



**University of
Zurich^{UZH}**

MTT: My Thesis Title

*Name Lastname
Zurich, Switzerland
Student ID: TBD*

Supervisor: Name Lastname, Name Lastname, Prof. Dr. Burkhard Stiller

Date of Submission: Month DD, YYYY

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland



Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmuehlestrasse 14, CH-8050 Zurich, Switzerland
URL: <http://www.csg.uzh.ch/>

Declaration of Independence

I hereby declare that I have composed this work independently and without the use of any aids other than those declared (including generative AI such as ChatGPT). I am aware that I take full responsibility for the scientific character of the submitted text myself, even if AI aids were used and declared (after written confirmation by the supervising professor). All passages taken verbatim or in sense from published or unpublished writings are identified as such. The work has not yet been submitted in the same or similar form or in excerpts as part of another examination.

Zürich,

Signature of student

Abstract

Acknowledgments

Contents

Declaration of Independence	i
Abstract	iii
Acknowledgments	v
1 Introduction	1
1.1 Motivation	1
1.2 Thesis Goals	1
1.3 Methodology	2
1.4 Thesis Outline	3
2 Fundamentals	5
2.1 Background	5
2.1.1 Ultra Wideband	5
2.1.2 UWB MAC	7
2.1.3 UWB PHY	8
2.1.4 Two-way ranging	12
2.1.5 Wireless Sensor Networks (WSN)	13
2.1.6 k-connected graphs	15
2.2 Related Work	15
2.2.1 Artwork Tracking	15
2.2.2 Sensor Networks	16
2.2.3 Wireless ranging	16

3 Design	19
3.1 Hardware	19
3.1.1 Microcontroller	19
3.1.2 UWB shield	20
3.1.3 Humidity and temperature sensor	20
3.1.4 Accelerometer and Gyroscope	20
3.1.5 Tag technical plan	21
3.2 Architecture	22
3.2.1 Responsibilities	22
3.2.2 Dataflow	25
3.3 Network	29
4 Implementation	31
4.1 Tag	31
4.1.1 Temperature and humidity module	31
4.1.2 Gyroscope	32
4.1.3 Network	34
4.1.4 Two-way ranging	35
4.1.5 BLE	37
4.1.6 Job Handler	38
4.1.7 Combining modules	40
4.2 App	41
5 Evaluation	49
5.1 Experiment 1: Static	49
5.1.1 Results	50
5.1.2 Conclusion	54
5.2 Experiment 2: Temperature	57
5.2.1 Results	59

CONTENTS	ix
5.2.2 Conclusion	61
5.3 Experiment 3: Gyroscope	62
5.3.1 Reults orientational read	62
5.3.2 Results angular velocity read	64
5.3.3 Colclusion	66
5.4 Experiment 4: Distance	68
5.4.1 Results	68
5.4.2 Conclusion	70
5.5 Experiment 5: Real World experiment	70
5.5.1 Results	71
6 Final Considerations	73
6.1 Summary	73
6.2 Conclusions	73
6.3 Future Work	73
Abbreviations	77
List of Figures	77
List of Tables	80
List of Listings	81
A Contents of the Repository	85

Chapter 1

Introduction

Research questions:

- What are the advantages and disadvantages to using UWB over BLE for fast network communication.
- Can I build a system of sensors that detects disturbances in transportation and alerts the user.
- What is the most adequate network topology to be used in such a system.

1.1 Motivation

Certify is an international cooperative project between twelve partners situated in Switzerland and the EU. One of those partners is the university of zurich. Its focus is on the development of Internet of Things (IoT) systems for security, monitoring and detection. Next to certifications and the development of frameworks, Certify also consirnes itself with the integration of IoT devices.

One of multiple currently running pilots of Certify is the "Tracking and monitoring of artworks". The goal of this pilot is to enable the constant tracking and monitoring of artworks by attaching a device to it. This device allows for unique identification, by using cryptographic methods. It is also intended to act as a cluster of sensors that collect information about the surrounding of the artwork that are relevant to the wellbeeing of the artwork. The goal is to have constant data on the artwork throughout its lifecycle. This is intended to help with securing the artwork and helping with chain of custody monitoring.

1.2 Thesis Goals

The goal of this project is to develop a system that implements a localized version of the artwork tracking envisioned by the Certify project, ment for transportation in a truck.

Additionally the system will extend the Certify Projects goal by adding new detection methods and also informs the driver of the truck about potential problems.

The goal is to develop a system that tracks the state of artwork in a truck using different detection methods. The devices attached to the artworks, called tags, build a local decentralized network. The phone of the driver can query the network and displayed the collected metrics to them. If a metric is outside of the accepted norm, the system should alert the driver.

This thesis presents a proof of concept implementation. The used metrics are not intended to be a full representation of needed sensors to securely Transport art. Rather they are intended to show different types of sensors that can be used. This thesis assumes that the data transfer to a server using 4G, as planned by the Certify project, will work and will not implement it in this thesis.

1.3 Methodology

This Thesis was made in four stages:

Reserach

In a first step, the basis of the thesis had to be researched. This involved familiarizing with existing research on the topic of artwork tracking, local IoT networks and commonly used communication protocols. Existing artwork tracking methods need to be analyzed and evaluated, considering their strength and shortcomings during the transportation in a truck. The types of sensors that could be relevant need to be chosen, based on existing research, cost and availability. Options for the network-architecture inside the truck needed to be researched and compared, based on performance, stability and security. A communication protocol needed to be chosen, based on the same criterias.

Design

Once the fundamental knowledge for the project had been acquired, the system had to be designed. The design was chosen based on feasibility, security and stability.

Implementation

The design then was implemented in a simplified manner based on the material that was available. For this four tags were built, equipped with sensors, communication-capabilities and power supply. Then the required software was written, using existing implementations when possible and writing new code when required. A simple example app was also developed, based on an existing communications app published by Nordic Semiconductors and installed on a phone.

Evaluation

The developed system of tags and phone was tested in a series of five experiments. The first four experiments were intended to capture a specific part of the system, while the last was a general purpose test. The tests were performed in a manner that insured minimal external influence. The resulting data from the tests were analyzed using statistical methods. The goal was to determine the reliability of the system, find limitations and look for improvements.

1.4 Thesis Outline

Chapter 2

Fundamentals

2.1 Background

2.1.1 Ultra Wideband

IEEE

The Ultra Wideband (UWB) communication protocol was introduced in 2003 by the Institute of Electrical and Electronics Engineers (IEEE) as part of the IEEE 802.15.4 standard. In 2020 updates were made to the protocol when the IEEE 802.15.4z-2020 standard made improvements to the PHY layers of UWB connections. It achieved this by introducing a more robust timestamping system on the PHY layer. This is supplemented by changes to the MAC layer, that allow for the exchange of ranging information. The result is short frames, that are transmitted fast, between devices, leading to short bursts of communications that are fast, secure and ideal for ranging.

UWB works by using short radio frequency pulses, resulting in a large bandwidth. UWB is a lower power communication form. This prevents it from interfering with other communication forms it is sharing its wavelength with, such as WLAN or Bluetooth. Since UWB uses very short, distinct pulses over a short range, it has found use in ranging systems. UWB is split into high rate pulse (HRP) UWB and low rate pulse (LRP) UWB. Since ranging is part of this work and LRP is generally not used for ranging, I will not discuss it further in this thesis. Since UWB devices tend to be small and have a low energy consumption, in combination with the capability of ranging as well as data transfer, they have become popular as Internet of Things (IoT) devices.

The standard defines the PHY and MAC layer as well as frequency bands for communication. The 4z expansion tries to integrate UWB into the WPAN standard. In Section ... and ... I will discuss the PHY and MAC layer.

The sending devices emits pulses in a pre-set band of frequencies, using short bursts to transmit the bits. The signal forms a concave curve in this band, where the two points that are 10 db below the maximum power spectral density are called the lower- and upper-frequency point, see Figure 2.1. These two points must at least 500 Hz apart. The maximum power spectral density must be below the noise level. This process prevents conflicts with other communications, that use a single frequency with a high power spectral density and modulate signal transmission, such as WI-FI or Bluetooth. The UWB protocol has the added benefit of being useful for high accuracy localization.

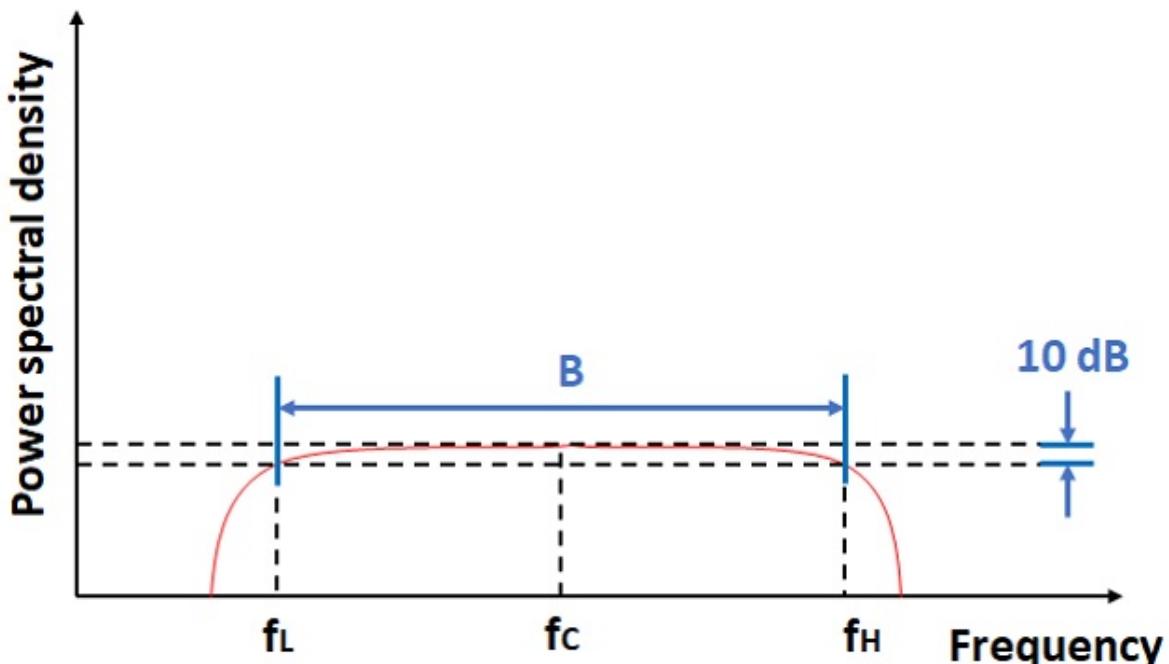


Figure 2.1: Power Spectral Density: Bandwidth B , lower-frequency f_L , upper-frequency f_H , [1]

UWB supported Nodes

The IEEE 802.15.4 standard distinguishes between two types of devices. Full-function device (FFD) are capable of connecting to multiple other devices, receive, transmit and coordinate. Reduced-function device (RFD) on the other hand can only connect to one other device and act as worker. In Topological terms RFDs can only operate as leaves, while FFDs can be any node in a network, including leaves. RFDs therefore are strictly worse, but make up for it by requiring fewer resources, such as memory and power. When FFDs work as PAN coordinators, they can use short addresses to address any node. The PAN also has a PAN identifier, to help communication accross multiple networks, while still using the short address. Each device also has a extended address, that is not assigned by any coordinator, that serves as a universal unique identifier (UUID).

2.1.2 UWB MAC

The Mac Layer is part of the Data link layer. The Mac Frame is the payload of the PHY frame. It carries information about the type of frame, frame-format, security mechanism, addressing and frame validation. The Mac Layer additionaly provides rules for beacon managment, chanel access.

Octets: 1/2	0/1	0/2	0/2/8	0/2	0/2/8	variable	variable	variable	2/4
Frame Control	Sequence Number	Destination PAN ID	Destination Address	Source PAN ID	Source Address	Auxiliary Security Header	IE	Frame Payload	FCS
Addressing fields						Header IEs	Payload IEs		
MHR						MAC Payload		MFR	

Figure 2.2: General MAC Frame Format [2]

MAC Frame Format

Figure 2.2 shows the composition of a UWB-MAC frame.

In the MAC header (MHR), the Frame Control Field includes information about:

- the frame-type
- if the Auxiliary Security Header Field is used and in what capacity
- if additional frames will follow
- if an acknowledgment message is expected
- if the message is between different PAN-Networks.
- of what type the receiver is (PAN coordinator, device, PAN-Network)
- the used frame-format standard
- where to find the source address

The Sequence Number counts up, helping to keep track of the order in which frames arive. The Adressing Fields carry the IDs of sender and recipient for the frame. The Auxiliary Security Header Field only exists if it was specified in the control Field. It contains additional information needed for the chosen security methode.

There are two parts to the information element (IE). The header IE specifies additional information about the frame, for example data formating information or chanel time

allocation. The payload IE specifies the length and data-type of the payload field. The payload contains the data that is sent. It and the IE are of variable length, depending of the frame-type and data-length.

The MAC footer (MFR) marks the end of the frame. It only contains the frame checking sequence (FCS), that can be used to detect corrupted frames using cyclic redundancy checks.

2.1.3 UWB PHY

PHY Channel

The IEEE 802.15.4z amendment defines 16 channels for communication for HRP UWB. A channel is defined by its center frequency. UWB devices can transmit on three different bands, high band, low band and sub-gigahertz. For each band there is one channel that is mandatory to support, if a device supports the band. The other channels are optional, but if two devices want to communicate with each other they need to use the same band. The bands, 16 channels and their ranges and which channels are mandatory can be found in table (see table 2.1).

Channel number	Center frequency (MHz)	HRP UWB band	Mandatory
0	499.2	Low band	✓
1	3494.4		
2	3993.6		
3	4492.8		✓
4	3993.6		
5	6489.6		
6	6988.8		
7	6489.6		
8	7488		
9	7987.2		✓
10	8486.4	High band	
11	7987.2		
12	8985.6		
13	9484.8		
14	9984		
15	9484.8		

Table 2.1: HRP UWB Frequency and Channel Assignments [2, 3]

Scrambled timestamp sequence

The 4z amendment added the option to include a scrambled timestamp sequence (STS) into the frame. The STS is a cyphered sequence that includes the timestamp and is used for ranging. It is meant to increase the accuracy and integrity of the ranging results.

Before transmission receiver and sender exchange a randomly generated key. The key is then used to encrypt the timestamp using the advanced encryption standard (AES) with 128 bits. This ensures that the signal has not been intercepted and changed, to manipulate the ranging result. Devices that support STS are called HRP-enhanced ranging capable devices (HRP-ERDEV).

Pulse Repetition Frequency

The pulse repetition frequency (PRF) is the frequency at which bursts are sent by the transmitter. The mean PRF is the average PRF while sending the payload (power switching service data unit PSDU). The higher the mean PRF, the shorter the airtime of each frame and allows for faster communication. HRP-ERDEV use a different mean PRF than general devices. They can work in Base pulse repetition frequency (BPRF) operating at mean PRF 64 MHz or in higher pulse repetition frequency (HPRF) mode operating above BPRF (Table 2.2).

Standard	HRP UWB mode	mean PRF
802.15.4	Non HRP ERDEV	3.9 MHz, 15.6 MHz, 62.4 MHz
802.15.4z	HRP-ERDEV BPRF	62.4 MHz
	HRP-ERDEV HPRF	124.8 MHz, 249.6 MHz

Table 2.2: HRP UWB Mean PRF (Based on IEEE 802.15.4 and IEEE 802.15.4z, [2, 3])

Symbol Encoding

UWB sends symbols by transmitting a burst of pulses that encode the symbol. Since the pulses have clean edges, the arrival time can be measured precisely. This leads to the burst having two ways to carry information([4]):

- Binary phase-shift keying (BPSK): Encoding zeros and ones shifting the pulses phases so the burst peak for one has an opposite amplitude to the other. Figure 2.4 shows the signal 101 binary phase-shift keyed. Each bit is set twice, to detect problems with transmission.
- Burst position modulation (BPM): Changing the timing of the burst so it falls into a different time-slot inside of the possible burst position. Figure 2.3 shows how the burst can be placed in a BPM-interval. The burst can't be placed in the guard interval. The guard exists to minimize inter-symbol interference from the signals taking multiple paths.

One or both of these encoding-strategies can be used in a uwb transmission. The position of the pulses inside of the burst (see figure 2.4) relative to each other can be used to detect the presence of multipath effects and adjust for them. Using this, precise arrival times for the whole signal can be calculated.

Non-HRP ERDEV use BPM and BPSK. Some HRP-ERDEV can use only BPSK, using a higher PRF and therefore reducing airtime.

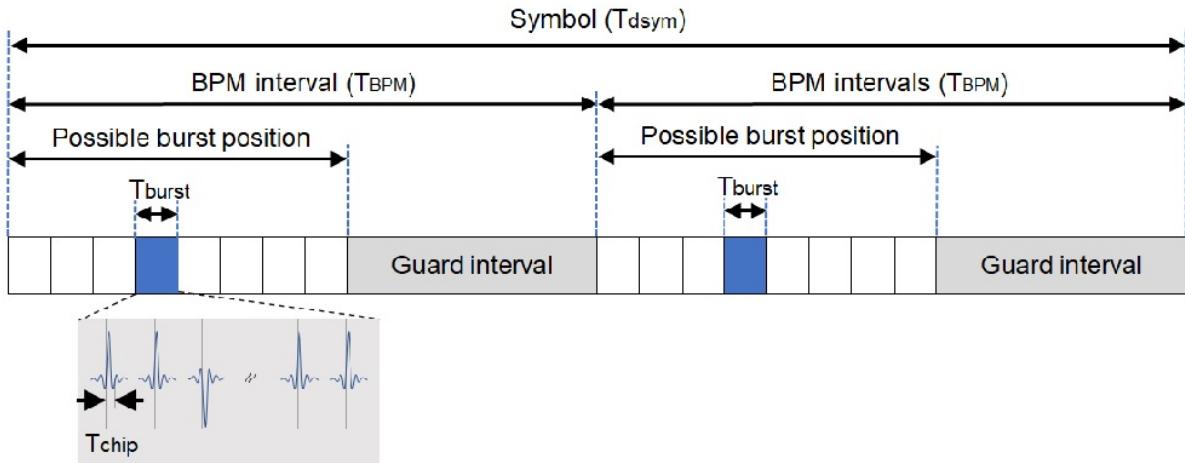


Figure 2.3: HRP UWB PHY Symbol Structure [1]

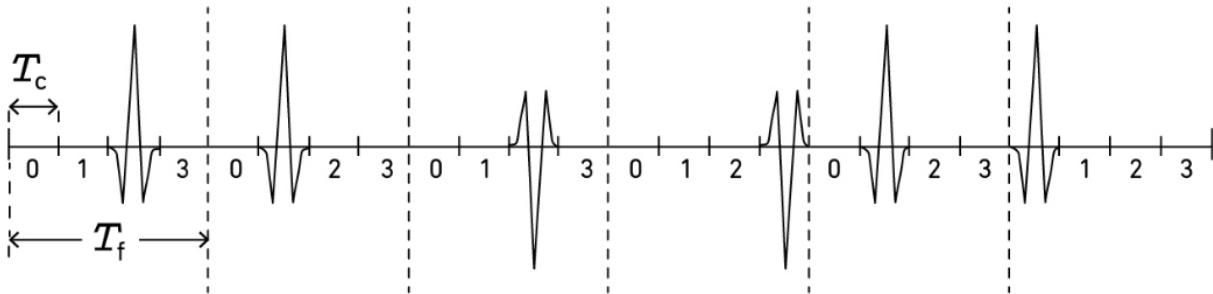


Figure 2.4: UWB signal transmission byte encoding, [4]

PHY Frame

Figure 2.5 shows a schematic view of a PHY frame as defined by the IEEE 802.15.4 standard. The Synchronization header (SHR) contains the information needed to detect the signal and adjust to its parameters. The PHY header contains meta information about the payload and its encoding. The PHY payload contains the data that is to be sent, namely the MAC frame.

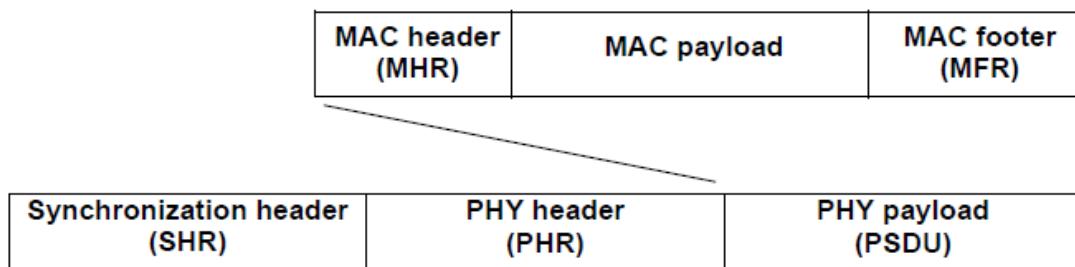


Figure 2.5: Schematic view of a PHY frame defined by IEEE 802.15.4 [2]

Figure 2.6 shows the synchronization header, consisting of two parts. The SYNC section is detectable by the receiver and informs it that a transmission has started. Depending on the predefined mode, pulses of different length. The sequence of pulses specify a set of

channels that can be used for communication. The preamble can also be used to identify a PAN coordinator.

The SHR ends with the Start of Frame Delimiter (SFD). It indicates that the synchronization has ended and the comming signals will be data, starting with the PHY header. It also contains a timestamp which can be used for ranging using time difference of arrival (ToA), see section ??

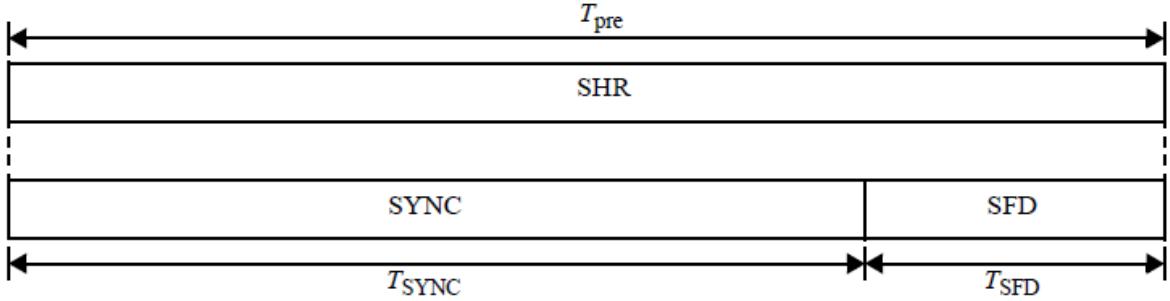


Figure 2.6: SHR Field Structure [2]

The PhY header contains all information needed to read the PHY payload (see Figure 2.7). The first bit defines the data rate that will be used during the payload transfer (see section 2.1.3). The next seven bits define the length of the frame, with a frame length of maximal 128 bytes. the 10th bit shows if ranging will be used with this frame. The next bit is reserved. Bits 11 and 12 define the preamble duration. It specifies how many repetitions are used, which can range from 16 to 4096. The last 6 bits are single error correct, double error detect (SECDED) bits that form a Hammock block and can be used to correct single bit errors and detecting, but not fixing, double bit errors.

The last part of the PHY frame is the The PHY payload (PSDU). This contains the the MAC frame, as defined in section 2.1.2.

