaWAN network while travelling at 110 km h^{-1} were unsuccessful. However, once connected, maintaining the connection was somewhat easier. Although the gateway receives the Join-request message and replies, the tracker was unable to receive the accept. This was consistent for both our tracker and the development board. If the tracker joined the network before travelling at high speeds, it was able to keep the connection and transmit data. This limitation may hinder applications where the user wants to track assets travelling by car, train, or other high-speed means of transportation.

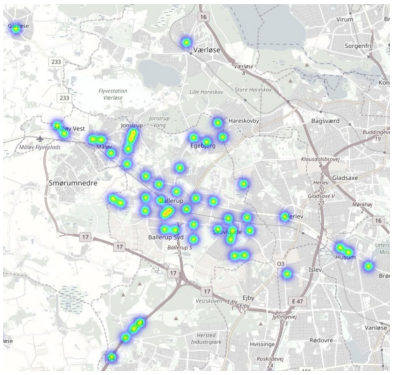
Fig. 57 displays the locations of gateways from three major LoRaWAN network operators in Denmark. The coverage information for Helium and TTN were obtained from their respective network coverage maps. Unfortunately, Cibicom has not provided an updated coverage map, citing rapid expansion outpacing their ability to update their maps, thus only a map of gateway placements in Ballerup was available. Despite requests for their latest map, no response was received during this writing.



(a) Map of Denmark showing the location of LoRaWAN gateways on TTN.



(b) Map of Denmark showing the LoRaWAN coverage of the Helium network.



(c) Map of Ballerup showing the LoRaWAN coverage of the Cibicom network.

Figure 57: Three maps showing the LoRaWAN coverage of three large LoRaWAN networks.

The three available maps indicate that coverage is rather limited, and our tests demonstrated that establishing a connection could be difficult even in areas with a high density of gateways. Fig. 58 below plots the placement of TTN gateways around DTU and includes radii of 2 km (gray) and 5 km (red) for each gateway.
Our tracker can connect within a 5 km radius in open terrain. However, given the densely urban nature of the area, a more conservative estimate would suggest that a 2 km range is more realistic. It is evident from the figure that there are potential dead spots, complicating asset tracking in the area.

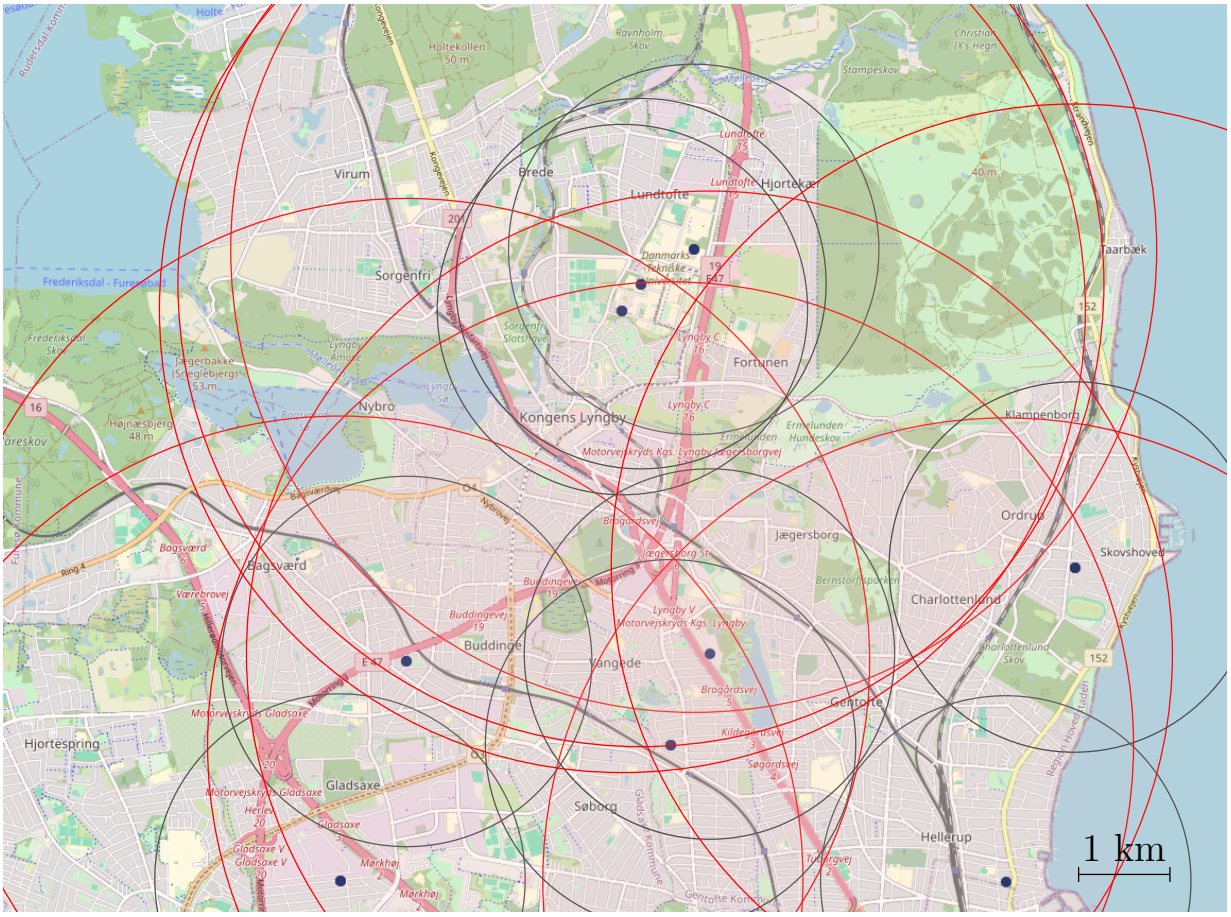


Figure 58: Locations of LoRaWAN gateways connected to TTN, with radii of 2 km (gray) and 5 km (red) mapped for each gateway.

5.2 Power consumption

As indicated by the project's title, a significant focus is to explore the feasibility and methods of developing an ultra-low power tracker. The following test will assess the power demands of the system and primary actions such as Wi-Fi and GNSS scans.

To measure the actual power consumption of the device, the *Power Profiler Kit II*[37] from Nordic Semiconductor has been used. For data visualisation the *Power Profiler* application from *nRF Connect for Desktop*[38] which is also developed by Nordic Semiconductor.

5.2.1 Test 4: Wi-Fi scanning

To evaluate the power consumption during Wi-Fi scanning, two tests were conducted.

[37]Power Profiler Kit II: https://www.nordicsemi.com/Products/Development-hardware/Power-Profiler-Kit-2
[38]nRF Connect for Desktop: https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-Desktop

In the first test, the device scanned for 5 Wi-Fi APs. As illustrated in Fig. 59, the scan started at time 0 and lasted approximately 3.3 s, with an average current consumption of 16 mA, resulting in a total charge consumption of 52.37 mC (14.44 mAh).

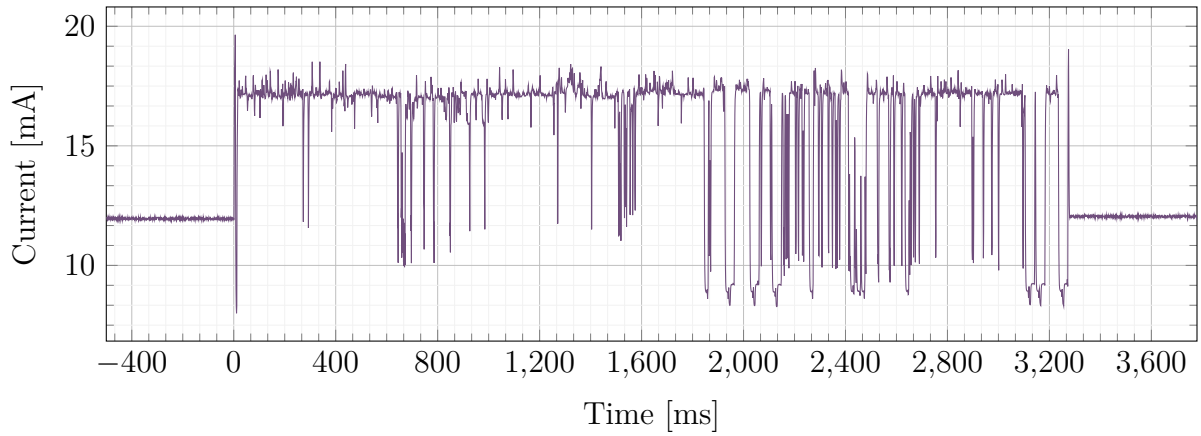


Figure 59: Power consumption during scanning for 5 Wi-Fi AP.
Average consumption: 16 mA, time: 3279 ms, charge: 52.37 mC (14.54 mAh).

