Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Networked Systems and Services

Karlsruhe Institute of Technology
Institute of Radio Frequency Engineering and Electronics

Sensor and position data collection in UWB radio systems

Bachelor's Thesis

Author
Zoltán Barnabás Marcsek

Supervisor
Dr. Balázs Matolcsy

December 10, 2021

Contents

HALLGATÓI NYILATKOZAT

Alulírott *Marcsek Zoltán Barnabás,* szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2021. december 10.

Marcsek Zoltán Barnabás
hallgató

Kivonat

Napjainkban egyre nagyobb szerepet kapnak a különféle szenzorrendszerek. Legyen szó akár okos otthon, okos város, vagy ipar 4.0 megoldásokról, mindegyiknek szerves része az adatgyűjtés, és az adatok elemzése. A használt rendszerek jellemzően állandóan telepített szenzorokból állnak, amik egy adott helyen végeznek méréseket. Ez sokszor előnyös lehet, például ha ismertek a kritikus helyek és ezekre kerülnek telepítésre a szenzorok, azonban egy terület általános felmérésére nem kifejezetten alkalmasak az ilyen rendszerek. A felmérendő területről könnyebben kaphatunk átfogó képet, ha a fixen telepített szenzorok helyett mozgó eszközöket használunk. Ebben az esetben adott terület felméréséhez sokkal kevesebb mérőeszköz is elég lehet, mivel egy mérőeszköz több pontban is végezhet méréseket.

A szakdolgozatomban bemutatom a pozicionálás fontosságát a szenzorhálózatokban valamint a mozgó mérőeszközök előnyeit és kihívásait. A legnagyobb kihívás a pozíció meghatározása, ezért erre alkalmas helymeghatározási megoldásokat ismertetek, külön kiemelve az ultra-szélessávú (UWB) rádió technológiát. Ezután egy mérőrendszert tervezek, amely képes UWB rádió használatával egy mérőeszköz pozíciójának meghatározására, és ugyanezen rádiókapcsolaton keresztül a mérőeszköz által végzett mérések eredményének továbbítására. A mérőeszközhöz megtervezem és megépítem a szükséges áramkört, valamint a méréshez és kommunikációhoz szükséges firmwaret. A mérési eredményeket egy központi szerver tárolja egy adatbázisban, és ezeket egy API-n keresztül a helyi hálózaton elérhetővé teszi. Az adatok kezelésére és a mérési eredmények megtekintésére egy webfelület szolgál, ahol a mérési eredmények hőtérképen láthatók.

Abstract

Nowadays, sensor systems play an increasingly important role in our everyday lives. Whether talking of smart home, smart city, or industry 4.0 solutions, data collection and data analysis are an integral part of any of these. The systems used typically consist of permanently installed sensor nodes that perform measurements at a specific location. This can often be advantageous, for example if the critical locations are known and the sensors are installed on them, but such systems are not particularly suitable for a general survey of an area. It is easier to get a comprehensive picture of the area to be surveyed by using moving devices instead of fixed nodes. In this case, far fewer measuring devices may be sufficient to survey a given area, as a measuring device can take measurements at several locations.

In my thesis I present the importance of positioning in sensor networks and the advantages and challenges of moving sensor nodes. The biggest challenge is determining the position, so I describe suitable positioning solutions, with special emphasis on ultra-wideband (UWB) radio technology. I then design a measurement system that is able to determine the position of a measuring device using a UWB radio. The system can also transmit the results of the measurements made by the measuring device via the same radio connection. I design and build the necessary circuit for the sensor node and the firmware for the measurement and communication. The measurement results are stored in a database on a central server and made available on the local network via an API. A web interface is used to manage the data and view the measurement results, where the measurement results can be viewed on a heat map.

Chapter 1

Introduction

1.1 The importance of sensor systems

With the rapid spread of the Internet of Things (IoT), wireless sensor networks have found their way into many segments of our lives. We rely on this constant flow of real-time data to make everything more comfortable, more efficient, or safer. Smart thermostats enable us to control the temperature in every individual room to be just perfect and can cut down on energy waste when we don't need it. The fourth industrial revolution (Industry 4.0) makes factories and logistics more efficient by the real-time analysis of production data and by connecting separate production sites through the cloud.

When analyzing the data gathered by sensor networks, location plays an important role. The location where a measurement was taken will change the meaning of the measured data, and will potentially alter the reactions taken. Location information provides the context for the measurement, as in different environments different values are considered normal, and different deviance may be allowed from this baseline. Location information may also help understand wireless coverage, or even gas or heat leakage.

Most sensor systems solve the problem of gathering location information by having stationary sensor nodes, whose position is stored in a central database. This can greatly reduce the nodes' complexity but severely limits the spatial resolution of the gathered data. Increasing this resolution would require an impractical amount of sensor nodes, whose locations would have to be individually determined during the initial setup of the system, thus increasing the required time and effort. An increased number of sensor nodes poses additional challenges for the communication medium, but this can soon be solved by the Massive IoT capabilities of 5G networks.

By using mobile sensor nodes, a higher sensor coverage can be achieved with fewer nodes since a single sensor node would be able to take measurements at many different locations. This would give a better understanding of the whole covered area, but mobile nodes pose new challenges. In the absence of fixed power sources, sensor nodes must be designed to operate from battery power, and

depending on application requirements as well as available battery capacity, may need to utilize careful power management. An even bigger challenge is posed by the loss of location information, thus the need for a positioning system emerges.

1.2 Positioning methods

Positioning systems serve to determine the location of a mobile device through a set of anchors. The location of the anchors must be known, with the highest possible accuracy, since any position error here will directly cause erroneous positioning. For the purposes of this thesis, only radio-frequency-based positioning methods will be discussed in detail, however many of the outlined principles are applicable to visible light-based and audio wave-based positioning with little modification. For the purposes of better illustration, the concepts will be explained in two dimensions, but the same methods are easily extended to provide a three-dimensional position.

1.2.1 Angle of arrival

The position of an object can be determined with several different methods. The most trivial concept is the angle of arrival (AOA) method, where the mobile device emits a radio signal, which is detected by at least two anchors. By determining the angle of arrival of the signal at all anchors, the position of the signal emitter can be determined through triangulation, as illustrated in Figure 1.1. The main drawback of this method is the need for an antenna array at each anchor in order to detect the angle of arrival, which is often not feasible, although recent WiFi access points often feature three or more antennas, making AOA positioning a possibility.

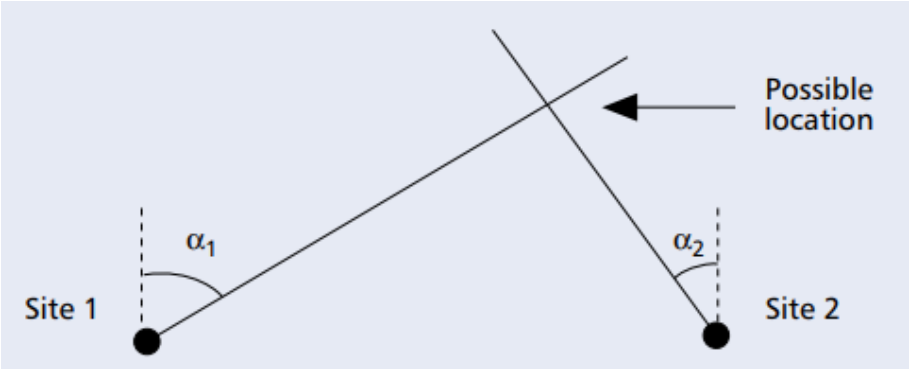

Figure 1.1: Angle of arrival positioning with two anchors — source: [18]

1.2.2 Time of arrival

When the angle of arrival cannot be determined, the positioning can rely on time measurement. The time of arrival (TOA) positioning method relies on measuring the time the radio signal takes to travel between the anchor and the mobile device, and by multiplying this time with the velocity of light, the distance between the mobile device and each of the anchors can be determined. By acquiring at least three such measurements, the position of the mobile device can be determined by finding the intersection of the range circles, as demonstrated in Figure 1.2. This method requires the clocks of all devices to be accurately synchronized, as 1 µs of offset can cause approximately 300 m error in the range measurements.

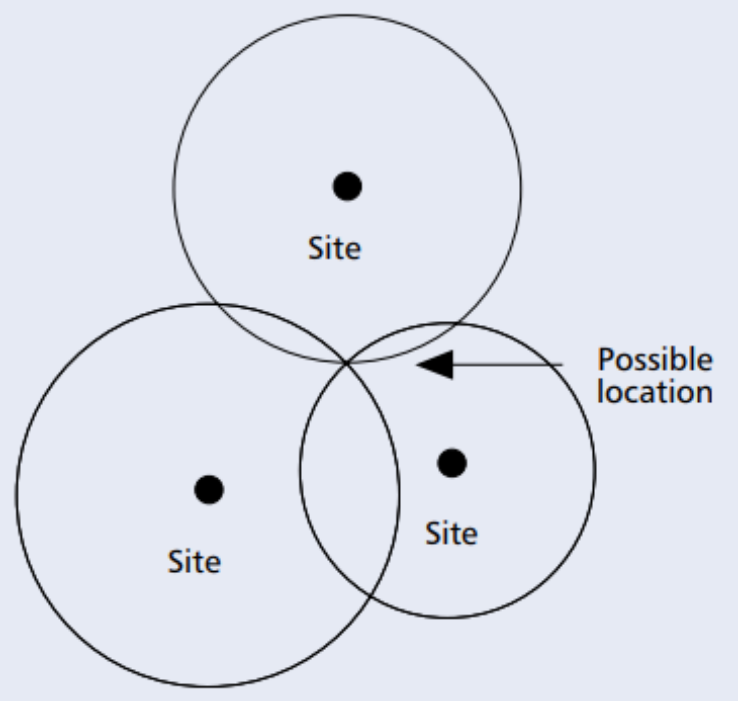

Figure 1.2: Time of arrival positioning with two anchors — source: [18]

1.2.3 Time difference of arrival

If the clock of the mobile device cannot be synchronized to the clocks of the anchors, the positioning may rely on the measurement of time difference. Positioning using the time difference of arrival (TDOA) method is done by measuring the difference of the radio signal travel times between the mobile device and each pair of anchors, thus essentially measuring the difference in the distances. In practice this usually involves the tag periodically transmitting a short message containing its identifier, and the anchors will each record the time when the message was received. The result of this process are a number of hyperbolic curves which intersect at the position of the mobile device as seen in Figure 1.3. This positioning method still requires the clocks of all anchors to be synchronized, but the clock of the mobile device can be arbitrarily out of sync. [18]

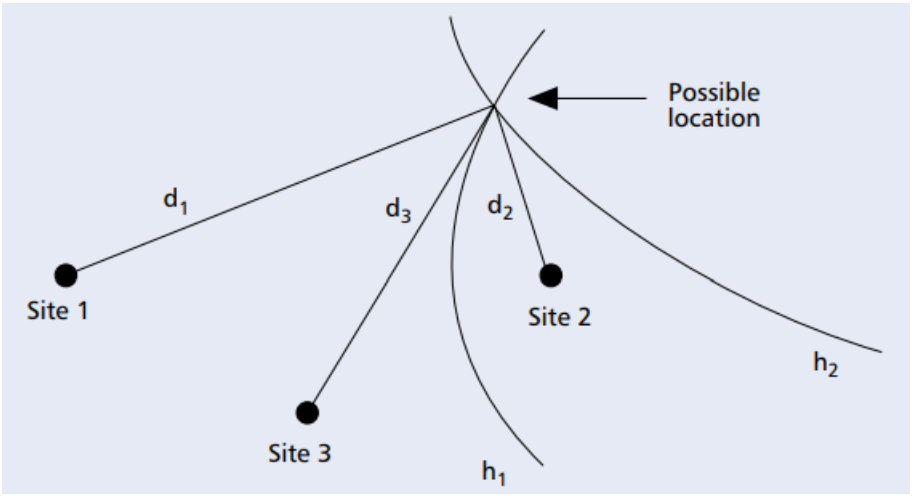

Figure 1.3: Time difference of arrival positioning with two anchors — source: [18]

1.2.4 Two-way ranging

If precise clock synchronization between the anchors is not feasible, the distance between the mobile device and each of the anchors can be determined through two-way ranging (TWR). For this method, the initiating side, which may be either the mobile device or an anchor, sends a radio signal, to which the other side replies with another radio signal and the time difference between sending the first message and receiving the reply (round trip time, T_{round}) is measured by the initiating device as illustrated in Figure 1.4. If the latency between receiving the first message and sending the reply (reply time, T_{reply}) is known, the propagation time can be determined:

$$T_{\text{prop}} = \frac{1}{2}(T_{\text{round}} - T_{\text{reply}})$$

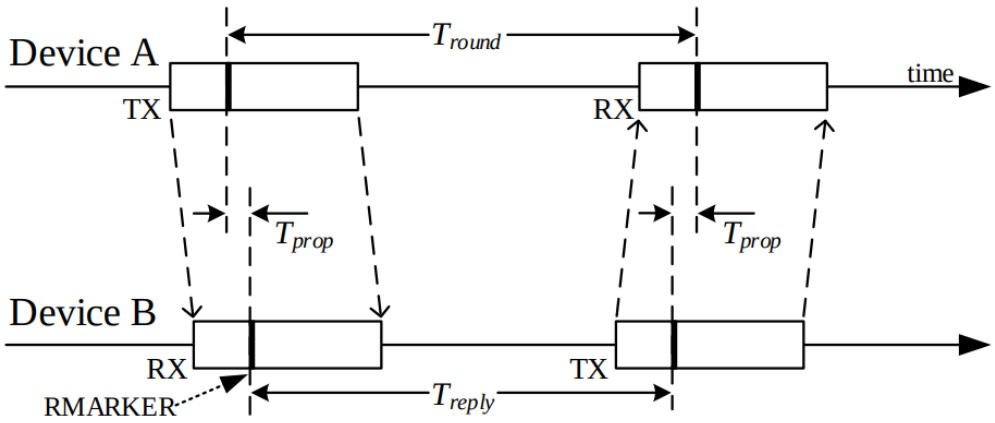

Figure 1.4: Time diagram of a single-sided two-way ranging message exchange — source: [7]

The location error can be reduced by performing two round trip time measurements in a method called double-sided two-way ranging (DS-TWR). In a DS-TWR measurement, the previously described single-sided two-way ranging (SS-TWR) is performed a second time, with the initiator and respondent roles swapped between the devices, reducing the error caused by oscillator errors between the devices. To reduce the number of required messages, the response of the first SS-TWR sequence may be used as the request of the second as illustrated in Figure 1.5 [7]. In this case, the propagation time is calculated using both round trip times and reply times:

$$T_{\text{prop}} = \frac{T_{\text{round1}} * T_{\text{round2}} - T_{\text{reply1}} * T_{\text{reply2}}}{T_{\text{round1}} + T_{\text{round2}} + T_{\text{reply1}} + T_{\text{reply2}}}$$

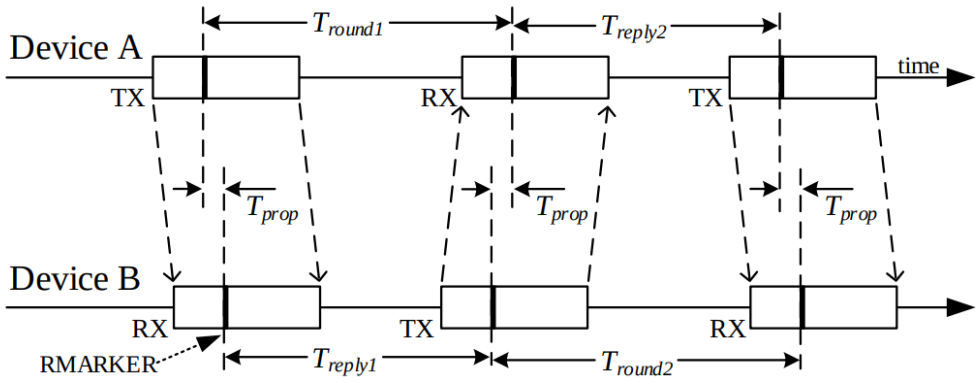

Figure 1.5: Time diagram of a double-sided two-way ranging message exchange — source: [7]

In case any form of precise time measurement is unfeasible, the location can be determined by measuring the signal attenuation between the anchors and the mobile device, thus determining the distance between the mobile device and the anchors. Unlike propagation time, radio signal attenuation is heavily influenced by any object between the two devices and the directional properties of the used antennas, this method yields a high location error.

1.2.5 Positioning systems

The positioning principles outlined in Section 1.2 may be implemented in many different positioning systems, differentiated by the available positioning anchors. Positioning systems may fall into three categories: satellite-based, terrestrial, and locally deployed systems. These systems provide different levels of accuracy and varying coverage, so the best positioning solution is highly dependent on the application requirements.

1.3 Satellite-based and terrestrial positioning systems

Globally deployed networks can be used for positioning in an outdoor environment. They have varying level of accuracy and usually utilize a radio with higher power consumption than local networks. Their main benefit is the effortless installation, since the anchors are already in operation and the operative tasks are carried out by the provider.

1.3.1 Cellular/LoRa positioning

In many cases the device to be located already carries some means of communication. When large areas need to be covered, the two most common means of communication are cellular networks. Using this communication infrastructure for positioning is an obvious simplification of the final system, since the need for a separate positioning radio is eliminated, thus reducing system cost and power consumption. This may be an important factor when it comes to mobile nodes. Positioning using cellular network is a tested, mature technology used by all our mobile phones to provide location information. Recently the IoT focused cellular protocols Narrowband-IoT (NB-IoT) and Long Term Evolution for Machines (LTE-M) have emerged as a low power and low bandwidth extension of cellular networks, making cellular positioning an even more attractive option. [6]

Another popular Low-Power Wide-Area Network (LPWAN) used by mobile IoT devices in rural environments is LoRaWAN. Its main advantages are the low cost radio modules needed for the communication, and the extremely low power consumption made possible by the proprietary LoRa modulation technique by Semtech. Semtech provides a geolocation cloud service for use with private LoRaWAN networks [1], but independent attempts have also proved the feasibility of LoRaWAN positioning on both private and public networks [1][11].

The main drawback of these positioning methods is their inaccuracy, the provided positions can have an error in the 10 m . . . 100 m range. Although this is acceptable for many applications, for precise positioning a separate system has to be used.