Bits: 0–1	2–8	9	10	11–12	13–18
Data Rate	Frame Length	Ranging	Reserved	Preamble Duration	SECDED

Figure 2.7: General PHR Field Format [2]

The 802.15.4z amendment contains optional changes to the PHY frame format if the participaiting devices are HRP-ERDEV devices. Figure 2.8 shows the newly allowed structures for a UWB frame. Configuration 1 is equivalent to the already existing PHY frame. The others additionaly contain a scrambled time stamp. This can be placed in different places after the SHR. Since UWB can also be used only for ranging without transmitting a message, configuration 3 only contains the SHR and STS, without a payload.

Additionally the PHR can be formatted differently (see gigure 2.9. The reserved field and preamble duration is removed to make more space for the frame length. This allows to send more data in one frame, increasing the throughput of the UWB communication.

STS packet configuration 0	SHR	PHR	PHY payload	
STS packet configuration 1	SHR	STS	PHR	PHY payload
STS packet configuration 2	SHR	PHR	PHY payload	STS
STS packet configuration 3	SHR	STS		

Figure 2.8: HRP-ERDEV Frame Structures [1]

Bits: 0	1	2–11	12	13–18
A1	A0	PHY payload length	Ranging	SECDED

Figure 2.9: PHR Field Format for HRP-ERDEV in HPRF Mode [3]

2.1.4 Two-way ranging

The IEEE 802.15.4z UWB standard describes two ranging methods, single-sided two-way ranging (SS-TWR) and Double-sided two-way ranging (DS-TWR). In both instances, the distance measurement is done by calculating the time of flight (ToF) of a signal sent between two device using timestamps and multiplying it with the speed of light. In this section both SS-TWR and DS-TWR will be discussed. In all other parts of the thesis, two-way ranging(TWR) refers to DS-TWR.

Single-sided two-way ranging (SS-TWR): During SS-TWR, one device sends a message to a second and measures the round-trip time (see figure 2.10). Device A sends a message to B and records a timestamp when the message was sent, T_{A0} . When device B receives the response, it also records a timestamp, T_{B0} . After some delay device B will send a response to A, that contains T_{B0} and the current timestamp T_{B1} . Device A on receiving the response records its timestamp, T_{A1} . The round trip time T_{round} can now be calculated using the timestamps from A:

$$T_{round} = T_{A1} - T_{A0} \quad (2.1)$$

The reply delay T_{reply} is calculated using the timestamps from B:

$$T_{reply} = T_{B1} - T_{B0} \quad (2.2)$$

The ToF can be calculated by subtracting these values. Since the messages was send the same distance twice, the ToF needs to be halved before multiplying it with the speed of light, to get the distance.

$$distance = \left(\frac{1}{2} \cdot T_{round} - T_{reply} \right) \cdot c_{air} \quad (2.3)$$

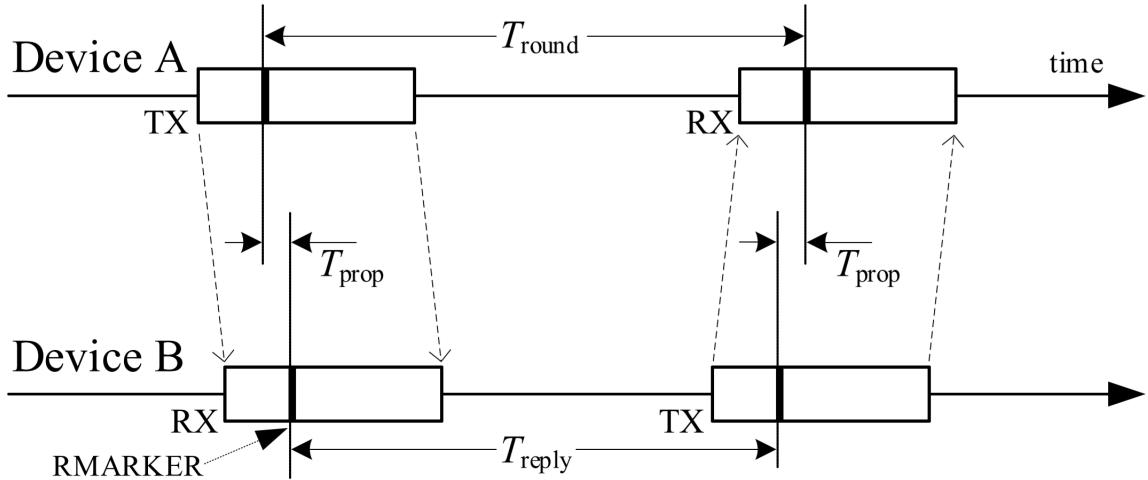


Figure 2.10: timeline of single-sided two-way ranging (SS-TWR)[3]

Double-sided two-way ranging (DS-TWR): In DS-TWR involves both device A and B performing a SS-TWR and calculating the average between the results. Figure 2.11 shows the two separate ranging sessions. Their result can then be combined to the average ToF for a single message:

$$T_{prop} = \frac{T_{Round1} \cdot T_{Round2} - T_{Reply1} \cdot T_{Reply2}}{T_{Round1} + T_{Round2} + T_{Reply1} + T_{Reply2}} \quad (2.4)$$

$$distance = T_{prop} \cdot c_{air} \quad (2.5)$$

The two ranging sessions can have one message overlapping. Figure 2.12 shows the timeline of an overlapping DS-TWR that only requires three messages.

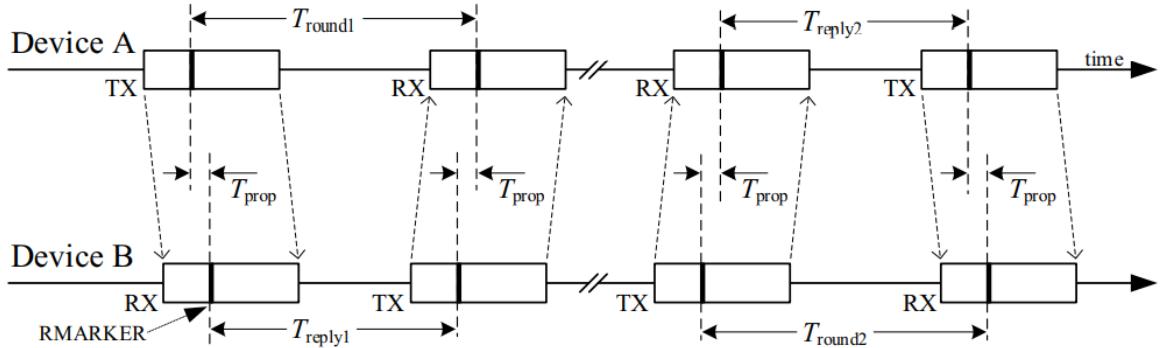


Figure 2.11: timeline of double-sided two-way ranging (DS-TWR)[3]

2.1.5 Wireless Sensor Networks (WSN)

Kevin Ashton coined the term Internet of Things (IoT) in 1999, although the idea predates this term and was before known as embedded internet or pervasive computing [?]. It

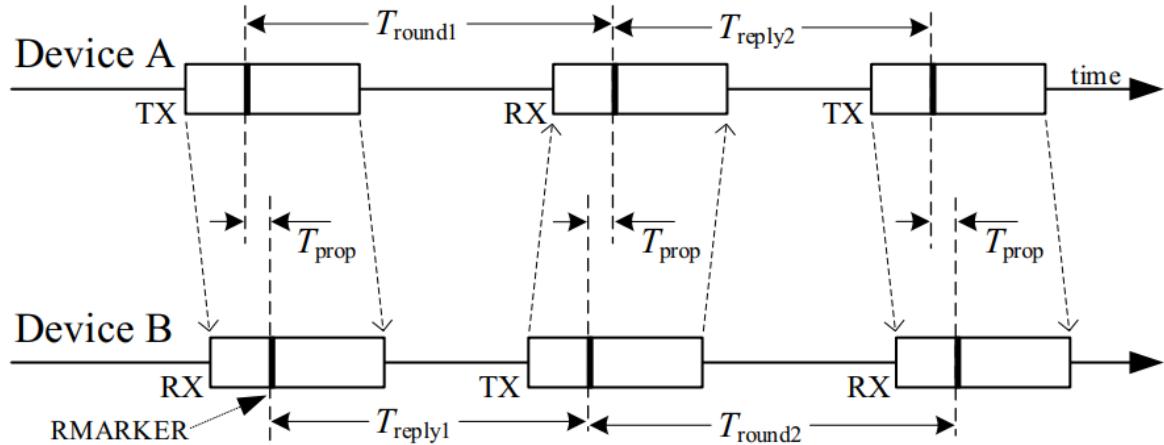


Figure 2.12: timeline of double-sided two-way ranging (SS-TWR) with three messages[3]

describes the ubiquity of digital devices and their seamless integration into the physical world and everyday life.

At the end of the 90s and during the 2000ends, the production of embedded systems in general and sensors in particular rose. This lead to falling prices and sensors becoming widespread. With the new availability of sensors, the concept of Wireless Sensor Networks (WSN) became more widespread. While not originating in during this time, the term itself was coined in 1980, the research community became more focused on the topic [?].

[?] determined four main challenges in the development of WSNs:

- Self organization: A large number of nodes should not require manual installation and maintenance
- Cooperative Processing: Sensor nodes have limited memory. Evaluating, compressing and transporting the data becomes a major challenge.
- Energy efficiency: WSNs often operate in places where no powersupply is available. The sensors therefore must run on limited, battery based energy.
- Modularity: WSNs should work for a large variety of applications and sensor node types.

The Ad hoc On-Demand Distance Vector Routing (AODV) Protocol was published in 1999 by E. Perkins and E. M. Royer [?]. It presents a routing algorithm for wireless ad hoc network, where routing is only established when needed and devices can be added to the network or leave at will. In doing this, it tackles the problem of self organization and modularity. A modified version of AODV is used in Zigbee to this day [?].

In the year 2000, Heinzelman et al. published the low Energy Adaptive Clustering Hierarchy (LEACH) algorithm [?]. Leach divides the sensors into clusters based on location. The clusters communicate with a network head using cluster-heads that collect and transmits the data of the cluster. New cluster-heads are elected periodically, to spread the increased energy drain, that comes from the data transfer to the network head. LEACH is used [?, ?] and improved [?] to this day.

2.1.6 k-connected graphs

In order to build a working positional model based on distance measurement, some background in graph theory is required. The k-connected subgraph of a graph, is a subgraph, where it takes at least k removal of vertices, to create two isolated subgraphs. A graph (V, E) can have multiple k-connected subgraphs. They build the set $S_{(V,E)}$.

A minimally weighted k-connected subgraph of a weighted graph $S_{(V,E,w)}$, is a k-connected subgraphs $(V', E', w') \in S_{(V,E)}$ that has the smallest sum of weights of all k-connected subgraphs.

A metric graph is a weighted graph that satisfies the triangle condition. Meaning for any three edges $e_{AB}, e_{BC}, e_{CA} \in E$ that connect three vertices $A, B, C \in V$, it holds that $e_{AB} + e_{BC} \geq e_{CA}$.

Finding minimally weighted k-connected subgraphs is a NP-hard problem. Kahn et al. [?] developed an approximation algorithm that finds weighted k-connected subgraphs in metric graphs. It gives an approximation of an approximation ratio of $\mathcal{O}(k \cdot \log(n))$. This means that the approximated solution w^{apr} and the optimal solution w^{opt} fulfill $w^{apr} < k \cdot \log(n) \cdot w^{opt}$. The algorithm puts all vertices in an order, which is determined at random, and assigns each vertex a rank based on the order. Each vertex then removes all edges, except for the k lowest weighted ones that connect it to a vertex with higher rank.

There are more precise approximation algorithms to find minimally weighted k-connected subgraphs [?, ?], but they are centralized, meaning the graph has to be known in its entirety by one actor.

2.2 Related Work

2.2.1 Artwork Tracking

Since art preservation is an old field and temperature, humidity, light and vibrations have been known to be detrimental to most artworks, especially paintings, most research in this direction is older than 20 years [5, 6, 7]. Still, the invention of new technologies, such as pattern recognition using artificial intelligence, improvement on existing tools like infrared imaging and a active need for solutions have kept the research into artwork preservation an active field [8, 9]. One such new technologies are sensor networks, which have become widespread in the field of art-preservation [10].

Artwork tracking during transportation has not been a major focus in academia. The most relevant related research was done by Fort et al. [11]. They developed a low-cost, low powered sensor node to track temperature, humidity, pressure and vibrations of artwork and wooden structures. The sensor node would then report its findings to a remote server. They confirmed the validity of their sensor in a series of experiments, that were performed in a static building. They also presented a theoretical framework for their sensor to be used in a transportation scenario, but they do not report having implemented or tested this system. Their sensor used an accelerometer to detect vibrations, and the

Bosch BME280 sensor to detect pressure, temperature and humidity. Their sensors did not build a network and were not queried, but reported their findings directly to either a wlan router or a ble-capable smart-device. The research of Fort et al. showed the value of low-cost sensors in the detection of threats to artwork.

2.2.2 Sensor Networks

Wireless Sensor Networks (WSN) have become a central aspect if IoT. Researchers have tried to focus on the most prevelant problems arrising from the development of NSWs, mainly power managment, security and privacy, data integrity and availability [12].

[13] researched WSNs outside of the controlled environment of a house. They propose a WSN that can track thevitals of mountainiers and call for help when measurements have dangerous values. They used an Arduuino Mega board equiped with a radio transiever, using LoRa with a star-topology, was used.

[?] created a WSN of NRF24101 board that is intended to monitor linear infrastructures like deepsee wires, using radio and wifi for communication. Using deep sleep they were able to optimize energy usage so the sensor is predicted to last five years on batery.

[?] used an acelerometer to detect vibrations in pipeline to discover leaks. They used a narrowband connection for communication and GPS for localization. Their sensors could query each other for data, to provide a more complete image of the situation.

2.2.3 Wireless ranging

[?] made an overview of publications envolving positioning systems for industrial settings. They looked at the positioning systems in papers using RFID, BLE, UWB, Wi-Fi and Zig-Bee. They found that UWB consistently reported the highest accuracy of these methods. UWB was the least affected by multipath-effects, although it was still the most common issue with this technology.

Early research of ranging using UWB was done by Gezici et al. [?, ?]. These papers gave an overview of the different positioning systems for UWB, angle of arival, recived signal strength, time of arrival and time difference of arival. Time of arival and time difference of arrival were studied further in these publications, presenting errorsources and mitigation tools.

Early research focused on augmentation of UWB ranging methods. [?] proposed using integer programs for midigating the error for ranging without line-of-sight. [?] tried to solve the same issue by using methods based on the statistics of multipath-effects. BiasSub and BiasRed was proposed to reduce the bias in time difference of arival, by appling of a well-known algebraic explicit solution for source localization [?]. [?] emproved uwb ranging by eliminating random error. They did this by pre-filtering, using a anti-magnetic ring to eliminate outliers and using the double-state adaptive Kalman filter to improve

position accuracy. Newer research has also begun incorporating neural networks into UWB positioning systems [?, ?, ?].

UWB localization has been used in many applied contexts. It has been proposed for pedestrian tracking [?], drone flying [?], robot navigation [?], navigation system for visually impaired people [?] and tracking people in buildings [?]. UWB positioning systems are particularly interesting for industrial IoT settings. [?] measured the performance of three different UWB antennas, Qorvo, Sewio, and Ubisense. They measured a lot of multipath-effects in such a complex environment. The mitigated this by employing a Bayesian filtering method. [?] used UWB positioning in combination with Real-time kinematic positioning, to track workers while monitoring the factory. The goal was to trigger an alarm if a dangerous situation occurred.

Chapter 3

Design

This section presents the principle design of the monitoring system. In the section 3.1 the components used are presented. Section ?? describes the functionalities and responsibilities of the system components. In Section ?? the network topology and data-flow is discussed.

3.1 Hardware

This section describes the hardware used in the project. The setup consists of two distinct components: the artwork-tags, of which there are four, and one Phone that provides the interface to the user. The tag itself consists of 4 components:

1. nRF52840 Microcontroller
2. DWM3000 UWB Shield
3. DHT22 temperature and humidity sensor
4. MPU6050 accelerometer and gyroscope

3.1.1 Microcontroller

The fundament of the artwork-tag is build by the nRF52840 DK microcontroller developed by Nordic Semiconductors. It is part of the nRF52 series of microcontrollers intended for development. The nRF52840 DF is specialized for ble communication, for which it already includes the necessary components. It is compatible with the nRF52 Software Development Kit (SDK), also developed by Nordic Semiconductors. The SDK makes it possible to use the ble functionalities and to control the pins. It also includes implementations for a plethora of pin based protocols. It contains 58 pins, 48 of which are data-pins and manage the power supply for additional modules, which includes 3.3 and 5 Volt supply pins. 32 of the pins are installed in the same way as the pins on the Arduino uno,

making it compatible with many peripherals that were designed with this common board in mind, such as the dwm3000. The remaining ten pins are enough to attach the sensors to. The nRF52840 DK includes a USB-B port that is used for Powersuply. Additionally it is connected to two pins and is used for UART-communication and for debugging. The nRF52 was chosen since it was available and previous projects have been done with it in combination with the DWM3000 shield. As a result a lot of initial setup was already available.

3.1.2 UWB shield

For communication between the tags as well as distance measurement the DWM3000 UWB-shield developed by Qorvo was chosen. The DWM3000 is a commonly used device for research involving UWB [14, 15, 16]. It allows low level access, but includes an SDK written in C that makes a lot of the processes transparent to the user, if they wish. The SDK uses the Serial Peripheral Interface (SPI) for communication between the shield and the microcontroller.

3.1.3 Humidity and temperature sensor

For humidity and temperature sensors I decided to use the DHT22(AM2302) produced by Guangzhou Aosong Electronic Co. [17]. It is a commonly used sensor in IoT monitoring systems [18]. The vendor claims a temperature range from -40° to 80°Celsius with a precision of 0.5°. [18] could experimentally confirm that errors did not exceed 0.1°Celsius. They also concluded that the sensor is slow in detecting temperature change. This is also confirmed by the user manual [17], that states a read-interval of less than 2s is not possible.

The humidity sensor can detect the full range from 0% to 99.9% humidity, with an advertised maximum error of 2 percent-points [17]. I could not find any research that confirmed or denied these claims.

The DHT22 sensor uses three pins from the microcontroller, two pins for power supply and ground and one for single-bus communication. Since no SDK for this type of communication has been built for the nRF52 board series, it had to be implemented manually by reading the high and low voltage on the communication pin and, detecting headers and footers and parsing the binary messages. Dmitry Sysoletin published a project on github (https://github.com/DSysoletin/nRF52_DHT11_example) that handles the communication between an nRF52840 and a DHT11 sensor. Since the communications is mostly the same, I used his implementation, but changed the parsing of the actual data to fit the encoding used by the DHT22.

3.1.4 Accelerometer and Gyroscope

The MPU6050 sensor produced by InvenSense Incorporated provides accelerometer and gyroscope data. The accelerometer reports the acceleration in the three cardinal directions

in meters per second. The Gyroscope reports the rotation around the three euclidean axis in degrees per second. In this project the accelerometerdata was not used, just the gyroscope.

The MPU6050 uses 4 pins, two for power supply and ground and two communication. The Sensor communicates using the I2C protocol, a serial synchronous communication system. The microcontroller acts as the master and would in theory support multiple workers on the same bus. Here only the MPU6050 uses I2C and is therefore the only worker. While the nRF52 SDK does not supply a I2C API, it offers a Two Wire Interface (TWI) implementation that is compatible with the I2C protocol. It even used to offer MPU6050 specific support in the older SDK. I was able to port this older code to the current SDK.

3.1.5 Tag technical plan

The microcontroller builds the base of the Artwork-Tag. The other devices are attached to it over the available pins. In the nRF52 SDK each data pin is assigned an integer value. These often correspond with the name of the pin according to the nRF52830 DK manual, but not always. I will use the names given in the manual to describe the pins. Pins are the methods by which a microcontroller controls its peripherals.

Some pins are intended for power supply. On the NRF52840 these pins are located in section P1, see table 3.1. The three VDD pins supply electricity with a Voltage of 3.5 Volts. A secondary power supply that uses 5 Volts is also available. What voltage is needed depends on the peripheral. In this case, the DHT22 runs on 3.5 Volts, while the MPU6050 is made for 5 Volts. The P1 section also contains two ground pins, that need to be connected to the peripherals and a reset pin, so restart the microcontroller. The last pin is not connected (N.C.). There are additional ground pins in sections P4 and P24 of the board.

The other pins are called data pins. By using voltage modulation these pins can transfer data and therefore be used for communication. The nRF52840 has a I/O voltage of 3.3 Volt. This means that a voltage of 3.3 Volt corresponds to a *Logic high* and 0 Volts represents a *Logic low*. This allows the data pin to transfer communication in a binary encoding. How a signal is interpreted is defined by the used communication protocol. The MPU6050 for example uses the I2C protocol and uses 2 datapins. This defines that one pin is used for a serial clock and the other pin transmits data. For the data transmission the protocol defines what a package looks like. This includes the start condition, the voltage characteristics that signal the beginning of a package, addressing, data encoding, acknowledgments and stop condition.

The DHT22 sensor does not use a given communication-protocol. It uses one data-pin to report its sensor data. How that data is encoded to high and low voltage is specified in the user manual [17] and has to be implemented manually.

The DWM3000 shield is mounted on the 32 pins meant for arduino one connections. All pins are forwarded and can be used by other devices, in a common arduino-stackable

style. If they are data-pins they will share the data. Table 3.1 shows which devices use which pins. The only pin shared by multiple devices are power and ground pins. The microcontroller supplies enough power to support this.

The sensors are attached to the same powersource and ground as the shield, but use different data-pins. The DWM3000 leaves enough pins unused that both sensors could be attached to them. Since it is not visible which pins the shield leaves free, I decided to use data-pins that are not attached to the DWM3000 in any way. Table 3.2 shows how the sensors are connected to the remaining open pins.

3.2 Architecture

In order to discuss how the dataflow works, first the section 3.2.1 will establish what services are implemented in each part of the system. The section 3.2.2 will explain what triggers events and how they are handled inside the system.

3.2.1 Responsibilities

The system consists of the tags, the sensor network and the phone. These parts all have their own responsibilities.

Tag: The tag is responsible to manage its sensors. It has to do correct setup and converts its output into a understandable form. The tag can perform ranging with all its neighbours. Additionally the tags are responsible to search for networks to join, and react appropriately to network requests, be those queries for sensor data, ranging requests or network management jobs. The tags provide a unique, secure universal identifier, to be used by queries or the network. How this is done is part of the certify project and will not be discussed in this thesis. The tag is also responsible for its own power management. This is not the focus of this thesis and will only be mentioned when relevant. A guideline on powermanagement will not be provided here.

Network: The network is responsible to keep track of all tags taking part in the network. It offers a joining protocol for new devices and remains stable when devices leave or become unavailable. It offers the possibility for phones to connect to the network. It ensures queries from phones get transported to the correct tag and the answers to the correct phone. It ensures a network topology that corresponds to a graph that is at least 3-connected. On request it returns a list of connected devices to the phone.

Phone: The phone connects to the network via the provided method. It offers a graphical user interface (GUI) to be used by the driver. The GUI offers a method for the driver to set the acceptable ranges for all sensor data. Additionally it offers a method to set query intervals-length. The phone is responsible to query sensor data for each tag and measurement once in each interval. The phone has to evaluate the answer. The phone has to report the results to the driver using the GUI. If a parameter falls outside of the acceptable range for its type, the phone is responsible to alert the driver to this fact.