The second test involved scanning for a single Wi-Fi AP, as shown in Fig. 60. In this case, the scan duration was around 1.1 s with an average current consumption of 17 mA, consuming a total of 18.69 mC.

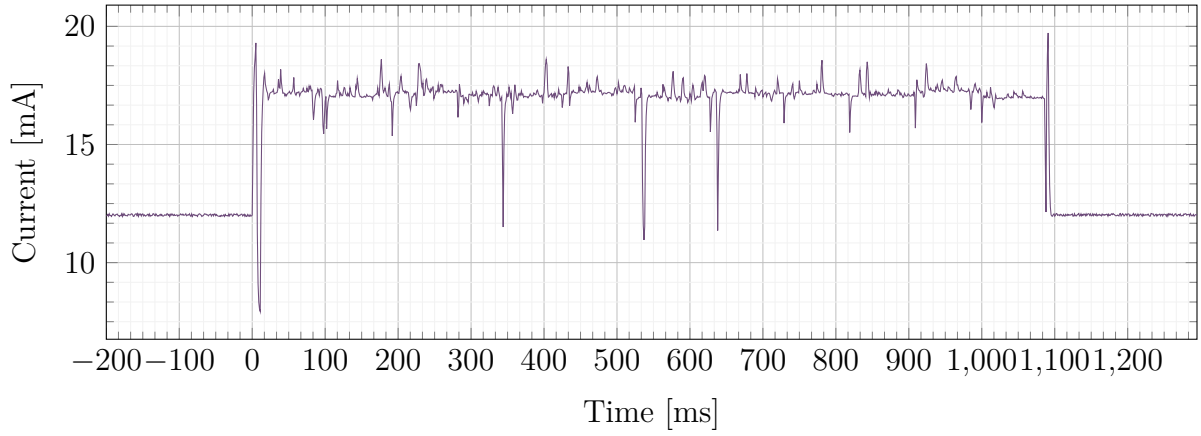


Figure 60: Power consumption during scanning for 1 Wi-Fi AP.
Average consumption: 17 mA, time: 1095 ms, charge: 18.69 mC (5.19 mAh).

These tests demonstrate that the duration and the number of Wi-Fi APs scanned significantly impact the power consumption of the device. Scanning for more APs results in higher power consumption, underscoring the importance of optimising scan parameters for ultra-low power applications.

5.2.2 Test 5: GNSS scanning

As mentioned in Sec. 5.1.2, we will utilise the LR1110 Development Kit to test GNSS scans. Since the development board is connected to an STM32 Nucleo board, it is crucial to carefully examine the power consumption to ensure that we only measure the chips required for performing a GNSS scan.
To accurately measure the power consumption of the STM32 Nucleo, the JP6 jumper must be removed (see Fig. 39) and an ammeter connected (ST, 2020, sec. 6.6, p. 23).

The results of a test scanning for 4 GNSS SVs are shown in Fig. 61. The scan period lasted approximately 17.2 s and consumed a total of 213.8 mC (59.38 mAh). Another test indicated that scanning for GNSS SVs and retrieving information from one more SV (5 in total) takes about 1 s longer, consuming approximately 14 mC (3.88 mAh) more power.

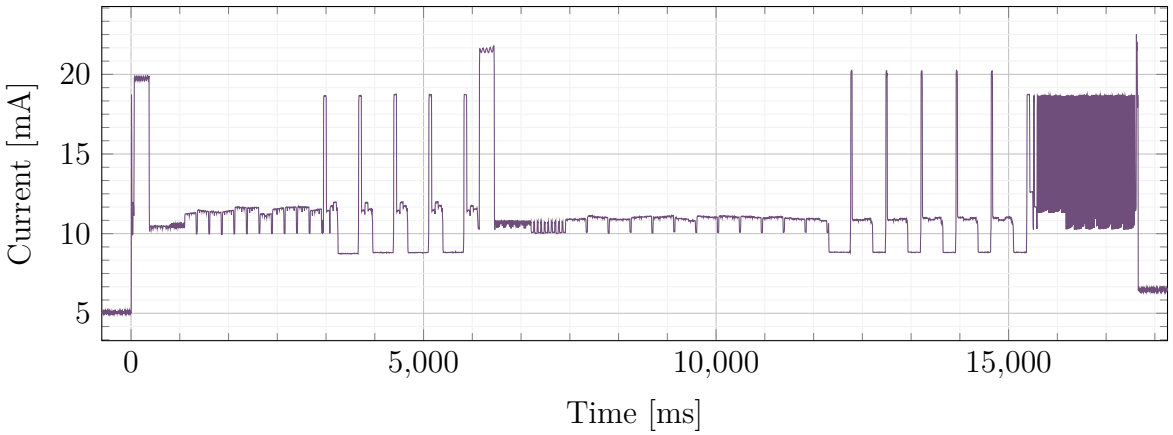


Figure 61: Power consumption during scan where 4 GNSS SVs were found. The ripple at the end is where the tracker retrieves the information of 4 SVs.
Average consumption: 11.6 mA, time: 17.2 s, charge: 213.8 mC (59.38 mAh).

The tests show that GNSS scanning consumes significantly more power compared to Wi-Fi scanning and that the power consumption increases with the number of scanned SVs. Further testing is necessary to determine the relationship between the number of SVs scanned, the accuracy of the geolocation, and the overall power consumption.

5.2.3 Test 6: LoRaWAN transmission

In this test, we want to investigate the power consumption characteristics during the transmission of data via LoRaWAN.

Fig. 62 shows the consumption profile during the transmission of 6 bytes via LoRaWAN.

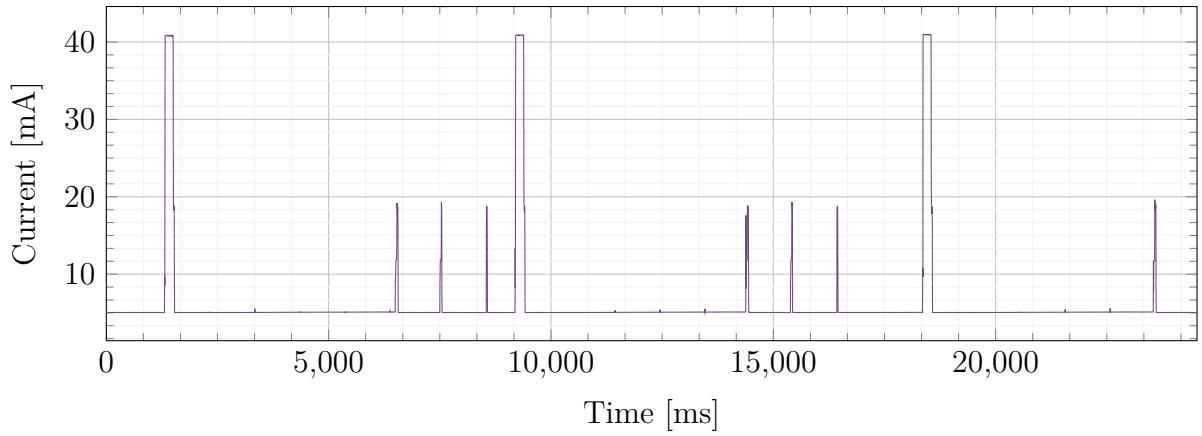


Figure 62: Power consumption during sending 6 bytes via LoRaWAN.
Average consumption: 6 mA, time: 24.5 s, charge: 148.59 mC (41.27 mAh).

The overall duration of the transmission is around 24 s, resulting in a total charge consumption of 148.59 mC (41.27 mAh). The data indicates that the transmission process is marked by three prominent current spikes, with the latter two always having three smaller spikes preceding them, suggesting a pattern can derived - possibly between the number of bytes and LoRaWAN class. However, the specific events corresponding to these spikes are not fully understood, and further testing and analysis are necessary to understand the pattern.

5.2.4 Test 7: Sleep

This test examines the power consumption of the tracker during sleep mode, a critical phase for conserving battery life, as the tracker is most likely spending most of its time in sleep mode.

Among the components on the board, several remain powered even when the tracker is in sleep mode. From the datasheets, the trackers theoretical idle power consumption is:

- Voltage regulator: 8 µA ([6], p. 1)
- Accelerometer (LIS2DE12): 3 µA ([4], p. 16)
- RF splitter (SKY13588-460LF): 5 µA ([8], p. 2)
- GNSS LNA (BGA524N6E6327): 0.2 µA ([7], p. 2)
- LR1110 1.5 µA ([1], p. 15)
- STM32L476 (Stop 2 mode) 1.5 µA ([2], p. 1)

Adding these power consumptions, the estimated current draw when the tracker is in sleep mode totals 19.2 µA.