1.3.2 GNSS positioning

The most well-known positioning system in the world, the Global Positioning System (GPS) is a satellite-based positioning system developed for the U.S. military, and later released for civilian use with some restrictions. Although GPS is used colloquially as a general term for satellite-based positioning, most modern applications rely on other similar systems as well, e.g. GLONASS (Russia), Galileo (Europe) and BeiDou (China). These systems are collectively referred to Global

[1] LoRa Cloud™ Geolocation (retrieved: 28.11.2021)

Navigation Satellite Systems (GNSS), and by simultaneously using multiple satellite constellations the accuracy can be improved. Unfortunately for civilian use the accuracy is limited to a few meters, more accurate positioning is reserved for military use with encrypted packets. In spite of this limitation, accuracy can be drastically improved by applying correction data with the use of Real Time Kinematics (RTK), which supplements GNSS data with additional phase-correction information gathered by a stationary base station nearby, whose location must be precisely known. Using this method the accuracy of GNSS positioning can be improved to be in the 0-10cm range [12]. Figure 1.6 shows location accuracy using the same GNSS receiver with and without RTK correction, here we can observe the error decreasing from over 1.5 meters to under 25 cm

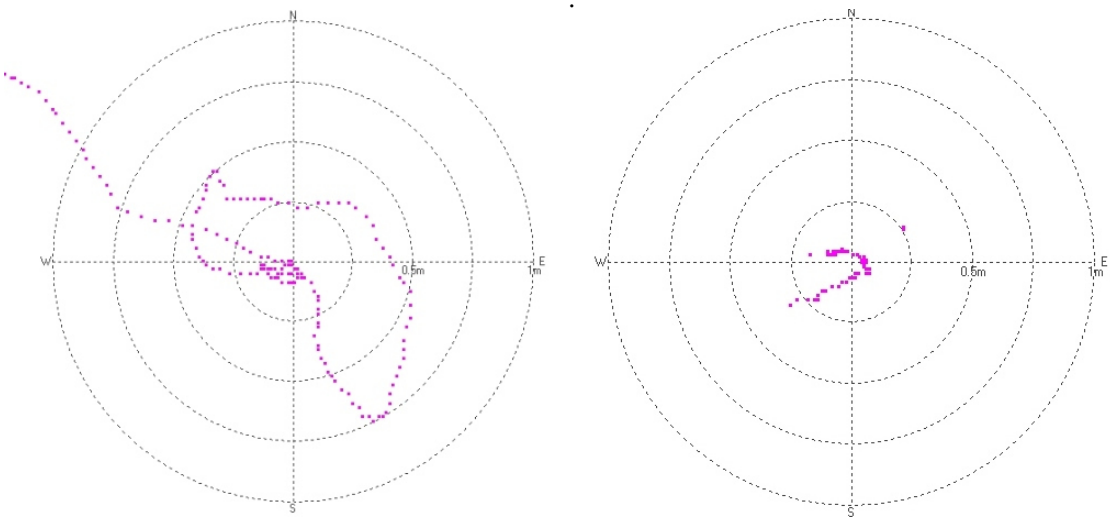

Figure 1.6: Position wandering with no correction (left) and with RTK correction applied (right) — source: Sparkfun: What is GPS RTK? (retrieved: 28.11.2021)

Although RTK supplemented GNSS positioning provides adequate accuracy for almost all positioning requirements, and the system is available for easy implementation, this method has two key downsides. The relatively large power consumption of GNSS receivers and the need for an additional data link to receive RTK correction information and transmit measurements may pose additional design challenges and increase system complexity. An even more important deficiency of this system is its unsuitability for indoor positioning. Buildings heavily attenuate the weak signals of the GNSS satellites, thus reducing the location accuracy, or even make positioning impossible.

1.4 Locally deployed positioning systems

Where the use of terrestrial or satellite-based positioning systems is impossible, deploying a local positioning system will be necessary. These systems are often referred to as Real-time Location Systems (RTLS), and their function is to determine the real-time location of objects or people - often referred to as tags. The

term RTLS does not have a strict definition, it is usually used to describe a locally deployed positioning system which is commonly used for indoor positioning. Satellite-based and cellular positioning systems are not commonly referred to as RTLS, as they are deployed globally or regionally. RTLS are most often deployed in buildings, where external signals may be attenuated to the point where they cannot be reasonably utilized, thus GNSS and cellular positioning methods are not feasible.

The deployment of RTLS involves the distribution of anchors across the area where the system is to be used. The location of these anchors have to be individually determined to establish the reference system for the RTLS. The most obvious solution is the use of stationary anchors, whose location is determined using simple distance measurement techniques, like a tape measure or laser ranging. A less obvious, and more complex system can be constructed with mobile anchors, but in this case a separate positioning system have to be used to determine the accurate position of all anchors in real time. One possible implementation of this is the use of GNSS+RTK to track the position of the anchors, while the location of the tags will be determined by the RTLS. The main advantage of such a system would be lower power consumption for the tags, and flexibility of the arrangement of the anchors. This principle is used by the Apple AirTag system, where the anchor network comprises of iPhones, and the AirTags can operate with a single CR2032 coin cell battery as the power source for an extended amount of time - "AirTag is designed to keep going more than a year on a standard battery"[2].

1.4.1 WiFi

WiFi (IEEE 802.11) networks are commonly deployed all over the world, to provide wireless internet access for mobile devices, providing thorough coverage in densely populated urban areas. This makes them an obvious consideration for use as the basis of an RTLS. Proximity information gathered from WiFi access points are already used by smartphones to enhance GNSS position. This can be realized by building a database containing the approximate locations of WiFi access points, and then continuously monitoring the nearby networks and their signal strength (RSSI). An openly available database usable for these purposes is provided by WiGLE, and the data is gathered by volunteers installing the application on their smartphones[3]. Figure 1.7 shows a map of European WiFi access points created by WiGLE, one can observe the increased density of WiFi access points in urban centers. The main advantage of using WiFi access points as RTLS anchors is that no additional hardware needs to be deployed, if a reasonably modern WiFi network with good coverage is available, but the location of the access points will have to be determined to improve accuracy. Even a single access point can provide reasonably accurate location information using a multi-antenna constellation found on many recent routers (see Figure 1.8). The positioning error can be reduced below a meter with the use of multiple access points [17].

[2] Apple AirTag (retrieved: 28.11.2021)
[3] WiGLE (retrieved: 28.11.2021)

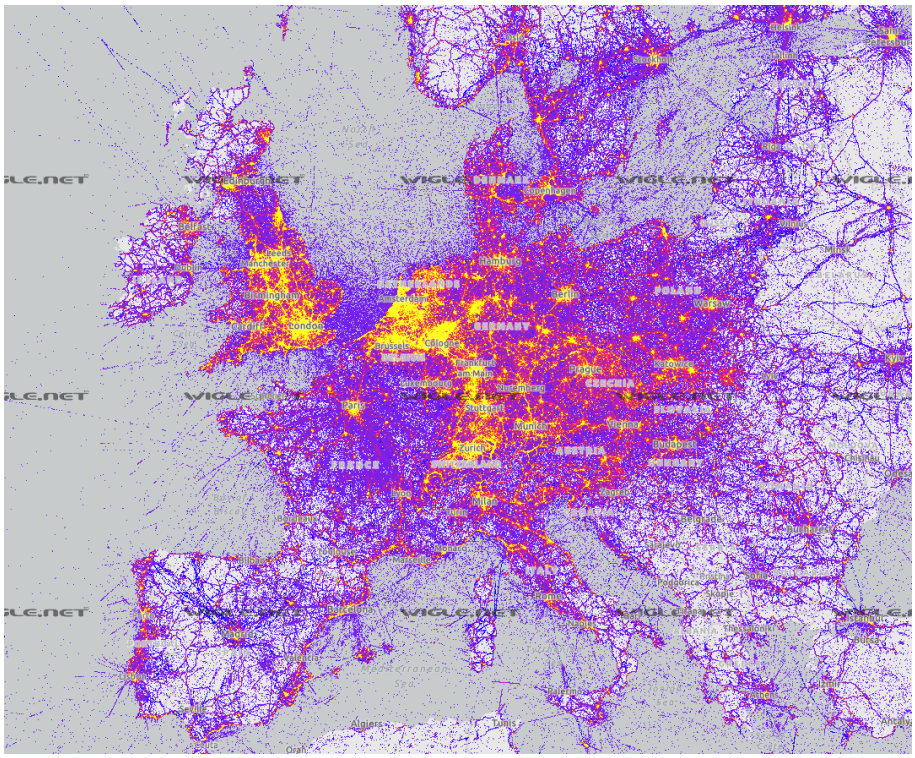

Figure 1.7: WiFi access points in Europe, yellow color indicates high density — source: `wigle.net`

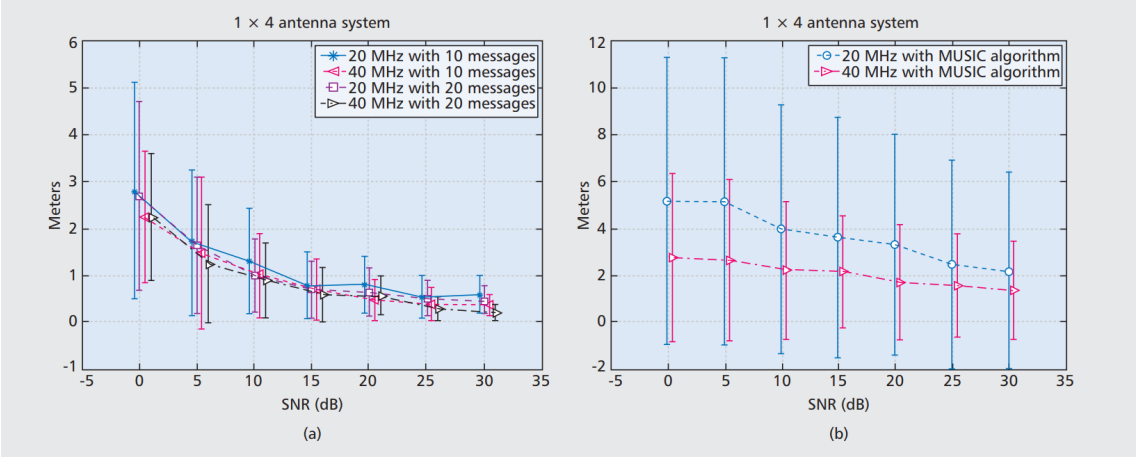

Figure 1.8: WiFi positioning accuracy with a) ToA approach proposed in a paper by Chouchang Yang and Huai-Rong Shao [17]; b) state-of-the-art ToA approach using multiple signal classification (MUSIC) approach

Unfortunately these results cannot be easy replicated in real-world scenarios, and many applications demand more accurate positioning that WiFi cannot currently provide. Another drawback of WiFi based positioning is the relatively high power consumption of WiFi radios compared to other short range networks such as Bluetooth. It should also be considered, that WiFi can be considered a critical infras-

tructure and maintenance of the positioning systems may cause disruptions to the whole network.

1.4.2 Bluetooth

Another widespread wireless technology is Bluetooth (IEEE 802.15.1) that is mostly used to communicate with wireless peripherals and audio devices. Communication with wireless, battery powered peripherals is made possible by the development of Bluetooth Low Energy (BLE). The extremely low power consumption of BLE devices is made possible by the reduced transmit power and data rate. The BLE technology enables the creation of low powered positioning anchors, referred to as beacons in BLE positioning terms, which periodically transmit their universally unique identifiers (UUID), a process called advertising. The beacons can be powered by a battery for months or even years depending on the beacon advertising interval, making the system easy to deploy. The BLE beacons are relatively simple devices, whose only role is to advertise their UUID, the position calculation in this case is performed by the tag. Because BLE achieves the low power consumption in part by decreasing transmit power, the coverage provided by each beacon is relatively low, thus the quantity of needed beacons becomes high [4]. This requirement is offset by the low price of BLE beacons, which together with the widespread adoption of Bluetooth capable devices (e.g. smartphones) makes the technology a very appealing and widely used indoor positioning method.

The main downside of BLE beacon based positioning is the low accuracy provided by the technology. Since BLE positioning is based on signal strength, any obstruction in the signal path will lead to more significant errors and the low power nature of BLE makes the technology unsuitable for ranges greater than a few meters[4]. Figure 1.9 illustrates the RSSI fluctuation of BLE beacons by distance. Any RSSI fluctuation for a given distance observed on the graph directly leads to positioning errors, one can also observe that above approx. six meters the graph flattens out, thus reasonable positioning accuracy further away from the beacons cannot be achieved.

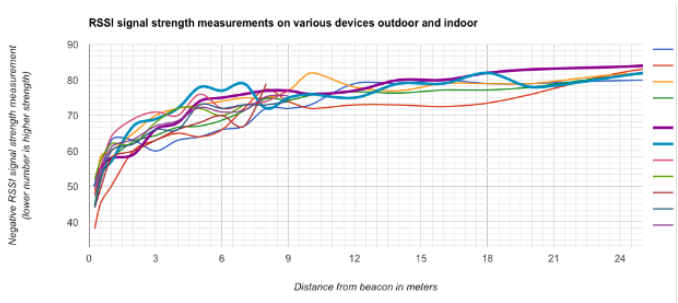

Figure 1.9: Bluetooth RSSI variation by distance — source: Locatify

[4]Locatify: BLE Beacons for Indoor positioning – Beacon limitations (retrieved: 20.11.2021)

1.4.3 Ultra-wideband radio

The use of ultra-wideband (UWB) radios in positioning systems is becoming increasingly popular, as the technology offers many benefits compared to the previously discussed WiFi and Bluetooth based positioning. UWB radio based positioning systems offer higher accuracy[5] and lower power consumption, thus making it a very appealing candidate for RTLS networks. Although the use of UWB technology is mostly limited to positioning, the technology is capable of transferring other payloads as well, making it well suited for use in wireless positioning sensor networks. For the above outlined reasons the UWB technology was selected as the base of this thesis project, and will be discussed in more detail.

1.5 Introduction to UWB radio

A radio communication technology is considered to be ultra-wideband, if its absolute bandwidth is higher than 500 MHz, or its relative bandwidth is higher than 20%, e.g. its bandwidth is greater than 20% of the center frequency of the transmission [14]. Even though the term UWB was not used until the late 1980s, the first UWB radio transmission was performed by Guglielmo Marconi in 1894 when he experimented with a spark gap radio transmitter. A spark gap transmitter produces very short burst of electromagnetic pulses resulting in a high bandwidth, as it can be recognized through the Fourier-transform. During the 1990s the UWB radio technology was utilized by the U.S. military in various radar system developments, and has been released for unlicensed civilian use in the early 2000s.

1.5.1 Advantages of UWB radio technology

One of the main error sources in radio-positioning systems is the effects of multi-path propagation of electromagnetic waves. The previously discussed radio-positioning techniques are all based on the assumption, that the measured radio wave takes a direct path between the two ends of the communication, and so measurements of this radio wave can lead to conclusions about the direct path between the endpoints. Unfortunately in a real world scenario this cannot be guaranteed, as the signal received will be a result of the interference of the radio wave following the direct path, as well as any reflected, scattered, or refracted radio waves from nearby objects or the ground, as illustrated in Figure 1.10. This results in the originally transmitted signal being received multiple times in close succession, from different angles, with different signal strengths, and different propagation times, leading to measurement errors.

When using UWB radio positioning, the affects of multi-path propagation can be minimized, as the wide bandwidth of the signal allows for short pulses, which in turn make it possible to differentiate between the direct signal and the reflected signals. By only processing the first received signal from any message, we can

[5] Locatify: Precise Indoor Positioning is finally here (retrieved: 22.11.2021)

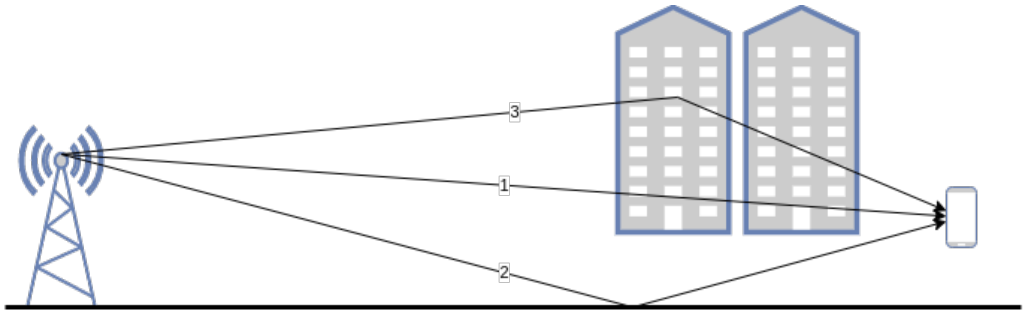

Figure 1.10: Different signal paths in multi-path propagation, 1) direct signal, 2) ground-reflected signal, 3) signal reflected from a nearby building

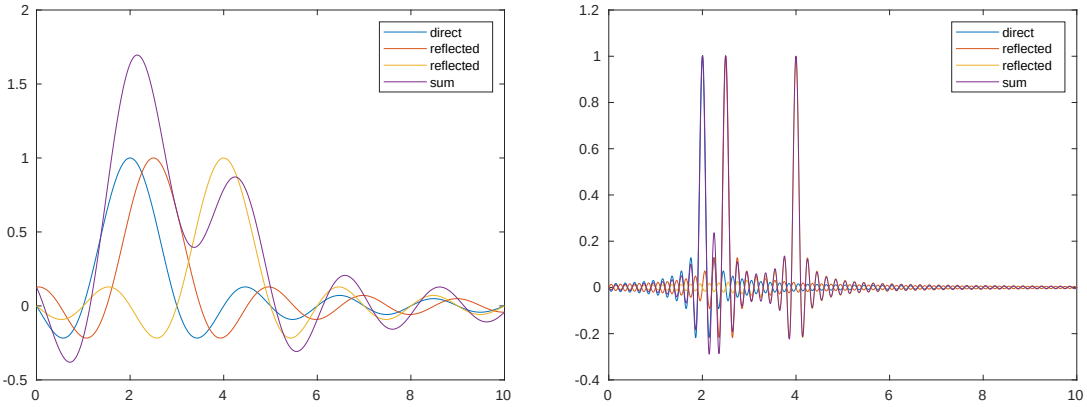

Figure 1.11: Effects of multi-path propagation on a narrow-band (left) and a wide-band (right) impulse

make sure that the calculations will lead to the properties of the actual distance between the endpoints. The effects of multi-path propagation on the signal are illustrated in Figure 1.11. The bandwidth of the wide-band impulse on the right is approximately ten times bigger than that of the narrow band impulse on the left, leading to noticeable improvement in accuracy.

Another key benefit of UWB radio technology also stems from its bandwidth. According to the Shannon-Hartley theorem, the data carrying capacity of a transmission channel can be calculated as

$$C = B \cdot log_2 \left(1 + \frac{S}{N}\right)$$

where C is the channel capacity, B is the signal bandwidth, and S/N is the linear signal-to-noise ratio. From this formula we can deduce that channel capacity increases linearly with the signal bandwidth, while assuming a constant noise floor an exponential increase in transmission power is necessary for the same increase in capacity. This makes it possible to achieve high data transfer rates coupled with low power consumption when using UWB technology.[5] In practice commercially available UWB transceivers are capable of achieving data transfer rates of around 6.8 Mb/s [7], which is between the practical data rates achievable

over Bluetooth and WiFi. The limiting factor for the maximum achievable data rate over UWB radio is the low transmit power. As UWB signals span a wide frequency range, the used frequencies often overlap with other services, thus high power UWB transmission could cause disruptions to many services operating in the same frequency range. To minimize the risk of signal interference the maximum transmit power of UWB signals is limited to -41.3 dBm/MHz [14], which results in less then 0.5 mW of transmit power for a bandwidth of 500MHz. This is a very low value, when compared to the maximum transmit power of 2.5mW of commonly used class 2 Bluetooth devices[13], or the 100 mW maximum transmit power of WiFi.

1.5.2 Application of UWB radio technology

Current applications of UWB radio technology are mainly limited to positioning systems, with a noticeable increase in adoption as both Apple and Samsung have recently included UWB transceivers in their smartphones. The previously mentioned AirTags from Apple rely on the nearby iPhones with UWB transceivers acting as positioning anchors to provide low power global coverage. UWB radio technology can also be used to unlock a car based on the proximity of a digital car key installed on a smartphone[6]. In an effort to enable the interoperability of UWB based positioning systems, the FiRa consortium was formed in 2019, and at the end of 2021 the members include the main producers of UWB radio transceivers as well as producers of consumer electronics devices[7].

The goal of this thesis project is to demonstrate the use of UWB radio technology not only for positioning, but also for simultaneous data transmission. This enables the development of a system capable of real-time mapping various sensor information over the area covered by the anchors using only a single radio system, thus decreasing system complexity.

[6] https://www.bmw.com/en/innovation/bmw-digital-key-plus-ultra-wideband.html
[7] https://www.firaconsortium.org/about/members

Chapter 2

System design

2.1 High level concept

The goal of this thesis is to design and construct a measurement and positioning system based on UWB technology. The project involves the design of a UWB tag equipped with various sensors, capable of determining its position and transmitting the location information and measurement results over the same UWB link that was used for positioning. To supplement this measurement device, a gateway device will also be constructed. The gateway will handle the stream of information coming from the measurement device and store all this data. It would also be used to provide a web frontend to ease human interaction by visualizing the stored data. This concept is illustrated in Figure 2.1, showing four positioning anchors, a sensor tag and a gateway.