		Pin	DWM3000	DHT22	MPU6050
		P4	✓		
		P3	✓		
P1.10			✓		
P1.11			✓		
P1.12			✓		
P1.13			✓		
P1.14			✓		
P1.15			✓		
GND			✓		
P0.02					
P0.26			✓		
P0.27					
P1.01			✓		
P1.02			✓		
P1.03			✓		
P1.04			✓		
P1.05			✓		
P1.06			✓		
P1.07			✓		
P1.08			✓		
P1.10			✓		
VDD	P1				
VDD					
RESET					
VDD			✓	✓	
5V			✓		✓
GND			✓	✓	✓
GND			✓		
N.C.					
P0.03	P2		✓		
P0.04			✓		
P0.28					
P0.29					
P0.30					
P0.31					

Table 3.1: Adruino compatible pin assignment

	Pin	DWM3000	DHT22	MPU6050
P6	P0.00			
	P0.01			
	P0.05			
	P0.06			
	P0.07			
	P0.08			
	P0.09			
	P0.10			
P24	P0.11			✓
	P0.12			✓
	P0.13		✓	
	P0.14			
	P0.15			
	P0.16			
	P0.17			
	P0.18			
	P0.19			
	P0.20			
	P0.21			
	P0.22			
	P0.23			
	P0.24			
	P0.25			
	P1.00			
	P1.09			
	GND			

Table 3.2: Non-Adruino compatible pin assignment

The certify project also plannes to collect the sensor data on remote servers using a 4G connection. The plan is to equip each tag with antennas to allow it to send the data directly to the server itself. Since this is not a part of this thesis, the responsibility for the tag to do this was not added to a list. A known problem with this plan is, that a 4G connection is not always possible. Since small tags have very limited memory, the plan to store the sensor data on the tag is not pheasible. If the setup presented in this thesis is used, it would allow for the storage of the data on the phone, which has a much larger memory. This again was not added, since it is not part of this thesis.

3.2.2 Dataflow

How a tag connects to the network is described in the section 4.1.3. Figure 3.1 shows a sequece diagram ob the setup and main observation loop of the system. On the top the communicating parts are listed.

- Human is the driver of the truck
- Phone is the phone used by the Human
- Network consists of all the tags that are used and the network they build.
- Connected Tag is part of this network, but is listed seperately. It represents the tag that is communicating to the phone
- Tag j is also part of the Netowrk. It represents the tag that is queried during the observation loop

Phone and Human communicate by using a GUI. Phone and Connected Tag commuincate using BLE. Every communication inside the network happens using UWB. This includes the communication between the Connected Tag, the network and Tag j.

When the phone wants to connect to the network, it looks for advertised BLE devices. It then displays the devices to the user and lets him pick one. The phone then pairs to the chosen tag, making it the connected tag and the phones connection to the network of tags. Once connected to the network, the phone will prompt the human to enter the parameters. These consist of:

- Upper and lower limit for sensor data, like temperature and humidity
- Maximal displacement value for distance and gyro. These values represent the maximal difference in registered values that is allowed for positional measurements.
- Time between measurements. This gives the time period that will pass between measurements for each device and measurement type.

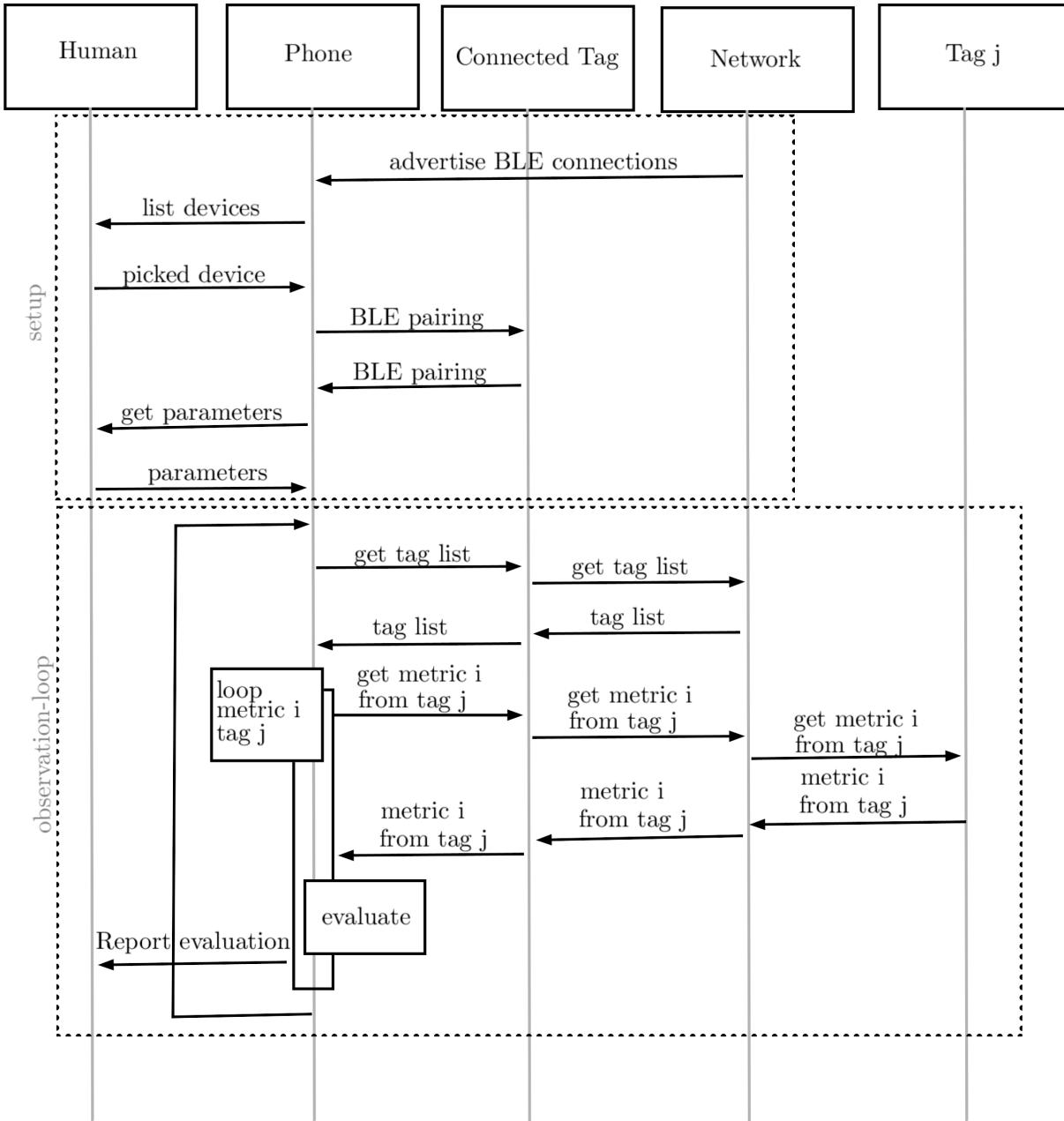


Figure 3.1: Sequence diagram of setup and oberservation loop. Setup is performed once, oberservation loop repeats until stopped.

Once the parameters chosen, the user can start the observation.

Each iteration of the observation loop begins with a call to the network for a list of all tags currently in the network. Since the tag network is a dynamic sensor network, the tags in the network can theoretically change. In practice this should only happen, when artwork is unloaded or if a tag becomes faulty. The request for the list is transmitted to the connected tag over BLE, which then queries the network for all connected devices. The response is returned to the Phone. The phone then starts a nested loop, iterating over the list of tags and the list of metrics captured by the system. For each measurement and tag combination (i,j) the phone contacts the connected tag for the value, which in turn queries the network. Once the message has arrived at the tag j, tag j gets measurement

i. In case of sensors this entails contacting the sensor and requesting a value. If metric i is a distance measurement, tag j will commence a two-way ranging operation over UWB with all its registered neighbours and will report the list of distances, together with the tag-addresses they correspond to. Metric i is then transported over the network back to the connected tag and finally to the phone. The phone must then evaluate the retrieved data.

During the evaluation process, the phone creates an evaluated measurement, and marks it as problematic or unproblematic. What the evaluation looks like depends on the metric.

- For most metrics, like humidity and temperature, the evaluated measurement is equivalent to the received measurement. It is then checked, if the measurement falls into the acceptable measurement parameters, set by the human. If it does not, the evaluated value is marked as problematic.
- Some metrics require comparison to the previous data. The gyroscope reports the current orientation of the tag. This is then compared to all previous measurements and the maximal angular difference forms the evaluated measurement. If the evaluated metric is bigger than allowed by the set parameters, the measurement is marked as problematic. After evaluation the original measurement is added to the list of previous measurements.
- The distance measurement has a unique evaluation process, which is described in section 3.3.

Once the data evaluation is done, the evaluated measurement is presented to the user over the GUI, together with the address of the tag it belongs to. If the evaluated measurement is problematic, the driver is alarmed.

Distance evaluation

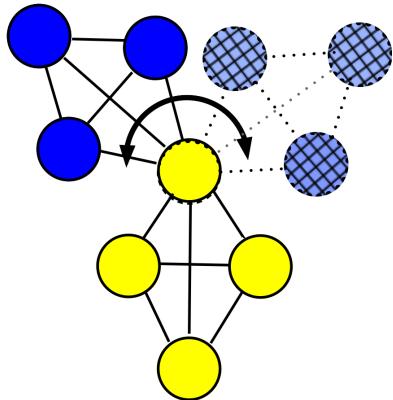
The goal of the distance evaluation is to build a working model of where every tag is. To achieve this, a quadratic program is solved, to get the coordinates of all tags. The steps to do this are as follows:

1. Get a list of all current tags, $T := \{t_1, t_2, \dots\}$.
2. For each tag, get the last known distance measurements and put it into a set $S_D := \{(t_i, t_j, d_{ij})\}$, where t_i is the tag which measured, t_j the tag that was measured to and d_{ij} the distance measured.
3. If a tag has no distance measurements, remove it from the list.
4. Assign each tag t_i a position in a 3D coordinate system, (x_i, y_i, z_i)
5. Pick one random tag t_o .
6. Set the values x_o, y_o, z_o all to 0.

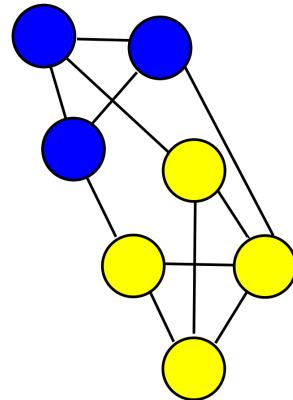
7. Create the objective function: $L(X, Y, Z) = \sum_{t_i, t_j, d_{ij} \in S_D} |(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 - d_{ij}^2|$. For x,y and z use variables for all but the six values set in step (6).
8. Solve the quadratic program consting of the Objective function L and no constrains.

Quadratic Programs in general are NP-Hard, but Quadratic Programs with a convex function can be solved efficiently. $(a - b)^2$ and c are convex functions. The sum of a convex function is always a convex function. The objective function in (7) only sums up convex functions and is therefore convex itself. The quadratic program can therefore be solved efficiently.

By setting the values of tag t_o to zero, the results of the quadrate function becomes grounded. It is not strictly necessary, but without it the returned solution could have values anywhere in the Euclidean space. by setting one tag to the coordinates at the origin, the solution will place the other tags near that region. There are still an infinite amount of solutions to this quadratic function, since all solutions can be rotated around any axis and still return the same objective function.



(a) 3-edge connected graph



(b) 2 connected graph

Figure 3.2: Left: Five dots, all having at least two connections, still blue can move independently. Right: minimal 2-connected graph, no movement possible.

For a point to be clearly placed in Euclidean space, three distances to other points have to be known. This alone is not sufficient to insure a unique result. The left of Figure 3.2 illustrates this point in two-dimensional space. Every circle is connected to two others, still the blue circles can move without the whole figure moving. What is needed to keep every point static is for known distances and tags to build a four-connected graph (three in two dimensions). The left of Figure 3.2 shows a solution of the problem on the right by creating a three-connected subgraph.

Once the coordinates for all tags are found, they are compared to previous results. For each tag the phone calculates by how much it has moved. The evaluated measurement is the distance of the tag that has moved the most. If the evaluated measurement is larger than the maximal allowed displacement, the measurement is problematic.

3.3 Network

For the presented network to work, tags need to be ranked. This means that for each tag pair i, j , one can either say that $\text{rank}(i) < \text{rank}(j)$ or $\text{rank}(j) < \text{rank}(i)$. To achieve this, the universal unique identifiers are used. No matter what form the UUID has, it can be converted to an integer, by simply interpreting its binary code as one. Since the UUIDs are unique, no two tags will have the same resulting integer. When referring to the rank of a tag in this section, the integer representing the UUID is intended.

The tags inside the truck, while not connected to a phone, form a decentralized mesh using UWB for communication. Each tag keeps two lists, a list of known devices and a list of neighbours. When a new tag joins the network, it sends a joining request over UWB, containing its universal unique id (UUID), using a weak signal. All tags in the network that receive this request add the new device to their list of known devices. If the new device also has a higher rank, they additionally add it to their list of neighbours. They then answer by sending their own UUID and address back to the new tag. By waiting an amount of time that correlates with their UUID, the tags in the network can ensure, that their answers don't overlap. The new tag adds the received addresses and UUIDs to its known device list. If the rank of the added tag is also has a higher rank than the new tag, it will add it to the list of neighbours. If the new tag now has four neighbours, it stops. Otherwise it will repeat the process with a increasingly stronger signal, until it has either found four tags with higher rank or reached maximum signal strength. Afterwards it starts advertising its BLE connection. This concludes the network joining process.

A user with a phone can connect to any of the advertised BLE connection. Once that happens, the tags in the truck will switch from their decentralized mesh to a star-topology, with the connected device serving as the coordinator. The coordinator will inform all tags about its new status, by sending a message using a strong UWB signal. The tags will then acknowledge this message in order of rank. The tags in the network will still keep their stored neighbours and known devices. The coordinator records a list of all acknowledgments, thus creating a list of all devices in the network.

The phone can request the list of all tags from the coordinator. The phone can now also query the tags in the truck by sending the query to the coordinator over BLE, which then will pass it directly to the appropriate tag using BLE. For all sensor data, this is a simple call and response request.

If a distance measurement is queried, the tags takes the following steps:

- It conducts a UWB two-way ranging session with each tag in the neighbour list.
- It reports those results to the coordinator tag.
- It orders all received distances.
- It keeps the tags with the four lowest distances and deletes the rest from the neighbour list.

The first time a distance is requested, the tag will perform more ranging sessions than necessary to build a 4-connected graph. Afterwards it only perform ranging with four other tags, unless a new device was added. If a ranging session does not report a result, because a tag left or became unavailable, the tag adds ads the tag with the shortest previously measured distance and higher rank from the list of known devices back intro the list of neighbours.

This design mirrors the algorithm proposed by [?, khan2007simple]nd presented in section 2.1.6. It creates an approximation of a minimally weighted k-connected subgraph, based on the measured distances. This is allowed, since the distances are in euclidean space, which when mapped to a graph forms a metric graph. As discussed in section , a four-connected graph is needed to uniquely identify the position of each tag. The graph should be minimally weighted, so that measurements are between tags that are as close as possible to each other. This reduces the multipath effect and therefore increases precision.

If the tags are not connected to a phone and report their data to a remote server, they can still use the same distance-measurement, to approximate the k-connected subgraph. The quadratic program can then be calculated on the server.

Chapter 4

Implementation

In this section, the implementation that was used for the experiment is discussed. In section 4.1 the implementation of the tags is presented. Section 4.2 is about the implementation of the App.

4.1 Tag

The software of the tags consists of n modules:

1. Temperature and humidity sensor
2. Gyroscope
3. UWB network
4. Two way ranging
5. BLE communication
6. Job handler

The following subsections will discuss the first five modules, followed by how they interact using the job handler module. The section 4.1.7 discusses challenges from combining these modules and how they were solved.

4.1.1 Temperature and humidity module

This module is responsible for managing the DHT22 humidity and temperature sensor. It is responsible to setup the sensor during initial startup and to provide the sensors measurements when queried. The DHT22 sensor communicates using only one data pin, pin 13, which will be referred to as the data pin in this section. Dmitry Sysoletin created an

implementation ?? for the DHT11 sensor together with the nRF52840 board that build the basis for this implementation, by addapting it to the DHT22 and adding functionalities needed by the job handler module.

Since the DHT22 is a very simple sensor, using single bus communication, not much setup is needed. The evaluation of the sensor data requires that the voltage of the pin is read out in pre-defined intervals, when reading the sensor data. To do this, a clock is required. This resource has to be reserved an initiated at startup. This is the only setup that is required for the DHT22 sensor.

To initiate a sensor-read the voltage of the data pin is set to 0. When the sensor is in standby mode, the data pin is on *logic high*, and when set to *logic low*, the sensor will respond with a read of its current value. A schematic view of a sensor read of the DHT22 can be seen in Figure 4.1. The temperature and humidity module will then check the Pin State in intervals of 5ms, until a *logic low* is registered, signalling that the sensor has registered the request. The module will now monitor the pin state, waiting for *logic low* followed by a *logic high*, this beeing the start condition of the data transfer.

The data is transferred in five chunks of eight bits. Each bit is preceeded by a prolonged *logic low* state, that is detected by the module. The module then proceeds to write the state of the data pin into a 8-bit buffer, *logic high* corresponding to a 1 and *logic low* to 0.

Once all five chunks are read, the communication has ended and the module can verify the data. The first two bytes correspond are combined to form the temperature information in celcius, the second and third form the humidity. Both values are multiplied by 100 and stored in a 16-bit integer. This doesn't loose data, since the sensor only measures up to a precision of 1 after the decimal point. The data beeing stored in an integer help with data transfer. It will be converted back on the phone. The fifth chunk contains the parity and is used to accept or reject the humidity and temperature values. If the process fails at any state, -100°C is returned for the temperature and -100These form both impossible values, since humidity can't be negative and the DHT22 sensor can only detect temperatures as low as -20°C.

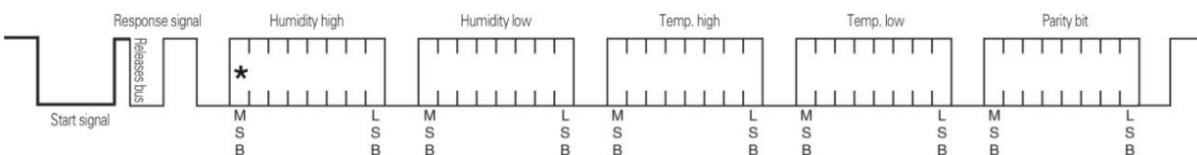


Figure 4.1: Signal of a DHT22 sensor-read as presented in the manual [17].

4.1.2 Gyroscope

This module manages the MPU6050 gyroscope and accelerometer. It is responsible to setup the sensor and report its result. An implementation for the MPU6050 was present in the nRF52 15.3.0 SDK, but is no longer available for the nRF52 17.1.0 SDK, used in this project. The old implementation was ported to this project. This consisted of replacing deprecated parts of the SDK with updated ones and adding newly required flags to the build.

MPU6050 sensors use the I2C communication protocol. The nRF52 SDK does not include an implementation for this protocol, but has a Two Wire Interface (TWI) implementation that is compatible with the I2C protocol. During startup the TWI module has to be initialized. This is handled by the SDK, but requires some parameters to be passed.

- The Serial Clock Line (SCL) defines what pin will be used for the clock shared in the TWI. This implementation uses pin 11.
- The Serial Data Line (SDA) defines which pin is used for the data communication. Pin 12 was used.
- The frequency which the TWI uses. It is defined in MPU6050 data sheet, and is 100 kHz [?].
- The Interrupt priority is a rank that determines, how easily this process can cause an interrupt. It is set to high.

After the TWI service is initiated with these parameters, it is enabled, ensuring that its resources are locked and can not be used by other services.

Afterwards the results from the sensor can be read using the TWI service again. The TWI-TX requires the address of the read device and a registry where to write the MPU6050 datasheet [?]. The address of the sensor is the same for all MPU6050 sensors and can be found in the manual. It sets a flag to true once the sensor has written the data, which then can be read using the TWO-RX function. The result consists of three 16-bit integers, representing the angular velocity around the X, Y and Z axis, shown in figure 4.2.

Returning this data when queried has only limited use. It represents a measurement of the current situation. The caller is more interested what happened since the last query. Two different implementations for the read of the gyroscope were used during the experimental phase of this thesis. One would try to return the current orientation of the tag. This read will be called the *orientational read*. The other would return the maximal registered angular velocity since the last read. This will be called the *angular velocity read*.

To achieve the orientational read, three orientational variables x_{angle} , y_{angle} and z_{angle} keep track of the current rotation around their corresponding axes. During setup, all three angles are set to zero. The MPU6050 is read out periodically in between calls. The elapsed time since the last read is multiplied with the angular velocity at this moment around the axis and is added to the orientational variables. When the gyroscope module is queried for its measurement, x_{angle} , y_{angle} and z_{angle} are returned.

The angular velocity read is achieved in a similar manner. Three angular velocity variables x_{max} , y_{max} and z_{max} are created and set to zero during initiation. The MPU6050 is read out periodically and its values are compared to the angular velocity variables. If any of the angular velocity values is smaller in absolute magnitude than the corresponding read value, it is replaced by that read value. When the gyroscope module is queried, the values of x_{max} , y_{max} and z_{max} are returned. The angular velocity variables x_{max} , y_{max} and z_{max} are then set to zero again.

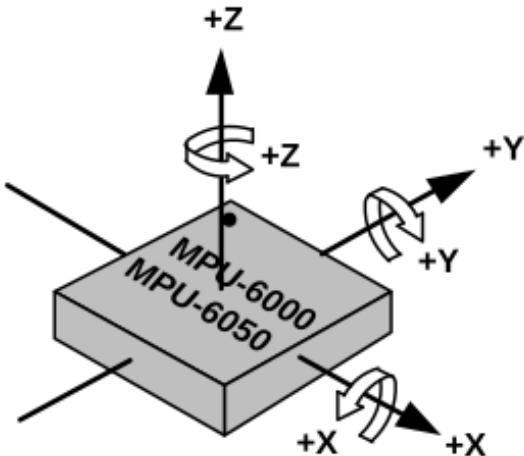


Figure 4.2: Schematic view of the MPU6050, showing the direction of the three axis X,Y,Z.

4.1.3 Network

The network module is responsible for the management of the network. This consists of: sending requests to join a network, managing requests to join a network, keeping track of its neighbours, transmitting messages and sending messages. Since only four devices were used in this implementation, the processes for the network are much more simplified, than presented in design chapter. A 4-connected minial graph of 4 vertices must neccarily include that all the nodes are fully connected. This leads to a simplified network architecture. Since this implementation was build to run experiments and not to be used in real-world applications, a lot of security measures were canceled. Messages are not encrypted and devices are not authenticated. All messages are assumed to reach their destination and no devices is expected to become unavaliable.

The Network-module is based on the implementation of [?]. It is based on published examples from Qorvo, the producer of the DWM3000 shield. It uses the DWS3000 SDK to communicate with the DWM3000.

The DWM3000 uses the Serial Peripheral Interface (SPI) protocol. This requires some resources that have to be reserved and some configurations that need to be set. This is the first thing that happens during the setup of the UWB network. Next the interrupt-priorities and the communication speed of the SPI connection are configured. Then the DWM3000 is reset, to insure no cross-effects from previous sessions are possible. Then the board is told to initialize. After that the used configurations are send to the shield. This includes information like chanel number, preamble codes, data rates and header modes. The SDK contains many pre-defined configurations. All configurations that allow for RX and TX and that use scrambled timestamp (STS) work for this usecase. It is crucial that all tags use the same configurations. For this implementation the same configurations were used, as in [?]. The configurations can be seen in table 4.3. The setup finishes with initiating the LEDs, that serve no critical service, but are usefull for debugging.