This means that a battery with a capacity of 1200 mAh (assumed that it will not perform any scanning and transmission) will last for:

$$\begin{aligned}\text{Battery Life} &= \frac{1200\,\text{mAh}}{19.2\,\mu\text{A}} \\ &= 62\,500\,\text{h} = 7.13\,\text{yr}\end{aligned}$$

Despite theoretical calculations suggesting minimal consumption, practical measurements using the Power Profiler Kit II indicate a higher than expected current draw of approximately 1 mA during sleep (Stop 2 mode). This results in a battery life of 50 days in sleep mode. Numerous tests have been conducted to reduce this power draw, but as of this writing, the lowest recorded consumption remains around 1 mA. We have verified that STM32 and LR1110 enter sleep mode correctly, indicating that the excessive power consumption must be originating from some component on the PCB. Further investigation needs to be done.

Assuming the tracker performs geolocating 4 times a day it will last for:

Daily Wi-Fi scan consumption: $14.54\,\text{mAh} \cdot 4 = 58.18\,\text{mAh}$
Daily LoRaWAN transmission consumption: $41.27\,\text{mAh} \cdot 4 = 165.08\,\text{mAh}$
Daily sleep mode consumption: $19.2\,\mu\text{A} \cdot 24\,\text{h} = 0.46\,\text{mAh}$

$$\begin{aligned}\text{Battery Life} &= \frac{1200\,\text{mAh}}{58.18\,\text{mAh} + 165.08\,\text{mAh} + 0.46\,\text{mAh}} \\ &= 5.36\,\text{d}\end{aligned}$$

The investigation into the tracker's power consumption during sleep mode has highlighted significant differences between theoretical predictions and practical measurements. Addressing this issue is essential before a final product can be realised.

5.3 Minor tests

In addition to the above tests that evaluate core functionalities and performance metrics, a series of minor tests were conducted to assess the reliability of the solution and the rest of the product requirements. The results of these tests provide an understanding of the tracker's capabilities and limitations in practical scenarios.

5.3.1 Reliability test: Keep-alive transmissions

To assess the reliability of the tracker in sending data and the web application in receiving data, a series of tests were conducted using different keep-alive intervals: 2 hours, 5 hours, and 24 hours. The objective is to determine whether data is received at the expected intervals to verify uninterrupted transmission from the tracker, indicating no resets. In the event of a reset, an unexpected Join-request would be observed, and subsequent keep-alive messages would be out of synchronisation with the expected intervals.

All tests were carried out successfully, suggesting that the system is robust and reliable under longer operation times.

5.3.2 Remaining requirements and wishes

Other minor tests were conducted to confirm or deny/assess some of the remaining requirements and wishes.

- **Location accessible by phone/tablet (R1)**
  The web application can successfully be opened on a phone or tablet. However, the application is not yet responsive, and has been designed from a PC's perspective. Consequently, it performs better on tablets due to their larger screens, although it remains fully functional on phones.
- **Low battery alarm (R2)**
  A low battery alarm was successfully sent and received when the device was powered with a power supply, and the voltage was decreased to the low battery limit (3 V).
- **Configurable ping interval and LoRaWAN class (R3, R4, W4)**
  We could successfully configure the ping interval and LoRaWAN class from the web application. An downlink was scheduled, and upon receiving the tracker, it changed its configuration.
- **Location triggered by movement (R5)**
  It was verified that the STM32 receives an interrupt from the accelerometer upon movement and sets an interrupt flag to initiate a scan procedure after the expiration of the current ping interval.
- **Data accessible via API (W1)**
  Using Postman[39], each endpoint was tested and demonstrated the capability to receive requests and return the appropriate data or error message.
- **Possibility to view battery charge and temperature (W2, W3)**
  The battery charge and temperature of the tracker are correctly sent and updated with each keep-alive message. However, further tests are required to assess the accuracy of this data and determine if further calibrations are necessary.
- **Size and weight (R10, R13)**
  The size and weight were measured. The PCB's dimensions are 49 mm × 33 mm × 7 mm, and weighs 6 g, thereby satisfactorily meeting R10 and R13.

5.4 Comparison of LPWAN communication protocols

This section provides a comparative analysis of three prominent LPWAN communication protocols: LoRaWAN, **L**ong-**T**erm **E**volution **M**achine Type Communication (LTE-M), and **N**arrow**b**and **I**nternet **o**f **T**hings (NB-IoT). The comparison is based on key specifications such as data rate, range, latency, power consumption, and cost, as summarised in Tab. 18.

[39]Postman: https://www.postman.com/

Table 18: Specification comparison between LoRaWAN, LTE-M and NB-IoT.
Data sources: [1](Adelantado et al., 2017), [2](IOT), [3](Bosson), [4](Smartmakers), [5](DevAcademy), [6]Airtime calculator[40], ToA is calculated with a payload of 0 bytes, [7]All gateways forward messages, see Fig. 9, [8]Fig. 57, [9]Fig. 63, [10](Dasari and Das, 2020), [11](Lesund, 2022). Under normal conditions using LTE-M will on average have a lower power consumption than NB-IoT, but NB-IoT is better in poor coverage conditions. [12](Labs, 2017).

Parameter	LoRaWAN	LTE-M	NB-IoT
Maximum data rate (UL/DL)	27/27 kbps[1]	4/7 Mbps[2]	127/159 kbps (Cat-NB2)[3]
Typical range estimate	10 km[4]	11 km[5]	15 km[5]
Latency	50-600 ms[6]	50-100 ms[5]	1.5-10 s[5]
Mobility/call reselection	Supported[7]	Supported[5]	Not supported[5]
Duty cycle restrictions	Yes	No	No
Licensed spectrum	No	Yes	Yes
Coverage	Sparsely[8]	Good[9]	Good[9]
TX current	24-44 mA[10]	-[11]	74-220 mA[10]
RX current	12 mA[10]	-[11]	46 mA[10]
Idle current	1.4 mA[12]	-[11]	6 mA[10]
Sleep current	0.1 µA[10]	-[11]	3 µA[10]
Price	$[12]	$$$[12]	$$[12]

From the table, it is evident that while each protocol shares some similarities, they also has distinct specifications that cater to different use cases. LoRaWAN offers the lowest price point and power consumption, making it suitable for cost-sensitive applications. However, it has poor coverage, but its use of an unlicensed spectrum allows for the deployment of private gateways, which can be beneficial for specific scenarios. LTE-M is ideal for applications that require high data rates, low latency, and mobility, but it has the highest price point of the compared protocols. It operates on a licensed spectrum, which ensures reliable communication but takes away control from the user, as they must rely on the service provider's infrastructure and policies. As seen in Fig. 63 the coverage is excellent in Denmark, but this varies between countries. NB-IoT strikes a balance between LTE-M and LoRaWAN. It is more affordable than LTE-M and offers a better range. However, its lower data rate and higher latency make it suitable only for static devices where real-time communication is not critical.

[40]LoRaWAN airtime calculatorhttps://www.thethingsnetwork.org/airtime-calculator

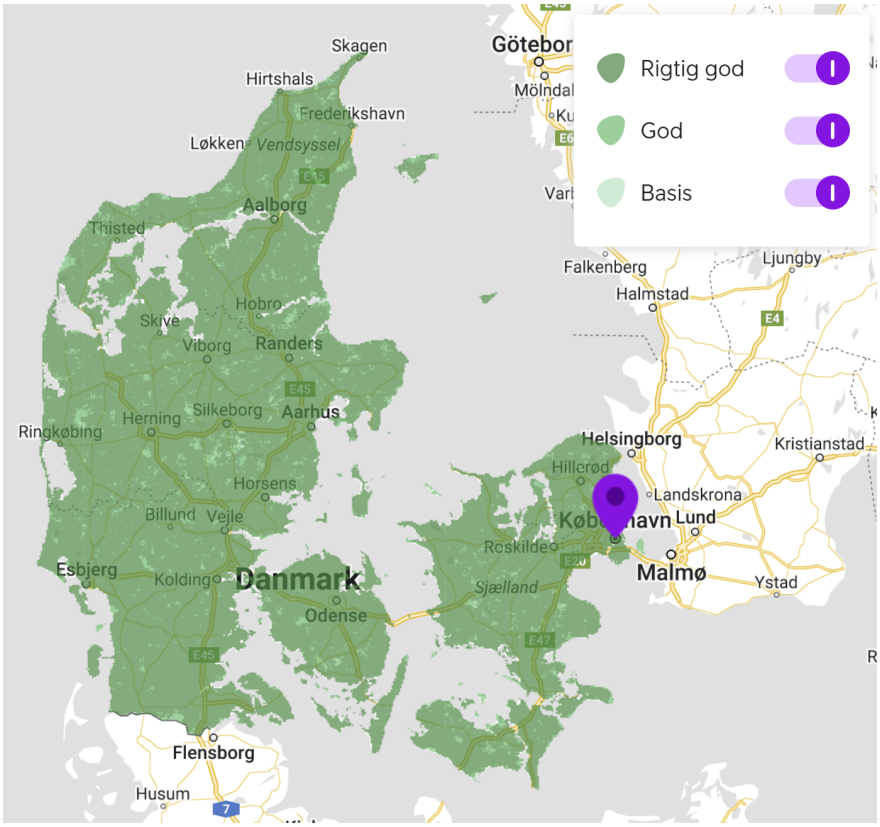


Figure 63: Coverage map of Telia's NB-IoT and LTE-M network in Denmark[41].