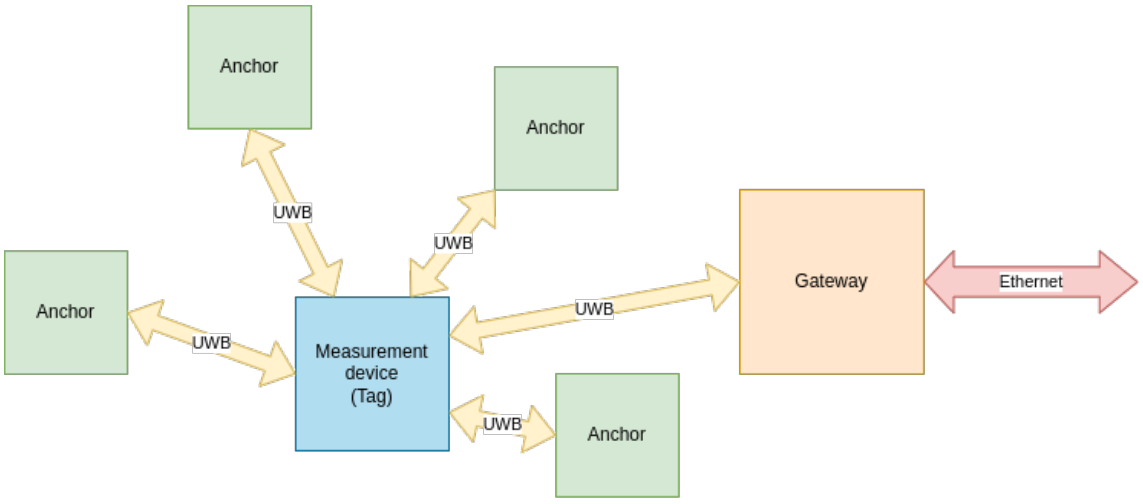

Figure 2.1: High level system concept

2.2 UWB platform

In order to continue the design process on a lower level, a UWB ecosystem had to be chosen. The two main options are the products from NXP and Qorvo (formerly Decawave). Other implementations of UWB radios do exist, but they are not as widespread (e.g. 3dB 3DB6830) or not open for individual developers (Apple U1 chip). From the two options I have eliminated NXP, since their products are either available through a non-disclosure agreement (NDA) making them unsuitable for a thesis project, or they are too expensive.

Qorvo's lineup of UWB transceivers consists of the DW1000 and the DW3xxx series, the latter getting released as recently as the first half of 2021, thus not easily available at the beginning of this project. For these reasons, the DW1000 UWB radio transceiver would be used for this project, which is a UWB radio transceiver IC controllable via the serial peripheral interface (SPI). As the DW1000 is only a transceiver, an additional microcontroller (MCU) would be required for the positioning application. Such a combination is already available from Qorvo, in the form of the DWM1001C module, which integrates the DW1000 UWB transceiver and an nRF52832 MCU from Nordic Semiconductor with antennas for both chips on a compact module, as seen in Figure 2.2. The DWM1001C also incorporates a 3-axis accelerometer from STMicroelectronics connected to the nRF52832 MCU through I^2C bus, and the DW1000 transceiver is connected to the MCUs SPI interface. The internal connections and the external inputs and outputs (IOs) of the module are illustrated in Figure 2.3.

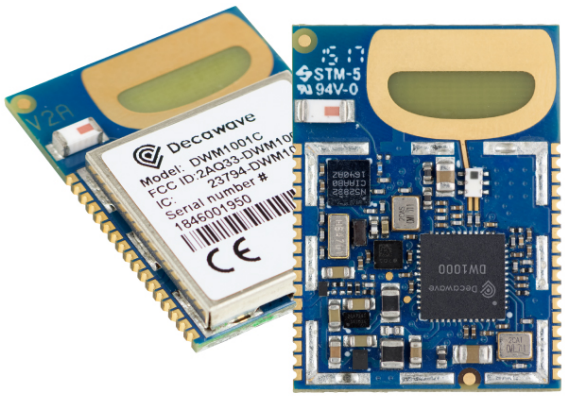

Figure 2.2: DWM1001C module with and without the metal shield, revealing the internal layout — source: DWM1001C product page on Decawave.com (retrieved: 29.11.2021.)

The DWM1001C module is supplied with a positioning system developed by LEAPS labs, called PANS. This system provides support for positioning based on two-way ranging, as well as transferring IoT data between the tags and a gateway. It provides access to the transferred data and position information through an MQTT broker running on the gateway. The structure of a PANS network is illustrated in Figure 2.4.

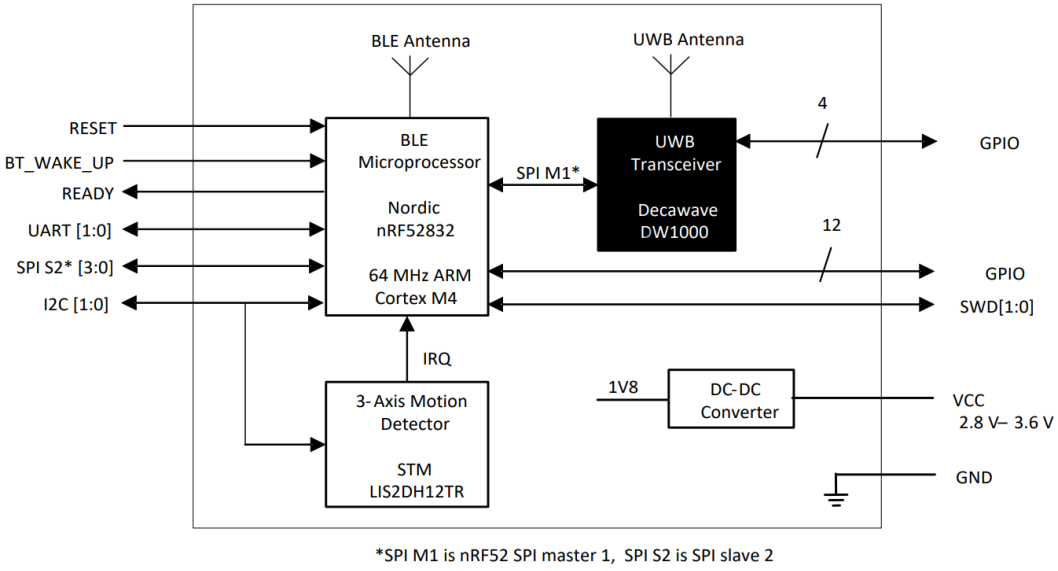

Figure 2.3: Internal block diagram of the DWM1001C module, showing the connection between components — source: [8]

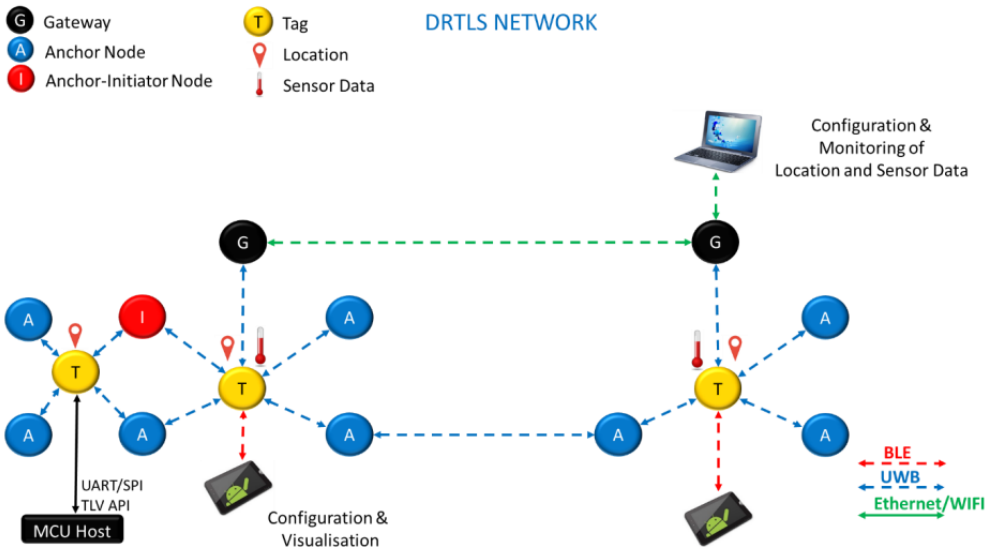

Figure 2.4: PANS RTLS network structure

2.3 Tag concept

The design of the sensor-equipped tag comprises of two parts, a hardware that accommodates the UWB radio module and any necessary peripherals, and a firmware, that manages measurement and UWB communication. As earlier discussed, the central component of the tag would be the DWM1001C module, which can handle both the UWB communication and interfacing with attached sensors.

2.3.1 Tag requirements

First the design requirements for the tag have to be determined, aside from the UWB connectivity. During the planning of the project we determined, that the following components would make the tag-to-be-designed a versatile demonstration platform, with a good selection of sensors for various measurements:

- Acceleration sensor — already present on the DWM1001C module
- Light intensity sensor
- Temperature sensor
- Relative humidity sensor
- WiFi received signal strength indication (RSSI) sensor
- Lithium battery support for portability
- Button to trigger measurements

2.3.2 Sensor selection

Since the DWM1001C module already contains an accelerometer, the main focus of the selection were the light intensity, temperature and relative humidity sensors. The accelerometer integrated on the module is an LIS2DH12 sensor by STMicroelectronics (see Figure 2.3), and is connected through I^2C bus to the nRF52832 MCU. By examining the block diagram one can observe that this same I^2C bus is also routed to external IO pads on the module. This makes it possible to connect the external sensors to the same I^2C communication bus as the internal accelerometer, leading to a more elegant system architecture.

2.3.2.1 The I^2C protocol

The I^2C protocol uses a serial data (SDA) and a serial clock (SCL) connection, and while the SDA line is bidirectional (e.g. it may be driven by either the controller or the target), the SCL line is always driven by the controller. In the I^2C protocol all communication is initialized by the controller, and only the target identified by the target address must respond. Data transfer is organized in bytes followed by an acknowledgment (ACK) bit, and the start and end of the communication are marked with the start and stop conditions. The first byte of the transfer is always transmitted by the controller, and contains the 7 bit target address (although 10 bit addressing is implemented is I^2C , it is not relevant for the purposes of this thesis). The least significant bit (LSB) of the first byte determines whether the controller will write to (0) or read from (1) the target, thus later referred to as R/W bit. The following bytes may be transmitted either by the controller (R/W bit is 0) or by the addressed target (R/W bit is 1). The structure of an I^2C data transfer is illustrated in Figure 2.5.

Sensors acting as I^{2}C targets are usually logically structured to have multiple internal registers with internal addressing. These registers may contain configuration values, in which case configuration is altered by writing the register, or may contain measurement results, and will respond with the currently stored measurement results when read. Since I^{2}C only specifies addressing the target device, internal addressing can be performed such that a write to a specific memory address would contain the address as the first data byte, and the data in the following byte (or bytes), while reading from an address would involve executing a write access first with the desired internal register address as a data byte, with a read request following it to perform the actual data transfer.

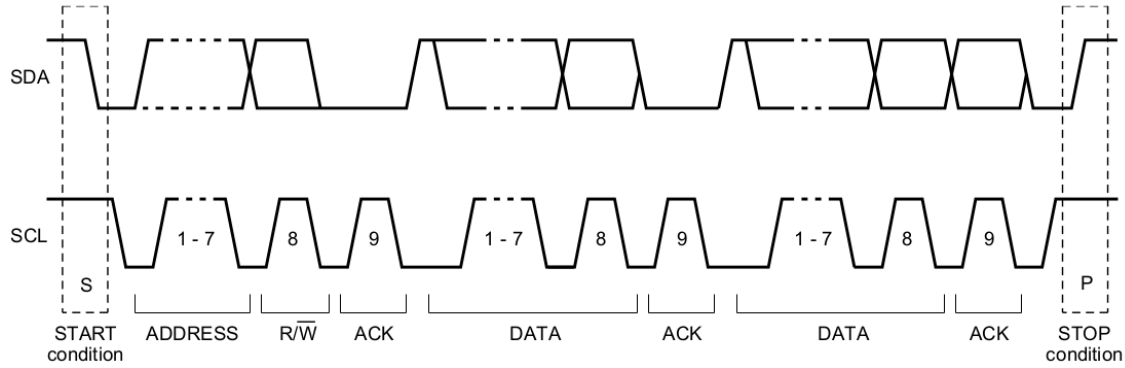

Figure 2.5: I^{2}C data transfer — source: [10]

2.3.2.2 Selection criterion

During the selection of the external sensors the main criterion was the ability to interface them over I^{2}C bus. As I^{2}C is one of the most popular interfaces for low data-rate sensors, this need still left many options to be considered. A more practical requirement was the availability of a hand-solderable package, since the tag hardware would be eventually constructed by myself. The unfortunate circumstances of the current global semiconductor-shortage posed the final, most restrictive filter, since most of the suitable sensors were unavailable for purchase at the time of design.

Considering the requirements discussed in the paragraph above the following sensors were selected:

2.3.2.3 Vishay VEML7700

The Vishay VEML7700 is a high accuracy ambient light sensor with I^{2}C interface, and a simple pad layout, its external appearance and internal structure can be seen in Figure 2.6. The sensor is capable of providing ambient light levels in Lux, which makes absolute measurements possible. The VEML7700 features a 16 bit analog to digital converter (A/D converter), and configurable gain (0.125x to 2x) and integration times (25ms to 800ms), leading to a high dynamic range with a maximum measurable light level of 120000 lx and a minimum resolution of 0.036

lx. Accuracy is further increased by temperature compensation using the integrated temperature sensor.

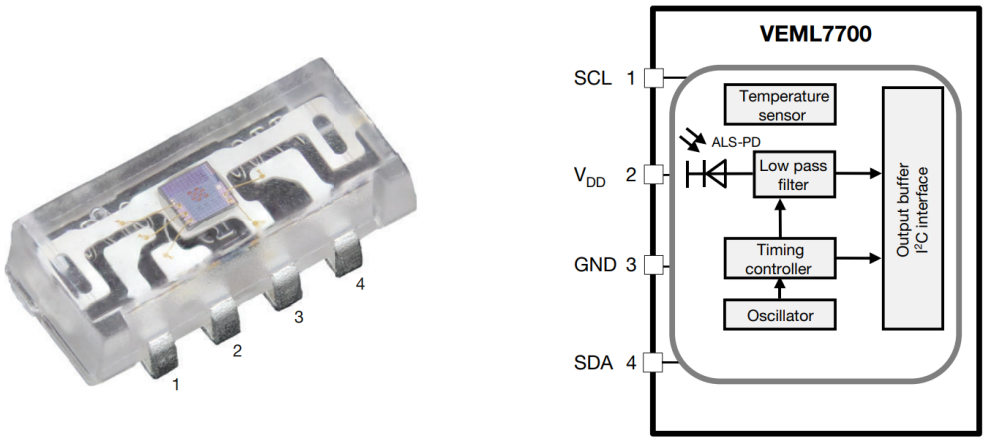

Figure 2.6: The VEML7700 sensor, in its transparent plastic mold (left) and the sensors internal structure on a block diagram (right) — source: [16]

2.3.2.4 TE connectivity HTU21D

The HTU21D is a digital relative humidity and temperature sensor produced by TE connectivity. It features an I^2C interface for control and measurements. It is capable of measuring relative humidity through a 12 bit A/D converter with a typical accuracy of ±2%. Temperature measurement is performed using a 14 bit resolution A/D converter, with a typical accuracy of ±3%. The accuracy fluctuation over the allowed humidity and temperature range are shown in Figure 2.7. The packaging of this sensor is not particularly suitable for hand soldering, but it is also available on a breakout module, that makes it easier to handle. Using the module comes with the added benefit of better thermal isolation between the sensor and the main circuit board, thus reducing the effect of nearby components' heat on the measurement.

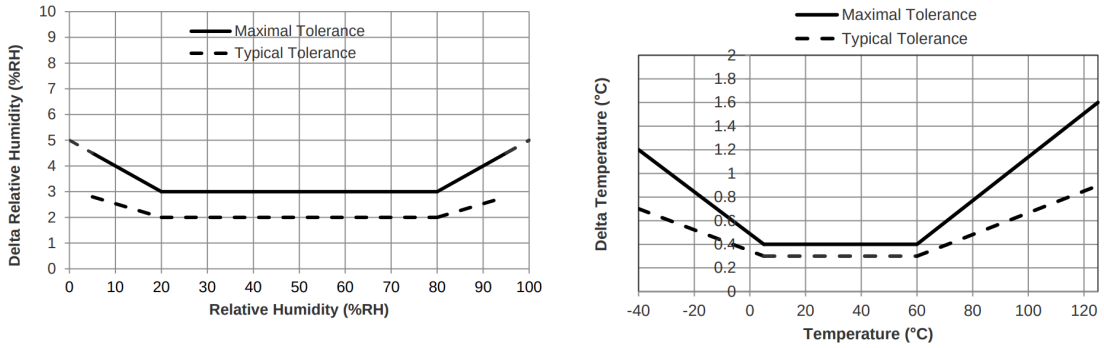

Figure 2.7: Relative humidity measurement tolerance of HTU21D over the measurable humidity range (left) and temperature tolerance over the allowed temperature range (right) — source: [2]

2.3.3 WiFi RSSI measurement

The measurement of WiFi RSSI is not as trivial as temperature, humidity and light intensity, since no single sensor exists with this purpose. For the other measurements the use of a dedicated sensor was the best solution, for WiFi RSSI the sensor had to be constructed as well. For this task the Espressif ESP8266 MCU was selected.

2.3.3.1 Espressif ESP8266

The Espressif ESP8266 is a very popular MCU with built-in WiFi connectivity. It is based on a 32 bit Tensilica L106 CPU operating at speeds up to 160 MHz and its built-in radio supports the 802.11 b/g/n standards in the 2.4 GHz frequency band[1]. Similar to the other sensors, the ESP8266 would also act as an I^2C target[2] device, so that the interfacing with all sensors would be performed in a uniform manner. The ESP8266 unfortunately does not feature an I^2C hardware peripheral, so the I^2C communication protocol would have to be implemented by bit-banging, so the MCU would react to GPIO interrupts, and in response would change a GPIO output with the correct timing, according to the I^2C protocol. Although it seamed to be feasible to bit-bang the I^2C protocol with the ESP8266 MCU as its 160 MHz maximum CPU clock frequency is multiple orders of magnitude higher than the standard 100 kHz operating frequency of I^2C , I have decided to conduct some preliminary testing in order to determine if the interrupt and GPIO handling latency were low enough to react to I^2C clock edges in time. For 100 kHz I^2C clock frequency the compound latency would have to be under 5 µs, the half of the I^2C clock period. This requirement stems from the fact that output data has to be set on the falling edge of the clock, and input data has to be read on the rising edge. The test program consisted of a single interrupt service routine (ISR) triggered by the falling edge on an input line. In the ISR an output would be set to high state, then back to low state. The test was conducted using a logic analyzer with 12 MHz sampling frequency, which would result in a time resolution of 83.3 ns, short enough to determine the latency of the reaction. The result of the experiment is shown in Figure 2.8. The measured latency of 5.5 µs unfortunately makes the bit-banging method unfeasible.

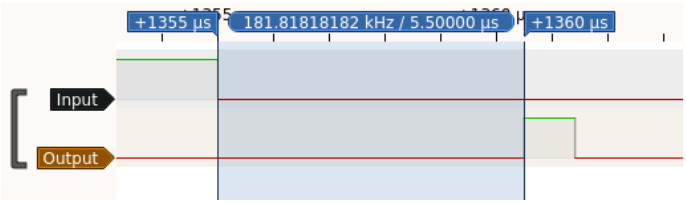

Figure 2.8: Reaction latency to falling edge input interrupt on ESP 8266

[1] Espressif ESP8266 data sheet

[2] I^2C master and slave devices have been renamed to controller and target in the 7th revision of the I^2C specification [10].