Figure 4.3: Configurations of the DWM3000 for UWB communication

Description	Value
Channel number	5
TX preamble length	128 symbols
RX preamble acquisition chunk size	8 chunks
TX preamble code	9
RX preamble code	0

The Certify project uses unique, falsifiable identifiers for its tags. Since this is not available for the tags used here, the device-ID was used instead. It serves as a 8-bit long address for the purposes of this implementation. Each tag also keeps a list of all known addresses, called neighbours.

The address *0x3F* was used, when a tag wants to join a network. This was chosen since none of the used devixes had this device-ID, and it corresponds to a question mark when using ASCII encoding. When a tag wants to joind a message, it sends the adress *0x3F*, followed by the message 'findnet', and its own adress. It then start listening for answers. If the listening timed out without any answers, it sends the message again.

For the network to function, the receiving and sending of messages is critical. The UWB listener function from project [?] was modified. It waits for a listened message from the shield. If it receives a message, it copies it to a buffer. It then checks the first bit of the message for the receiver adress. If the receiver adress is equivalent to the tags own adress, it passes the message on to the job-handler module for further evaluation. Otherwise the message is discarded. An exception is made, if the receiver-adress is "*0x3F*", indicating that a tag is looking for a network. In that case, the network module adds the tag to the list of neighbours. It then waits for a time proportional to its own adress, before continuing. Since addresses are unique, this ensures that no two tags responde to the new tag at the same time. Afterwards it sends a new message, beginning with the adress of the new tag, followed by the string 'NEW' and its own adress. This way it can be added to the neighbours of the new tag as well.

For sending messages, the implementation of [?] was modified. It sets the DWM3000 to TX, passes a int-buffer and lets it transmit, before returning to RX mode. Due to limitations discussed in section 4.1.7, the message length could not exceed 10 bytes.

4.1.4 Two-way ranging

The two-way ranging module is responsible for measuring its distances to the tags in the neighbourhood. Since it also uses the DWM3000 shield, it requires no additional setup.

When the two way ranging module gets a distance request, it loops over the list of neighbours, performing two-way ranging with each of them. First it sends a prepare-rainging request to the neighbour it wants to performe ranging with, before performing the ranging. It then sends the result back over the network to the requesting tag with the following format:

$$a_r \text{DST} a_t a_n c d_{tn} \quad (4.1)$$

with

- a_r : The adress of the requesting tag.
- DST: The string "DST", indicating the purpose of the message.
- a_t : The own adress of the tag performing the measurement.
- a_n : The adress of the neighbour that the distance was measured to.

- c : A boolean. If false, this is the last neighbour measured for this query.
- d_{tn} : The distance to measured.

The reason for each measurement triggering its own response is the message-length limitation mentioned in section 4.1.3.

When a tag receives a prepare-ranging request intended for another device, it enters a short sleep. This is because ranging involves multiple messages being sent between both participants. This would unnecessarily drain energy from the tags that are not involved. Because of that they sleep for the expected duration.

If the tag is the intended receiver for the prepare-ranging message, it will enter the preparation part of the two-way ranging module. It will function as device A in respect to figure 2.12. In a first step, it will clear all RX and TX buffers. It then sets the expected RT and TX antenna delays, d_{rx} and d_{tx} . They represent the expected time loss during receiving or transmitting messages and are device specific. These delays will automatically be taken into account, when calculating the timestamps. It then sends the first polling message and immediately starts waiting for a response. The polling message is a constant string with no changing data. The DWM3000 will automatically store the transmission and reception timestamps, there is no need to retrieve it right away. When the response is received, it checks if it is the expected response. If it is, the two timestamps $T_{TX_1}^A$ and T_{RX}^A are retrieved. The final transmission time $T_{TX_2}^A$ is calculated by adding a constant c_A to T_{RX}^A :

$$T_{TX_2}^A = T_{RX}^A + c_A \quad (4.2)$$

The final message is then prepared, containing all three timestamps $T_{TX_1}^A$, T_{RX}^A and $T_{TX_2}^A$. The message is loaded into the message buffer of the DWM3000 and a delayed transmission is started. The delayed transmission takes the timestamp $T_{TX_2}^A$ and will start the transmission once that timestamp is reached. Afterwards all caches are cleaned and the tag returns to its previous state, listening for requests.

The tag that performs the ranging responds to device B in figure 2.12. Once it has sent the the prepare-ranging message to its neighbour, it will enter the receiving part of the two-way ranging module. As device A, device B will also start by setting its antenna delays d_{rx} and d_{tx} and clear all its RX and TX buffers. It will then start polling for a message. Once a message from device A is received and validated, it will retrieve the timestamp when the message was received, $T_{RX_1}^B$. Device B will add a constant c_B to this timestamp to get T_{TX}^B :

$$T_{TX}^B = T_{RX_1}^B + c_B \quad (4.3)$$

It will then start a delayed transmission for the response message at T_{TX}^B . The response is a constant string without any data. Once the response is sent, device B starts to listen for messages again. When the final message is received from device A and validated, $T_{TX_1}^A$, T_{RX}^A and $T_{TX_2}^A$ are extracted from the message. Device B also retrieves its final timestamp, $T_{RX_2}^B$. Once this is done, the time of flight for a single message can be calculated, and

from that the distance:

$$T_{round1} = (T_{RX}^A - T_{TX_1}^A) \quad (4.4)$$

$$T_{round2} = (T_{RX_2}^B - T_{TX}^B) \quad (4.5)$$

$$T_{reply1} = (T_{TX}^B - T_{RX_1}^B) \quad (4.6)$$

$$T_{reply2} = (T_{TX_2}^A - T_{RX}^A) \quad (4.7)$$

$$ToF^{AB} = \frac{(T_{round1} \cdot T_{round2}) - (T_{reply1} \cdot T_{reply2})}{T_{round1} + T_{round2} + (T_{reply1} + T_{reply2})} \quad (4.8)$$

$$distance = ToF^{AB} \cdot c_{air} \quad (4.9)$$

The distance is then returned, all caches cleared and the module continues with the next distance measurement, if any are remaining.

The TX and RX antenna delay d_{rx} , d_{tx} are different for each device. Qorvo supplies a default value, but it is the same on all devices. Since the antenna delays are multiplied with the speed of light, even small mistakes in calibration can lead to big errors. According to Qorvo, without the calibration of antenna delays, a measurement can be off by up to 40 cm [?]. This will be a constant bias and not change over measurements.

Qorvo has published a manual on how to calibrate their devices [?]. They have not published a codebase that implements this process. The calibration process published by Qorvo required things that were not part of this project:

- A synchronized clock, shared over all devices, without significant clockdrift
- A UART connection to a computer
- A pipeline performing statistical analysis and coordinating the devices.

Since implementing this calibration process would have been out of scope for this thesis, a simpler version was designed. The tags were set up in a tetrahedron, so each tag was 30 cm apart from each other. Then one tag would perform two way ranging with another tag, chosen at random. The result would be shared between both tags. If the result was larger than 30 cm, d_{rx} or d_{tx} would be chosen at random and increased. If it was lower, d_{rx} or d_{tx} would be decreased. Then the second tag would start a new ranging session with a random tag. This system was left running for over one hour, until all distances measured were in the range of [27 cm, 33 cm].

4.1.5 BLE

The BLE module is responsible for the communication between the UWB network and the phone. It advertises the tag to the phone and receives messages from the phone and sends messages to the phone using BLE. The nRF52840 microcontroller is equipped with a antenna with BLE capabilities. The nRF52 SDK includes libraries for the management of this antenna. It also includes the *ble_app_uart* example project. This project offers a ble connection, handles the pairing process. Once connected, it forwards all

incomming communication to a USB-UART mduoule connected to a computer. Input from the computer ver USB-UART is sent as a message to the paired device. The *ble_app_uart* example project was took as a basis to build the BLE-module.

The nRF52 SDK for BLE requires the use of the S140 SoftDevice. The S140 SoftDevice is a BLE protocol stack that can be used for the 811, 820, 833 and 840 series of nRF52 boards. In order for the SoftDevice to be available, a memory 156 kilobyte segment of memory has to be reserved for it, starting at memory segment 0x0. The SoftDevice then has to be flashed to the board.

During startup, the BLE module has to initialize a few services and reserve some resources. Firstly a nRF clock has to be reserved for the BLE module. Then the powermanagement for the SoftDevice has to be initiated, before the BLE stack inside the SoftDevice can be initialized. Next the Generic Access Profile (GAP) and the Generic Attribute Profile (GATT) have to be prepared. The information what functions to call when the SoftDevice receives data has to be set, as well as the advertized name, the UUID, timeout durations and what to do on faults. The advertized name was left unchanged from the *ble_app_uart* example, "Nordic_UART".

Once the SoftDevice is initialized and the tag has connecteced to the UWB network, the BLE connection can be advertized. The advertisement function of the nRF52 SDK was used for this.

The BLE module listens for queries sent from the Phone to the tag using BLE. To achiev this, a query-handler function was passed to the SoftDevice during initiation. All incomming messages will be passed to this function by the SoftDevice. When a query is received, the BLE module interprets the message. It checks what is beeing queried and transformes it into a job, readable by the Job Handler module. The BLE module also offers a service to send messages to the phone. This service uses the nRF52 SDK to load the message into a the SoftDevice and send it to the phone.

4.1.6 Job Handler

The job-handler module connects all other module. It takes job structs (see figure 4.4, interprets which module is responsible for handeling them and calles the job together with the relevant data. The job struct consits of a field for the job-type, that tells the job-handler what type of job this is. It also includes fields to store data, that is needed for the job.

```

1  struct job {
2      enum job_types type;
3      uint8_t* data;
4      int length;
5  };

```

Figure 4.4: Job struct

There are 14 total job types. The following list while decribe the meaning of them, as well as how they are handled by the job-handler:

- **search for network:** This job is triggered after setup. The tag is not connected to the network. It will be passed to the Network module without any additional data.
- **join network request:** This job comes from the Network module, when it receives a request from another tag to join the network. It will be passed back to the Network module, with the data of the new device's ID.
- **set network and address:** This job comes from the Network module. It informs the network connection has been established. The job is handed back to the Network module, with the received message, to be added to the list of neighbours.
- **ble temp hum request:** This job comes from the BLE module, where a query for temperature and humidity has been registered. The requested tag is extracted from the job. If the request is for this tag, the job is handed to the Temperature and Humidity module. Otherwise it is passed to the network module, to be transmitted to the requested tag.
- **temp hum request :** This job comes from the Network module and informs that a request for a temperature and humidity read has been made. It is passed to the Temperature and Humidity module, together with the requesting tag's address.
- **temp hum answer:** This job comes from the Network module and carries the response to a temperature and humidity request. It is passed to the BLE module, together with the measurement, which will be passed to the phone.
- **ble gyro request:** This follows the same logic as "ble temp hum request", but with the gyroscope module.
- **gyro request:** This follows the same logic as "temp hum request", but with the gyroscope module.
- **gyro answer:** This follows the same logic as "temp hum answer"
- **ble distance request:** This job comes from the BLE module. The phone has queried for a distance. If the queried tag is not this tag, the message is passed to the Network module. Otherwise, it is passed to the Two-Way Ranging module.
- **distances request:** This job comes from the Network Module. It requests a distance measurement. The job is passed to the Two-Way Ranging module, together with the requesting tag's address.
- **distances prepare:** This job comes from the Network Module. It informs that another tag is requesting a ranging session. If the ranging session is with this tag, the job is passed to the Two-Way Ranging module. Otherwise the tag goes to sleep for a short time.
- **distances answer:** This job comes from the Network module. It reports that a distance measurement has been returned. The job is handed over to the BLE module, together with the content of the message.
- **ble get known devices:** This job comes from the BLE module. It requests a list of all neighbours. The job is transferred to the Network-Module.

4.1.7 Combining modules

Each module except for the job-handler module was developed in separate projects, to ensure operability. Afterwards the modules were merged into one project. The Network module was chosen as the base project, that the other projects were merged into. This was chosen since the Network module was based on [?], which intern was based on a example published by Qorvo. The Qorvo example uses a lot of shorthand, magic numbers and development shortcut, that are not easily readable to developers outside the firm. The Network module whas therefore chosen as a basis, since merging it into another project would likely be cumbersome, since parts would easily be forgotten or interact poorly, without the knowledge or understanding of the developer. Combining the modules came with several chalanges, that described in this section.

The Qorvo example that builds the basis of the Network module uses the pin-mapping PCA10056. This is the pin mapping for boards that include the NRF52840 board, but not the NRF52840 development board, that this example was made for and is used in this thesis. The NRF52840 board does not contain the nesecary pins to attack a DWM3000 board to it. This wrong pin-mapping leads to mistakes that the Qorvo example has to work around.

When switching to the correct pin-mapping, PCA10040, the Network mduole would no longer work, since those work-arounds now introduced mistakes now. Since fixing the Qorvo example code would have been cumbersome, it was decided to instead change the other modules that used pins, the Gyroscope module and the Themperature and Humidity module. The pins for those modules, pin 11, 13, and 13. where hard coded into the modules, instead of using the pin-mapping.

The nRF52 SDK offers a rich selection of tools, such as SPI and TWI communication, clocks, ble capabilities, SoftDevice, UUIDs. These tools are all enabled or disabled in the sdk_config file. Merging in general requires only to enable the tools needed by the merged module.

Three mdoules requie a nRF clock,, Two-Way Ranging, Temperature and BLE. The nRF SDK offers exactly three clcoks slots, so all of them have to be enabled with the appropriate clock type. Each module has to be adapted, so it uses its assigned clock-slot.

The nRF52 SDK can suport up to three SPI or TWI conenctions simultaniosly, nemaed SPI0, SPI1, SPI2, TWI1,TWI2 and TWI3. SPI and TWI share their memory, so SPI0 can not be used while TWI0 is used and vise-versa. Since the DWM3000 uses two SPI connections and the MPU6050 uses one TWI connection, exactly enough resources remain, for both devices to run simultanisously. SPI0 and SPI1 were used for the DWM3000 and TWI3 for the MPU6050.

All other SDK resources were non-conflicting. They were ported from the original module implementation to the merged one without change.

As most embeded systems do, the nRF52840 requires static memory allocation during flashing. The available memory is seperated into flash-memory and random-access-memory (RAM). Some memory segments are required by every runnable system:

- **FLASH, vectors:** The interrupt vector table defines the interrupt handlers for the system, like resets, faults.

- FLASH, **init**: The initialization routine sets up clocks, pins and other peripherals.
- FLASH, **text**: This section contains the executable code in machine language.
- FLASH, **data**: This section contains the initial values for all global values..
- **rodata**: This section contains the constant variables, that will not change at runtime.
- RAM, **data**: During startup, the initial values for changable global variables are copied to this section. They can change at runtime.
- RAM, **bss**: This section contains the global variables that do not have initial values.
- RAM, **stack** and **heap**: The stack and heap that build the runtime environment.

Neither the MPU6050 nor the DHT22 require any additional memory segments. The DWM3000 and the BLE module both require additional memory segments.

The BLE module requires the SoftDevice to be added to memory. The Softdevice requires 156 KB of Flash and 10.7 KB of RAM. Those reserved memory segments need to be the first one in both Flash and RAM. This additionally requires SoftDevice observers for System on Chip (SoC), BLE, state and stack. Additionally a segment to house the nRF52 SDK memory allocator is required, nrf_balloc. These segments are rather small, never exceeding 32 bytes.

The DWM3000 shield requires two additional memory segments, fConfig in Flash and nrf_balloc in RAM. Qorvo does not publish what the fConfig module is for, but it is required for the shield to work.

Since the base project was made for the DWM3000 shield, it had to be adapted to additionally fit the segments needed for the BLE module. This mainly consisted of moving all segments to later address-spaces to add room for the SoftDevice reserved memory. All other memory segments had to be added as well. To make room for this, the Flash memory had to be expanded.

The Qorvo example implementation for the DWM3000 shield uses some work-arounds. An example of this is the "NRFX_SPIM3_NRF52840_ANOMALY_198_WORKAROUND_ENABLED" present in the SDF configuration. These workarounds let the SPI communication with the shield perform certain memory manipulation. If these workarounds are necessary is doubtful, but fixing them would have been out of scope for this thesis. The workarounds do generally have no effect on the implementation, with one exception. When the DWM3000 receiver sends a message longer than 10 bytes to the microcontroller over SPI, it intrudes on the SoftDevice RAM. This behaviour was found experimentally, the responsible code could not be located. Since the system can be implemented with the restriction of 10 byte messages, this was done.

4.2 App

Nordic Semiconductor, the maker of the used microcontrollers, published the code to a simple app that allows for BLE communication with their devices. It is called nRF Toolbox. It is intended to pair with the *ble_app_uart* example, published in the nRF52

SDK. Since this example code was used as the basis for the ble communication used in this project, it was addapted to work with this project.

The App contains different modules, intended for different examples, among them the Universal Asynchronous Receiver/Transmitter (UART) module (see 4.5). It is intended to be used with the ble_app_uart example. When opem it shows the ble services that are currently beeing addvertised and allows the user to connect to one of them 4.6. It then opens a window similar to phone messangers, were the keyboard can be used to tpye messages, that are sent to the connected devices.

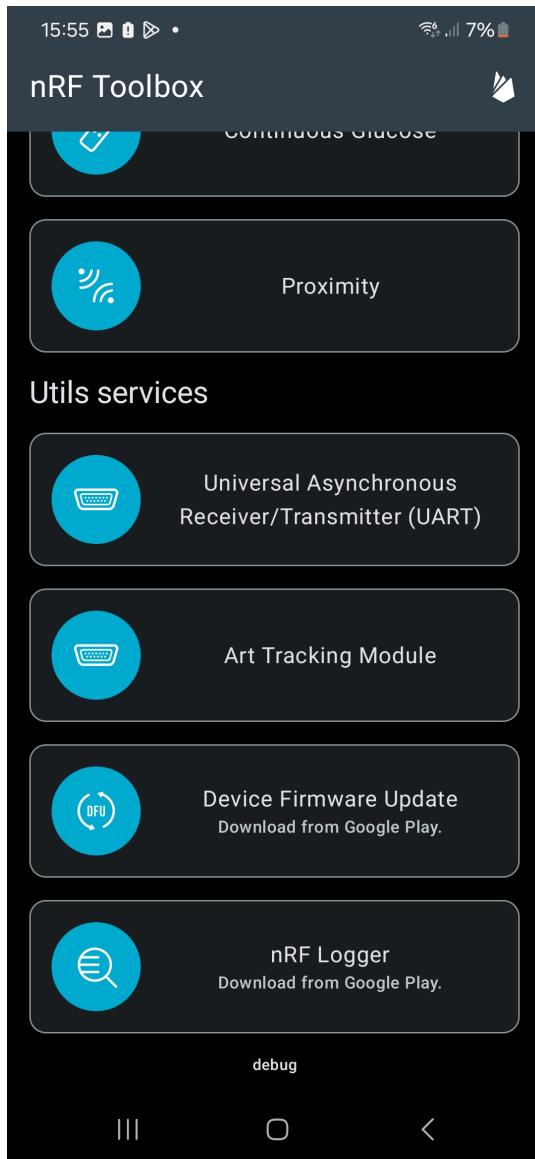


Figure 4.5: nRF Toolbox module menue, with the added Art Tracking Module

Since the development of an application was not the primary focus of this thesis, it was decided to take the nRF Toolbox app and add a new module for art-traking to it. The UART module searved as the basis for this new module, since it had a lot of usefull services already implemented. As with the UART module the art-tracking module opens up the

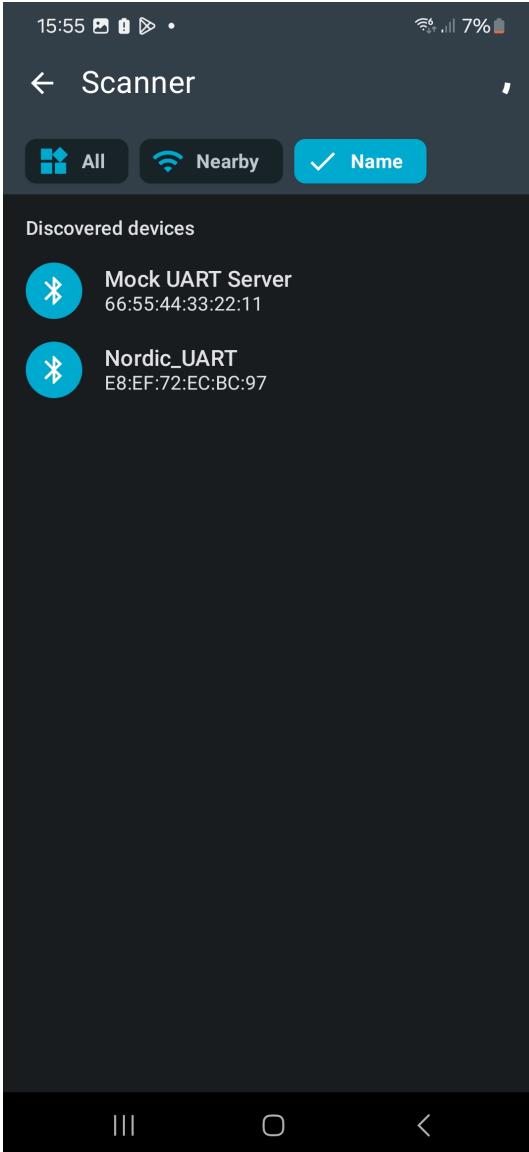


Figure 4.6: nRF Toolbox shows available devices to connect to

same connection page 4.6, that allows the user to select the art-tracking and connect to it.

Once connected, the observation screen is shown (figure 4.8). At the bottom seven parameters can be set: *time*, *max Temp*, *min Temp*, *max Hum*, *min Hum*, *max Angle*, *max Dist*. The parameters *max/min Temp/Hum* represent the expected range of humidity and temperature. Any measurement outside these parameter will be considered a dangerous value by the app. The tollerated difference in angle compared to the previous measurement is set by *max Angle*, larger differences are considered dangerous values. Distance measurement work analogously with *max Dist* in meters. The *time* set defines the time that passes enbetween measurements in seconds. The default is set to 350 seconds. This means that the time that passes between, for example, the temperature measurements on tag 2 are 350 seconds.