In conclusion, there is no one-size-fits-all solution. Each protocol has unique benefits, and the choice should be based on the specific needs of the deployment environment and application requirements.

[41] Telia's NB-IoT and LTE-M network map: https://coverage.ddc.teliasonera.net/coverageportal_dk?profile=IOT

6 Suggestions for future research

As mentioned, the development of our geolocation tracker has revealed several areas for further exploration and improvement. This section outlines, in list form, recommended tests and implementations prioritized by importance, with the most critical items listed first.

- **Fix power issues with hardware**
  Addressing the power consumption issues identified during testing should be the priority to ensure the device meets the originally envisioned ultra-low-power benchmarks.
- **Miniaturisation of components**
  During the initial development phase, larger components were used due to their cost-effectiveness and ease of handling. This pragmatic choice allowed multiple PCB iterations within our budget constraints. However, future iterations of the PCB design will focus on incorporating smaller components once all functional parameters have been validated. This transition aims to optimise the design for compactness.
- **Minimum supply voltage and its effect on performance**
  Explore the lowest minimum voltage that can power the PCB and how it affects the performance.
- **Investigate sleep modes**
  Investigate how different sleep modes affect the performance and power consumption of the device. This includes evaluating various low-power modes to determine the most efficient option for extending battery life while maintaining functionality.
- **Reliability testing**
  The tracker needs to be tested for extended periods, exceeding 24 hours, to ensure reliability and performance under prolonged usage conditions.
- **LoRaWAN payload size impact on battery consumption**
  Analysing how payload size affects power consumption will provide insights into optimising data transmission, balancing the need for comprehensive data with energy efficiency.
- **Store results or sent immediately**
  Evaluate the power consumption trade-offs between storing scan results and transmitting them in batches versus sending each scan result immediately. Determine which approach minimises overall power usage.
- **Passive or active GNSS antenna**
  Comparative tests between passive and active GNSS antennas will be conducted to evaluate their impact on power consumption and overall performance.
- **Relationship between accuracy and power consumption when sending RSSI with Wi-Fi scans**
  Investigate the balance between accuracy and power consumption when sending RSSI data during Wi-Fi scans.

- **LoRaWAN parameter impact**
  Explore whether manually adjusting LoRaWAN parameters can lead to better power consumption than relying on Network Server optimisations and how it affects range and duty cycle.
- **Optimal time for assistance data syncing**
  Determining when fetching new data is more power-efficient than running a GNSS scan, ensuring the device remains energy-efficient while maintaining accuracy.
- **Responsive design web application**
  Enhancing the web application's frontend to be fully responsive will improve user experience across various devices. This involves refining the interface to ensure it functions seamlessly on different screen sizes and resolutions.
- **User test**
  The current web application is an integration prototype, ready to test the full integration of the product with users. This testing will provide valuable insights into how well the product functions as a cohesive whole and how it can be refined to better meet user needs.
- **Duty cycle analysis**
  Assessing the current duty cycle, utilisation will ensure compliance with regulatory limits and identify opportunities for optimising communication intervals to enhance power efficiency.
- **Geofencing**
  Implement geofencing functionality to allow the device to trigger specific actions when entering or leaving predefined geographic areas. This feature can improve the relevance of location-based notifications and actions.
- **Investigate other low-power geolocation chips**
  Investigating the potential benefits of using the newer LR1121[42] chip will determine if it offers improvements in performance, power consumption, or other critical parameters compared to the current LR1110 chip. The main difference between LR1121 and LR1110 is the support for satellite S-band. LR1121 can also be better used in cross-domain positioning scenarios, such as sea transportation (Joffe, 2022). For smart home applications and medium-range indoor and indoor-to-outdoor wireless applications, the LLCC68 from Semtech can be recommended. Its physical size of $4 \times 4\,\text{mm}$ (QFN) is also smaller compared to LR1110 and will therefore reduce the size of the end device. Another geolocation chip is the nRF7000[43] from Nordic Semiconductor. Like the LR1110 it also includes Wi-Fi sniffing and GNSS scanning for geolocating.
- **Form factor (length vs width)**
  Evaluating different form factors will help determine which design — longer or wider — is most beneficial for user comfort and device performance.

[42] LR1121: https://www.semtech.com/products/wireless-rf/lora-connect/lr1121
[43] LR1121: https://www.nordicsemi.com/Products/nRF7000

- **Firmware update over the air**
  Implement **F**irmware **U**pdate **O**ver **T**he **A**ir (FUOTA) to allow for remote updates of the device's software. This feature will enhance the ability to deploy updates and improvements without needing physical access to the device.
- **Implement Hall effect logic**
  The Hall effect sensor has been successfully implemented on the board, with tests confirming its operation. The next step involves determining its specific application and programming the associated logic. This process will be conducted collaboratively with users to understand and address their specific needs effectively.
- **Integrate bluetooth**
  A new firmware update for the LR1110 chip has introduced **B**luetooth **L**ow **E**nergy (BLE) capabilities to the chip for geolocation. Add this functionality, and test power consumption and accuracy compared to Wi-Fi and GNSS scan.
- **Click interrupt**
  Integrating a click-interrupt feature using the LIS2DE12 sensor will enhance user interaction. This functionality will allow the tracker to wake or configure based on user taps, improving usability and extending battery life by reducing unnecessary activity.

7 Conclusion

This thesis describes the successful development, design, implementation, and test of a geolocation tracker utilising the LR1110 chip and LoRaWAN technology.

We developed a complete ecosystem featuring a custom-made PCB with the LR1110 chip capable of scanning for Wi-Fi access points but unfortunately not GNSS SVs. The data from the scan can be transmitted via LoRaWAN to a cloud solver for position calculation, and to our custom-developed web application, capable of storing and displaying the retrieved position.

Hold up against the product requirements, our tracker solution fulfils 13 out of 14 specified requirements ("need to have") and 6 out of 7 wishes ("nice to have"), as detailed in Tab. 19. Not all requirements could be fulfilled during the 5 months of this project. Despite achieving significant milestones, our project fell short of achieving the ultra-low power consumption requirement mentioned in the project title. However, we are confident that both this and the GNSS issue are hardware-related and that with additional time, an improved PCB which can successfully perform GNSS scans, and can be categorised as having an ultra-low consumption, can be developed.

Table 19: Product requirements deducted from empirical data obtained from interviews and desk research. Green colour denotes fulfilled requirements and wishes, while red signifies those yet to be achieved.

Category	Requirements	Wishes
Functional requirements	Location accessible by phone and tablet	Data accessible via API
	Low battery alarm	View battery charge
	Configurable ping interval	View temperature
	Remote configuration of ping interval	User adjustable LoRaWAN class
	Location triggered by movement	
	Accuracy of 20 m	
Performance requirements	LoRa range of 5 km in open terrain	
	2 years battery life and 1000 positions on one battery charge[44]	
Physical requirements	Shape that enables easy case design	
	Size of max 50 × 40 × 10 mm	
	External antennas	
	External battery	
	Weight of 20 g without battery and antennas	
Sustainability requirements	Unit price < 1000 DKK	Rechargeable battery
		Open-source

The development process highlighted the steep learning curve of working with embedded programming, the LR1110 chip and LoRaWAN-enabled devices.

Furthermore, the thesis investigated the advantages and limitations of LoRaWAN technology and the market demand for battery-powered ultra-low-power geolocation devices, showing a real-world demand.

Our findings suggest that while LoRaWAN and the LR1110 chip offer compelling advantages, such as potentially being the lowest power option on the market, the technology also presents significant challenges. Product developers and users must carefully consider these factors before developing and deploying LoRaWAN-based solutions. While LoRaWAN and the LR1110 chip demonstrate promise for certain use cases, the complex setup and development process, fragmented market landscape with incompatible network operators, diverse regional parameters, and, for now, limited coverage limits it to cases tailored to specific geographic areas.