2.3.3.2 Espressif ESP32

After determining that an I^2C target device implementation on the Espressif ESP8266 is unfeasible, a more recent WiFi capable MCU by the same manufacturer was considered. The Espressif ESP32 is based on a dual core Tensilica Xtensa LX6 CPU running at a maximum clock frequency of 240 MHz and it has integrated 802.11 b/g/n capable WiFi as well as a Bluetooth 4.2 radio support. The main improvement from the perspective of this project is the presence of a hardware I^2C peripheral with I^2C target capabilities, so the GPIO interrupt based bit-banging solution would no longer be necessary. These components are illustrated on the internal block diagram of the ESP32 MCU in Figure 2.9. Since the ESP32 is more recent than the ESP8266, it is supported by the ESP-IDF framework from Espressif, which provides drivers for the on-chip peripherals, as well as a FreeRTOS based real-time operating system (RTOS). As the CPU on the ESP32 is dual core, Espressif has developed a symmetric multiprocessing (SMP) scheduler for FreeRTOS, a feature which was only recently implemented in the mainline FreeRTOS project with the help of Espressif, among others.

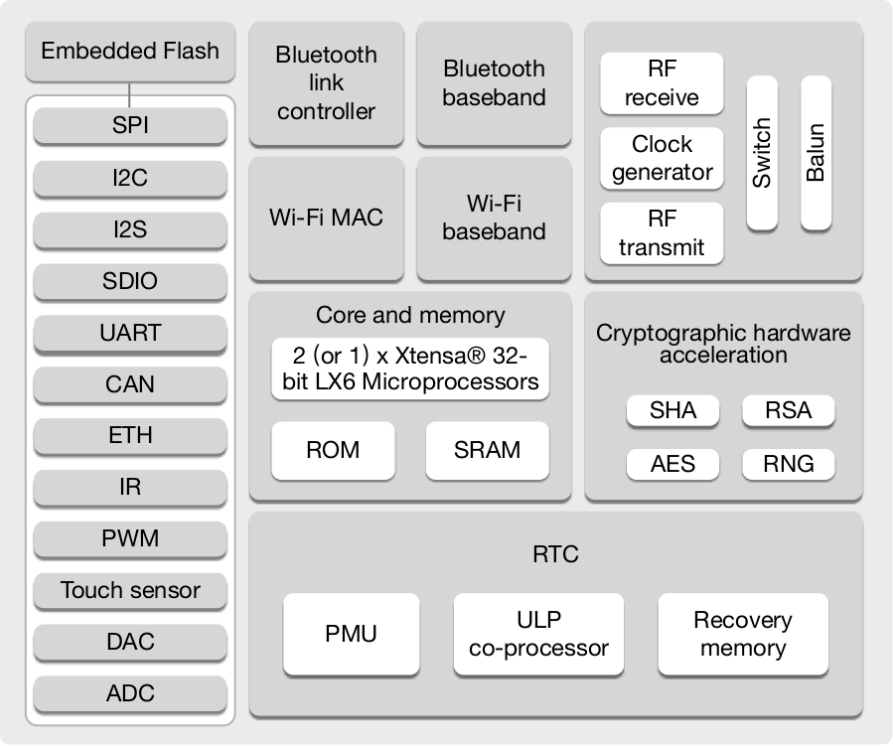

Figure 2.9: Espressif ESP32 block diagram — source: ESP32 data sheet

2.3.4 Measurement trigger

At this point all the sensors have been selected, but there is no way to activate the tag. To trigger measurements a momentary push button was connected to

a GPIO of the nRF52832 MCU. Pressing the button will trigger an instantaneous measurement and immediately transmit the results as they become available.

At this point the measurement and communication functions of the tag are complete, thus an overview is provided in Figure 2.10. Note that the I^2C addresses displayed by the sensors are shown in binary 7 bit format, as in some cases the address indicated in the data sheet is the address shifted left by one bit (e.g. expanded with the R/W bit as zero) leading to possible confusion. To facilitate programming of the nRF52832 and ESP32 MCUs the appropriate communication and reset signals are also indicated.

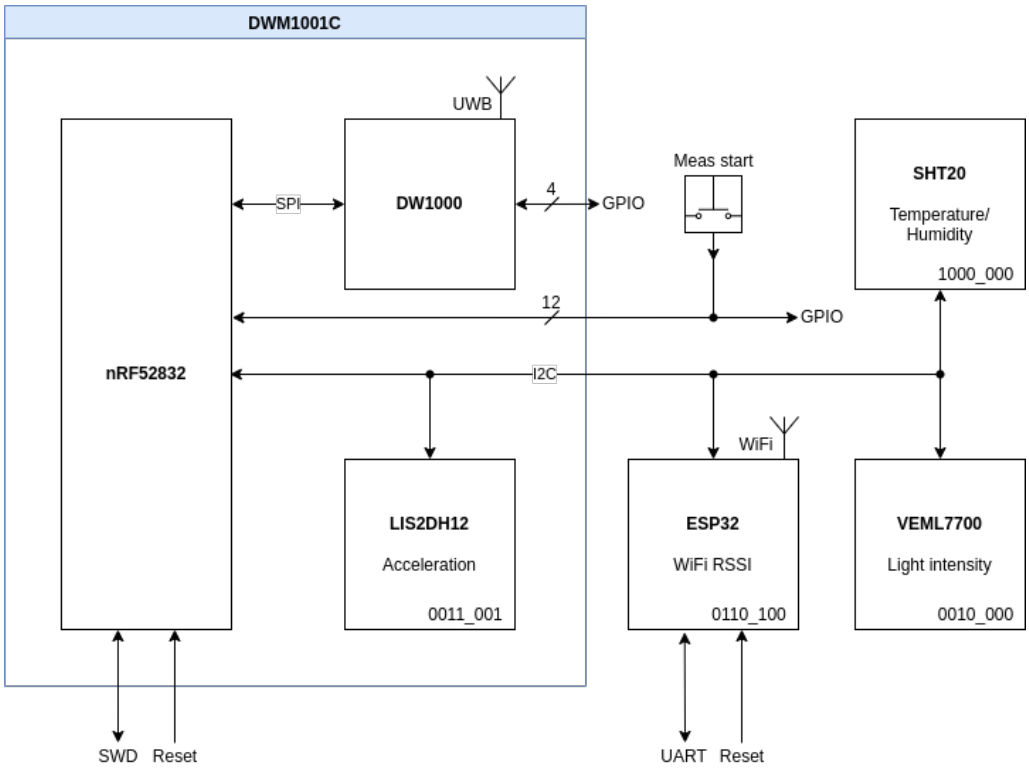

Figure 2.10: Block diagram of the sensor tag, showing the main internal components of the DWM1001C module

2.3.5 Power supply

The tag has to be portable, thus the need for a battery is evident. Due to its high power density, a lithium-polymer battery was chosen, but with lithium batteries care must be taken during charging and discharging, as they can react quite violently to accidental overcharge, over-discharge, and over-current conditions. As these limits are specific to battery models, I have decided to use batteries with built-in protection circuitry, hence any connected battery would be safely protected according to its specifications. As an additional layer of protection a reverse-polarity protection circuit and a self resetting polymeric fuse (poly-fuse) was also added.

To handle the charging requirements of the battery a lithium battery charger IC was used, and a USB type C connector was selected as the charging connector. During charging the power source will be automatically changed from the battery to USB power. Since the voltage from the USB connector is 5 volts, and the battery voltage can be as high as 4.2 volts, a regulator must be used to power the 3.3 V components. In order to increase efficiency and battery life, a switching regulator was used instead of a linear low dropout (LDO) regulator. As the battery voltage will not go under 3 volts and every component can safely operate at this voltage the boosting of the battery voltage is not necessary, so a buck converter was used for the power supply.

2.4 Gateway and anchor concept

With the tag design outlined, we must focus on the two missing components of the positioning and measurement system. One of the main advantages of the DWM1001C module is its capability to behave not only as a tag, but also as an anchor or a gateway-bridge node.

2.4.1 Anchors

As the anchors do not require any specific external circuitry to be connected to the DWM1001C module besides a power supply, I have opted to use the DWM1001-DEV development board available for the module pictured in Figure 2.11. The DWM1001-DEV features a buck converter to enable powering the device through its micro USB connector, and is equipped with a JLink debug probe used for programming and debugging the nRF52832 MCU on the DWM1001C module, while providing a USB-UART link to communicate with the MCU. The DWM1001-DEV also features a lithium battery charger to enable powering the module from a battery, but since the anchors will remain stationary this feature will not be used for the purposes of this project.

2.4.2 Gateway

The gateway handles all measurement results and location information sent by the tag, and has to be able to store and manage all data. The PANS RTLS solution used by the system supports the collection of sensor data through a DWM1001C module connected to a Raspberry Pi 3 model B. The previously mentioned DWM1001-DEV development board conveniently also features a Raspberry Pi compatible 26 pin header, so it can be directly connected to the GPIO header on the Raspberry Pi 3.

As the PANS driver for the Raspberry Pi 3 is based on the Raspbian Linux distribution, the functions of the gateway will be implemented using software available for Linux. The PANS driver makes any incoming data available through an MQTT broker, which only provides temporary storage for the messages. As permanent

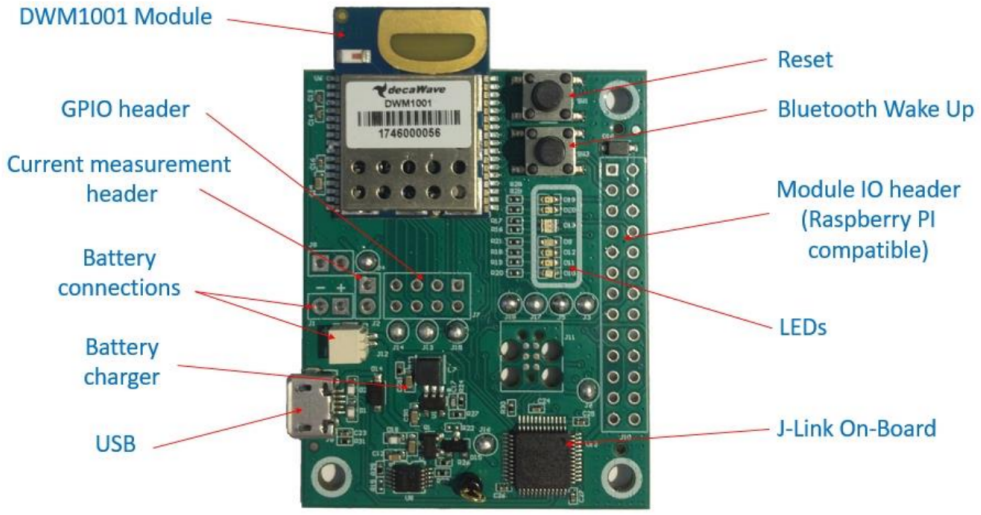

Figure 2.11: The DWM1001-DEV development board used as anchors and gateway — source: DWM1001-DEV data sheet

and structured storage is needed, all data will be stored in a relational database. For this purpose the MariaDB server was selected, as it is an open-source alternative to the popular MySQL server. The visual interface for the gateway will be served through an HTTP server, and the web interface will be capable of initializing a new measurement sequence, and visualize the data points from a selected measurement sequence on a heatmap. There will also be an option to export the data points of a measurement sequence for further analysis.

2.5 Complete system

The block diagram of the whole is shown in Figure 2.12, showing the DWM1001-DEV modules as the anchors, and in the gateway with the Raspberry Pi 3. The main internal components of the tag are also visible.

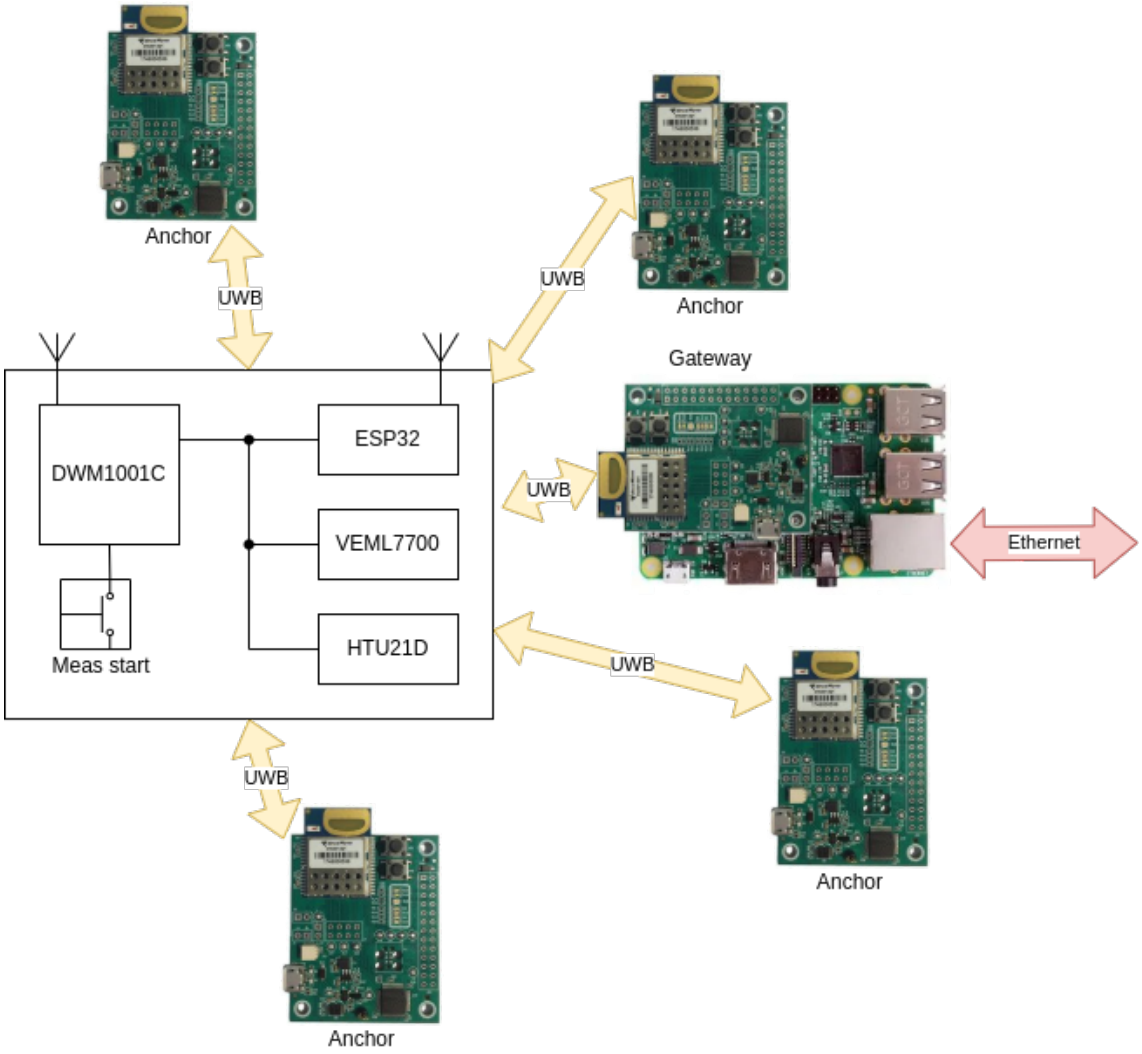

Figure 2.12: Block diagram of the complete system

Chapter 3

Tag firmware

With the components selected for the tag, the circuit was assembled as a prototype using the DWM1001-DEV development module and breakout boards for the ESP32, the HTU21D temperature and humidity sensor, and the VEML7700 ambient light sensor. Having this functional prototype was key to starting the development of the firmware, as it allowed me to easily alter the connections without having to get a new circuit board manufactured, thus greatly speeding up the development.

The development of the firmware can be divided into three parts, based on the required functionality: the tag has to measure the selected environmental conditions, it has to determine its position, and it has to be able to transmit these results to the gateway.

3.1 Positioning

The previously mentioned PANS RTLS stack was used to handle the UWB transceiver. This software stack is supplied with the DWM1001C module and is capable of acting both as an anchor and a tag, and in anchor mode, it can even be used as a bridge in a gateway. This makes it possible to create a complete positioning solution with every component using the same core module, greatly simplifying the architecture. In both modes, the PANS stack can provide control interfaces over UART and Bluetooth Low Energy, utilizing the wireless capabilities of the used nRF52832 MCU. When configured in tag mode, the PANS stack allows us to determine the position of the device, using the SS-TWR positioning technique, as described in Section 1.2.4. Although the DW1000 UWB transceiver is capable of carrying out TDOA measurements (explained in Section 1.3), due to the strict time synchronization required between all anchors, it is not supported by the PANS stack. For the purposes of this system, SS-TWR provides a reliable and accurate source of positioning, and even though it generates more radio traffic than TDOA, these added packets can be utilized to carry measurement data. As the PANS RTLS is based on IEEE 802.15.4, a PANID (network identifier) has to

be chosen for all participating nodes, and an anchor has to be selected for the initiator role, which controls the formation and timing of the network.

3.1.1 SS-TWR implementation

The single-sided TWR ranging is used in the PANS RTLS to determine the position of the tags. In order to enable the presence of multiple tags, the time-division multiple access (TDMA) technique is used to provide every node access to the radio channel. With TDMA, time is divided up into frames (referred to as superframes in PANS terminology), and every frame is divided up into time slots. During each time slot, a different message may be transmitted, usually by different devices. The structure of a PANS superframe is shown in Figure 3.1. The superframe consists of 30 0.5 ms time slots for anchor beacon messages, 2 service message slots, 15 3 ms time slots for TWR packets, and 30 0.5 ms time slots for bridge node beacon messages. The remaining time in the superframe is used as idle time to signal the switch between the different message types and a longer idle time between the superframes. The anchor beacon messages are used by the tags to determine the network topology and select the anchors used for positioning. The two service message slots are used for either Almanac messages, network service messages, or anchor uplink/downlink data. Each TWR slot contains a single TWR request transmitted by the tag and up to four replies from the anchors selected during the anchor beacon time slots. For the positioning to succeed, the tags need at least three anchors in range, but positioning accuracy can be improved if four anchors reply to the TWR request. At the end of each TWR frame, up to 34 bytes of IoT data can be exchanged. Finally, the bridge node beacon messages are used to notify tags about available downlink IoT data [9]. The ranges obtained during the TWR process and the anchors' position obtained from the anchor beacon messages are used to calculate the position of the tag.

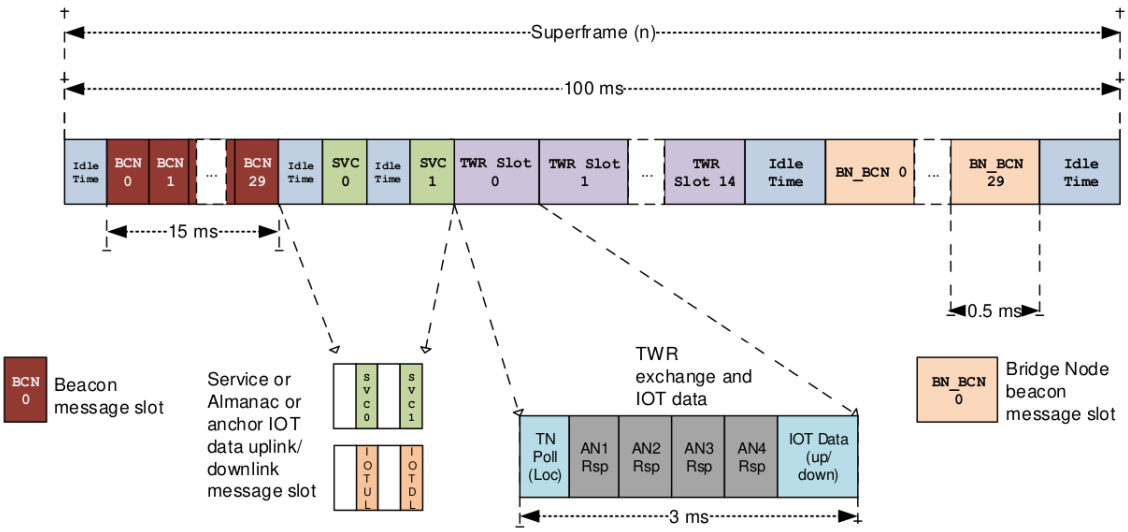

Figure 3.1: Structure of a PANS superframe — source: [9]

3.2 Measurement

The measurement capabilities of the tag are what set this system apart from a general UWB-based RTLS. As described in Section 2.3, the tag is equipped with an ambient light sensor, a temperature and relative humidity sensor, and a WiFi RSSI measurement module, in addition to the accelerometer already present on the DWM1001C module. As all these sensors are connected through a single I^2C bus, the main task of the firmware is to write the suitable measurement setting into the sensors' registers and read out the measurement results.