When the user presses the *Start Service* button, a services starts that poeriodically queries

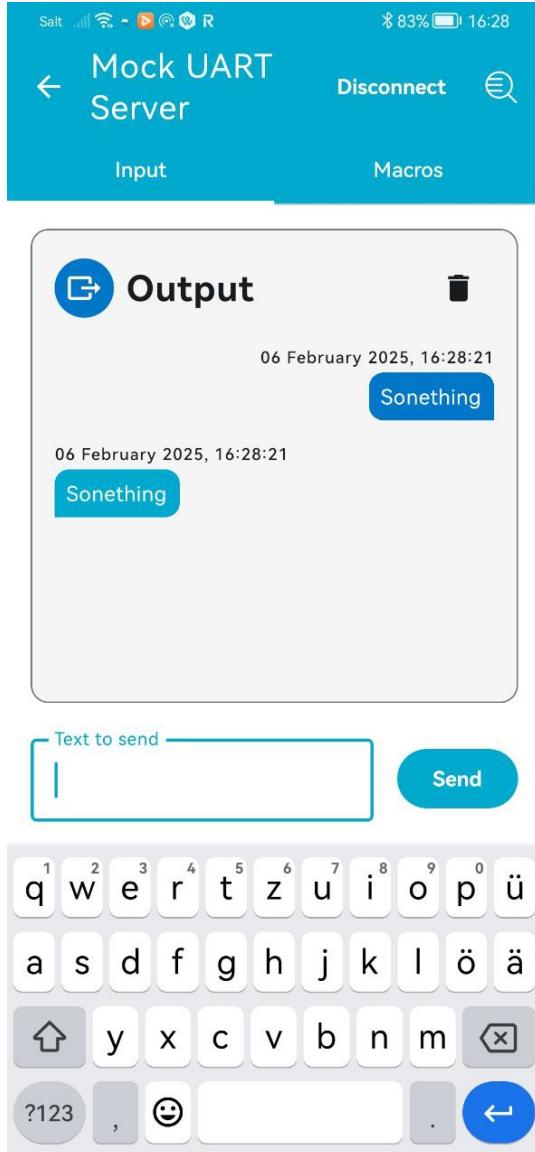


Figure 4.7: nRF Toolbox UART module screen

the tags for the Measurements. Figure 4.9 shows the measurement loop. Each sensor is assigned a character. T for temperature and humidity, G for gyro and D for distance. Each tag has a number, here from one to four since four tags were used in the experiments. The loop concatenates these two characters and sends the resulting query to the connected tag. Then the next tag-number is prepared for the next query. Once all tags have been queried for a sensor, the tag-number starts with the first again and the next sensor is queried. In between calls the app waits. The call time for distance-measurement is fixed at 80 seconds. Distance measurement takes longer than the other sensors, since for every device three measurements need to be conducted. Additionally the sensors that do not participate in a ranging session are sleeping for a quite generous amount of time, to ensure they don't distract the ranging session. 80 seconds has been chosen, since it allows enough time for all the ranging to happen, plus two repeats per sensor in case the ranging session fails. For the other sensors the waiting time in between queries is calculated from the remaining set time, after the ranging time is deducted.

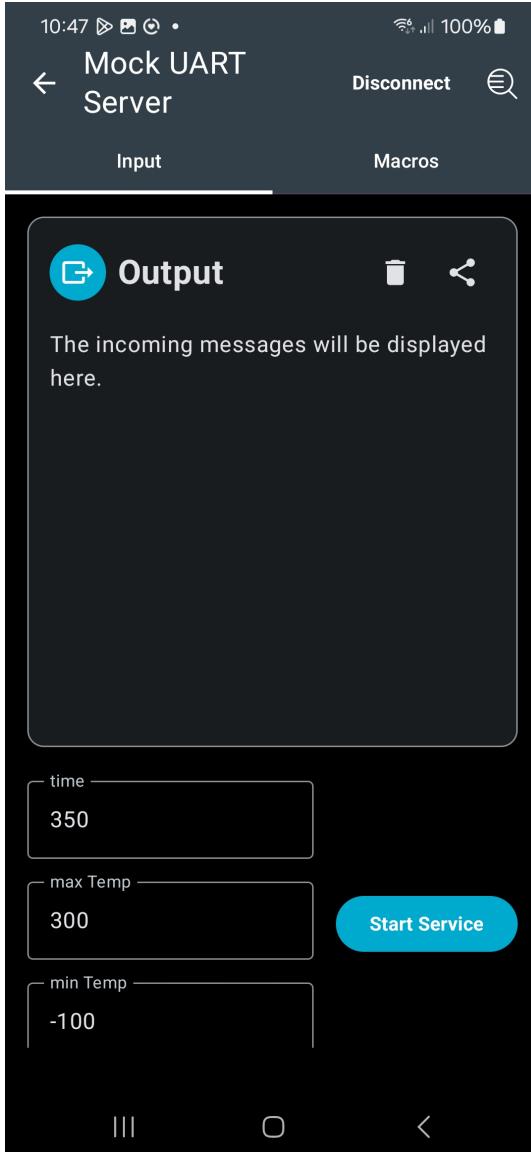


Figure 4.8: Art Tracking module oberservation screen before measurements

Once the process has started, the queries will appear in the chat window on the right side of the screen. The responses are on the right side, see figure 4.9. If the response is inside the set parameters, the message bubble will appear blue. If the measured value is considered a dangerous value, the text bubble will appear red (see figure 4.10). Since the message display is programmed in a asynchronous way, it can happen, that the answer to a query appears before the querry itself, if the querried tag is the same as the connected tag. The service can be stopped by pressing the *start service* button again or by exiting this screen in any way.

The query-answers are appended to a file that is safed in the app-storage. The information appendedd consists of: the queried tag, the returned values, a timestamp and if the value was unproblematic. This functionality is intended for experimental evaluation. In a real word application, this data should be periodically backed up on a server in a compressed manner. When pressing the share-button on the top right of the message-box 4.10. It

```

1  private val sensors = listOf("T", "G", "D")
2  private val devices = listOf("1", "2", "3", "4")
3  private var measurement_type = 0
4  private var tag = 0
5  private var timeBetweenCals: Long = 3750
6
7  private val runnable = object : Runnable {
8      override fun run() {
9          if (tag >= list2.size) {
10              tag = 0
11              measurement_type += 1
12          }
13          if (measurement_type >= list1.size) {
14              measurement_type = 0
15          }
16          val textToSend = "${list1[measurement_type]}${list2[tag]}"
17          artRepository.sendText(textToSend, MacroEol.LF)
18          tag += 1
19          if(list1[measurement_type] == "D"){
20              handler.postDelayed(this, 80000)
21          } else {
22              handler.postDelayed(this, timeBetweenCals)
23          }
24      }
25  }

```

Figure 4.9: Section from the ArtMetricService.kt, main measurement loop

will open the Android native share functionality, to share the file over mail, an installed messenger, save it to onedrive or send it over Bluetooth. In this project all files were sent with email. Pressing the trashcan next to it will delete the chat and empty the file. This allows the user to distinguish between different testing session.

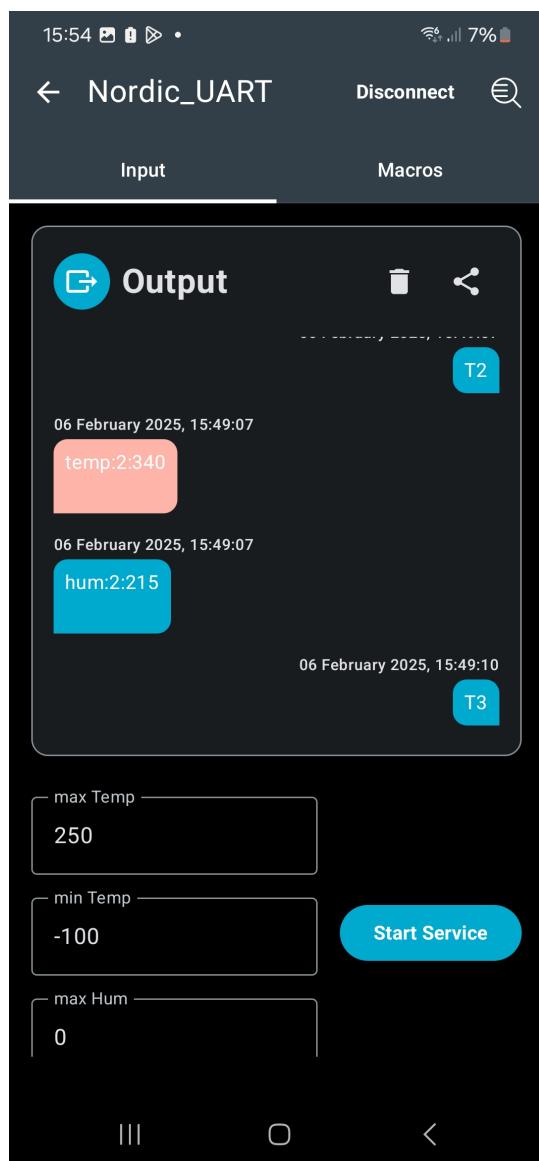


Figure 4.10: Art Tracking module, queries and responses

Chapter 5

Evaluation

Five experiments were performed to validate the functionality of the tags. The first is non specific and ment to test the setup in a stable environment. Experiments two to four are intended to test the detection of unwanted circumstances. Experiment five is tests the system in a real-world environment. The experiment results were stored on the phone and then exported using email. The analysis of the data and creation of graphs was then performed using a Jupiter Notebook, using Pandas and Pyplot for datamanagment and the creation of graphs.

For all experiments the query-frequency was set to 330s. The measurements queries are spread across this timeframe. Each experiment lasted between 40 minutes and one hour. All experiments were performed two to three times. In each section only the data-set from the first experiment run is presented fully. The other experiments will be mentioned only, if they have differing data or to confirm an unexpected datapoint.

The tags used were programmed as described in Chapter 4. The same four tags were used for all experiments. They will be referred to as Tag-1, Tag-2, Tag-3 and Tag-4.

5.1 Experiment 1: Static

The four tags where placed on the corners of a 80 cm by 50 cm rectangle on a wooden table. Figure 5.1 shows a schematic view of the setup. Each tag was turned on sequentially and given enough time to establish the network. The phone then was connected to Tag-4. The parameters in the app were left unchanged. The default parameters are large enough, that no measurement should be able to trigger a warning. Orientational reading was used for the output of the gyroscope. The setup was then left untouched for 35 min. The goal of this experiment was, to gauge by how much the measurements can vary in a static environment.

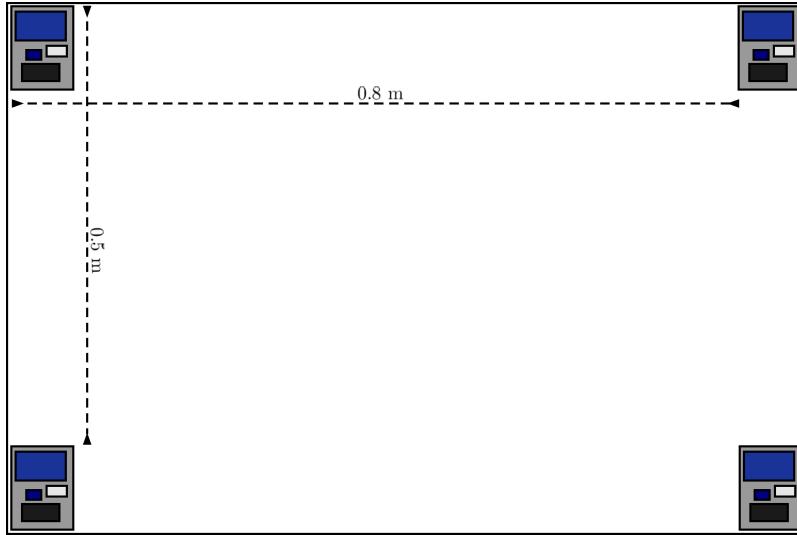


Figure 5.1: Schema of the setup of experiment 1.

5.1.1 Results

In experiment one, all measurements are expected to be unchanging. Table 5.1 shows the mean values for temperature, humidity and angle during the experiment by tag. Figures 5.2, 5.3, 5.4, 5.5 shows the change of these values over time.

Table 5.1: Mean and Variances for Temperature and Humidity Data by Tag during experiment 1.

Tag	Temp	Hum
Tag-1	22.06	32.56
Tag-2	21.90	33.93
Tag-3	22.06	32.94
Tag-4	21.87	32.80

Mean

Tag	Temp	Hum
Tag-1	0.02	0.03
Tag-2	0.05	0.04
Tag-3	0.03	0.06
Tag-4	0.03	0.05

Variance

Figure 5.2 shows the recorded temperature during experiment 1. The tags are color coded and use different line-styles. To make it easier to distinguish the lines, the Y axis only displays the relevant section, rather than starting at 0°. The time at the bottom represents the timestamp at which the measurement arrived at the phone. All four tags have a mean temperature between 21.8 and 22.1 °C. The variance are also small, tag two having the highest one with 0.05 °C variance. The graph shows that all tags have a rising temperature. The increase is quite small with tag two having the biggest increase of 0.5 °C over 20 minutes. When the experiment was repeated,, the means stayed similar between the tags and the variance became only smaller. The trend in temperature changed from upwards to downwards, when the experiment was repeated.

Figure 5.3 shows the change of humidity over time. Again, the relevant section of the y-axis is shown, rather than the full 0% to 100%, to increase readability. The humidity of all sensor was similar as well. The highest humidity was recorded by Tag-2 with 33.93% .

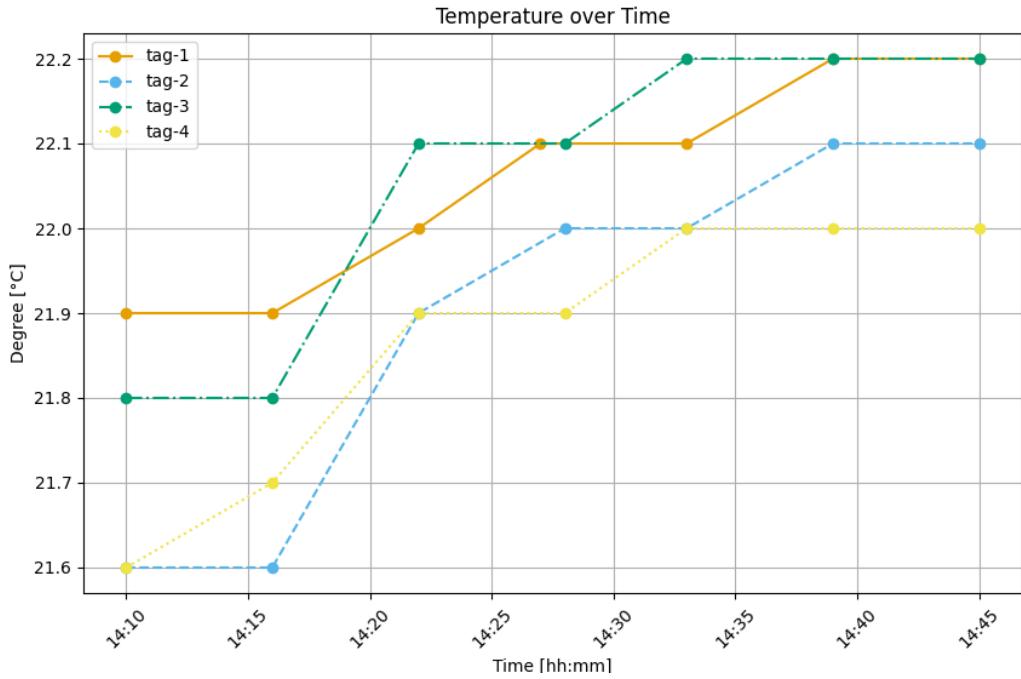


Figure 5.2: Experiment 1, temperature over time.

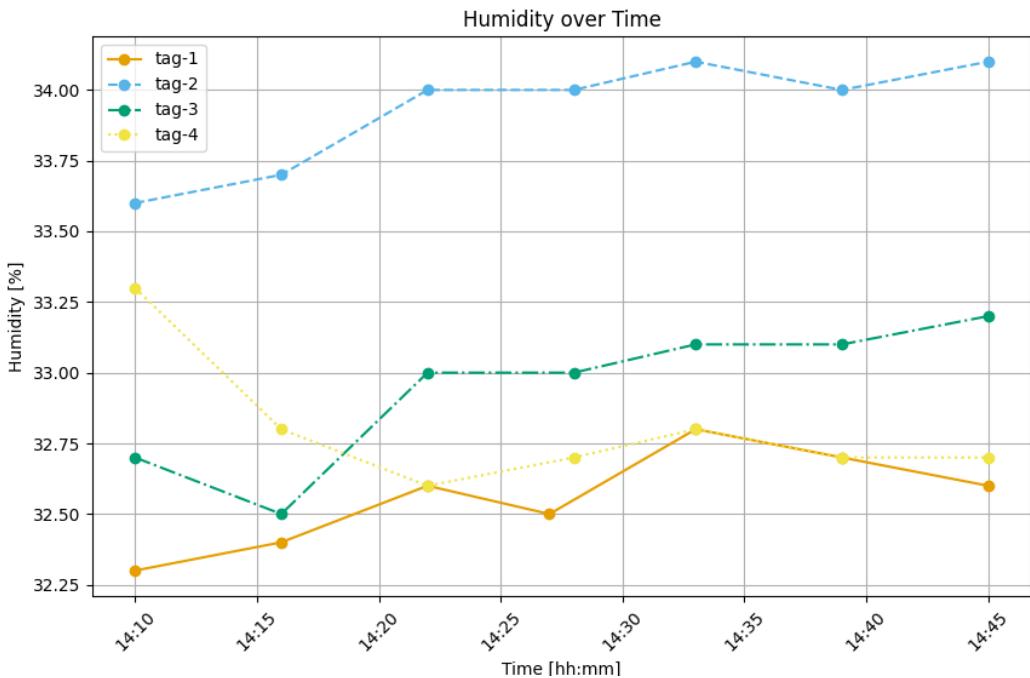


Figure 5.3: Experiment 1, humidity over time.

The lowest was recorded by Tag-1 with 32.56%. The variance is small, with Tag-3 having the biggest variance with 0.06% pt. During the first experiment, humidity increased by a small amount. When the experiment was repeated, the humidity dropped during the experiment.

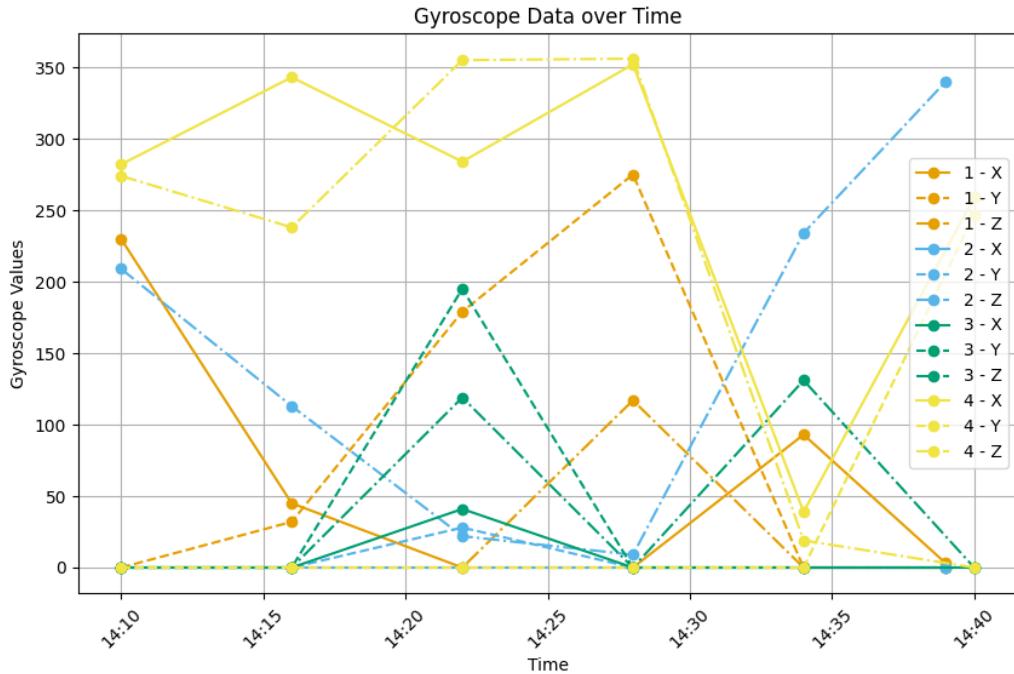


Figure 5.4: Registered value of the gyroscope over time during experiment 1.

Since all tags were stationary during the experiment, the gyro sensor was expected to be unchanging. This is not what occurred. The graph 5.4 shows the values of the Gyroscope during experiment 1. Orientational read was used, so the values should correspond to the angle around the given axis. Each tag is assigned a color, and all angle measurements are shown in that color. All X-axis measurement are displayed using a filled line. Axis-Y uses dotted lines. Point-dotted lines represent the angles around axis-Z. Looking at the graph 5.4 it is clear, that the measurement shows a wide range of angles for each tag and axis. The angle of axis-X, Tag-1 (filled orange line) for example, jumps from a value of 230° to 45° , 0° , then stays at 0° for one measurement, goes up to 93° and drops down again to 3° . As can be seen with this example is, that the measurements also don't fall a clear trajectory. Tag-1 switches between rising and falling. The only exception is tag 2 around the x axis, which stays at 0 for the whole measurement duration.

Since angle measurements fall into modular arithmetic, it "wraps around" at 360° , means can only meaningfully be taken if the angles are in a small range. Since this is not the case for most tags, the only mean that is meaningful is tag 2 axis-x, which has a mean of 0 and a variance 0.

Table 5.2, shows the mean, expected value and variance of the measured distances. The tag listed in the row is the queried tag that initiates the distance measurement, and the row corresponding to the responding tag. By looking to the measurements diagonally opposed to each other, one can see that the measured distances are the same, independent of who initiated the measurement, up to a range of two centimeters. The measurements from Tag-3 to Tag-1 is the highest, with 0.024m. All other variances are negligibly small, being below 0.005m. This shows that the measured distances are constant and stable, except for the measurement from Tag-3 to Tag-1. The distances measured do not correspond

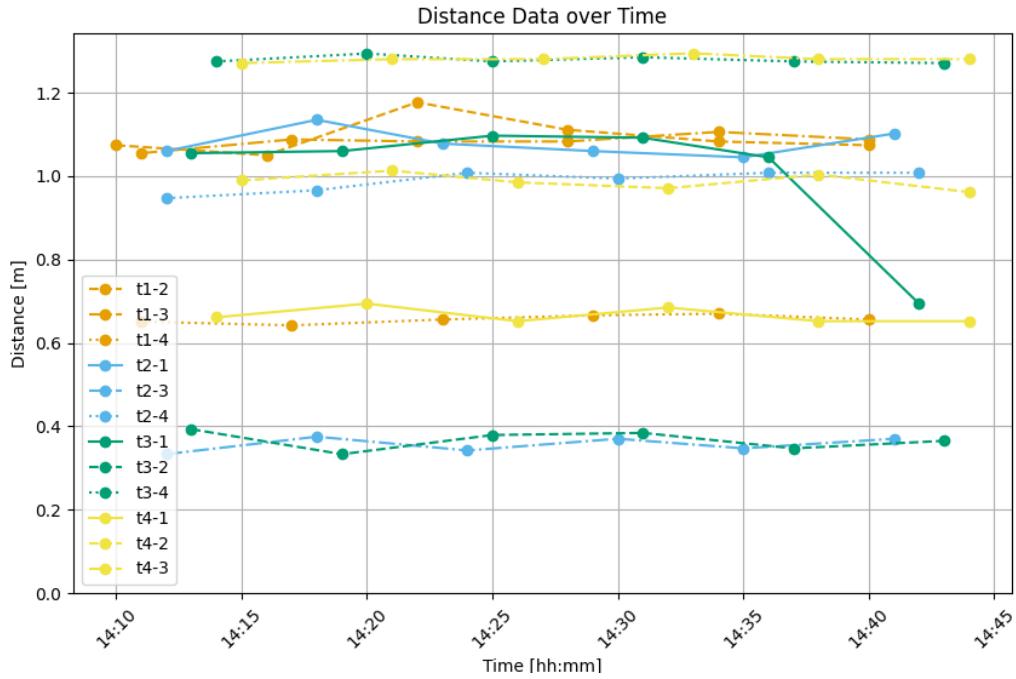


Figure 5.5: Experiment 1, distance over time.

to the actual distances the tags had to each other, also seen in table ???. The measured distances can be as far of as 0.5 meters. The two larger distances, 0.8 and 0.94 meters, correspond to the two larger measured values for each tag, while the smallest measured value always corresponds to the smallest distance, 0.5 meters. The two larger values are not always ordered correctly, 0.94 meters sometimes beeing measured smaller then 0.8 meters. In repeated experiments, all these facts stayed true.