In conclusion, while our thesis has demonstrated the possibilities of developing cost-effective, low-power end devices with the LR1110 chip, it is essential to acknowledge the shortcomings of the LR1110 chip and LoRaWAN technology; the high level of development difficulty and

[44] LiPo battery: 1200 mAh, 3.7 V, 6 × 34 × 50 mm

the critical factor of network coverage. As highlighted, coverage is not universally assured, necessitating either careful selection of deployment areas or the capability to establish custom gateway infrastructure.

References

Adelantado, Ferran et al. (2017). "Understanding the Limits of LoRaWAN". In: *IEEE Communications Magazine* 55.9, pp. 34–40. ISSN: 0163-6804. DOI: 10.1109/mcom.2017.1600613. URL: http://dx.doi.org/10.1109/MCOM.2017.1600613.

Bäumker, E, A Garcia, and Peter Woias (Nov. 2019). "Minimizing power consumption of LoRa® and LoRaWAN for low-power wireless sensor nodes". In: *Journal of Physics: Conference Series* 1407, p. 012092. DOI: 10.1088/1742-6596/1407/1/012092.

Bosson, Michael (2024). *NB-IoT vs LTE-M: Comparing the two IoT technologies*. URL: https://onomondo.com/blog/nb-iot-vs-lte-m-a-comparison-of-the-two-iot-technology-standards/ (visited on 06/20/2024).

Charlie (2023). *SSID-based Wi-Fi locationing: Comparing performance with other location services*. URL: https://devzone.nordicsemi.com/nordic/nordic-blog/b/blog/posts/ssid_2d00_based-wi_2d00_fi-locationing-comparing-performance-with-other-location-services (visited on 06/20/2024).

Dasari, Mallesham and Samir Das (2020). *CSE570 Spring 2020, Wireless and Mobile Networks, IoT/Sensor Networks (LoRa & LoRaWAN)*. URL: https://www3.cs.stonybrook.edu/~mdasari/courses/cse570/lora.pdf.

DevAcademy (2024). *LTE-M and NB-IoT*. URL: https://academy.nordicsemi.com/courses/cellular-iot-fundamentals/lessons/lesson-1-cellular-fundamentals/topic/lesson-1-lte-m-and-nb-iot/ (visited on 06/20/2024).

Dhaker, Piyu (2018). *Introduction to SPI Interface*. URL: https://www.analog.com/en/resources/analog-dialogue/articles/introduction-to-spi-interface.html (visited on 02/15/2024).

Grusin, Mike (2024). *Serial Peripheral Interface (SPI)*. URL: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all (visited on 02/15/2024).

Guide, LoRa Networking (2024). *Long range tests*. URL: https://development.libelium.com/lora_networking_guide/long-range-tests (visited on 05/26/2024).

Haidarzhy, Valerii (2023). *LoRaWAN MAC Layer: Definition, Architecture, Classes, and More*. URL: https://sirinsoftware.com/blog/lorawan-mac-layer-definition-architecture-classes-and-more.

Holtzblatt, Karen and Hugh Beyer (2016). *Contextual design*. eng. Second edition. Interactive technologies. Morgan Kaufmann. Chap. 9. ISBN: 0128008946.

Houde, Stephanie and Charles Hill (1997). "Chapter 16 - What do Prototypes Prototype?" In: *Handbook of Human-Computer Interaction (Second Edition)*. Ed. by Marting G. Helander, Thomas K. Landauer, and Prasad V. Prabhu. Second Edition. Amsterdam: North-Holland, pp. 367–381. ISBN: 978-0-444-81862-1. DOI: https://doi.org/10.1016/B978-044481862-1.50082-0.

Iba, Takashi, Ayaka Yoshikawa, and Konomi Munakata (2017). "Philosophy and methodology of clustering in pattern mining: Japanese anthropologist Jiro Kawakita's KJ method". eng. In: *Proceedings of the 24th Conference on Pattern Languages of Programs*. The Hillside Group, pp. 1–11. ISBN: 1941652069.

IOT, Telenor (2024). *LTE-M vs NB-IoT – A Guide Exploring the Differences between LTE-M and NB-IoT*. URL: https://iot.telenor.com/iot-insights/lte-m-vs-nb-iot-guide-differences/ (visited on 06/20/2024).

Joffe, Scott (2022). *Why is LoRa Edge™ a Revolutionary Technology*. URL: https://www.mokolora.com/why-is-lora-edge-a-revolutionary-technology/#:~:text=Comparison%20of%20LR1110%20and%20LR1120&text=Or%20more%20accurately%2C%20the%20LR1110,S%2Dband%20and%202.4%20GHz. (visited on 06/17/2024).

Labs, Link (2017). *Cost of Building With LPWAN Technologies*. URL: https://www.link-labs.com/blog/costs-in-iot-lte-m-vs.-nb-iot-vs.-sigfox-vs.-lora (visited on 06/20/2024).

Lesund, Martin (2022). *LTE-M vs NB-IoT Field Test: How Distance Affects Power Consumption*. URL: https://devzone.nordicsemi.com/nordic/nordic-blog/b/blog/posts/ltem-vs-nbiot-field-test-how-distance-affects-power-consumption (visited on 06/20/2024).

LoRa Alliance, Inc. (2015). *LoRaWA Specification*. URL: https://resources.lora-alliance.org/technical-specifications/lorawan-specification-v1-0 (visited on 02/25/2024).

Network, The Things (2024). *What are LoRa and LoRaWAN?* URL: https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/ (visited on 05/09/2024).

Novatel (2024). *What are Global Navigation Satellite Systems?* URL: https://novatel.com/tech-talk/an-introduction-to-gnss/what-are-global-navigation-satellite-systems-gnss (visited on 02/15/2024).

Overflow, Stack (2023). *2023 Developer Survey*. URL: survey.stackoverflow.co/2023/#technology-most-popular-technologies (visited on 02/17/2024).

Pachuca, Pedro (2020). "Indoor Wi-Fi Geolocation with LoRa Edge™". In: *Inside Out: Semtech's Corporate Blog*. URL: https://blog.semtech.com/indoor-wi-fi-geolocation-with-lora-edge.

Rohde and Schwarz (2024). *Understanding UART*. URL: https://www.rohde-schwarz.com/products/test-and-measurement/essentials-test-equipment/digital-oscilloscopes/understanding-uart_254524.html?change_c=true (visited on 02/15/2024).

Scupin, Raymond (1997). "The KJ Method: A Technique for Analyzing Data Derived from Japanese Ethnology". eng. In: *Human organization* 56.2, pp. 233–237. ISSN: 0018-7259.

Secretary, ISO Central (2018). *Geographic information — Imagery sensor models for geopositioning*. en. Standard ISO/IEC TR 19130-1:2018(E). Geneva, CH: International Organization for Standardization. URL: https://www.iso.org/standard/66847.html.

Semtech (2019a). *An In-depth Look at LoRaWAN® Class A Devices*. Tech. rep. URL: https://lora-developers.semtech.com/uploads/documents/files/LoRaWAN_Class_A_Devices_In_Depth_Downloadable.pdf.

Semtech (2019b). *An In-depth Look at LoRaWAN® Class B Devices*. Tech. rep. URL: https://lora-developers.semtech.com/uploads/documents/files/LoRaWAN_Class_B_Devices_In_Depth_Downloadable.pdf.

Semtech (2019c). *An In-depth Look at LoRaWAN® Class C Devices*. Tech. rep. URL: https://lora-developers.semtech.com/uploads/documents/files/LoRaWAN_Class_C_Devices_In_Depth_Downloadable.pdf.

Semtech (2020). *LoRa Basics™ Modem-E Transform Asset Management*. URL: https://info.semtech.com/hubfs/LoRa_Basics_modem-E_infographic_final_secure.pdf (visited on 06/14/2024).

Semtech (2021). *LoRa Edge™ Asset Management System Location Performance Overview Application Note*. URL: https://www.semtech.com/uploads/design-support/AN1200.68_LoRaEdgeManagementSystemLocationPerformance.pdf (visited on 06/19/2024).

Semtech (2022). *AN1200.66, Application note: PCB Design Guidelines*. Version 1.1. URL: https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/3n000000qSpB/cgE7bugHAizCivyqDQiD9u3p_rIhE4pdFo6Ts0ZQIrg.

Semtech (2023a). *LR1110 Transceiver User Manual*. Version 2.0. URL: https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/RQ000001DmRJ/.BD3tRRoM1xW5W1VpTVe2rkizt6u76_g7xowpMoNq6U.

Semtech (2023b). *LR1110DVK1TBKS*. Tech. rep. URL: https://www.semtech.com/products/wireless-rf/lora-edge/lr1110dvk1tbks.

Siebeneicher, Hannes (2024). *Universal Asynchronous Receiver-Transmitter*. URL: https://docs.arduino.cc/learn/communication/uart/ (visited on 02/15/2024).