3.2.1 Light intensity

The VEML7700 digital ambient light sensor was selected to measure light intensity. This sensor features a 16 bit A/D converter with configurable gain and integration time, in effect providing two settings to control the step size of the D/A converter. The gain can be configured to the discrete values of 1/8, 1/4, 1, and 2, while the integration time can be set to 25 ms, 50 ms, 100 ms, 200 ms, 400 ms, and 800 ms. These settings allow for high range light intensity measurement, from 0.0036 lx step size with 236 lx maximum illumination to 1.8432 lx step size with 120796 lx maximum illumination as shown in Tables 3.1 and 3.2 [15].

	Gain 2	**Gain 1**	**Gain 1/4**	**Gain 1/8**
IT = 800 ms	0.0036	0.0072	0.0288	0.0576
IT = 400 ms	0.0072	0.0144	0.0576	0.1152
IT = 200 ms	0.0144	0.0288	0.1152	0.2304
IT = 100 ms	0.0288	0.0576	0.2304	0.4608
IT = 50 ms	0.0576	0.1152	0.4608	0.9216
IT = 25 ms	0.1152	0.2304	0.9216	1.8432

Table 3.1: VEML7700 light intensity resolution with every integration time (IT) and gain combination, in lux — source: [15]

	Gain 2	Gain 1	Gain 1/4	Gain 1/8
IT = 800 ms	236	472	1887	3775
IT = 400 ms	472	944	3775	7550
IT = 200 ms	944	1887	7550	15099
IT = 100 ms	1887	3775	15099	30199
IT = 50 ms	3775	7550	30199	60398
IT = 25 ms	7550	15099	60398	120796

Table 3.2: VEML7700 maximum possible illumination with every integration time (IT) and gain combination, in lux — source: [15]

In order to utilize the full range of the sensor while measuring with the highest possible resolution, an automatic ranging algorithm was developed loosely based on the automatic ranging method described in [15]. The ranging process starts by configuring the lowest gain (1/8) and shortest integration time (25 ms), thus achieving the highest possible maximum illumination range, and taking a measurement. If the result is below a set threshold, the process is repeated with higher gain, then longer integration time values, while a suitably high result is achieved, or the maximum of the gain and integration time settings is reached, as illustrated in Figure 3.2. The reason for increasing the gain first and only raising integration time when the highest gain setting is reached is to keep the measurement time as low as possible, as higher integration times result in longer measurement times. The limit for the light intensity used by the automatic ranging algorithm was chosen to be 10000. Since the sensor uses a 16 bit A/D converter, the returned values are between 0 and 65535, and the biggest increase between subsequent gain stages is 4x (from 1/4 to 1), and with the 10000 limit it can be reasonably assured, that the light intensity deemed too low would still not be out of range using the next gain setting. This algorithm ensures that any light level below the measurement limit of the sensor will be measured with the highest possible resolution.

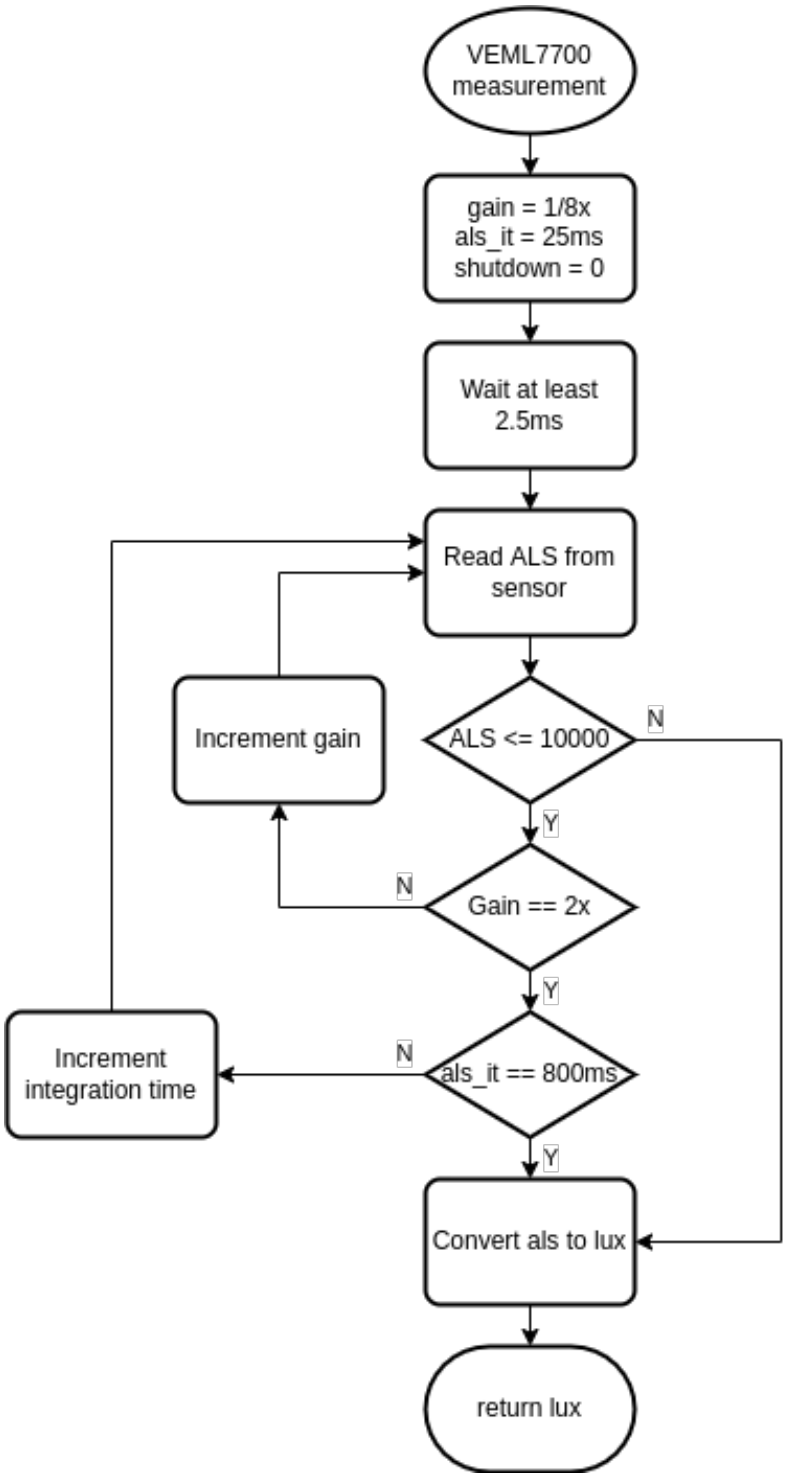

Figure 3.2: VEML7700 automatic ranging algorithm

3.2.2 Temperature and humidity

For the measurement of temperature and relative humidity, the HTU21D sensor was selected. The HTU21D is equipped with a 12 bit A/D converter for the relative humidity sensor and a 14 bit A/D converter for the temperature sensor. The resolution of the A/D converters can be configured lower as the maximum in order to shorten measurement times, but as accuracy is more important in our current use case than measurement speed considering the lower resolution would only reduce measurement times by a total of 56 ms, the highest possible resolution was chosen for both converters. The results of the measurements are provided in a left-justified format and have to be scaled appropriately to obtain true measurements, as described in the data sheet [2].
Relative humidity scaling (RH_{ADC} is the value returned by the sensor, RH is the relative humidity in %):

$$RH = -6 + 125 \cdot \frac{RH_{\text{ADC}}}{2^{16}}$$

Temperature scaling (T_{ADC} is the value returned by the sensor, T is the relative humidity in °C):

$$T = -46.85 + 175.72 \cdot \frac{T_{\text{ADC}}}{2^{16}}$$

As the relative humidity sensor is calibrated for 25 °C temperature, the result has to be further compensated:

$$RH_{\text{compensated}} = RH - 0.15 \cdot (25 - T)$$

3.2.3 Acceleration

The LIS2DH12 digital three-axis accelerometer comes integrated on the DWM1001C module, and features 16 bit A/D converters for each of the three axes. The original purpose of the accelerometer is to be used by the PANS stack to detect if the tag was stationary, allowing a reduced position update rate when position is not changing. Although this feature is a great way to reduce the average power consumption of the tag, it has been disabled in the firmware as data transfer uses the positioning frames, thus a lower position update rate would result in a lower data transfer capacity. In the current application the sensor is used as a simple three-axis accelerometer. After enabling all axes and setting the resolution and update rate the acceleration values can be read out from the sensors registers. In order to obtain the actual the value has to be scaled according to the data sheet:

$$a = \frac{a_{\text{ADC}}}{16} \cdot 12$$

Where a is the acceleration in units of mg (e.g. 1/1000 of Earth's gravitational acceleration), a_{ADC} is the value provided by the sensor. As the result is left-justified, the division by 16 provides the actual 12 bit result which at ±16 g range has to be multiplied by 12 due to the gain of the sensor.

3.2.4 WiFi RSSI

As mentioned previously, the WiFi RSSI sensor had to be created using an ESP32 MCU, as no dedicated sensors are available for this task.

3.2.4.1 ESP32 firmware requirements

With a suitable platform selected for the WiFi RSSI sensor, the measurement firmware had to be designed. Since this firmware has to fulfill only a single task, the requirements are quite simple. It has to:

- act as an I^2C target,
- scan for nearby WiFi networks upon request over I^2C ,
- and make the scan results available through I^2C .

The device must be able to supply the SSID, RSSI, and operating channel of maximum 32 nearby WiFi networks. According to the IEEE 802.11-2020 standard, the maximum allowed length for a WiFi RSSI is 32 octets (Section 9.4.2.2), the maximum channel number is 14 (table 15-6), while for RSSI the only restriction is that "[RSSI] shall be a monotonically increasing function of the received power"[3], thus the actual range of RSSI is manufacturer-dependent, but measurements with the same device provide a good understanding of signal strength differences. In case of the ESP32 MCU RSSI is a signed integer between -100 and 0. These parameters will be necessary to determine communication parameters and memory allocation.

3.2.4.2 ESP32 firmware design

In order to perform the functions outlined in 3.2.4.1, the program will consist of three main parts:

- The main entry function will initialize the peripherals, the I^2C interrupt handler, and the WiFi scan task
- The WiFi scan task will perform a WiFi scan when unblocked
- The I^2C interrupt handler will interpret commands from the I^2C controller (the nRF52832 MCU in this case), and reply to them, or unblock the scan task upon a scan request

At startup, the main function initializes the I^2C peripheral as a target, enables the I^2C receive interrupt with the correct ISR, and configures the I^2C target address. This was selected to be `0x34` so as to avoid any address collision with the previously selected sensors. The WiFi peripheral is also initialized by the main function in WiFi station mode and an RTOS task is created for the WiFi scan function. This process is illustrated on the left side of Figure 3.3.

The WiFi scan task waits in a blocked state for a notification, and when a notification is received, performs a scan of nearby access points. At the end of the scan, the SSID, the RSSI, and the channel of maximum 32 access points will be stored, sorted in descending order by RSSI. The number of found access points is also stored, the end of the scan is signified with a flag and the task re-enters blocked state. This process is illustrated on the right side of Figure 3.3.

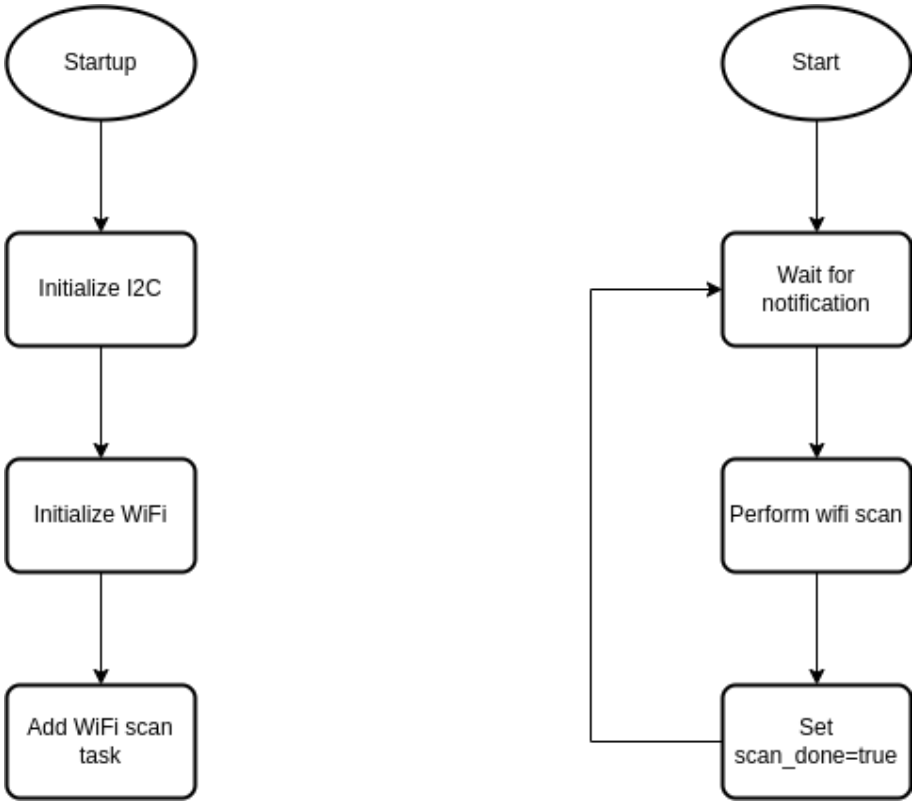

Figure 3.3: Flowcharts of the ESP32 main function (left) and the WiFi scan task (right)

The I^2C ISR is invoked every time a data is received over the I^2C bus. In order to access the WiFi scan results, start a WiFi scan, or query the status of the device, the internal register logic described in 2.3.2.1 was implemented. The registers are described in Table 3.3. When invoked, the I^2C ISR will clear the I^2C transmit buffer as a safety measure in order to prevent any misinterpretation caused by left over bytes from earlier transactions, then decode the register address, and following the register description will either send a notification to the scan task in order to start a scan, or place the appropriate reply into the transmit buffer. Before returning, the ISR must also clear the I^2C interrupt flag, so as to avoid handling the same event multiple times. This function is illustrated in Figure 3.4.

Register address	Function
0x00–0x19	SSID of networks 0-31 as a 32 element array of unsigned 8 bit integers, should be followed by a 32 byte read access
0x20–0x39	RSSI of networks 0-31 as a single signed 8 bit integer
0x40–0x59	Channel number of networks 0-31 as a single 8 bit unsigned integer
0xF0	Scanning status, number of found networks if a scan has already been performed, or 0x80 otherwise
0xF1	Trigger, any access to this register will start a WiFi network scan

Table 3.3: Register description of the ESP32 WiFi RSSI sensor

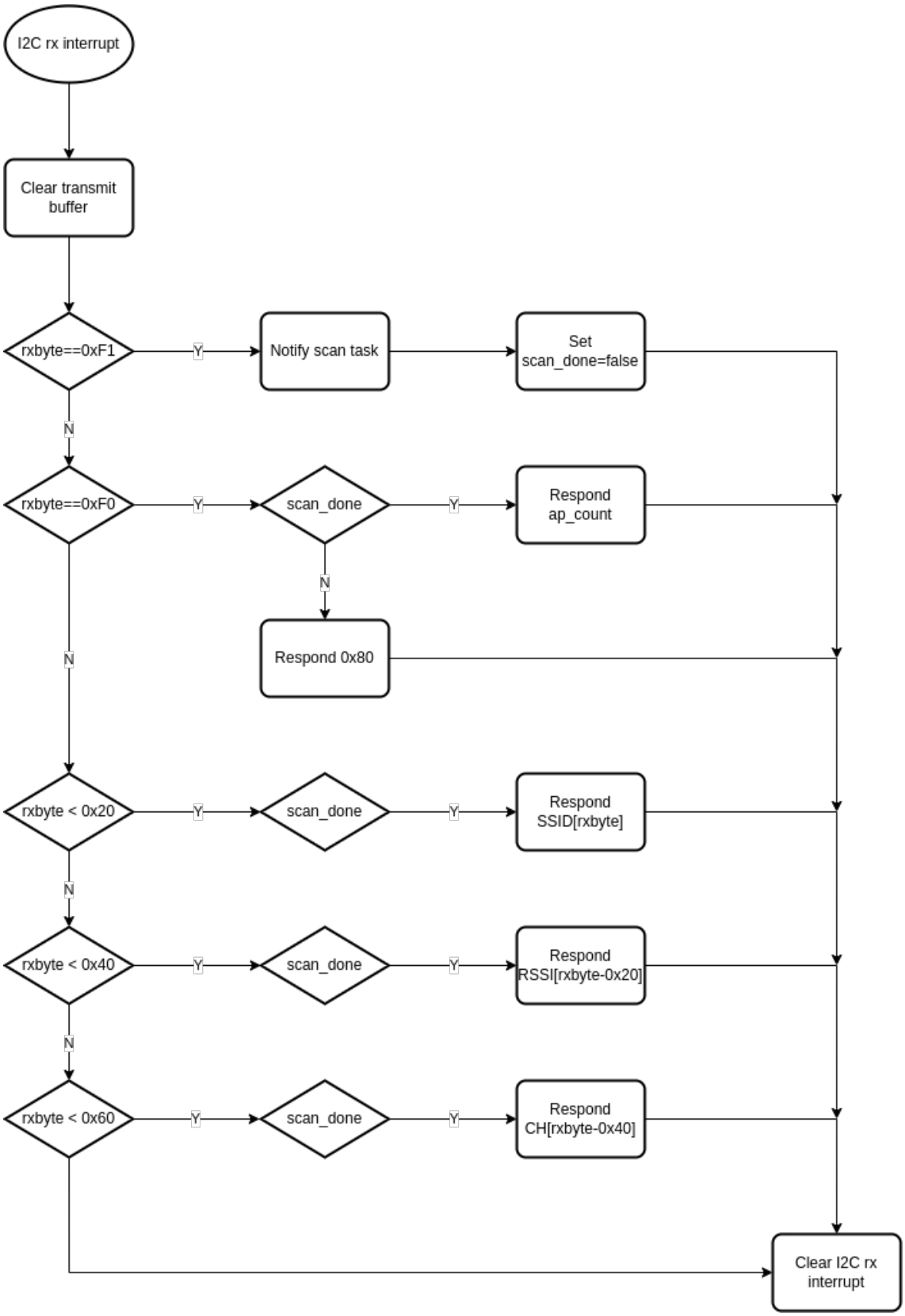

Figure 3.4: Flowchart illustrating the I^2C ISR

3.2.4.3 Communication with the ESP32 WiFi RSSI sensor

The WiFi RSSI sensor was created with a simple I^2C interface. When measuring RSSI, first the scan is started with the `0xF1` command, then the status can be polled by reading the `0xF0` register. This returns either `0x80` when the scanning is still in progress, or the number of found networks. This count can later be used to retrieve the SSID, RSSI, and operating channel of the networks.

3.2.5 Triggering

The measurement process is started by pressing the button on the tag. The button is connected to a digital input of the nRF52832 MCU and a falling edge interrupt is enabled for that input. This allows for an event-driven program structure and eliminating the need to periodically poll the state of the input, resulting in a more efficient solution.

When the button is pressed, first a WiFi scan is triggered on the ESP32 and while the scan is running, the other sensors are sampled, and then the WiFi records are read as well.