Figure 5.5 shows the measured distance over time. A label i-j informe that tag-i initiated the measurement, and the distance between tag-i and tag-j was measured. All measurements initiated by Tag-1 are orange. The measurments of Tag-2 are blue, Tag-3 green and Tag-4 yellow. The second tag that is involved in the measurement is signified by the line. Measurements to Tag-1 use filled lines. Measurements to Tag-2 use dashed lines, Tag-3 uses dashed and dotted lines and Tag-4 uses dotted lines.

All lines except for 3-1 are horizontal and show little variance. Measurement 3-1 is also stable until the last measurement, where a datapoint that is 0.35m lower than all previously recorded data is measured. One can also see that not all measurements start at the same time. The first measurement of distance 1-2 was registered at 14.10, while the first measurement of 4-3 was recorded at 14.15. Each distance was measured seven times and with equidistance measurement times.

The measurement pairs i-j and j-i report the same distance, but with different tags initiating the measurement. To better compare these pairs, figure 5.6 shows six subplots of figure 5.5 containing only each of these pairs. The graphs show that the measurement pairs are consistently close together. One outlier happens when Tag-3 measures the distance to Tag-1 at very end of the measurements. The measured value drops 0.35m bellow the previous mean of 1.30m. When repeating this experiment and during other

Table 5.2: Mean, expected values and variance of distant measurements, experiment 1.

	Tag-1	Tag-2	Tag-3	Tag-4		Tag-1	Tag-2	Tag-3	Tag-4
Tag-1	0.0	1.094	1.084	0.657	Tag-1	0.0	0.8	0.94	0.5
Tag-2	1.080	0.0	0.356	0.989	Tag-2	0.8	0.0	0.5	0.94
Tag-3	1.007	0.367	0.0	1.279	Tag-3	0.94	0.5	0.0	0.8
Tag-4	0.666	0.987	1.281	0.0	Tag-4	0.5	0.94	0.8	0.0

Mean

Expected values

	Tag-1	Tag-2	Tag-3	Tag-4
Tag-1	0.0	0.002	0.000	0.000
Tag-2	0.001	0.0	0.000	0.001
Tag-3	0.024	0.001	0.0	0.000
Tag-4	0.000	0.000	0.000	0.0

Variance

experiments, these outliers happened again, a bit less frequently then twice per hour. The outliers always affected a measurement involving tag 1.

Since the pairs i-j and j-i report the same data and this fact is consistent in the measurements, they can be combined into one graph. Figure 5.7 shows the distances over time for all combined pairs i-j and j-i, called $i==j$. Graphs like this will be called combined graphs in this report. Since initiating and receiving tag can no longer be distinguished, the line colors and types have no assigned meaning. The two pairs 2=3 and 1=4 corresponding to the two low distances of 0.5m can be seen at the bottom. The pairs 1=3 and 2=4 that represent the highest distance of 0.94m do not separate and are mixed together with 1=2 and 3=4.

Table 5.3 shows the means and variances of the combined tag pairs. Since the measurements of $i=j$ are the same as $j=i$, only the upper triangle of the distance-matrix is needed. The table shows, that the variances are very low for all pairs, except 1=3.

Table 5.3: Statistics of the combined distance measurements between tags for experiment 1

	Tag-2	Tag-3	Tag-4		Tag-2	Tag-3	Tag-4
Tag-1	0.112	0.265	0.177	Tag-1	0.001	0.013	0.000
Tag-2		0.368	0.824	Tag-2		0.000	0.000
Tag-3			0.385	Tag-3			0.000

5.1.2 Conclusion

The temperature measurement seem to be working as excpected. All four tags show the same temperature, within a small margin of error. Variance is low, showing a consistent

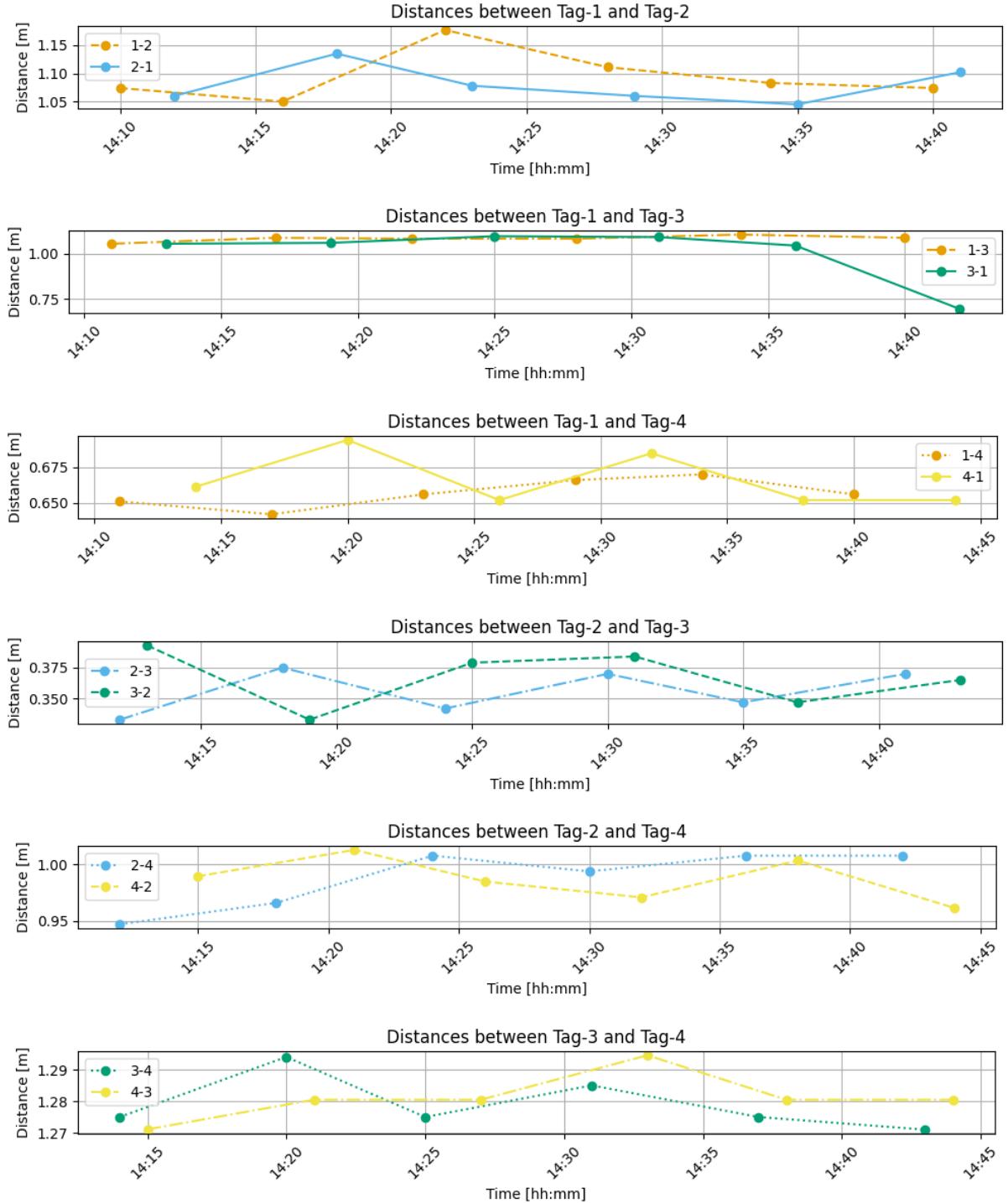


Figure 5.6: Experiment 1, distance over time, for all pairs i-j.

temperature measurement.

Two possible explanations were found for the increase in temperature during the experiment. One possible explanation is given by the fact, that this was the first experiment performed in the day, and the room temperature was slightly increasing because of the presence of a person, that was not present before. An alternative explanation is, that the microprocessors produced heat that was detected. The fact that the temperature decreased

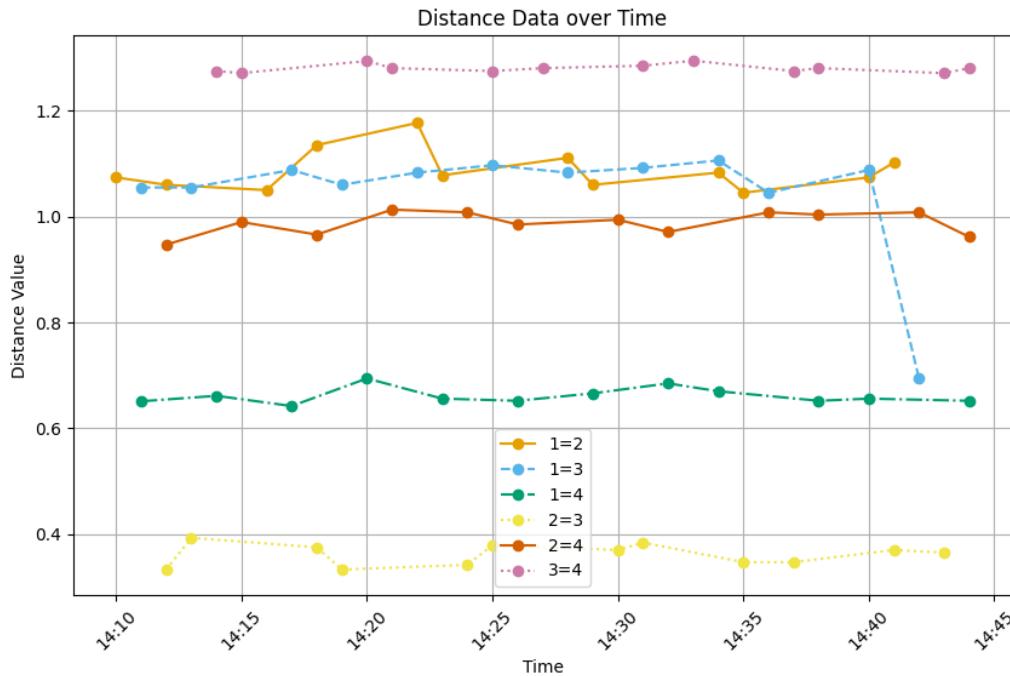


Figure 5.7: Experiment 1, distance over time, for combined pairs $i=j$.

during subsequent experiments, favors explanation one, since there would be no reason for the microcontrollers to stop producing heat. The decrease itself can be explained, since during setup, the person performing the experiment was close to the sensor, while during the experiment the person stayed in a different part of the room. The decrease in temperature was smaller, this difference in closeness to body heat could explain the difference.

The humidity sensor similarly produced satisfactory results. All four tags presented the same humidity, only with small margins of error. The variance is again satisfactory, since it is very small, being below 0.05. The slight increase in humidity can again be explained by this being the first experiment of the day, and the person performing the experiment having wet hair from the rain. This again weakens the microcontroller heat theory, since rising temperature without adding moisture would only decrease the humidity.

The decrease in humidity in subsequent experiments lacks a clear explanation. It is a very weak trend, so factors global factors could explain the difference. Changing weather conditions could account for the difference. Another proposed explanation arises from the setup of the system. During setup, each tag was touched repeatedly to put them into position. The person performing the experiment tends to have clammy hands, that feasibly could lead the sensors to detect additional humidity at the beginning of the experiment. Without additional data, no one explanation can be favored over the other. Since the decrease in temperature was small, this is not considered an issue for this system.

The Gyroscopic sensor data does not produce any meaningful result. The measured orientation of the tags varied widely, while the physical tags stood still. A possible explanation of this is, that the gyroscope used in the implementation has consistent biases. Since the angular velocity is evaluated often and then added to the current angle, small errors would accumulate over time. The time between measurements was 5 minutes and

30 seconds. A bias of only $\frac{12}{11}^\circ$ would correspond to an accumulated error of 360° over this timespan. Since rotational position is inherently circular, wrapping around at 360° , unless there was no variance next to the bias, the values would end up randomly scattered over the range of $[0^\circ, 360^\circ]$. The MPU6050 outputs only integers, so any bias at all would have this effect.

Be bias hypotheses is additionally strengthened, by the existence of Tag-2 axis X, that stays at 0 over the course of the measurement. While this could indicate a faulty sensor, during later experiments using rotational velocity readings, see section 5.5, Tag-2 axis X did produce meaningful results. While this doesn't disprove, that Tag-2 axis X was faulty during this experiment, it makes it more reasonable to assume, that it has a bias of 0.

A possible reason for the bias in angular velocity was considered, in the rotation of the earth. After some consideration, this thesis was dropped, since the angular velocity introduced by the earth would account for no more than $\frac{1}{240}^\circ$ around the X or Y axis, if standing on the equator, where the effect is strongest.

The bias explanation seems to be a reasonable and explains the measured results. As a consequence, the orientational read has to be considered useless.

The distance measurements have mixed results. The fact that the tag pairs produce the same results is good. Double-sided two-way ranging is used, so during each ranging session both tags conduct single-sided two-way ranging and the results are combined. It is therefore expected, that the device that initiates the ranging does not matter.

The fact that ranging sessions involving Tag-1 occasionally produce inconsistent results can not directly be explained. Different locations were used for the experiments and the tags did not always have the same position. This means that an explanation involving multi-path effect based on position can be rejected. The possible explanation involves a fault on the nRF52840 microcontroller or the DMW3000 shield. Since the final calculation relies on the timestamps recorded during the ranging, a possible explanation would be, that the clock of the nRF52840 sometimes faults, or that there is an issue with the clock line of the SPI connection.

The fact that the resulting distances are wrong is troublesome. The proposed design that would calculate the position by solving a quadratic program relies on somewhat accurate distance measurements. The likely reason for the distance measurements producing wrong results is the simplified calibration, that was used for the d_{rx} , d_{tx} values, see Section 4.1.4. The fact that the distances still sort themselves into high and low values correctly indicates, that some calibration has worked, but it is not granular enough to work for small distances. The distance measurements can currently not be used to build a model of the tag positions. If they can be used to detect movement can not be determined by the static experiment and requires the introduction of movement, see Experiment 4 ??

5.2 Experiment 2: Temperature

The four tags were placed in the same 80 cm by 50 cm rectangle as in experiment one. One tag placed on an elevated surface, 4 cm above the table, next to the tag on the table, seven candles were placed (see figure 5.9). Next to the Tag-2 thermometers detectors

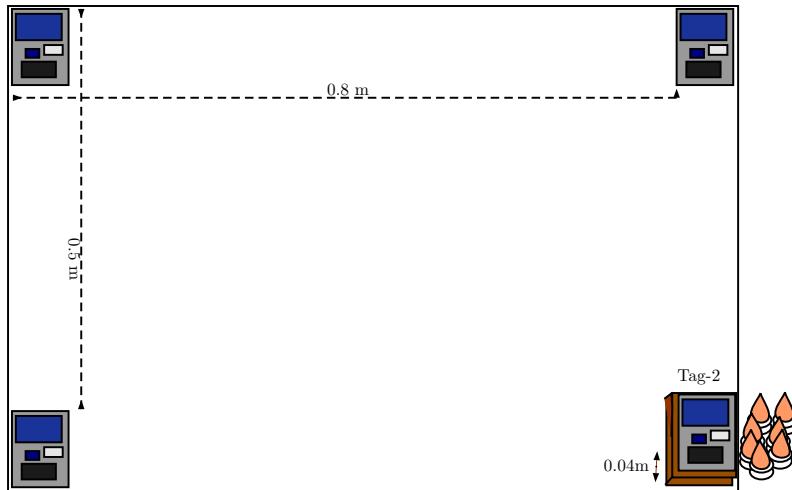


Figure 5.8: Schema of the setup of experiment 2.

were placed. Figure 5.8 shows a schematic view of the setup. Each tag was turned on sequentially and given enough time to establish the network. The phone then was connected to one tag. The max Temperature parameter in the app was changed to 35°C. After 20 minutes the candles were lit. The experiment was then left alone for another 30 minutes. The independent thermometers were filmed during the process, to allow for later review and comparison. The goal of experiment 2 was to test the temperature detection capabilities of the system.



Figure 5.9: Photo of elevated Tag-2, candles and thermoeter used in experiment 2.

5.2.1 Results

Experiment two introduced heat-sources to the system. Since the main setup was the same as experiment 1 5.1.1, many of the findings are the same. In this section, only differences in results are discussed. If a metric is not mentioned, one can assume it behaved the same as for experiment 1 (see section 5.1.1).

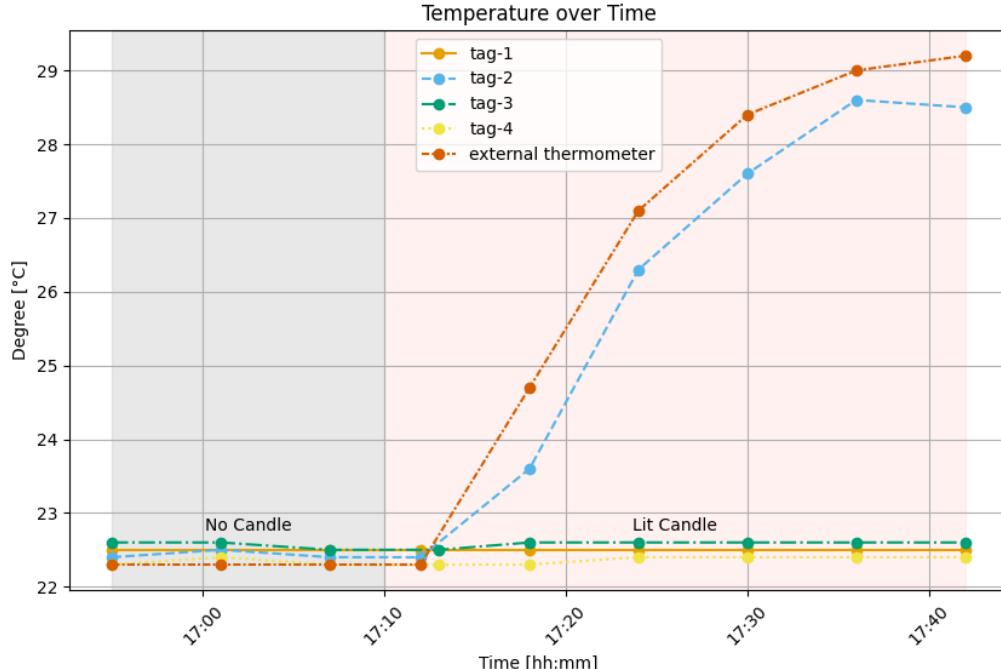


Figure 5.10: Experiment 3, temperature over time, mith external measurement added.

The progression of the external thermometer and the internal temperature sensor can be seen in figure ???. The candles, that functioned as the heat source, were lit at 15.10. The section of time before the candle was lit has a gray background. After the candle was lit, the background becomes red. The measurements from te external thermometer are shown with a red dashed and dotted line. They were extracted manualy from the video. The datapoints correspond to the datapoints when Tag-2 measured.

Before the candle was lit, Tag-1, Tag-2, Tag-3 and Tag-4 recorded mean temperatures of 22.5°C, 22.4 °C, 22.6 °C and 22.3 °C respecivly. The variances were all bellow 0.01°. Once the candle was lit, Tag-1, Tag-3 and Tag-4 continued with similar temperature, havin mean temperatures of 22.5°C, 22.6°C and 22.4°C over the whole duration, with variance remaining under 0.01°C.

After the canle was lit, Tag-2 started to deviate from the other tags. During the next measurement of tag 2, at 15.12, both the external thermometer and the temperature sensor on tag 2 had not yet registered any change, remaing at 22.4 °C for the tag and 22.3 °C for the external thermometer. The recording showed the extrenal thermometer start rising 1 minutes later, at 15.13. During the next measurement at 15.18, the temperature-sensor registered a slightly increased temperature of 23.6 °C, while the external thermometer registered 24.7 °C. During the next measurement at 17.24 the tag reported 26.3 °C while

the thermometer showed 27.1 °C. The measured temperature of the external thermometer keeps climbing faster than the temperature sensor of Tag-2, until the end of the experiment, as seen in Figure 5.10. The difference in measured temperature between Tag-2 and the external thermometer neither exceeds 1 °C and gets smaller towards the end of the experiment, ending with a difference of 0.7 °C.

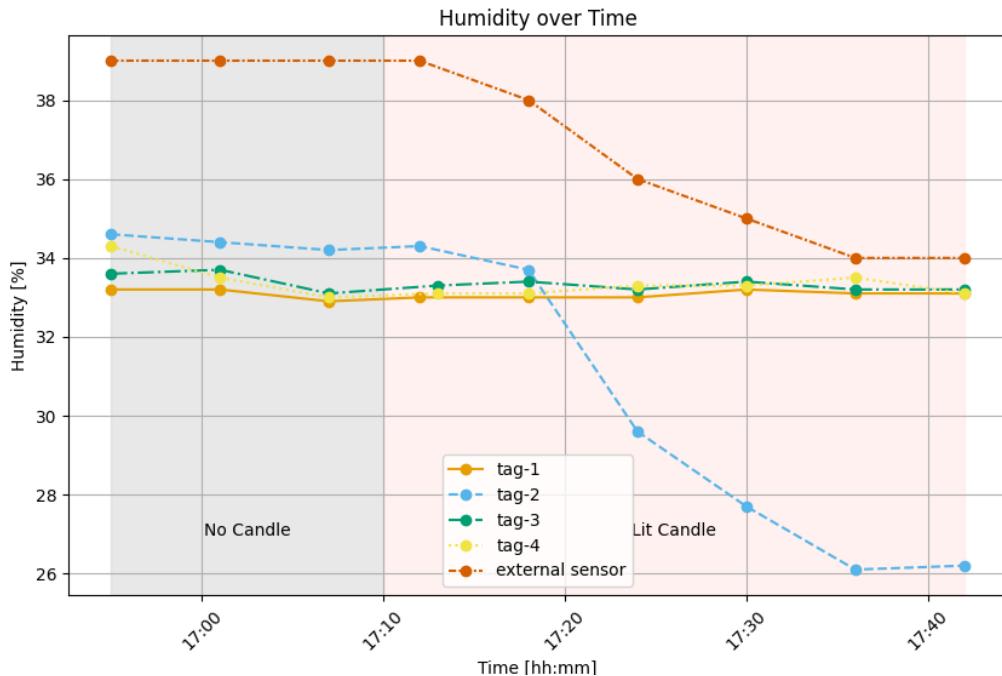


Figure 5.11: Experiment 3, humidity over time, with external measurement added.

Experiment 3 was intended to test the temperature and not the humidity. Luckily, the external thermometer also included a humidity sensor, that could retroactively be used for evaluation. Figure 5.11 shows the humidity over time, the gray and red sections again representing the time before and after the candle was lit. The external humidity sensor was added to the graph and is shown with a red dashed and dotted line. Since the external humidity sensor was initially not intended to be used, it is not particularly precise and does not display any digits after the decimal point.

Its values were again manually extracted from the video at the same points the times Tag-2 measured the humidity. Tag-1, Tag-3 and Tag-4 again show a constant measurement during the experiment, having mean humidities of 33.0% , 33.3% and 33.4% . Tag 4 has the highest variance of the group, with 0.2% , the others having variances below 0.1 %.

Tag-2 had a higher mean humidity of 34.4 before the candles were lit, with a variance of 0.04. The first measurement after the candle was lit was still 34.3 % . Afterwards the measurements started dropping, first with a small decrease to 33.7 % , followed by a large drop to 29.6 % , then 22.7 % and finally plateauing at 26.2 % .

The humidity sensor consistently shows a much higher humidity than the one on the tag. When the experiment starts at 15.10, the external sensor shows a humidity of 39 % . It stays on this value until the candle is lit. The first measurement after the candle is lit, at 17.12 still has a humidity of 39% . The external sensor then first notes a small decline of 1% pt, followed by a larger decline of 2% pt to 36 % and then decends with 1% pt at a

time until it plateaus at 34% .