Smartmakers (2024). *LoRaWAN Range, Part 2: Range and coverage of LoRaWAN in practice*. URL: https://smartmakers.io/en/lorawan-range-part-2-range-and-coverage-of-lorawan-in-practice/ (visited on 05/26/2024).

ST (2019). *AN4621, Application note: STM32L4 and STM32L4+ ultra-low-power features overview*. URL: https://www.st.com/resource/en/application_note/an4621-stm32l4-and-stm32l4-ultralowpower-features-overview-stmicroelectronics.pdf (visited on 06/18/2024).

ST (2020). *UM1724, User manual: STM32 Nucleo-64 boards (MB1136)*. URL: https://www.st.com/resource/en/user_manual/um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf (visited on 06/15/2024).

Team, The Things Network Global (2023). *New LoRa world record: 1336 km / 830 mi*. URL: https://www.thethingsnetwork.org/article/new-lora-world-record-1336-km-830-mi (visited on 02/15/2024).

Thagaard, Tove. (2004). *Systematik og indlevelse : en indføring i kvalitativ metode*. dan. Akademisk Forlag. Chap. 5. ISBN: 9788750038214.

Wikipedia (2024a). *Satellite navigation*. URL: https://en.wikipedia.org/wiki/Satellite_navigation (visited on 02/15/2024).

Wikipedia (2024b). *Serial Peripheral Interface*. URL: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface (visited on 02/15/2024).

Appendices

A Code

All code is available on GitHub at https://github.com/MagnusErler/WiFiLocationTracker.

B Essential hardware

1. LR1110 : https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/RQ000001Df4q/Mz_qfFd7oSbsmETxZ4BgwuQ5vIKrHp2vdfF23DzaC58
2. STM32L476RG : https://www.st.com/en/microcontrollers-microprocessors/stm32l476rg.html
3. NT2016SA 32MHz END4263A : https://www.digikey.dk/en/products/detail/ndk-america-inc/NT2016SA-32M-END4263A/8275433
4. LIS2DE12 : https://www.st.com/resource/en/datasheet/lis2de12.pdf
5. BU52077GWZ : https://fscdn.rohm.com/en/products/databook/datasheet/ic/sensor/hall/bu52077gwz-e.pdf
6. XC6220B331MR : https://product.torexsemi.com/system/files/series/xc6220.pdf
7. BGA524N6E6327 : https://www.infineon.com/dgdl/Infineon-BGA524N6-DataSheet-v03_05-EN.pdf?fileId=db3a304344e406b50144e44dfad302b9
8. SKY13588-460LF : https://www.skyworksinc.com/-/media/SkyWorks/Documents/Products/2201-2300/SKY13588_460LF_203512D.pdf

C Essential software

1. Visual Studio Code : https://code.visualstudio.com/
2. Git : https://git-scm.com/
3. GNU Make : https://gnuwin32.sourceforge.net/packages/make.htm
4. ARM GNU Toolchain : https://developer.arm.com/downloads/-/arm-gnu-toolchain-downloads
5. OpenOCD : https://gnutoolchains.com/arm-eabi/openocd/
6. Cortex-Debug : https://marketplace.visualstudio.com/items?itemName=marus25.cortex-debug

7. AppCAD : https://www.broadcom.com/info/wireless/appcad
8. Node.js & NPM : https://nodejs.org/en/download/prebuilt-installer

D Windows and Linux setup

D.A c_cpp_properties.json

```
{
    "configurations": [
        {
            "name": "STM32L476RG",
            "includePath": [
                "${workspaceFolder}/**"
            ],
            "defines": [
                "_DEBUG",
                "UNICODE",
                "_UNICODE",
                "USE_HAL_DRIVER",
                "STM32L476xx"
            ],
            "configurationProvider": "ms-vscode.makefile-tools
               ",
            "compilerPath": "${env:ARMGCC_DIR}/arm-none-eabi-
              gcc.exe",
            "compilerArgs": [
                "-mcpu=cortex-m4",
                "-mthumb"
            ],
            "cStandard": "c11",
            "cppStandard": "c++17"
        }
    ],
    "version": 4
}
```