3.3 Communication

With the measurements taken, all results together with the position of the tag have to be transmitted to the gateway. As multiple measurements are transmitted at once, these values need to be arranged in a packet which provides context for interpreting the data.

3.3.1 Sensor data packet

The sensor data packet contains all measurements taken by the sensor:

- Temperature [°C],
- Relative humidity [%],
- Ambient light intensity [lux]
- Acceleration on each of the axes [g],
- Position [m],
- SSID of the detected networks in a string array,
- RSSI of the detected networks,
- Channel on which the detected networks operate.

For the data packet structure JSON was chosen, as it provides a self-describing data packet, which contains not only the measurement results, but also the meaning of every value, thereby making the system easily adoptable to different measurement goals. Although the use of the JSON format does introduce a considerable overhead in packet size, it also provides the most flexibility for the system. An example of the produced JSON packet is shown in listing 1.

```
{
    "temp": 25.8,
    "relh": 49.1,
    "light": 582.590,
    "acc": [0.024, 0.000, 0.972],
    "pos": [0.819, 6.423, 0.724],
    "ssid": ["SSID1", "Another_SSID", "Third SSID"],
    "rssi": [-22, -33, -71],
    "ch": [5, 11, 1]
}
```

Listing 1: Example JSON packet produced by the tag

The size of a data packet depends on the number of nearby WiFi access points, their SSIDs, RSSIs, and channels. The size of the packet with no WiFi access points is 121 bytes, and since all other data is transmitted with a fixed number of decimal places, this size is constant. In addition to this the details of at most 32 WiFi networks can be transmitted, each containing the maximum 32 byte long SSID, RSSI between -99 and 0 (e.g. maximum 3 characters), and channel number between 1 and 14 (maximum 2 characters). This allows us to calculate the maximum packet size:

$$N = 121 + 32 \cdot [(32 + 3) + (3 + 1) + (2 + 1)] = 1465 \text{ bytes}$$

where N is the number of bytes in the packet, including the quotation marks for the SSIDs and the commas separating the elements of the arrays. This result is however far from realistic, as WiFi SSIDs are usually considerably shorter than the 32 byte limit, and in most cases the number of nearby WiFi networks is also considerably less than 32. In order to realistically estimate the packet size the SSID database of WiGLE.net (mentioned in Section 1.4.1) was used. The database contains the 325400 most popular SSIDs recorded by WiGLE.net along with the number of distinct networks using each SSID, leading to a total amount of 281336706 WiFi networks. From this data set the average length of an SSID can be estimated with reasonable accuracy, leading to the average length of 10.84 characters. It can also be reasonably estimated, that at once only 15 nearby WiFi networks will be detected, resulting in an estimated packet size of

$$N = 121 + 15 \cdot [(10.84 + 3) + (3 + 1) + (2 + 1)] \approx 434 \text{ bytes}$$

Although this is a reasonable packet size, unfortunately the PANS stack only supports a payload of 34 bytes. In order to circumvent this problem, the data packets have to be transmitted in multiple parts.

3.3.2 Packet fragmentation

The objective of packet fragmentation is to divide a data packet into multiple segments, in order to increase the maximum packet size transferable over a communication medium. The segment can be individually sent to the destination, and then reassembled to obtain the original packet. Packet fragmentation is commonly used in Internet Protocol (IP) communication, as the maximum transmission unit (MTU) of IP networks is usually maximized in 1500 bytes (although jumbo frames allow up to 9000 byte MTU). In the case of this project, the limiting factor is the 34 byte MTU posed by the PANS stack, while packet size can reach 1465 bytes.

3.3.2.1 Segment header

In order to ensure the segments can be reliably reassembled on the receiver side, each segment must have a header carrying information about the segments place in the data packet. If the transfer medium is ideal, e.g. all segments arrive at the destination, and they are received in the same order as they were transmitted, than a single bit would be enough to signify either the first or the last segment of a packet. In real-world systems however segments may not arrive in the same order as they were transmitted in, or not arrive at all, making the one bit method inadequate. If we accept that for increased reliability the header size has to be increased, a simple, but effective system can be developed.

The packet fragmentation system developed for the PANS RTLS contains a 3 byte long header, which contains the payload length carried by the segment, the total number of segments in the packet, and the index of the segment, as shown in Table 3.4. The first byte contains the payload size on the 5 least significant bits, as the maximum possible payload is 31 bytes. The 3 remaining bits are used as a version identifier, which ensures that if in the future a different frame structure would be developed, it would not result in any errors in the current system, as in this implementation any message with a version other than 0 will be ignored. Future systems would be able to use the version number to correctly interpret the frame, supporting multiple versions. The second byte stores the index of the segment in the sequence, thereby allowing for the reconstruction of a packet from out-of-order segments. The third byte of the header stores the maximum segment index in the sequence, thus the receiver can determine if all segments of the packet had been received. As both the segment index and the maximum segment index are allocated 8 bits the maximum segment index is 255, allowing for packets consisting of maximum 256 segments. With each segment carrying up to 31 bytes of payload, this system is capable of transferring data packets of up up to 7936 bytes, providing more than enough capacity for the current data packets with room for expansion.

7	6	5	4	3	2	1	0
Version				Payload size			
Segment index							
Max segment index							

Table 3.4: Header structure of the packet fragmentation system

3.3.2.2 Data transfer rate

Unfortunately there is a new bottleneck exposed by the use of packet fragmentation. As described in Section 3.1.1, IoT data is transmitted as part of the TWR slot assigned to the tag, thus only one data segment can be transmitted by a tag during the time of a PANS superframe. This results in only 10 segments transmitted per seconds, limiting the available data transfer rate to 310 bytes/s with the use of the packet fragmentation system. This can result in the transfer of a packet taking up to 4.7 seconds, but as the objective of this sensor system is to take measurements in multiple places to map an area, the data transfer would likely complete while the tag is being moved between subsequent measurement locations. If our earlier estimate for the size of a more realistic packet is used, the data transfer would only take 1.4 seconds.

The data transfer rate of this system is unfortunately orders of magnitudes slower than the achievable 6.8 Mb/s suggested by the manufacturer. This is due to the PANS RTLS being primarily optimized for large positioning networks. For this reason, the majority of the available bandwidth is consumed by the positioning messages, and only a short time segment is available for data transfer, as explained in Section 3.1.1. Unfortunately in order to solve this issue, the RTLS would have to be fundamentally redesigned.

3.4 Program structure

The structure of the program is determined by the PANS stack, as it handles the startup process, and provides a real-time operating system (RTOS) for the application. For this purpose the eCos RTOS is used, and provides preemptive scheduling among tasks. The application is divided into three parts: the main thread handles initialization and later handles all UWB radio events; while the measurement thread carries out the measurements and the transmission of the results. The measurement suspends itself after every measurement-data transmission cycle; and the GPIO ISR handling the button input resumes the measurement thread when a new measurement has to be performed, as shown in Figure 3.5.

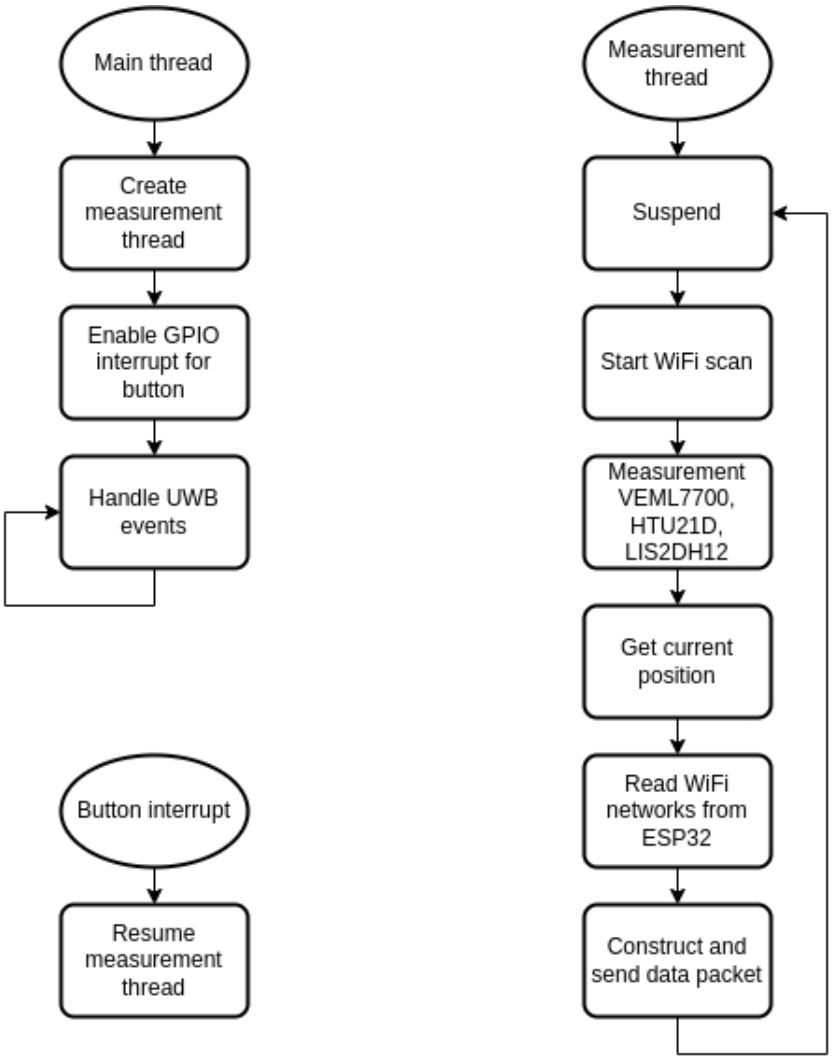

Figure 3.5: The three parts of the tag firmware: main thread, measurement thread and button ISR

Chapter 4

Tag hardware

With the firmware of the tag complete, it was time to design the hardware. This would house all components outlined in Section 2.3 in a more permanent form than the previously assembled prototype. The circuit was designed using the KiCad software. According to the concept, the circuit board would have to cfulfill the following requirements:

4.1 Requirements

- DWM1001C module with debug and download port
- ESP32 module with download port
- VEML7700 ambient light sensor
- HTU21D module
- Measurement trigger button
- Lithium battery with charging circuit
- 3.3 V power supply

4.2 DWM1001C module

The central part of the tag is the DWM1001C module, containing the DW1000 UWB transceiver, the nRF52832 MCU, and the LIS2DH12 accelerometer. For stable operation of the module, a pull-up resistor was connected to the inverted reset (RESETN) line, and a 10 µF as well as a 100 nF decoupling capacitor was placed near the power supply input line as seen in Figure 4.1. The module had to be connected to the I^2C bus, with pull-up resistors for both the SDA and the SCL lines as seen in Figure 4.2, which later proved to be unnecessary as the DWM1001C already includes the resistors required by the I^2C bus, as they are also needed for

the operation of the on-board accelerometer. The measurement trigger button was connected to a GPIO line of the nRF52832 MCU. Here care had to be taken, as the DW1000 transceiver has 4 GPIO lines which are also accessible on the module but have to be handled in a different way. Avoiding the DW1000 GPIOs, the button was connected to GPIO 12 of the nRF52832 MCU, as this line was also available on the DWM1001-DEV module and was used in the prototype, as seen in Figure 4.2. To stabilize the input signal from the button, a pull-up resistor and a capacitor to ground were connected, in effect forming an R-C low-pass filter attenuating the spurious signals created by the button. In order to provide debug access to the MCU, the ARM serial wire debug (SWD) signals are accessible through a 2x5 pin Cortex-M debug connector. As JTAG is not used, pins 6 and 8 are left unconnected, and because the firmware does not use the debugger detect function provided by the GNDDetect signal, pin 9 is also left unconnected, as seen in Figure 4.2.

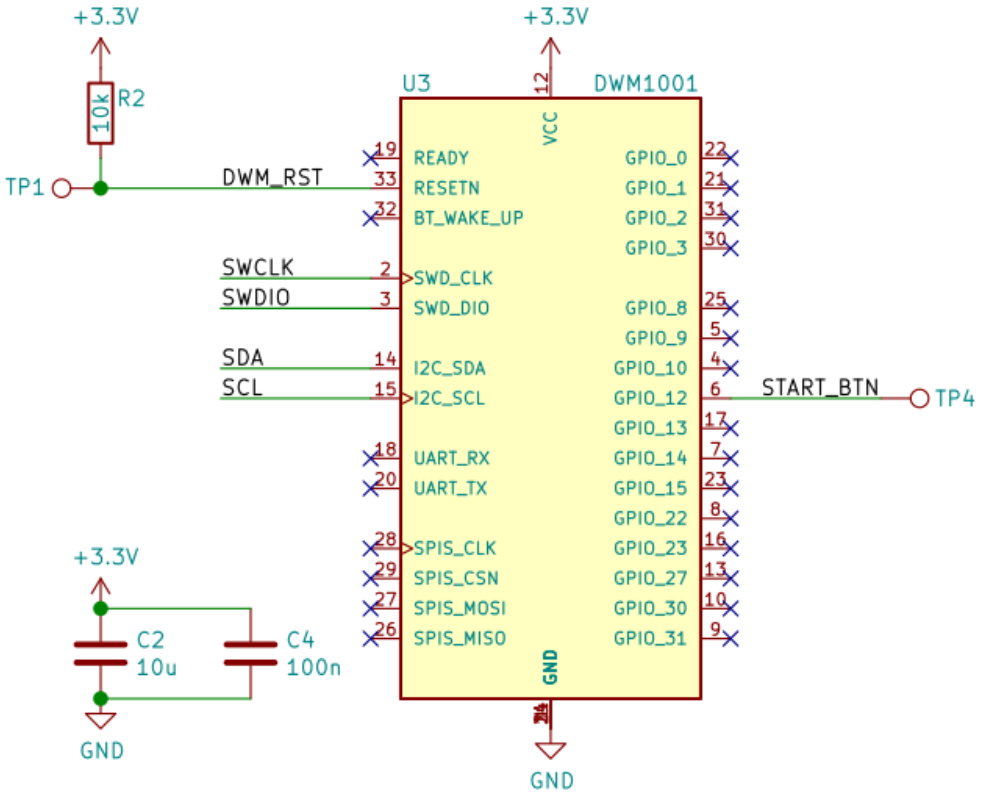

Figure 4.1: DWM1001C module connections and decoupling capacitors

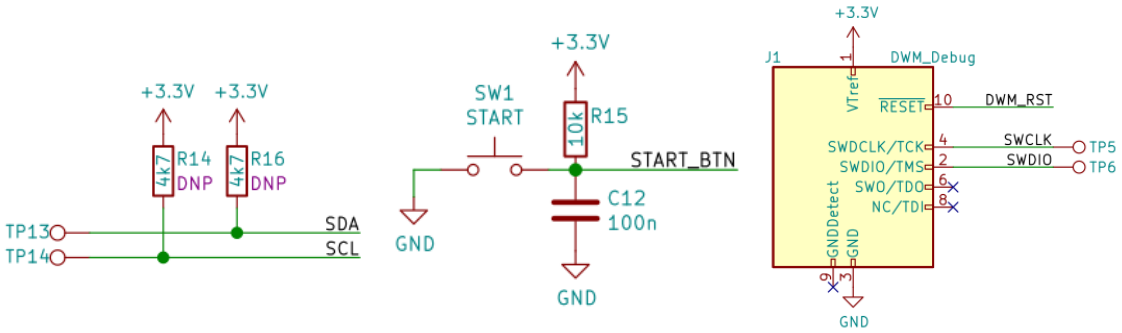

Figure 4.2: I²C pull-up resistors marked as do-not-populate (left), measurement trigger button with R-C filter (middle), and 2x5 pin Cortex-M debug connector (right).

4.3 ESP32 module

The ESP32-WROOM-32D module contains an ESP32 MCU with an on-board PCB trace antenna for the WiFi radio. Like with the DWM1001C, two decoupling capacitors were placed near the power input of the module, and a pull-up resistor was connected to the enable (EN) input to provide stability. The SDA and SCL signals of the I²C bus were also connected to two arbitrarily chosen GPIOs of the module, as the ESP32 is capable of internally connecting the I²C peripheral to any of its GPIOs. These connections are shown in Figure 4.3. In order to reprogram the ESP32 MCU, the UART boot mode has to be selected by pulling IO_0 low during startup. After this step, the MCU can be reprogrammed over UART, most commonly through a USB-UART converter like the FTDI FT232R. To accomplish the boot-mode selection, a circuit consisting of two transistors is commonly used, making it possible to automatically select the UART boot-mode using the data terminal ready (DTR) and the request to send (RTS) signals of the USB-UART converter. The receive (RX) and transmit (TX) signals between the programmer and the MCU are cross connected, and as an additional safety measure series resistors were connected to the data lines. The four necessary signals are accessible over a 2x3 pin connector, as seen in Figure 4.4.

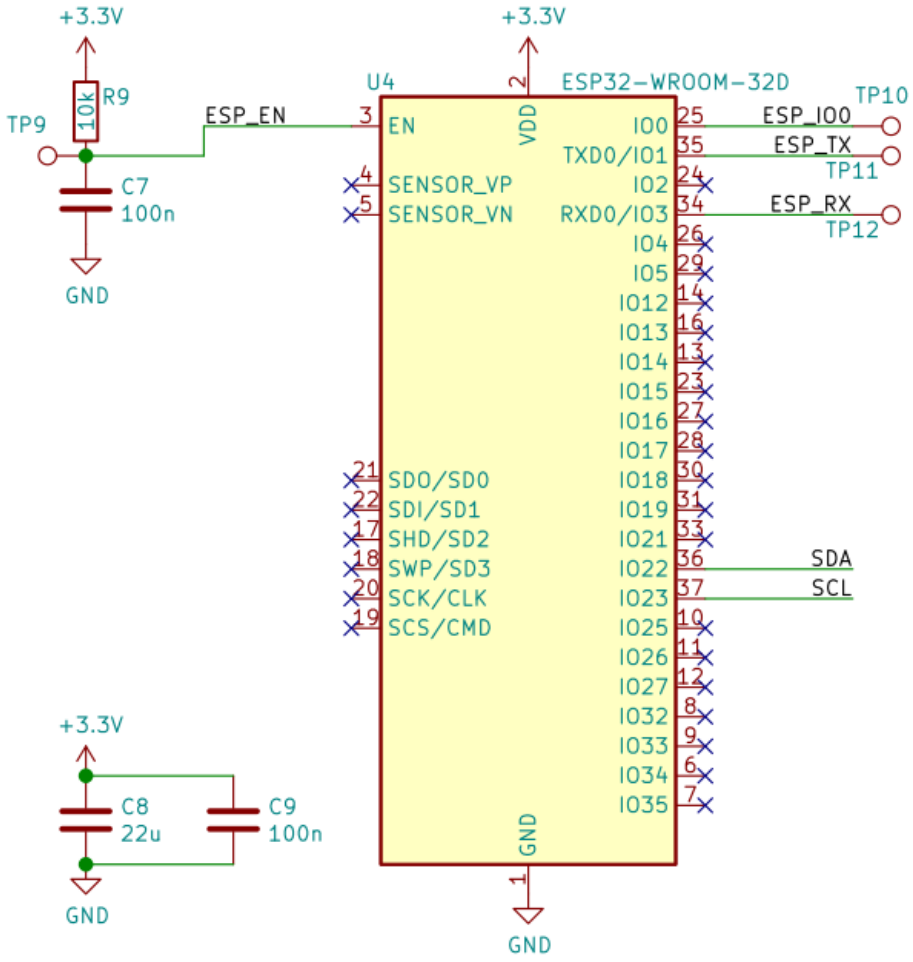

Figure 4.3: ESP32-WROOM-32D module connections

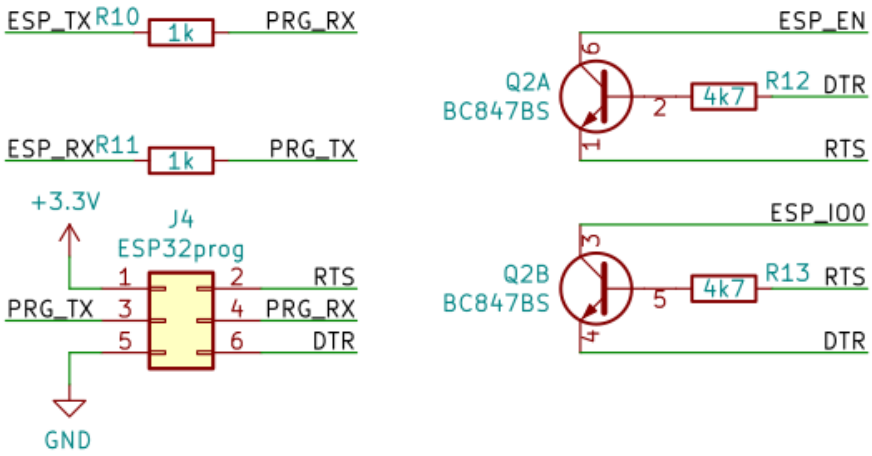

Figure 4.4: ESP32 boot-mode selection circuit and programming connector

4.4 VEML7700

As the VEML7700 ambient light sensor has significantly lower peak current consumption than the ESP32 and the DWM1001C, a single 100 nF decoupling capacitor is sufficient for stability. For communication, the VEML7700 was also connected to the I^2C bus, as seen in Figure 4.5. In order to provide design flexibility, the VEML7700 sensor can be installed in two positions, facing up and facing sideways. To accommodate this feature, I have created a double footprint for the sensor, which allows soldering it in both orientations as seen in Figure, facing the side of the board when installed sideways, as seen in Figure 4.5.