The humidity registered by Tag-2 and the external sensor forms a similar line. This two lines have a similar trajectory, but are not parallel. While the difference in registered humidity originally is around 4.6% pts, when both plateau, the difference has risen to 7.8% pts. The variance in the difference of the Tag-2 sensor and the external sensor is 2.1 % pts over the whole measurement period.

5.2.2 Conclusion

The fact that the three sensors that are far away from the candle don't show any sign of temperature increase was expected. Since warm air rises, and the tags were spread more than half a meter apart, it was not expected, that the heat from the candles would reach the tags. Even if hot air would not rise, the energy added to the system would be added with an efficiency of $O(d^{\frac{1}{3}})$, where d is the distance.

The tag that is close to the candle does notice the candle with a similar speed as the external thermometer. The fact that it takes a while for both the external sensor as well as the sensor of Tag-2 to register the heat, has two explanations. The first explanation is, that the external thermometer and the DHT22 sensor are both not high precision instruments and have a natural inertia. The second explanation is, that it takes a moment for the candles to fully burn and start heating up the air. most likely, a combination of both factors is responsible for the delayed start.

The observation that the external thermometer registers a higher heat than the internal sensor has two possible explanations. It could be a difference in registered value, due to different sensor reporting different results. It could also be, that the external sensor was actually hotter than the internal sensor. The internal sensor was mounted on a piece of cardboard and thus shielded a bit from the heat. The external sensor was also placed on the cardboard, but more directly exposed to the heat, since it was placed closer to the edge of the cardboard piece. Since initially both sensor have very similar values, the second explanation is more likely true.

The fact that the humidity dropped when the candles were lit should have been expected. The % humidity represents amount of water in the air, as a percentage of the maximal capacity of air. The capacity of air to carry water rises with temperature. So when the temperature rises, but no additional humidity is added, the percentage drops. This can clearly be seen happening in this experiment to Tag-2.

Since the temperature around Tag-1, Tag-3 and Tag-4 does not rise, neither does the humidity fall. This was verified by the humidity results of this experiment for those tags. Tag-2 starts with a slightly increased humidity. This is a further pointer to the theory, that the humidity of the experimenter during setup can be registered, since it took the experimenter a few minutes to set up everything around Tag-2 for experiment 2.

The difference in humidity between the tags and the external sensor lacks a clear explanation. Since the humidity function is not the main purpose of the external thermometer, it is possible that it was implemented poorly, thus leading to the difference. Further research is required to analyze the precision of the DHT22 sensors. Nevertheless, the sensor on Tag-2 shows clearly a happening phenomenon, indicating that the general implementation and setup is sound.

5.3 Experiment 3: Gyroscope

Again all four tags were placed on a 80 cm by 50 cm rectangle. Each tag was turned on sequentially and given enough time to establish the network. The phone then was connected to one tag. After 20 minutes one tag was turned by 90° clockwise. The experiment then ran for another 30 minutes. The goal of experiment number 4 was to test the detection of unwanted rotations.

Experiment 3 was performed in two differing manners. The orientational read was originally the only implementation for the gyroscope. After experiments one to four were evaluated, the lack of useful results from the gyroscope readings prompted a redesign of the sensor. This resulted in the development and implementation of the angular velocity read. Experiment 3 was repeated with the angular velocity read of the gyro.

For the orientational read, the maximal allowed angular difference was set to 30°. For the angular velocity read, the maximal allowed angular velocity was set to $100 \frac{\text{deg}}{\text{s}}$. The results for the orientational and the angular velocity read will be presented separately. The conclusion will talk about them both.

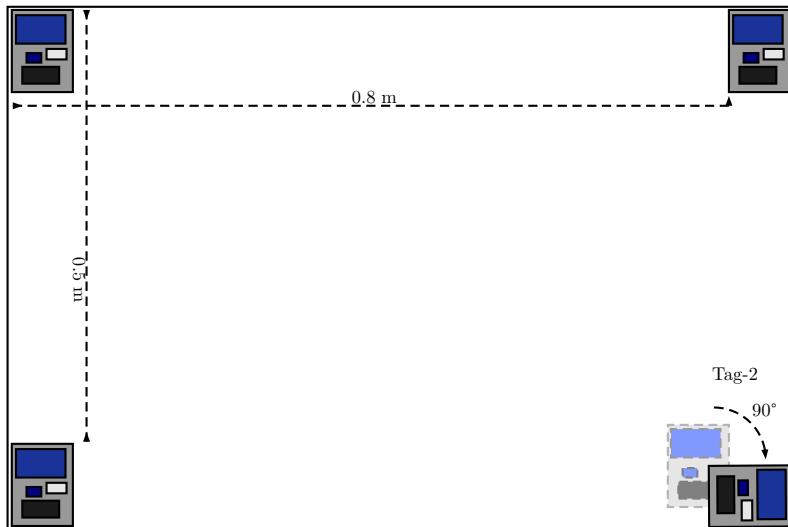


Figure 5.12: Schema of the setup of experiment 3.

5.3.1 Results orientational read

Experiment 3 was intended to check the functionality of the gyroscope. Temperature and humidity behaviour were the same as in the static experiment 5.1.1. As already seen during the evaluation of experiment 1, the gyroscope does not work as planned.

Figure 5.13 shows the values of the gyro over time. Tag 1 was rotated by 90° around the Z axis. The gray section marks the part of the experiment before the turn, while the red shows the results after the turn.

Tag-3 has a mean of 0 and 0 variance of 0 during the whole experiment for axes X and Y. Tag-4 has one measurement that has 0 mean and variance as well, at axes Y. Their is

no discernable change in the output of the gyro during or after this process in any of these measurements.

Some other orientation are also manly zero during this experiment. The orientation of the Z-Axis of Tag-3 is zero for six out of the eight performed measurements, only spiking once at the beginning for two reads. The other axes of Tag-1 have values 100° and 290° at the beginning and then stay zero for the rest of the experiment. The Y and Z axis of Tag-2 are also zero for almost all measurements except for one measurement at 22:11, where they registered an orientation of 235° and 5° respectively. This is especially unexpected, since Tag-2 was the one who was turned around axis Z.

Axis z of Tag-1 forms zig-zag line between values from 25° to 100 ° and 200 ° to 300 °. Tag-4 axis X and Z and Tag-2 Axis X follow no decernable pattern.

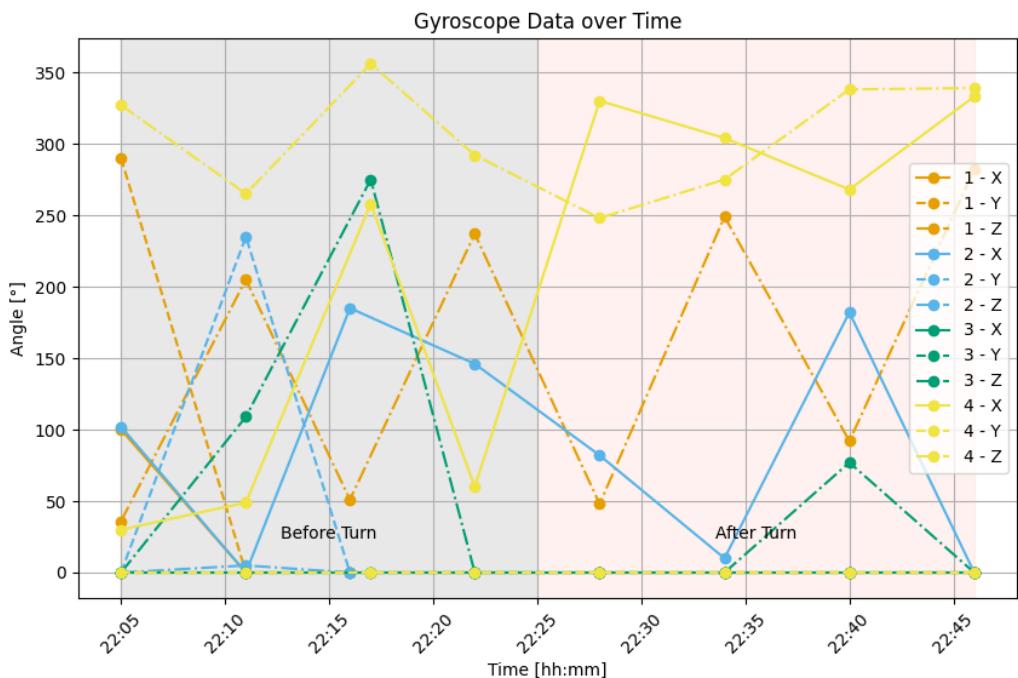


Figure 5.13: Experiment 4, gyroscope over time.

Figure 5.14 shows the distances over time for pairs, as discussed in section 5.1.1. Before the event, all measured distances are stable, except for one outlier in the distance 1=2. As in experiment 1 5.1.1 the distances are not equivalent with the physical distances in the experiment.

After the tag is turned at 22.26, all measurements involving Tag-2 change and then remain at the new distance. Distance 1=2 decreases from 0.71 m to 0.42, after a short jump to 1.4m. Distance 2=3 is on 0.92 before the measurement and 1.11m afterwards. It is also involved in a measurement during the turn, measuring 0.65m. Distance 2=4 is originally at 0.65m before the turn and ends up at 1.10m afterwards. The other pairs, 1=3, 1=4 and 3=4 don't change values significantly during this time, having variances of 0.001, 0.004 and 0.000 respectifly.

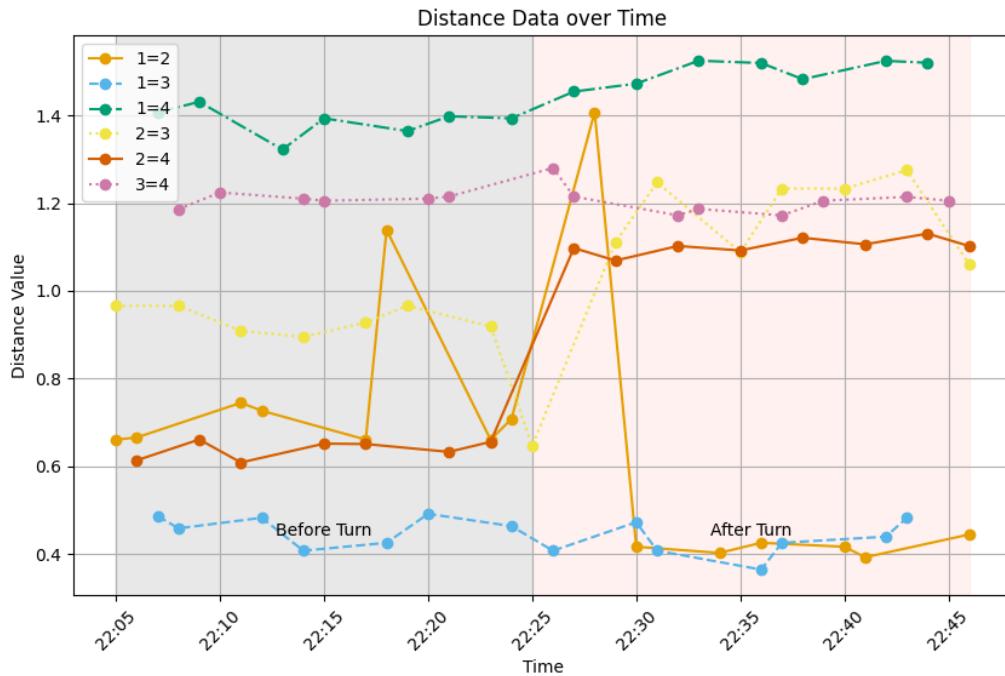


Figure 5.14: Experiment 3, gyroscope over time.

5.3.2 Results angular velocity read

As with the orientational read, this experiment had no results that differed from the static experiment, when analyzing temperature and humidity. The experiment was started at 10:01 and was terminated 10.40. Tag 2 was turned at 10.18 by 90°around the Y axis by hand. Figure 5.15 shows the angular velocity of all tags during the experiment. The angular velocity of tag i around axis v will be labeled as a_i^v . All axes of Tag-2 are shown in blue, with the angular velocity around the X axis, a_2^X , beeing a filled line, around the Y-Axis a dashed line and around the Z axis beeing dashed and dotted. The Y axis of figure5.15 is displayed using a log-scale, to better show the low values. The time before the utrn has a gray backgroud, while the time after the turn is displayed with a red background.

Tag-2 has constant angular velocity for all three measurements before the turn. a_2^X is $12\frac{\circ}{s}$, a_2^Y is between $14\frac{\circ}{s}$ and $17\frac{\circ}{s}$ and a_2^Z is a constant $20\frac{\circ}{s}$ before the turn. The first measurement after the turn reports angular velocities of $445\frac{\circ}{s}$, $6322\frac{\circ}{s}$ and $716\frac{\circ}{s}$ for axis X,Y and Z of Tag-2. Afterwards the values return back to their original values. a_2^X is $13\frac{\circ}{s}$, a_2^Y is $14\frac{\circ}{s}$ and a_2^Z $20\frac{\circ}{s}$, all constant measurements, after the turn.

Tag-1, Tag-3 and Tag-4 are in orange, green and yellow respectivly. They all keep a measurements with minimal fluctuation. Table 5.4 shows the mean values and variances of all tags and all axes. Tags-1 has no variance that exceeds 0.143, with a_1^Z even showing a constant value over all seven measurements, leading to a variance of zero. Tag-3 and Tag-4 both have two axes with variances of 0.143 and one axis with a variance of 0.905. The mean values of Tag-1, Tag-3 and Tag-4 are in the reange of $3\frac{\circ}{s}$ and $40\frac{\circ}{s}$, with five

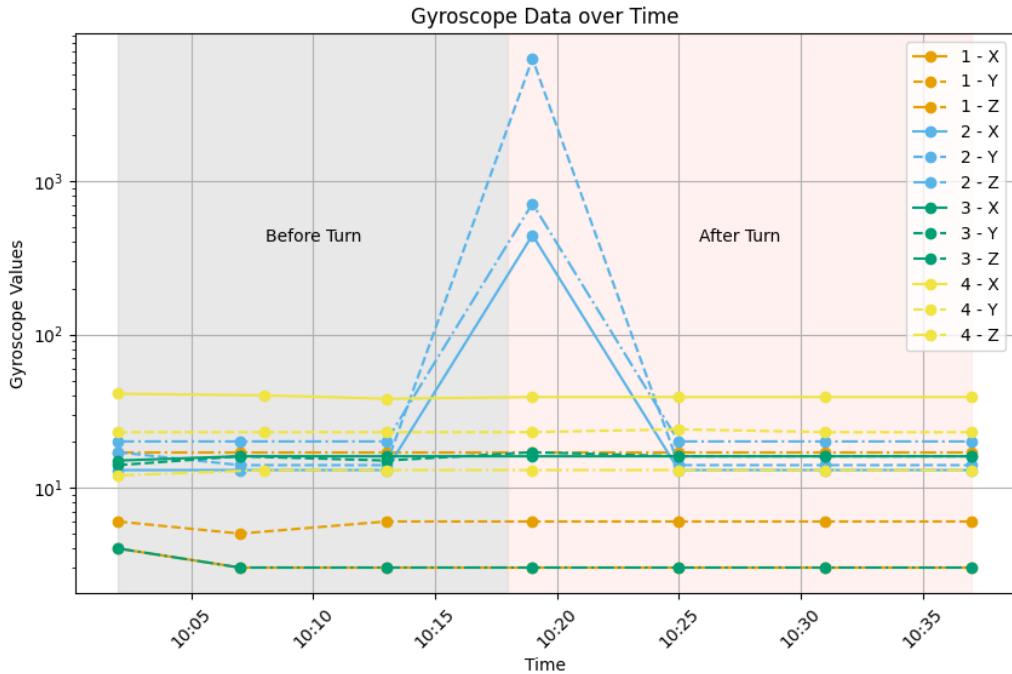


Figure 5.15: Experiment 3, gyroscope over time, using the angular velocity read.

falling into the range of $15\frac{\circ}{s}$ to $25\frac{\circ}{s}$. Tag-2 has variances between 740 and 16128.

Table 5.4: Summary of Gyroscope Data: Means and Variances of X, Y, Z Axes

tag	mean x	mean y	mean z	var x	var y	var z
tag-1	3.143	5.857	17.000	0.143	0.143	0.000
tag-2	23.3	41.3	68.0	741	5024	16128
tag-3	15.857	15.714	3.143	0.143	0.905	0.143
tag-4	39.286	23.143	12.857	0.905	0.143	0.143

Figure 5.16 shows the pairs of distances over time of experiment 3 using angular velocity read. The distances 1=2, 1=3, 1=4, 2=3 and 2=4 stay in a similar range over the whole experiment. Table 5.5 shows the mean and variance of the combined distance measurements. The variance in measured distances 1=2, 1=3, 1=4, 2=3 and 2=4 are all below one millimeter. The measured distances are between 0.39m and 0.87m. The measured distance 3-4 behaves very differently. It is bigger than all others, with 1.18m. It also has a comparatively high variance of 0.05m. Its lowest measurement is the first measurement after the turn. Looking at the split distance-measurement 3-4 in figure 5.17 shows, that the change in distance comes mainly from the measurements originating from tag 3. Distance measurements from Tag-3 to Tag-4 have a variance of 0.074m, while measurements from Tag-4 to Tag-3 have a variance of 0.014m.

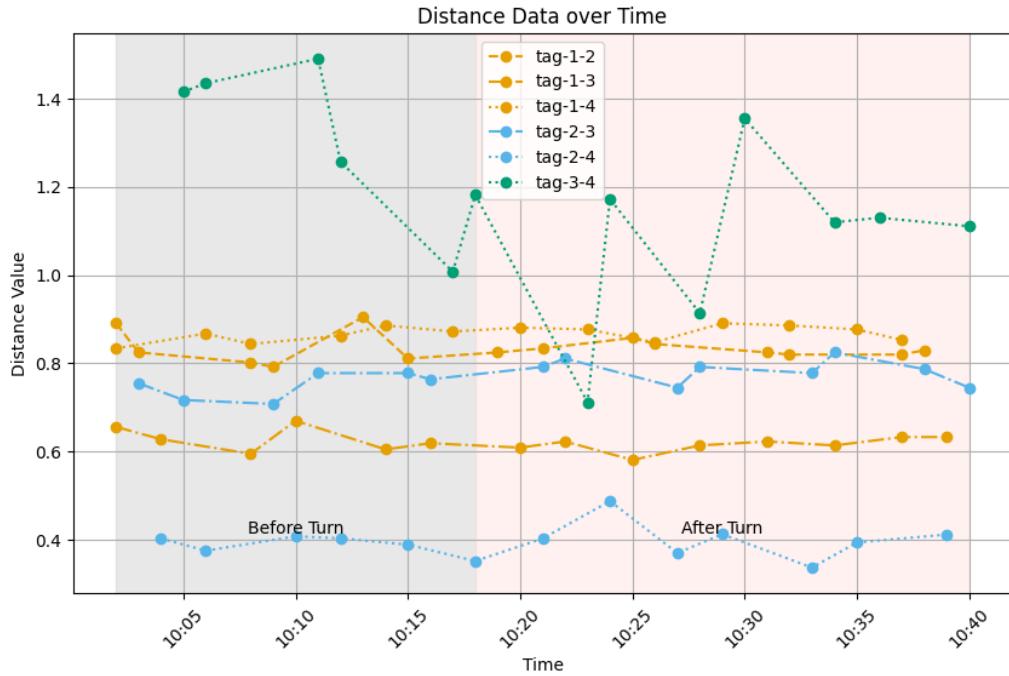


Figure 5.16: Experiment 3, combined distance over time, using angular velocity read.

Table 5.5: Statistics of the combined distance measurements between tags for experiment 3 with angular velocity read

	2	3	4
1	0.0834	0.622	0.868
2		0.770	0.396
3			1.177

Mean

	2	3	4
1	0.001	0.001	0.000
2		0.001	0.001
3			0.049

Variance



Figure 5.17: Experiment 3, split distance measurement between Tag-3 and Tag-4 over time.

5.3.3 Conclusion

The orientational read of the gyroscope produces puzzeling rezluts. Many values are zero for most of the time, but not consistently. This does not make any sense, since the orientational read does not reset after it has been read, rather it updates continuously. For a measurement to reutrn to zero after a wrong value, it would need to make the exact

same mistake again in reverse. This seems highly unlikely, especially for it to happen four times in one experiment. Additionally, the one orientation that should change, because it was physically turned, remains at zero before and after the turn. The most plausible explanation for this behaviour is, that there is an error in the implementation of the gyroscope module when performing read. Errors that could lead to such behaviour would be, if the orientational values could reset at runtime or if the wrong values are returned when queried.

The angular velocity read on the other hand produces useful results. The turn of tag 2 is very visible, while the other tags remain unaffected. The angular velocity around the X axis for Tag-2 reaches a maximal value of 6322 during this reading. While this might seem like a high value, it would mean that the turning of the tag, if performed at maximum speed constantly, would have taken 0.05s. Taking into account that this was a maximum speed and that a human can turn something quite fast, this seems like a reasonable result. Axis Y and Z show a rotational high velocity as well, even it is still smaller by a factor of ten than the rotation around the X-Axis. Since the sensor was attached on the cardboard by zip ties, it would have had some room to move during the turn. Additionally since the turn was performed by hand, it is reasonable to assume, that it was not performed perfectly around the X-axis. These two effects in combination could account for the high angular velocities around the unaffected Axis. These results show, that the angular read can be used to detect rotational movement on the Tag.

In addition we get an estimate for the biases of the MPU6050. It appears that the sensor outputs an angular velocity between $0 \frac{\circ}{s}$ and $50 0 \frac{\circ}{s}$. While this does not explain the result for the orientational read by itself, it may inform those readings.

During the orientational experiment, the Tag was turned around the Z-Axis, with an attempt being made to keep the MPU6050 sensor in the center. This moved the antenna of the DWM3000 board closer to the tag 1 and further away from Tag-3 and Tag-4. This behaviour was captured by the distance measurements. While the distance values are still not correct, even the differences, it captures something that actually happened in during the experiment.

Since the results of the orientational read were already known during the angular velocity read experiment, a decision was made to test, if a better turn could prevent the distance read. Instead of the Z-axis, the X-Axis was chosen for the turn, since it corresponds to the antenna position of the DWM3000 board. Additionally the turn was centered around the DWM3000 shield instead of the MPU6050. This results in the turn not being visible in the distance reads from the experiment.

It is unclear why the distance measurements between Tag-3 and Tag-4 were inconsistent, when originating with Tag-3. Both tags were not part of the experiment. If a repeated multipath-effect would appear, it should affect both measurements equally, since both participate in DS-TWR. Tag-4 served as the connection to the Phone in this experiment. It is possible, that a poor interaction between the BLE and UWB systems happened. It would be surprising, that these errors only appeared, when Tag-4 was participating, but not initiating the ranging session.

5.4 Experiment 4: Distance

The same 80 cm by 50 cm rectangle setup was used as in previous experiments. The tags were turned on sequentially, giving them enough time to build the network. The phone was connected to one tag. The max distance parameter was set to 20 centimeter. After 20 minutes, Tag-1 was moved parallel to the shorter rectangle line about 20 cm towards Tag-2 on the next corner. Figure 5.18 shows a schematic view of the experiment. The system was then left resting for another 30 minutes. The goal of experiment 4 was to test the detection of unwanted movement.