D.B launch.json

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Cortex Debug",
```

```
            "cwd": "${workspaceFolder}",
            "executable": "geolocation/
               build_stm32_l4_lr11xx_crypto_with_cred/
               app_stm32_lr11xx_crypto_with_cred.elf",
            "request": "launch",
            "type": "cortex-debug",
            "runToEntryPoint": "main",
            "servertype": "openocd",
            "device": "STM32L476RG",
            "configFiles": [
                "interface/stlink.cfg",
                "target/stm32l4x.cfg"
            ]
        }
    ]
}
```

D.C Web application setup

The application assumes that one uses TTN as their LoRaWAN provider. To set up the web application, follow the ReadMe in the `web_application` folder.

E PCB

E.A BOM

Table 20: BOM of the tracker.

Name	Designator	Footprint	Manufacturer Part	Manufacturer	Price [$]
			Chips		
STM32L476	U1	LQFP-64	STM32L476RET6	STM	2.924
LR1110	U2	QFN-50	LR1110IMLTRT	Semtech	10.540
SKY13588-460LF	U3	QFN-12	SKY13588-460LF	Skyworks	2.254
BGA524N6E6327	U4	TSNP-6	BGA524N6E6327	Infineon	0.286
XC6220B331MR	U5	SOT23-5	XC6220B331MR	Tech Public	0.179
LIS2DE12	U6	LGA-12	LIS2DE12	STM	0.469
BU52077GWZ-E2	U7	BGA-4	BU52077GWZ-E2	ROHM	0.177
			Resistors		
220 Ω	R3	R0402	0402WGF2200TCE	UNI-ROYAL	0.001
10 kΩ	R7	R0402	0402WGF1002TCE	UNI-ROYAL	0.001
200 Ω	R33	R0603	0603WAF2000T5E	UNI-ROYAL	0.001
			Capacitors		
100 nF	C1	C0402	CL05B104KO5NNNC	Samsung	0.001
47 pF	C2	C0402	0402CG470J500NT	FH	0.001
470 nF	C3	C0603	CL10B474KA8NNNC	Samsung	0.007
100 nF	C4	C0402	CL05B104KO5NNNC	Samsung	0.001

Table 20 BOM.

Name	Designator	Footprint	Manufacturer Part	Manufacturer	Price [$]
2.2 nF	C5	C0603	0603B222K500NT	FH	0.002
47 pF	C6	C0402	0402CG470J500NT	FH	0.001
47 pF	C7	C0402	0402CG470J500NT	FH	0.001
1 pF	C8	C0402	0402CG1R2C500NT	FH	0.001
6.8 pF	C9	C0402	CC0402CRNPO9BN6R8	YAGEO	0.001
47 pF	C10	C0402	0402CG470J500NT	FH	0.001
1.8 pF	C12	C0402	GJM1555C1H1R8BB01D	muRata	0.018
3 pF	C13	C0402	0402N3R0C500CT	Walsin	0.002
6.8 pF	C14	C0402	CC0402CRNPO9BN6R8	YAGEO	0.001
47 pF	C15	C0402	0402CG470J500NT	FH	0.001
1.5 pF	C17	C0402	GJM1555C1H1R5WB01D	muRata	0.028
1 pF	C18	C0402	0402CG1R2C500NT	FH	0.001
1.5 pF	C19	C0402	GJM1555C1H1R5WB01D	muRata	0.028
47 pF	C21	C0402	0402CG470J500NT	FH	0.001
1 pF	C22	C0402	0402CG1R0C500NT	FH	0.001
1.5 pF	C23	C0402	GJM1555C1H1R5WB01D	muRata	0.028
0.5 pF	C24	C0402	C1005NP0508CGTS	DARFON	0.008
12 pF	C25	C0402	0402CG120J500NT	FH	0.001
0.5 pF	C26	C0402	C1005NP0508CGTS	DARFON	0.008
1.3 pF	C27	C0402	RF15N1R3B500CT	Walsin	0.015
1.6 pF	C28	C0402	CC0402CRNPO9BN1R6	YAGEO	0.001
33 pF	C29	C0402	0402CG330J500NT	FH	0.001
0.7 pF	C30	C0402	0402CGR75C500NT	FH	0.001
3.3 pF	C31	C0402	0402CG3R3C500NT	FH	0.001
3.3 pF	C32	C0402	0402CG3R3C500NT	FH	0.001
100 nF	C33	C0402	CL05B104KO5NNNC	Samsung	0.001
1 µF	C34	C0402	CL05A105KA5NQNC	Samsung	0.004
100 nF	C35	C0402	CL05B104KO5NNNC	Samsung	0.001
100 nF	C36	C0402	CL05B104KO5NNNC	Samsung	0.001
100 nF	C37	C0402	CL05B104KO5NNNC	Samsung	0.001
10 µF	C38	C0603	CL10A106KP8NNNC	Samsung	0.005
1 nF	C39	C0402	0402B102K500NT	FH	0.001
1 µF	C40	C0402	CL05A105KA5NQNC	Samsung	0.004
10 nF	C41	C0402	CL05B103KB5NNNC	Samsung	0.001
4.7 µF	C42	C0402	CL05A475MP5NRNC	Samsung	0.005
1 nF	C43	C0402	0402B102K500NT	FH	0.001
1 µF	C44	C0402	CL05A105KA5NQNC	Samsung	0.004
1 nF	C45	C0402	0402B102K500NT	FH	0.001
1 nF	C46	C0402	0402B102K500NT	FH	0.001
100 nF	C47	C0402	CL05B104KO5NNNC	Samsung	0.001
10 µF	C48	C0402	CL05A106MQ5NUNC	Samsung	0.005