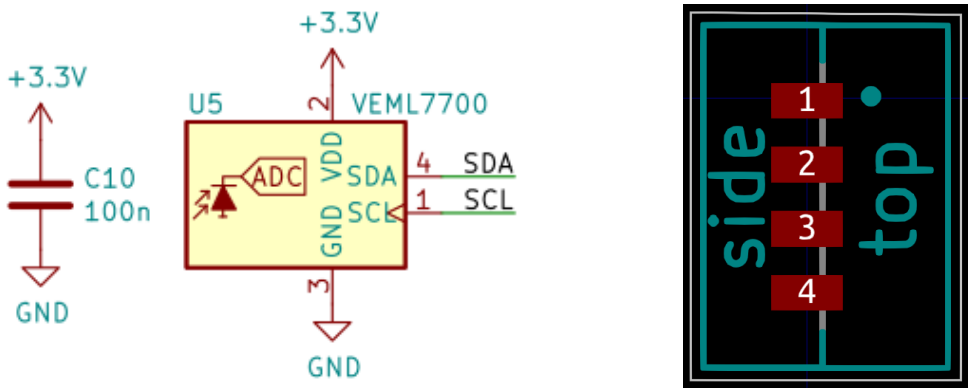

Figure 4.5: VEML7700 connections (left) and universal PCB footprint (right)

4.5 HTU21 module

Since most relative humidity and temperature sensors are either unsuitable for hand-soldering or are currently unavailable for purchase, an HTU21D-breakout module was used for this purpose. As this module already includes the necessary decoupling capacitors near the sensor IC, only the I^2C bus had to be connected to the module besides the power. Having the temperature sensor on a separate module comes with the benefit of better thermal isolation from the main circuit board, reducing measurement offset.

4.6 Battery

To enable the use of lithium-polymer batteries, the circuit has on-board battery clips supporting the 14500 battery size. Besides the battery clips, I have designed a 2 mm to 2.54 mm universal header for non-cylindrical batteries, to support a wider variety of connectors. The circuit is intended to be used with batteries with built-in protection circuits, but for additional safety a self-resetting fuse was installed, as well as a P-MOSFET for reverse-polarity protection, as seen in Figure 4.6.

As lithium batteries have special charging requirements, the MCP73821-2 lithium battery charger IC was used to enable charging from a USB port. Two 4.7 µF decoupling capacitors were installed to the input and the output of the IC, as recommended by the data sheet. The maximum charge current can be set with a resistor:

$$I_{\text{charge}} = \frac{1000V}{R_{\text{prog}}}$$

where I_{charge} is the maximum charge current in Amperes, and R_{prog} is the resistance of the programming resistor in Ohms. In this circuit a 400 mA charge current limit was set using a 2.5 kΩ resistor, thus the circuit can safely charge lithium batteries with 600 mAh or bigger capacity. To provide indication of the charge state, two LEDs were connected to the status output with current limiting resistors, providing charge, charge complete, and no battery indication, made possible by the tri-state status output of the charger IC. The complete charging circuit can be seen in Figure 4.6.

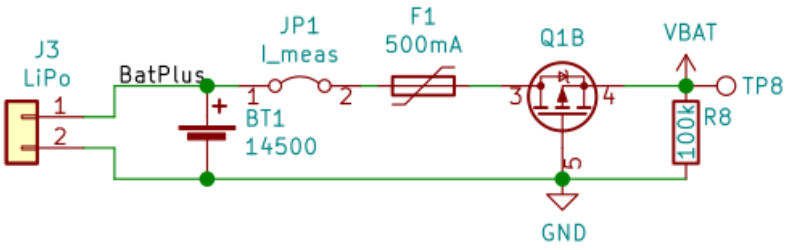

Figure 4.6: Battery connection and protection circuit

Besides the battery charger IC, a connector is also needed for charging over USB. For this a USB-C receptacle was chosen, to keep up with the evolution of USB and avoid the use of the obsolete micro-USB connector. The selected GCT USB4105 connector is a USB 2.0 compatible USB-C receptacle, e.g. only the signals needed for USB 2.0 operation are accessible. This reduces the pin count of the connector, making it easier to handle. As this connector is only used for power transfer, the communication signals are left unconnected, but in order to comply with the USB-C specification, 5.1 kΩ pull-down resistors were connected to the CC1 and CC2 lines, as seen in Figure 4.7. This declares the port upwards-facing (UFP), and allowing maximum 500 mA current to be consumed over USB.

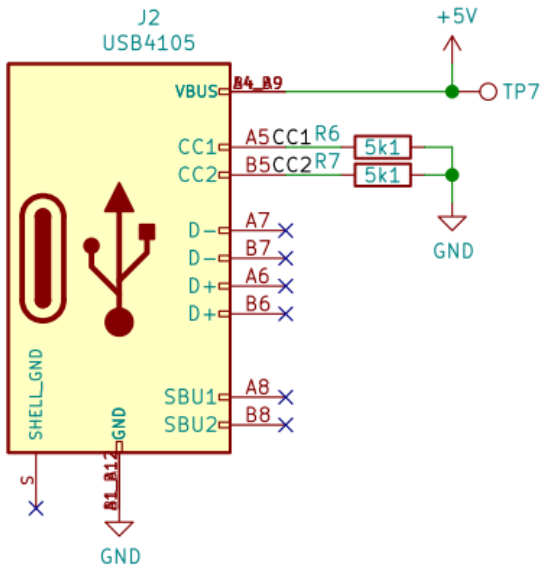

Figure 4.7: USB-C receptacle with pull-down resistors

4.7 3.3V power supply

As all used components are designed to operate from a 3.3 V power source, a regulator is needed to reduce the battery voltage to the allowed range. In order to make the circuit more efficient, a switch-mode power supply was chosen with buck-topology. The LM3671 is a compact buck-converter IC, requiring only one inductor and two capacitors to operate. The value of these components is provided in the data sheet for common output voltage options, and it is also recommended that X5R or X7R type capacitors are used to improve stability. Besides the required passive components an additional circuit was connected to provide automatic switchover when the USB power supply is connected. This is accomplished by another P-MOSFET and a diode, as seen in Figure 4.8.

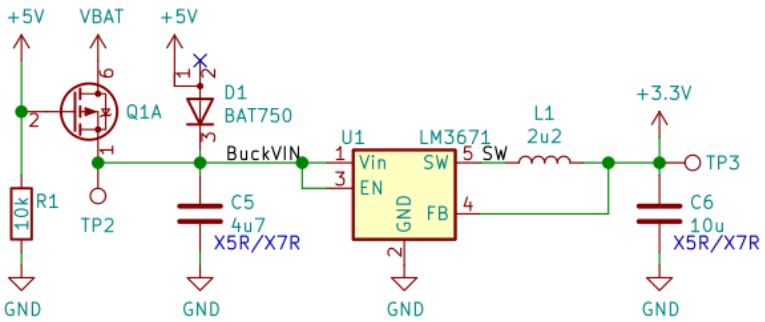

Figure 4.8: LM3671 buck-converter

4.8 PCB Layout

With the schematic complete (see Appendix A.2), the components had to be placed on a PCB, and the signals had to be routed between them. For most components the data sheet provided useful recommendations for the layout. For the buck-converter I aimed to place the inductor as close to the SW output of the LM3671 IC and connected it with a wide trace, so as to minimize the switching noise. In another attempt to reduce noise, and provide all components with a low-impedance connection to the ground of the power supply, the bottom copper layer of the PCB was kept as empty as possible, so that a continuous copper pour could be placed connected to ground. The empty areas on the top copper layer also have a ground pour, and the two layers are connected with multiple vias. Unfortunately these ground pours would greatly reduce the efficiency of the antennae on the DWM1001C and the ESP32 modules. In order to reduce this effect these modules have been placed such that the area with the antennae overhang the PCB.

The biggest component on the PCB is the 14500 battery holder. The holder consists of two identical parts for each end of the battery, which have to be spaced correctly to ensure a good connection. The size of the battery holder intuitively set both the size, and layout of the PCB. The power supply and battery charger were placed on one side of the battery, while the other side contained the two radio modules, the sensors, and the trigger button. In the corners of the PCB four holes were added for M3 screws, to enable mounting the PCB in a case. In order to make the testing of the PCB easier, test points were also added to every signal so that a logic analyzer or an oscilloscope can be connected if the circuit is behaving unexpectedly. With these additions, the PCB is finished, as seen in Figure 4.9.

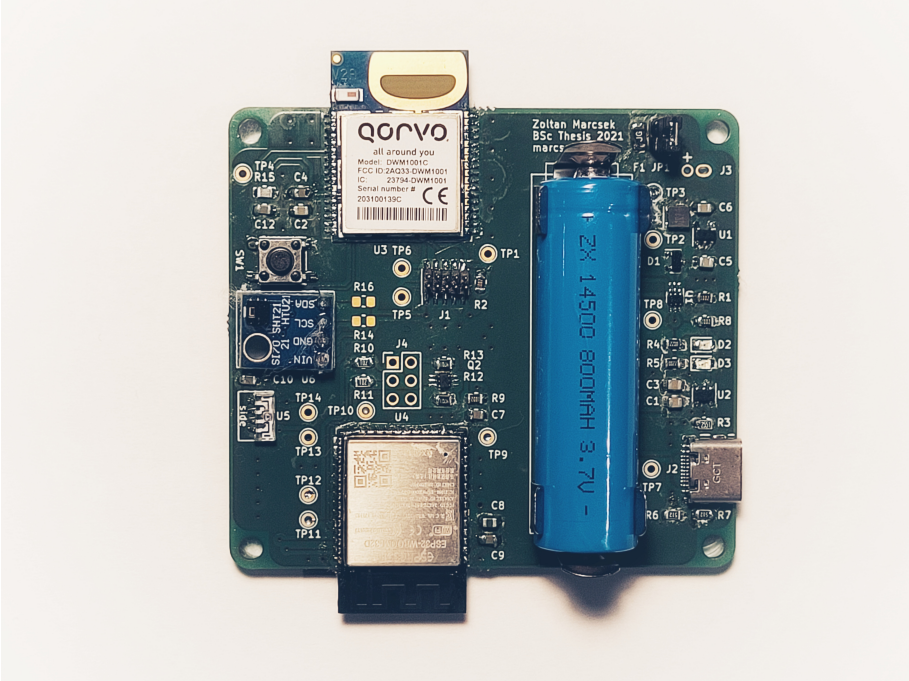

Figure 4.9: The finished, assembled tag with a 14500 lithium battery

Chapter 5

Gateway software

With the measurement tag completed, the gateway software had to be designed to handle the measurement results. The gateway software can be divided into three parts: the frontend, the backend, and the UWB bridge interpreting between the UWB network and the backend. As the gateway runs on a Raspberry Pi 3 single-board computer (SBC), the backend and the bridge were implemented in Python, to simplify the development. Unfortunately the Linux distribution supplied with PANS is too outdated for recent versions of the Python interpreter, and updating is strictly not recommended by the developers, as it would probably result in a malfunction of the PANS system. To circumvent this problem, I have used Docker for the development.

5.1 Features

The main feature of the gateway is to display measurement results on a heatmap. In order to make measurements more structured, the gateway software gathers measurement results in measurement sequences. Each measurement sequence can have a map associated to it, which is displayed under the heatmap. The sequences and maps can all be edited from the web-frontend.

5.2 Backend

The main component of the gateway is the backend. This part handles the storage and access of measurement results, measurement sequences, and maps. In order to store all data well structured, the MariaDB relational database was used. The database contains three tables for the measurement results (data points), the measurement sequences (measurements), and for the maps. For each data point the measurement results and the position are stored in JSON format for increased flexibility, and the time the data point was received is also stored as a UNIX timestamp. The measurement sequence of the data point is referenced with a foreign key. In the measurements table, for each measurement sequence a name

and an optional description is stored, as well as its state (active/inactive). The corresponding map is referenced in an optional foreign key, e.g. measurement sequences can exist without an associated map, if a map is not available for the area. A new map can be added later when it becomes available. Similar to measurement sequences, a name and an optional description is also stored for each map. Besides that the actual map is stored a base64-encoded image, with scaling and offset parameters to correct for any discrepancies between the map and the origin of the RTLS. The structure of the database is shown in Figure 5.1

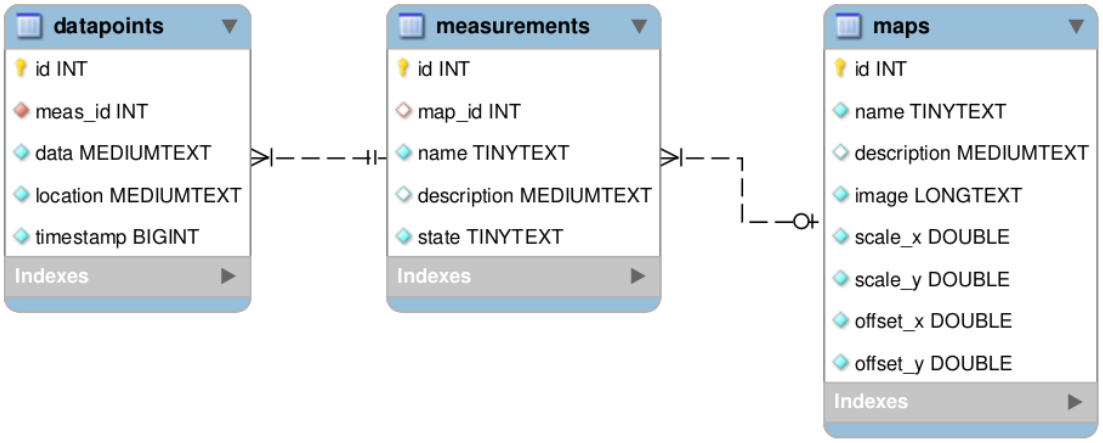

Figure 5.1: Entity-relation diagram of the database

To handle data access to the database, the backend component provides a REST API using the FastAPI python library. The available endpoints are described in Table 5.1. Data transfer takes places using the JSON data format in both directions.

Method	Path	Function
GET	/api/measurements	List measurement sequences
GET	/api/measurements/{id}	Get details of the measurement sequence with the specified id
POST	/api/measurements/{id}	Edit the measurement sequence with the specified id
POST	/api/measurements/new	Create new measurement sequence
GET	/api/maps	List maps
GET	/api/maps/{id}	Get details of the map with the specified id
POST	/api/maps/{id}	Edit the map with the specified id
POST	/api/maps/new	Create new map
GET	/api/datapoints?meas_id={id}	Get data points associated with the specified measurement sequence
POST	/api/datapoints	Create new data point associated to the currently active measurement sequence

Table 5.1: Description of the available API endpoints

5.3 UWB bridge

When viewed from the perspective of incoming data, the first component of the gateway is the UWB bridge. The PANS stack makes all data received from tags accessible through an MQTT broker, but as discussed in Section 3.3.2, the data packets are transmitted in multiple segments. The task of the bridge component is to collect all data frames received from the measurement tag, reassemble the received frames into complete data packets and send the complete frames to the backend.

5.3.1 MQTT

MQTT is a publish-subscribe protocol, where data producing nodes can publish data to specified URLs, and data consuming devices can subscribe to URLs in order to receive any data published to that endpoint. To connect the data producer and data consumer devices a broker is used. The task of the MQTT broker is to

receive all published messages, and transmit them to the data consumers according to their subscriptions. In an MQTT broker usually the last available message is stored for each endpoint, so that new subscribers can always receive the latest available information.

5.3.2 Data conversion

As data packets are available over MQTT from the closed-source PANS stack, the first step of the bridge module is to subscribe to the data endpoint of the measurement tag. Whenever a new frame is received, the interpreter either adds it to the current packet if it is not a start packet; or starts a new packet if it is a start packet. If a start packet is received while the previous packet is incomplete, then the incomplete packet is dropped. If the expected number of frames arrive for a packet, then the packet is deemed complete, and is sent to the backend in a `POST` request to the `/api/datapoints` endpoint. The operation of the packet assembler is illustrated in Figure 5.2.

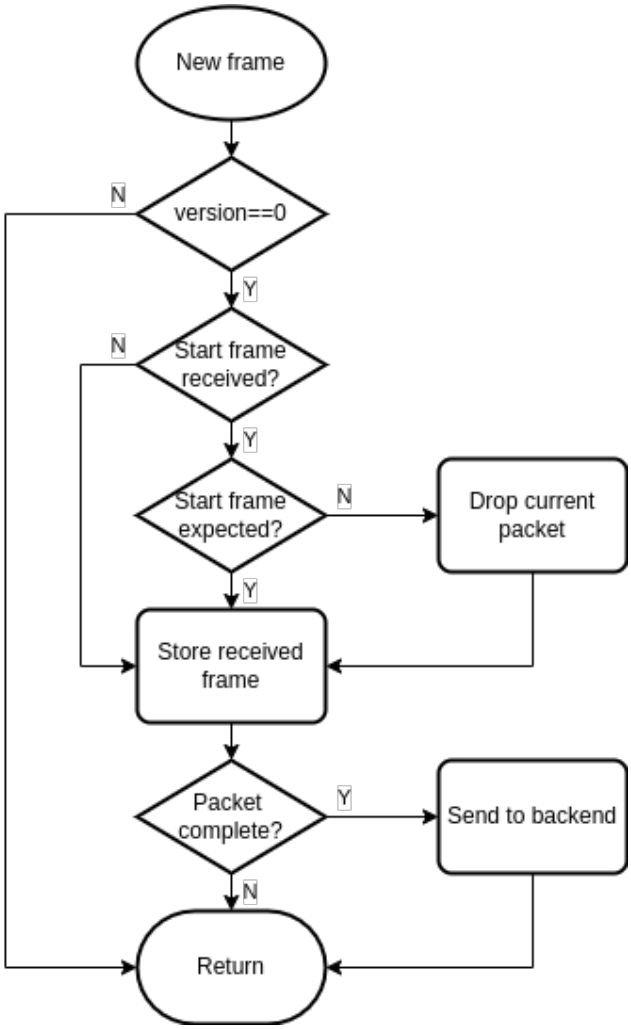

Figure 5.2: Flowchart describing the packet reassembly process

5.4 Frontend application

The frontend is the interface intended for human interaction, it provides convenient measurement sequence handling, map uploads, and visualization of measurements. It is accessible from a web browser, and in an effort to make the interface look nice, I have used the Bootstrap framework. To visualize the data points the heatmap.js library was used. Further data processing is enabled by data download, which is also available in the JSON format.