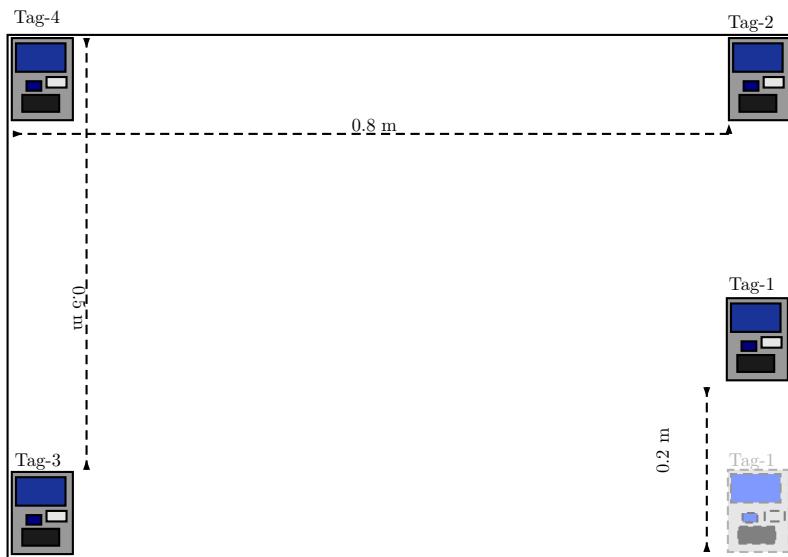


Figure 5.18: Schema of the setup of experiment 4.

5.4.1 Results

Experiment four was intended to test the distance measurement capabilities of the setup. Temperature and humidity and gyro behave as they do in experiment 1 5.1.1. For the gyroscopic data, the orientational read was used. It presents the same issues as in experiment 1 and 3. Humdity, Temperature adn Gyroscope will not be discussed further for tis experiment.

As with previous distance measurements, the pairs of distances moved together. Figure ?? shows the measured distance pairs of the 4 tags over time. At 14.24 Tag-1 was moved by 0.23 meters toward Tag-2. The time before the push has a gray background, while the time after the push has red one. The measured distances from Tag-1 to Tag-3 increases, while the distance to Tags-2 and Tag-4 dcreases. This represent what is happening in reality, since Tag-1 is now closer to Tag-2 and Tag-4 and further away from Tag-3 as before.

Table 5.6 shows the mean values of distance pairs before and after the move. The variances for all values, are low, confirming the choice to combine pairs during evaluation. As in

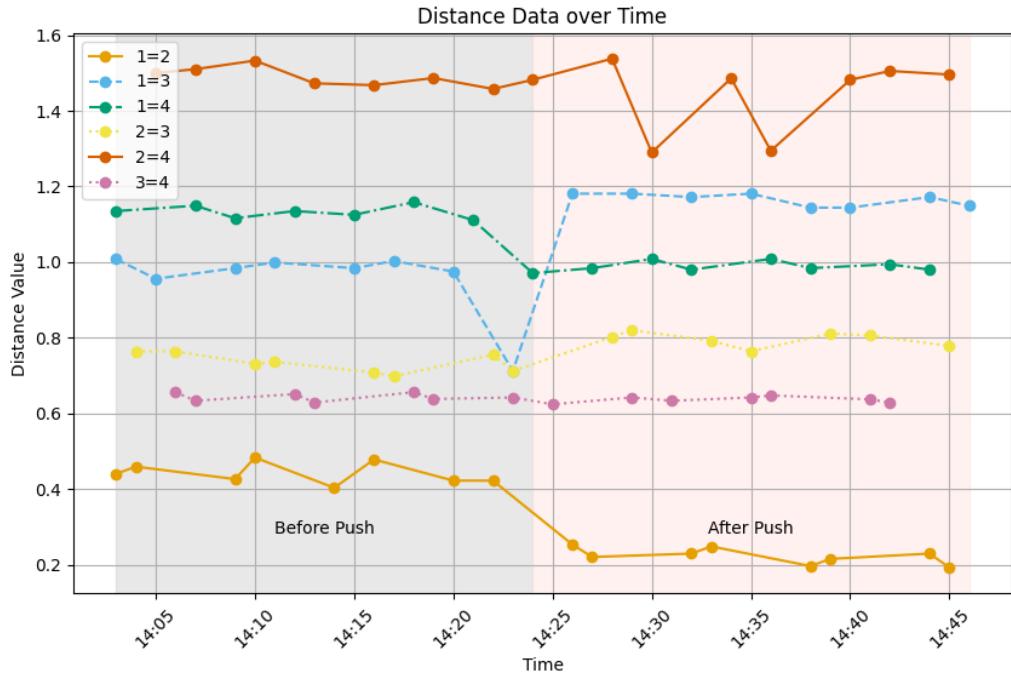


Figure 5.19: Experiment 4, pairs of distance over time.

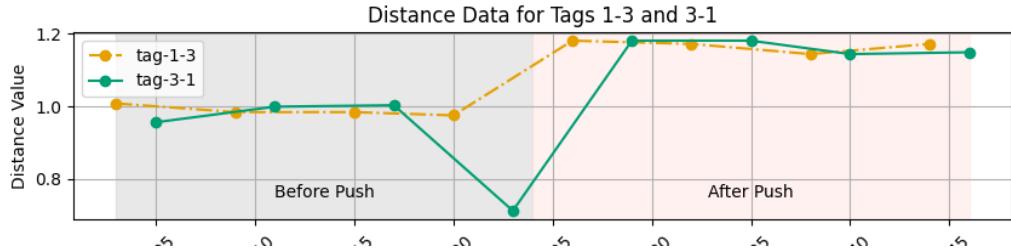


Figure 5.20: Experiment 4, distance between Tag-1 and Tag-3

experiment 1 and 3, the reported distances do not correspond to the what is physically happening. The difference in mean distance 1=2 before and after the push is 0.209m. This is close to the 0.23m that Tag-1 was acutally moved to Tag-2. The measurements show Tag-4 0.144m closer to Tag-1 after the push. The effect on tag 4 should be notissable but not as large as it is. Since the tag moves lateraly towards tag 4, the difference should only be 0.03 meters. The difference in measured distance between Tag-1 and Tag-3 is 0.213 meters. Figure 5.20 shows the distance measurements between Tag-1 and Tag-3. Measurement 1-3 and 3-1 move together as a pair, ecepct for the fourth measurement of distance 3-1. This also shows the higher variance of pair 1=3. If the outlier value is ignored, the distance difference between the means beofre and after the oush becomes 0.179 meters. This is still too large for the difference a latteral move, it should only be a 0.100 m difference. Their is also a small increase in the distance between tags 2 and 3, but which starts before tag 1 was moved.

Table 5.6: Statistics of the combined distance measurements between tags before and after the push for experiment 4

	Tag-2	Tag-3	Tag-4		Tag-2	Tag-3	Tag-4
Tag-1	0.442	0.953	1.133		0.223	1.166	0.989
Tag-2		0.734	1.490			0.796	1.447
Tag-3			0.654				0.635
Mean before move				Mean after move			
	Tag-2	Tag-3	Tag-4		Tag-2	Tag-3	Tag-4
Tag-1	0.000	0.010	0.000		0.000	0.000	0.000
Tag-2		0.001	0.001			0.000	0.010
Tag-3			0.000				0.000
Variance before move				Variance after move			

5.4.2 Conclusion

As was shown in Experiment 1, the distances do not match with the phisical distances. However, the differences in distance materialize correctly. The correct distances increase and decrease to match the physical reality. The most probable reason for the incorrect distances is the simplified calibration. The difference between the correct and measured distances never exceeds 0.4m, which is the assumed error by Qorvo, when using incorrect calibration.

The difference in distance is even closer, onyl beeing off by 0.1m at most. For now this error is still too large to be used unmodified in real world applications.

5.5 Experiment 5: Real World experiment

Experiment 5 was designed to check, how the setup would do in a real world applicaion. For this, all tags were put in a rucksack and taken on a journey. To protect the electronics from damage, each tag was put in a transparent cylindrical plastic bucket with radiu 012m and a hight of 0.14m and no lid. The buckets in the rucksack were stacked so there were two next to each other at the bottom and two on top.

First, the rucksack was taken by foot from Binzmühlestrasse 14 by foot to the Zurich Oerlokon train staiton. From their the journey continued via train to Zurich Main Station. After a short walk in Zurich Main Station, the journey continued via train to Lenzburg, were the jouney ended at the Lenzburg train station.

During the whole journey, the backpack was on the back of the person conductung the experiment and was nether put on the floor. This was done to protect the electronics from potential accidents by careless travelers during rush-hour.

5.5.1 Results

The walk by foot began at 17:43 and ended 17:51 when the S6 train to the main station was entered. The S6 arrived at the main station at 18:01. From their the track had to be changed and 18:08 the train to Lenzburg was entered. The train drove for 24 minutes, until the experiment ended 18:32 at the Lenzburg trainstation.

The following section will contain Figures for this journey. The figure all use the same color scheme to describe the journey. The first part by foot will have a gray background. The first trainride in the S6 will have a red background. The time in Zurich Main Station is blue and the final trainride to Lenzburg green.

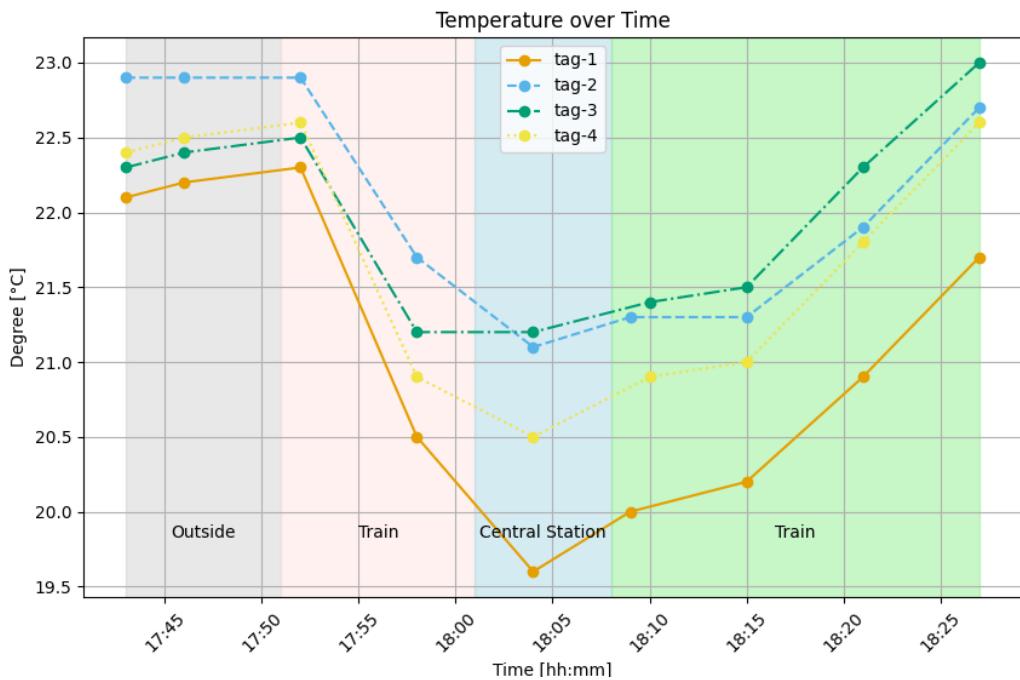


Figure 5.21: Temperature over time during experiment 5.

Figure 5.21 shows the measured temperatures during experiment 5. All four tags start with a high temperature between 22° and 23° . This remains true until the train is reached, where the temperatures start drop during the second measurement in the train to be a bit over one degree to a range of 20.5° to 21.7° . During the one measurement in the main station, most tags have their lowest values during the whole experiment, Tag-1 with 19.6° , Tag-2 with 21.1° and Tag-4 with 20.5° . The only tag without a significant drop during the main station is Tag-3, which records the same temperature of 21.2° as in the previous train. During the second train ride, all tags report a steadily increasing temperature.

Tag-2 starts with a temperature that is 0.5° higher than the other tags, who all start with very close values. Tag-1, Tag-3 and Tag-4 stay close until the first train ride. From there on Tag-1 drops to lower temperature measurements that are 1° lower than the lowest measurement of the rest, Tag-4, and stays there until the rest of the experiment. Tag-2 and Tag-3 report similar values during the Main Station and keep having similar values until the end. Tag-4 reports values that are close to Tag-2 and Tag-3, but always below it.

This becomes specially pronounced at the main station, where Tag-4 measures 20.5° , 0.6 °lower than Tag-2 and 0.7 °lower than Tag-3.

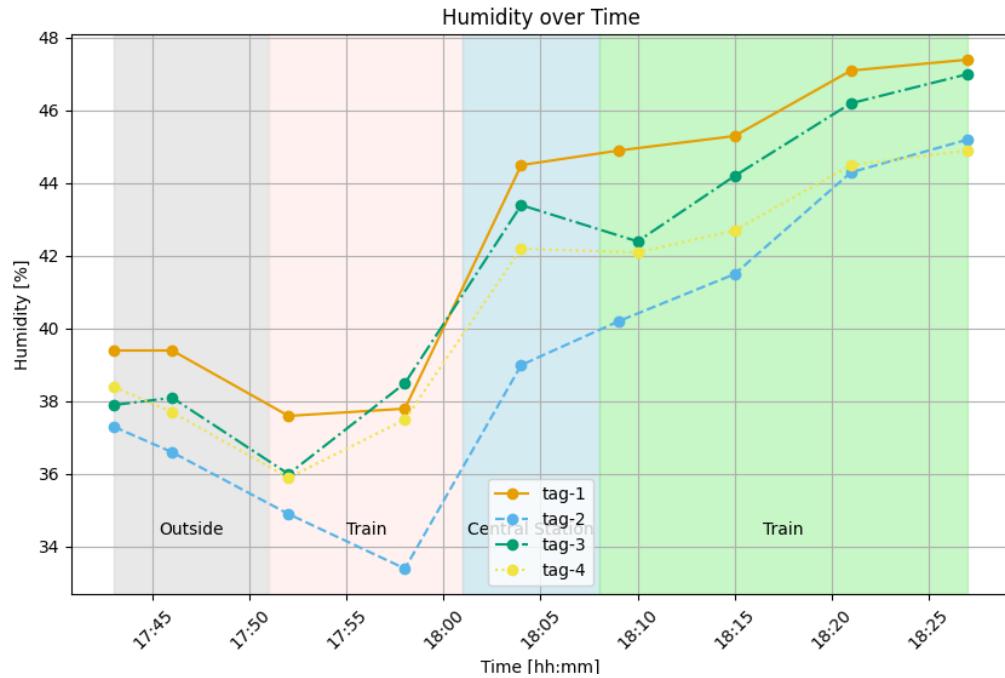


Figure 5.22: Humidity over time during experiment 5.

Chapter 6

Final Considerations

6.1 Summary

6.2 Conclusions

6.3 Future Work

Bibliography

- [1] E. Hsu, “An overview of the IEEE 802.15.4 HRP UWB standard,” https://blogs.keysight.com/blogs/tech/rfmw.entry.html/2021/07/28/an_overview_of_IEEE-J7ac.html, 2021, (Accessed: 2022-12-22).
- [2] “IEEE standard for low-rate wireless networks,” C/LM - LAN/MAN Standards Committee, IEEE, 2020.
- [3] “IEEE standard for low-rate wireless networks—amendment 1: Enhanced ultra wideband (UWB) physical layers (PHYs) and associated ranging techniques,” C/LM - LAN/MAN Standards Committee, IEEE, 2020.
- [4] “Getting back to basics with ultra-wideband (uwb),” Qorvo US, Inc, Tech. Rep., 2021.
- [5] M. F. Mecklenburg and C. S. Tumosa, “Mechanical behavior of paintings subjected to changes in temperature and relative humidity,” *Art in transit: studies in the transport of paintings*, 1991.
- [6] S. Michalski, *Paintings, their response to temperature, relative humidity, shock and vibration*. National Gallery, 1991.
- [7] D. Saunders and J. Kirby, “The effect of relative humidity on artists’ pigments,” *National Gallery Technical Bulletin*, Vol. 25, pp. 62–72, 2004.
- [8] B. Borg, M. Dunn, A. Ang, and C. Villis, “The application of state-of-the-art technologies to support artwork conservation: Literature review,” *Journal of Cultural Heritage*, Vol. 44, pp. 239–259, 2020.
- [9] E. Schito and D. Testi, “Integrated maps of risk assessment and minimization of multiple risks for artworks in museum environments based on microclimate control,” *Building and Environment*, Vol. 123, pp. 585–600, 2017.
- [10] J. Shah and B. Mishra, “Customized iot enabled wireless sensing and monitoring platform for preservation of artwork in heritage buildings,” *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. IEEE, 2016, pp. 361–366.
- [11] E. Landi, L. Parri, R. Moretti, A. Fort, M. Mugnaini, and V. Vignoli, “An iot sensor node for health monitoring of artwork and ancient wooden structures,” *2022 IEEE International Workshop on Metrology for Living Environment (MetroLivEn)*. IEEE, 2022, pp. 110–114.

- [12] K. Gulati, R. S. K. Boddu, D. Kapila, S. L. Bangare, N. Chandnani, and G. Saravanan, “A review paper on wireless sensor network techniques in internet of things (iot),” *Materials Today: Proceedings*, Vol. 51, pp. 161–165, 2022.
- [13] R. K. Garg, J. Bhola, and S. K. Soni, “Healthcare monitoring of mountaineers by low power wireless sensor networks,” *Informatics in Medicine Unlocked*, Vol. 27, p. 100775, 2021.
- [14] D. Coppens, E. De Poorter, A. Shahid, S. Lemey, and C. Marshall, “An overview of ultra-wideband (uwb) standards (ieee 802.15. 4, fira, apple): Interoperability aspects and future research directions,” *IEEE Access*, Vol. 10, pp. 1–23, 2022.
- [15] P. Leu, G. Camurati, A. Heinrich, M. Roeschlin, C. Anliker, M. Hollick, S. Capkun, and J. Classen, “Ghost peak: Practical distance reduction attacks against {HRP}{UWB} ranging,” *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1343–1359.
- [16] M. Stocker, H. Brunner, M. Schuh, C. A. Boano, and K. Römer, “On the performance of ieee 802.15. 4z-compliant ultra-wideband devices,” *2022 Workshop on Benchmarking Cyber-Physical Systems and Internet of Things (CPS-IoTBench)*. IEEE, 2022, pp. 28–33.
- [17] aosong, “Am2302 product manual.”
- [18] Y. A. Ahmad, T. S. Gunawan, H. Mansor, B. A. Hamida, A. F. Hishamudin, and F. Arifin, “On the evaluation of dht22 temperature sensor for iot application,” *2021 8th international conference on computer and communication engineering (ICCCE)*. IEEE, 2021, pp. 131–134.

Abbreviations

ABI	Application Binary Interface
AITF	Active Internet Traffic Filtering
AWS	Amazon Web Service
BloSS	Blockchain Signaling System
CIA	Confidentiality, Integrity and Availability
CSIRT	Computer Security Incident Response Team
DDoS	Distributed Denial of Service
DoS	Denial of Service
DNS	Domain Name System
DOTS	Distributed-Denial-of-Service Open Threat Signaling
ETH	Ether (Cryptocurrency)
EVM	Ethereum Virtual machine
IaaS	Infrastructure as a Service
IETF	Internet Engineering Task Force
IoT	Internet of Things
IPFS	Inter Planetary File System
ISP	Internet Service Provider
NFV	Network Function Virtualization
P2P	Peer to Peer
PoA	Proof-of-Authority
PoW	Proof-of-Work
REST	Representational State Transfer
RTT	Round Trip Time
SDN	Software-Defined Networking
SLA	Service Level Agreement
VNF	Virtualized Network Function

List of Figures

2.1	Power Spectral Density: Bandwidth B, lower-frequency f_L , upper-frequency f_H , [1]	6
2.2	General MAC Frame Format [2]	7
2.3	HRP UWB PHY Symbol Structure [1]	10
2.4	UWB signal transmission byte encoding, [4]	10
2.5	Schematic view of a PHY frame defined by IEEE 802.15.4 [2]	10
2.6	SHR Field Structure [2]	11
2.7	General PHR Field Format [2]	11
2.8	HRP-ERDEV Frame Structures [1]	12
2.9	PHR Field Format for HRP-ERDEV in HPRF Mode [3]	12
2.10	timeline of single-sided two-way ranging (SS-TWR)[3]	13
2.11	timeline of double-sided two-way ranging (DS-TWR)[3]	13
2.12	timeline of double-sided two-way ranging (SS-TWR) with three messages[3]	14
3.1	Sequence diagram of setup and oberservation loop. Setup is performed once, oberservation loop repeats until stopped.	26
3.2	Left: Five dots, all having at least two connections, still blue can move independently. Right: minimal 2-connected graph, no movement possible.	28
4.1	Signal of a DHT22 sensor-read as presented in the manual [17].	32
4.2	Schematic view of the MPU6050, showing the direction of the three axis X,Y,Z.	34
4.3	Configurations of the DWM3000 for UWB communication	34
4.4	Job struct	38

4.5	nRF Toolbox module menu, with the added Art Tracking Module	42
4.6	nRF Toolbox shows available devices to connect to	43
4.7	nRF Toolbox UART module screen	44
4.8	Art Tracking module observation screen before measurements	45
4.9	Section from the ArtMetricService.kt, main measurement loop	46
4.10	Art Tracking module, queries and responses	47
5.1	Schema of the setup of experiment 1.	50
5.2	Experiment 1, temperature over time.	51
5.3	Experiment 1, humidity over time.	51
5.4	Registered value of the gyroscope over time during experiment 1.	52
5.5	Experiment 1, distance over time.	53
5.6	Experiment 1, distance over time, for all pairs i-j.	55
5.7	Experiment 1, distance over time, for combined pairs i=j.	56
5.8	Schema of the setup of experiment 2.	58
5.9	Photo of elevated Tag-2, candles and thermometer used in experiment 2. . .	58
5.10	Experiment 3, temperature over time, with external measurement added. .	59
5.11	Experiment 3, humidity over time, with external measurement added. . .	60
5.12	Schema of the setup of experiment 3.	62
5.13	Experiment 4, gyroscope over time.	63
5.14	Experiment 3, gyroscope over time.	64
5.15	Experiment 3, gyroscope over time, using the angular velocity read. . . .	65
5.16	Experiment 3, combined distance over time, using angular velocity read. .	66
5.17	Experiment 3, split distance measurement between Tag-3 and Tag-4 over time.	66
5.18	Schema of the setup of experiment 4.	68
5.19	Experiment 5, pairs of distance over time.	69
5.20	Experiment 5, distance between Tag-1 and Tag-3	69
5.21	Experiment 5, distance between Tag-1 and Tag-3	71

List of Tables

2.1	HRP UWB Frequency and Channel Assignments [2, 3]	8
2.2	HRP UWB Mean PRF (Based on IEEE 802.15.4 and IEEE 802.15.4z, [2, 3])	9
3.1	Adruino compatible pin assignment	23
3.2	Non-Adruino compatible pin assignment	24
5.1	Mean and Variances for Temperature and Humidity Data by Tag during experiment 1.	50
5.2	Mean, excpected values and variance of distant measurements, experiment 1.	54
5.3	Statistics of the combined distance measurements between tags for experiment 1	54
5.4	Summary of Gyroscope Data: Means and Variances of X, Y, Z Axes . . .	65
5.5	Statistics of the combined distance measurements between tags for experiment 3 with angular velocity read	66
5.6	Statistics of the combined distance measurements between tags before and after the push for experiment 4	70

Listings

Appendix A

Contents of the Repository

The code repository contains the following content:

Installation

Operation