100 nF	C49	C0402	CL05B104KO5NNNC	Samsung	0.001
10 pF	C50	C0402	CL05C100JB5NNNC	Samsung	0.005
1.8 pF	C51	C0402	GJM1555C1H1R8BB01D	muRata	0.018
0.5 pF	C52	C0402	C1005NP0508CGTS	DARFON	0.008
1 nF	C54	C0402	0402B102K500NT	FH	0.001
470 pF	C55	C0603	0603B471K500NT	FH	0.002
0.5 pF	C56	C0402	C1005NP0508CGTS	DARFON	0.008
1.8 pF	C57	C0402	GJM1555C1H1R8BB01D	muRata	0.018
1 nF	C58	C0402	0402B102K500NT	FH	0.001
100 nF	C81	C0402	CL05B104KO5NNNC	Samsung	0.001

Table 20 BOM.

Name	Designator	Footprint	Manufacturer Part	Manufacturer	Price [$]
4.7 µF	C82	C0402	CL05A475MP5NRNC	Samsung	0.005
100 nF	C101	C0402	CL05B104KO5NNNC	Samsung	0.001
100 nF	C103	C0402	CL05B104KO5NNNC	Samsung	0.001
1 µF	C104	C0402	CL05A105KA5NQNC	Samsung	0.004
1 µF	C105	C0402	CL05A105KA5NQNC	Samsung	0.004
1 µF	C106	C0402	CL05A105KA5NQNC	Samsung	0.004
			Ferrites & Inductors		
10 Ω	FB1	L0402	CBW100505U100T	FH	0.004
600 Ω	L1	L0603	GZ1608D601TF	Sunlord	0.004
15 µH	L2	L0805	CMP201209JD150MT	FH	0.025
47 nH	L3	L0402	CMCC1005C47NJSP	Cybermax	0.004
47 nH	L4	L0402	CMCC1005C47NJSP	Cybermax	0.004
3.3 nH	L5	L0402	LQG15HS3N3S02D	muRata	0.006
8.2 nH	L6	L0402	LQG15HH8N2J02D	muRata	0.029
2.7 nH	L7	L0402	LQG15HS2N7S02D	muRata	0.006
6.8 nH	L8	L0402	SDCL1005C6N8JTDF	Sunlord	0.003
22 nH	L9	L0402	LQW15AN22NG00D	muRata	0.034
47 nH	L10	L0402	CMCC1005C47NJSP	Cybermax	0.004
9.5 nH	L11	L0402	SDCL1005C10NJTDF	Sunlord	0.003
33 nH	L12	L0402	LQW15AN33NJ00D	muRata	0.046
9.5 nH	L13	L0402	SDCL1005C10NJTDF	Sunlord	0.003
8.2 nH	L14	L0402	MGCI1005T8N2JT-LF	Microgate	0.004
3.6 nH	L15	L0402	LQG15HS3N3S02D	muRata	0.006
9.1 nH	L17	L0402	SDCL1005C10NJTDF	Sunlord	0.003
3.6 nH	L20	L0402	LQG15HS3N3S02D	muRata	0.006
			Crystals & TCXO		
32 MHz	X1	OSC-4	1XXD32000MBA	KDS	0.782
32.768 kHz	X2	FC-135R	Q13FC13500004	EPSON	0.185
32.768 kHz	X3	FC-135R	Q13FC13500004	EPSON	0.185
			Connectors		
Wi-Fi U.FL	RF1	FRF05002	U.FL-R-SMT-1(80)	HRS	0.080
GNSS U.FL	RF2	FRF05002	U.FL-R-SMT-1(80)	HRS	0.080
LoRa U.FL	RF3	FRF05002	U.FL-R-SMT-1(80)	HRS	0.080
			Antennas		
Wi-Fi antenna	ANT1		206560	Molex	3.080
GNSS antenna	ANT2		211140	Molex	2.690
LoRa antenna	ANT3		146153	Molex	2.300
Total					$26.805

E.B Wiring schematic

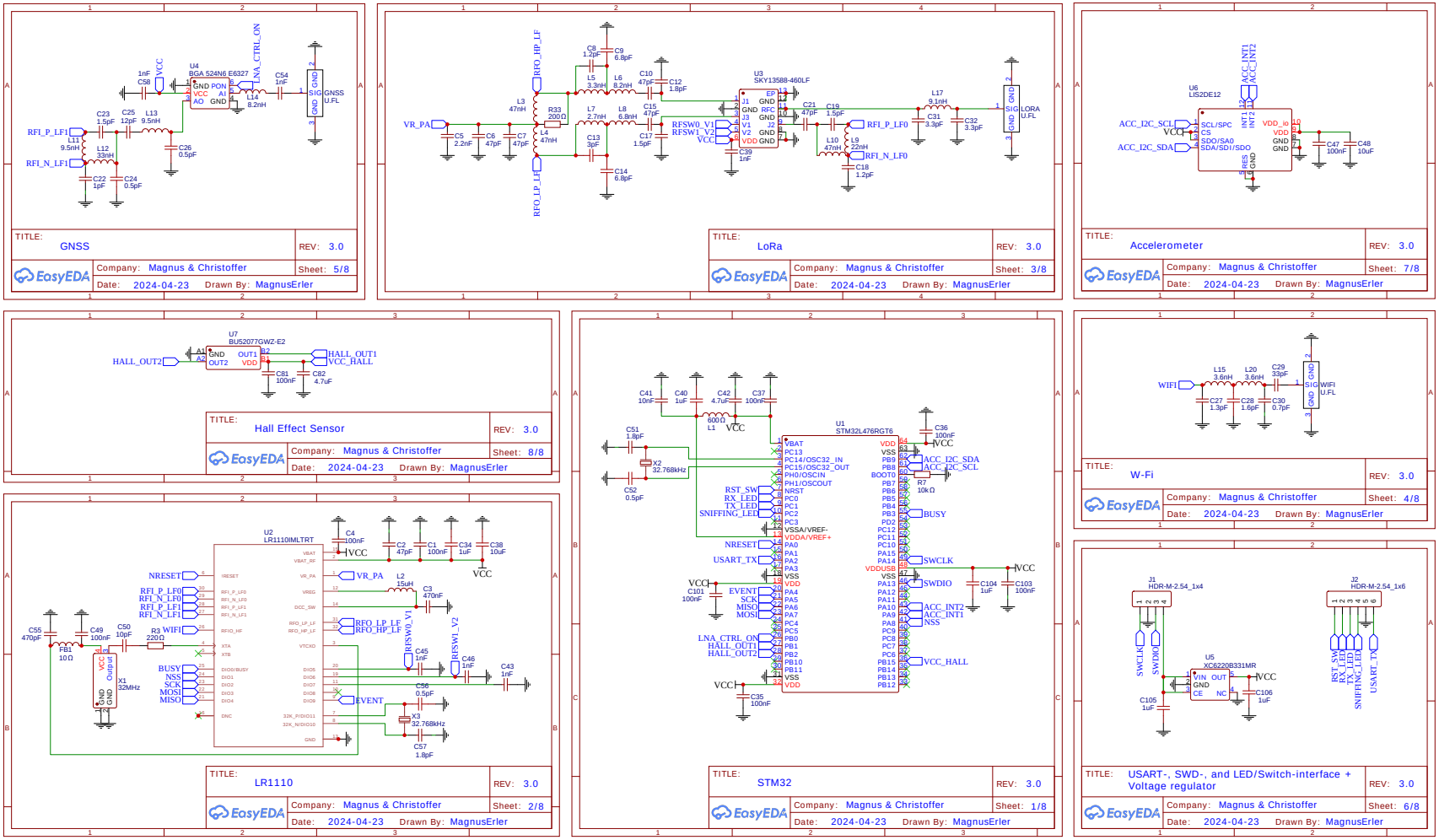


Figure 64: Overview of the final wiring schematic.

E.C PCB

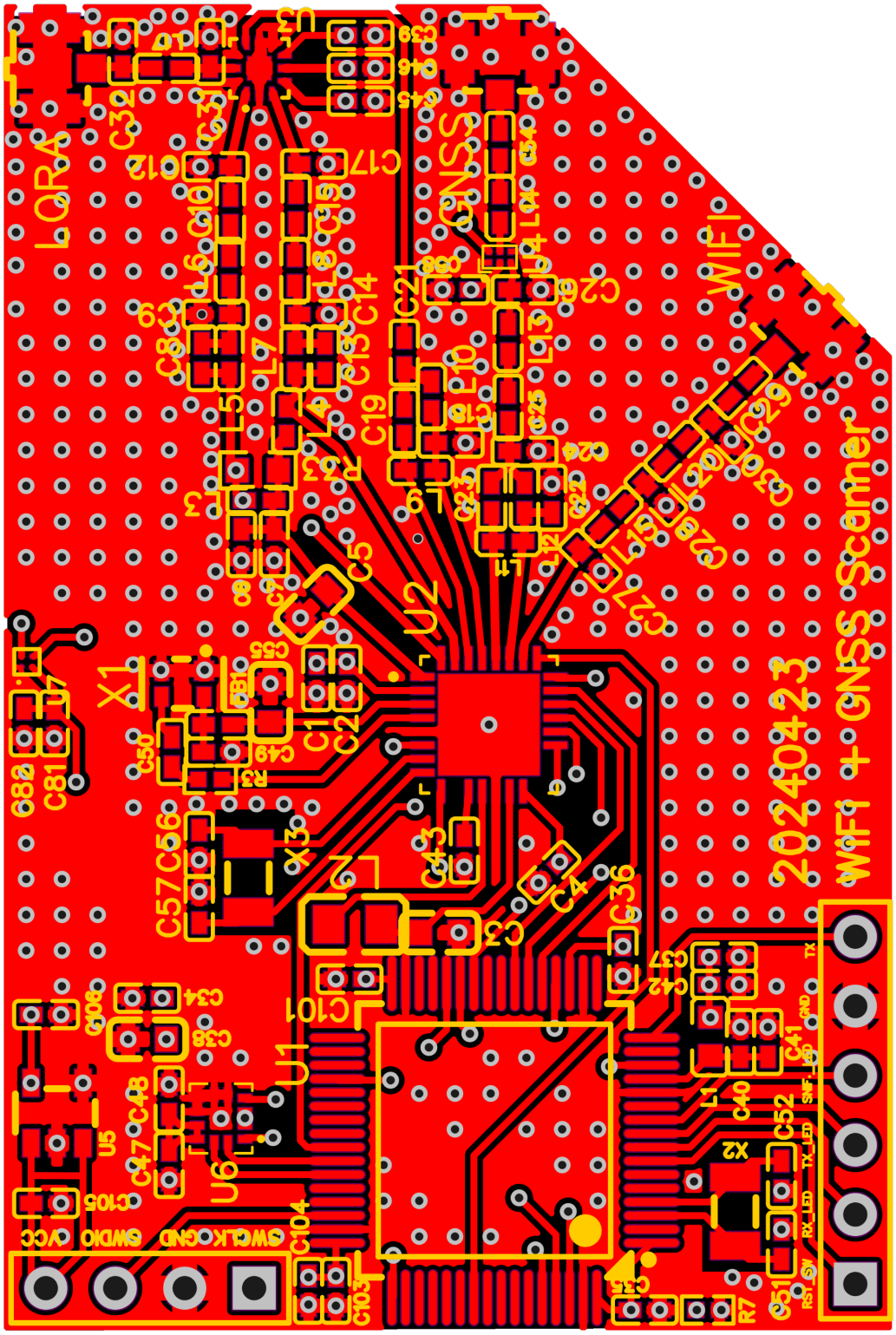


Figure 65: Overview of the third and current version of our PCB.

E.D 3D view

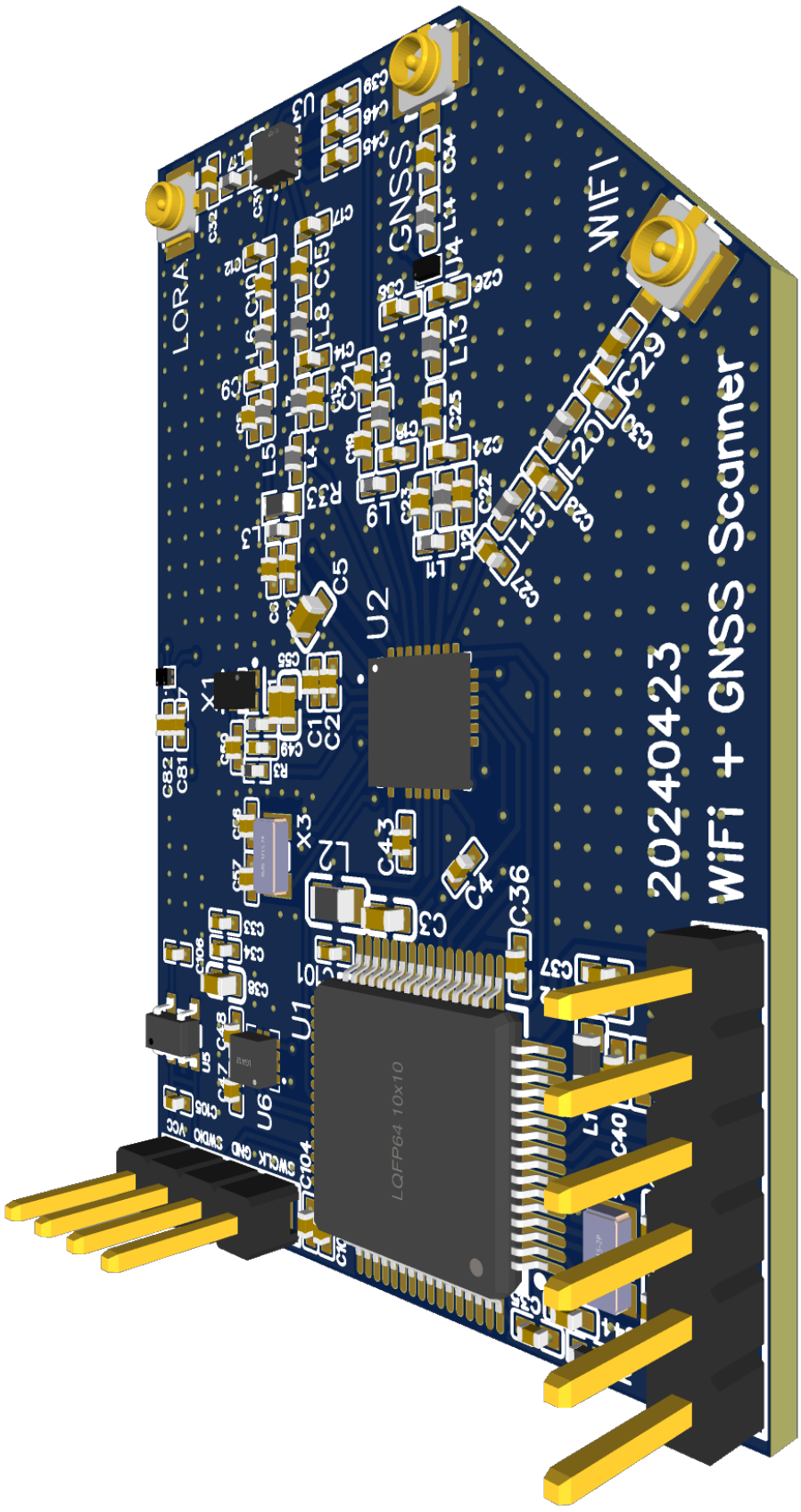


Figure 66: 3D view of the final PCB.

F Groups from affinity diagram

- Other potential functionalities
  - Modify the physical design
  - Alarms and triggers
- Integration with other systems
- Technical skills for installation
  - Does have
  - Does not have
- Market potential
  - Need for tracking
  - Alternatives to tracking
  - Future scenarios
  - Need for quality improvement
  - Lack of network connectivity
- Wireless technologies in current products
- Current product designs
- Product design concerning physical robustness
- One product, multiple designs
- Size vs battery life
- How to display data
- Importance of user involvement
- Data security and sharing of tracking data
- Legislation regarding the tracking of humans
- Tracking interval
- Alternative charging methods
- Mismatch between product specifications and reality
  - Real-life battery experiences
  - Promised battery life
- Economics

- Difficulties with product design for the public sector
- Operating costs
- Unit price
- Alternative business models
- Economic sustainability