On the main page of the application either the list of measurement sequences, or the list of maps is visible (see Figure 5.3). From this page a new measurement sequence or a new map may be added, or an existing can be viewed. In the measurement sequence view, the name and description of the measurement sequence are shown and can be modified, as well as a heatmap is visible of the selected data type (e.g. temperature), as seen in Figure 5.4. In the map view the name, description, and image of the selected map are shown (see Figure 5.5), and can be modified together with the scaling and offset parameters. For the creation of the maps I have used Inkscape, which provides an easy way to create accurate graphics including wall thickness and the placement of doors and windows.

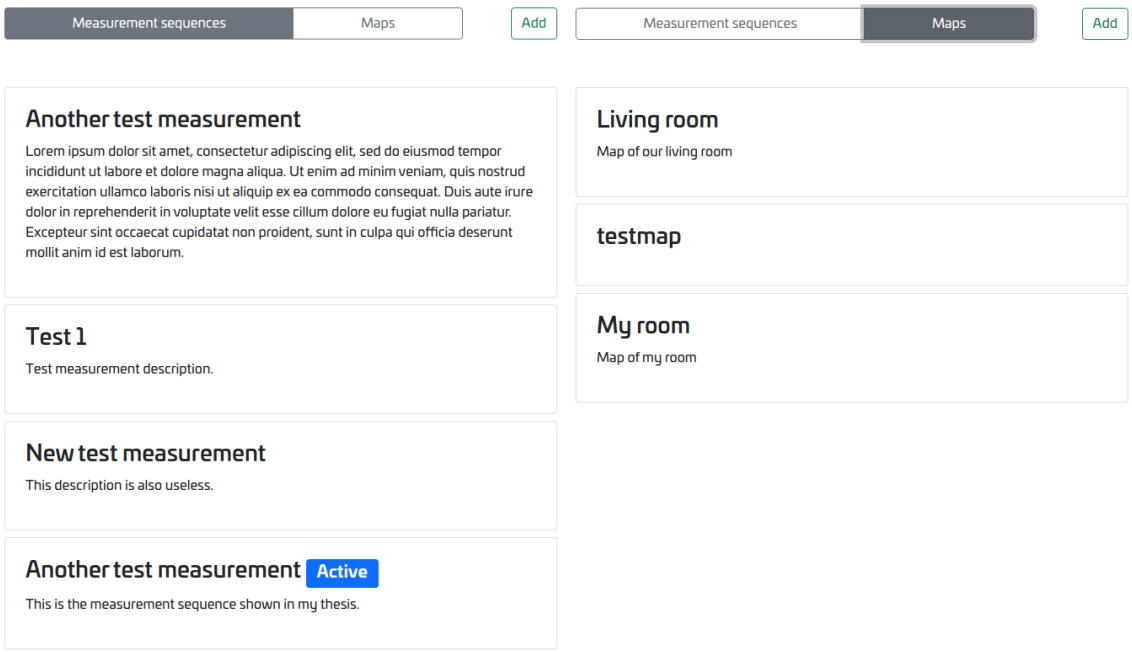

Figure 5.3: Measurement sequence list (left) and map list(right)

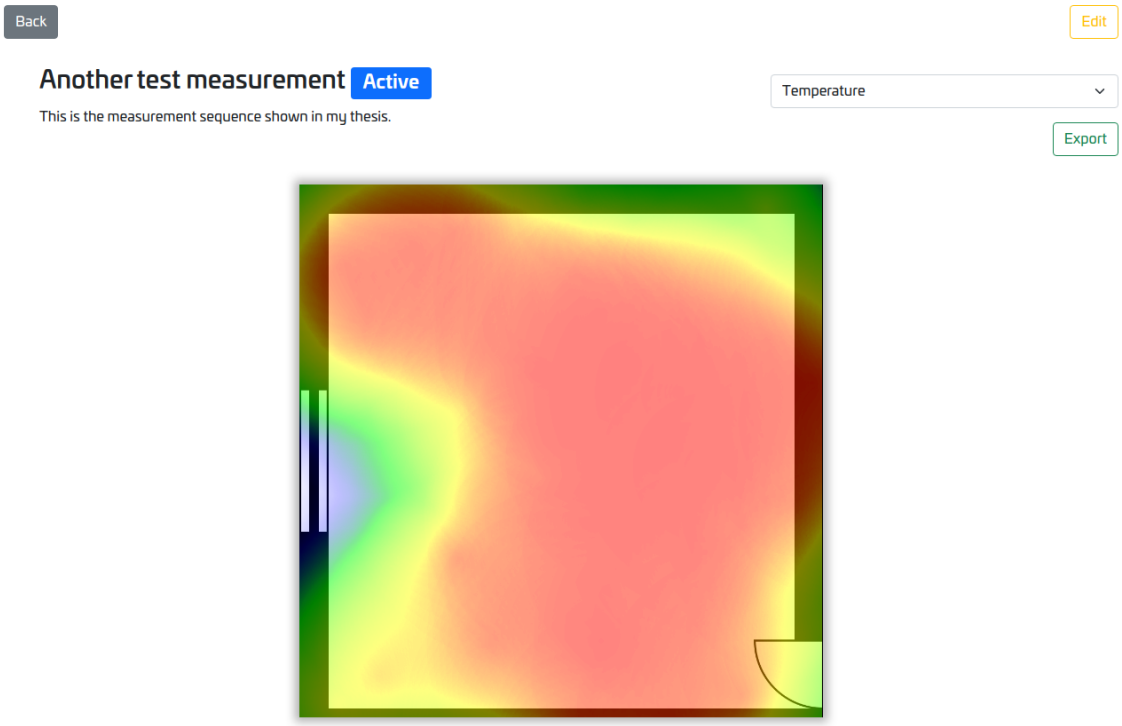

Figure 5.4: Measurement sequence view with heatmap

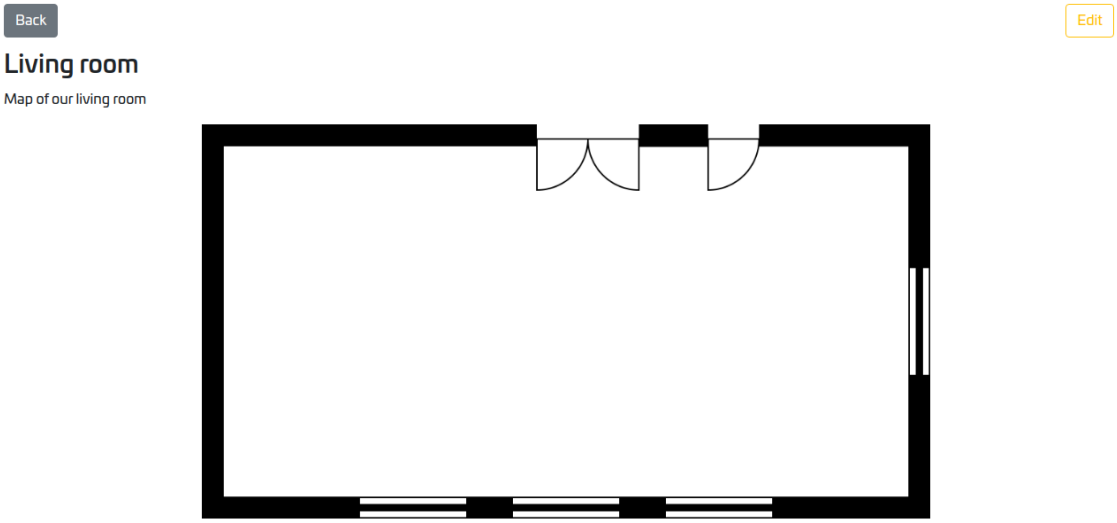

Figure 5.5: Map view

5.5 3D printed case

In order to protect the gateway and make it more presentable, I have designed and 3D printed a case for it. The design was inspired by the Mac mini, and was made in FreeCAD using the 3D model of the components in the gateway. The design process and the finished gateway are shown in Figure 5.6.

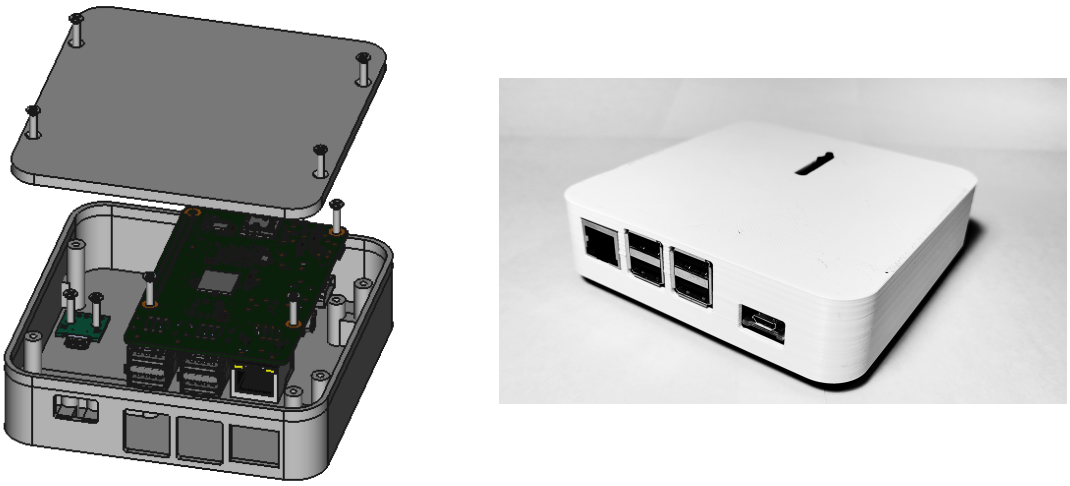

Figure 5.6: Gateway case during design (left) and completed with the gateway installed(right)

Chapter 6

Conclusion

In this thesis project, I have developed a UWB-radio based measurement system. It is capable of determining the location of a measurement device (tag) using UWB ranging and simultaneously transmitting measurement results over the same radio link. For this system, I have designed the hardware and firmware of the tag, as well as the software of the gateway. The tag is equipped with sensors for ambient light intensity, temperature, relative humidity, and WiFi RSSI and can be powered using a lithium-polymer battery. UWB connectivity is provided by the DW1000 transceiver from Decawave with the PANS RTLS stack, integrated on the DWM1001C module. The same module was used for the positioning anchors and the gateway, with the latter employing a Raspberry Pi 3 single-board computer as well. The gateway software can store all data received from the tag in a database, and make it accessible through an HTTP REST API. For human interaction a web frontend is also available, where measurement results can be visualized on a heatmap.

After the system has been completed, I have added the map of my room through the web interface, and created a new measurement sequence. Then I took measurements in multiple points of my room with the tag. The temperature heatmap (see Figure 6.1 left) is particularly interesting, as it clearly shows the cold air near the window at the left side of the room. Unfortunately the legend did not fit on the screenshot as it is placed in the bottom right corner of the screen. The blue color near the window represents 21.5 °C, and the red occupying most of the room means 23.4 °C. Another interesting measurement is the light intensity distribution throughout my room (see Figure 6.1 right), clearly showing the location of the lamp.

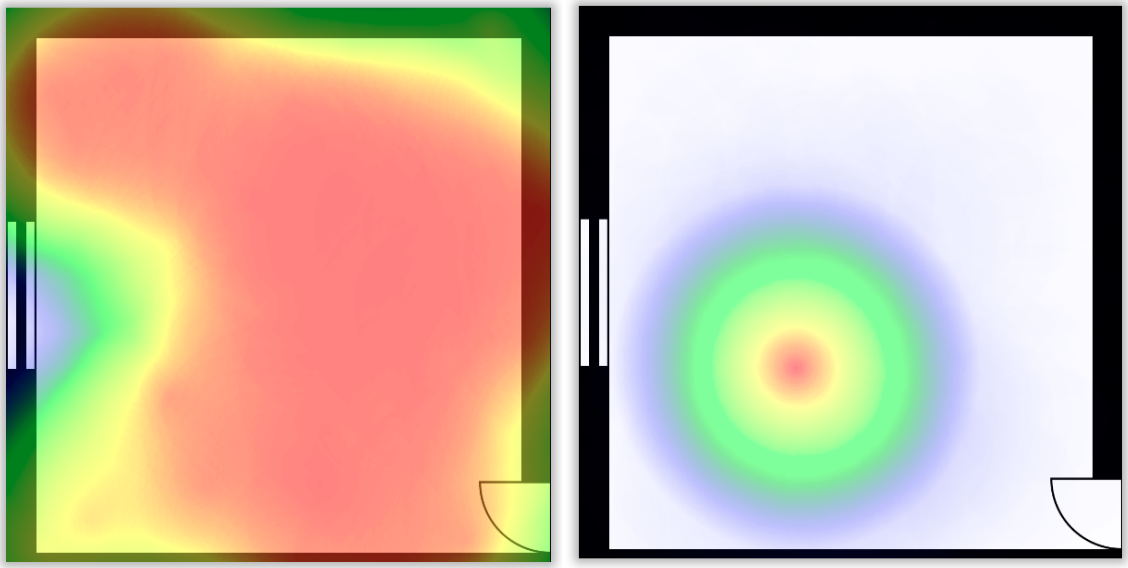

Figure 6.1: Temperature heatmap showing the cold air near the window (left) and light heatmap showing the light in my room (right)

6.1 Possible future work

The main flaw of the system is the slow data transfer rate, which is unfortunately a limitation posed by the PANS stack. A different system may allow for higher transfer speeds, in exchange for reduced expandability. Using a different communication system could also enable the use of the mesh communication capabilities of IEEE 802.15.4, thus the system would be expandable without the addition of proxy-gateways.

The tag hardware also has room for improvement. During the test measurements I realized that an LED indicating the state of the measurement would make the process more convenient. To expand the capabilities of the tag a digital microphone could be added, making it possible to measure noise level on worksites, or the acoustical characteristics of concert halls.

All in all the created system is suitable for carrying out various measurements, and is capable of creating data sets to analyze environmental conditions. A truncated data set produced by the export function is available in Appendix A.1.

Acknowledgements

I would like to thank my supervisor Dr. Balázs Matolcsy for all the provided support, especially during the writing process. I would also like to thank him for providing the development boards for the UWB modules, and for his quick assistance with parts purchases. In addition I would also like to thank my fellow student Márk Mihalik, for letting me use his own Raspberry Pi 3 for this project. Finally I want to thank my father for looking through my thesis and providing grammatical suggestions.

Bibliography

[1] Wongeun Choi, Yoon-Seop Chang, Yeonuk Jung, and Junkeun Song. Low-power lora signal-based outdoor positioning using fingerprint algorithm. *ISPRS International Journal of Geo-Information*, 7(11), 2018. ISSN 2220-9964. DOI: 10.3390/ijgi7110440. URL https://www.mdpi.com/2220-9964/7/11/440.

[2] TE Connectivity. *HTU21D data sheet*, 2017.

[3] IEEE 802.11 Working Group. Ieee standard for information technology–telecommunications and information exchange between systems - local and metropolitan area networks–specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pages 1–4379, 2021. DOI: 10.1109/IEEESTD.2021.9363693.

[4] Myungin Ji, Jooyoung Kim, Juil Jeon, and Youngsu Cho. Analysis of positioning accuracy corresponding to the number of ble beacons in indoor positioning system. In *2015 17th International Conference on Advanced Communication Technology (ICACT)*, pages 92–95, 2015. DOI: 10.1109/ICACT.2015.7224764.

[5] Rakhesh Singh Kshetrimayum. An introduction to uwb communication systems. *IEEE Potentials*, 28(2):9–13, 2009. DOI: 10.1109/MPOT.2009.931847.

[6] Xingqin Lin, Johan Bergman, Fredrik Gunnarsson, Olof Liberg, Sara Modarres Razavi, Hazhir Shokri Razaghi, Henrik Rydn, and Yutao Sui. Positioning for the internet of things: A 3gpp perspective. *IEEE Communications Magazine*, 55(12):179–185, 2017. DOI: 10.1109/MCOM.2017.1700269.

[7] Decawave Ltd. *DW1000 user manual*, 2017. URL https://www.decawave.com/dw1000/usermanual/.

[8] Decawave Ltd. *DWM1001C data sheet*, 2017. URL https://www.decawave.com/dwm1001/datasheet/.

[9] Decawave Ltd. *DWM1001 system overview and performance*, 2018. URL https://www.decawave.com/dwm1001/systemoverview/.

[10] NXP. *I2C bus specification and user manual rev.7*, 2021. URL https://www.nxp.com/docs/en/user-guide/UM10204.pdf.

[11] Nico Podevijn, David Plets, Jens Trogh, Luc Martens, Pieter Suanet, Kim Hendrikse, and Wout Joseph. TDoA-based outdoor positioning with tracking algorithm in a public LoRa network. *Wireless Communications and Mobile Computing*, 2018:1–9, May 2018. DOI: 10.1155/2018/1864209. URL https://doi.org/10.1155/2018/1864209.

[12] Seyed Reza Saghravani, Sa'ari bin Mustapha, Seyed Fazlolah Saghravani, Shaharin bin Ibrahim, and Mohd. Kamil bin Yusoff. Performance of real-time kinematic global positioning system and automatic level surveying for height determination - a comparison. In *2009 International Conference on Signal Acquisition and Processing*, pages 108–111, 2009. DOI: 10.1109/ICSAP.2009.37.

[13] Bluetooth SIG. *Bluetooth Core Specification v5.3*, 2021. URL https://www.bluetooth.com/specifications/specs/core-specification/.

[14] International Telecommunications Union. Recommendation sm.1755: Characteristics of ultra-wideband technology. https://www.itu.int/rec/R-REC-SM.1755, 2006.

[15] Vishay. *Designing the VEML7700 into an application*, 2019. URL https://www.vishay.com/docs/84323/designingveml7700.pdf.

[16] Vishay. *VEML7700 data sheet*, 2021. URL https://www.vishay.com/docs/84286/veml7700.pdf.

[17] Chouchang Yang and Huai-rong Shao. Wifi-based indoor positioning. *IEEE Communications Magazine*, 53(3):150–157, 2015. DOI: 10.1109/MCOM.2015.7060497.

[18] Yilin Zhao. Standardization of mobile phone positioning for 3g systems. *IEEE Communications Magazine*, 40(7):108–116, 2002. DOI: 10.1109/MCOM.2002.1018015.

Appendix

A.1 Exported data set


```
[
    {
        "id": 1,
        "data": {
            "temp": 25.8,
            "relh": 83.8,
            "light": 28.532,
            "acc": [
                -16.456,
                0.072,
                0.996
            ],
            "ssid": [
                "SSID-1",
                "SSID-2",
                "SSID-3",
                "SSID-4",
                "SSID-5",
                "SSID-6",
                "SSID-7",
                "SSID-8"
            ],
            "rssi": [
                -32,
                -38,
                -74,
                -86,
                -88,
                -91,
                -91,
                -93
            ],
            "ch": [
                5,
                5,
```

```
                11,
                11,
                11,
                1,
                11,
                6
            ]
        },
        "pos": [
            1.378,
            1.089,
            1.985
        ]
    },
    {
        "id": 2,
        "data": {
            "temp": 25.2,
            "relh": 42.3,
            "light": 87.837,
            "acc": [
                -16.432,
                0.012,
                0.972
            ],
            "ssid": [
                "SSID-2",
                "SSID-1",
                "SSID-3",
                "SSID-6",
                "SSID-7",
                "SSID-4",
                "SSID-5",
                "SSID-8",
                "SSID-9",
                "SSID-10"
            ],
            "rssi": [
                -40,
                -42,
                -68,
                -81,
                -82,
                -85,
                -86,
                -90,
                -92,
```

```
                -92
            ],
            "ch": [
                5,
                5,
                11,
                1,
                11,
                11,
                11,
                6,
                1,
                1
            ]
        },
        "pos": [
            2.051,
            0.544,
            0.045
        ]
    },
    {"Truncated"}
]
```

A.2 Schematic