University of Zurich[UZH]

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

# 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

ifi

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 All passages taken verbatim or in sense from published or unpublished writings are identified as such.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

# Chapter 1

# Introduction

Artworks are sensitive to many external factors, such as humidity, temperature, and vibrations. While these factors are generally well controlled in museums and storage, art pieces also need to be moved between these buildings. Artworks include a variety of objects, ranging from photographs and paintings to the pieces needed for modern installations. The transportation, therefore, needs to be secure and flexible. Sensors play a critical role in the safe transportation of these objects.

## 1.1 Motivation

Certify is an international cooperative project between twelve partners 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 concerns itself with the integration of IoT devices.

One of the 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 artwork by attaching a device to it. This device allows for unique identification using cryptographic methods. It is also intended to act as a cluster of sensors that collect information about the artwork's surroundings that is relevant to the well-being of the artwork. The goal is to have constant data on the artwork throughout its lifecycle. This is intended to help secure the artwork and help with chain of custody monitoring.

## 1.2 Thesis Goals

This project aims to develop a system that implements a localized version of the artwork tracking envisioned by the Certify project, meant for transportation in a truck. Additionally, the system will extend the Certify Projects goal by adding new detection methods and informing the truck driver 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 driver's phone can query the network and display the collected metrics to them. The system should alert the driver if a metric is outside the accepted norm.

This Thesis presents a proof of concept implementation. The used metrics are not intended to be a complete representation of the sensors needed for security 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 be implemented 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 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 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 criteria.

#### 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 was then implemented in a simplified manner based on the available material. For this, four tags were built and equipped with sensors, communication capabilities, and a 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 ensured minimal external influence. The resulting data from the tests were analyzed using statistical methods. The goal was to determine the system's reliability, find limitations, and look for improvements.

## 1.4 Thesis Outline

This Thesis is structured as follows:

Chapter Two presents the fundamental knowledge researched for this Thesis. Chapter Two also presents previous research on tracking artwork and sensor networks. Chapter Three presents the design of the system and its inner workings and capabilities. Chapter Four shows the implementation that was developed and used for this project. Chapter Five describes the experiments performed. It presents the results of the experiments and discusses them. Chapter Six summarizes the findings of this Thesis and discusses the most important aspects in a conclusion.

# Chapter 2

# Fundamentals

In this chapter, the fundamental knowledge that was researched during this thesis is presented. Section 2.1 introduces the background knowledge required for this thesis and Section. Section 2.2 presents the current state of research on topics surrounding this thesis.

## 2.1 Background

This Section describes the theoretical background used in this thesis. It covers key aspects, such as communication protocols, two-way ranging, sensor networks, and required graph theory.

### 2.1.1 Wirerless 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 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 2000s, the production of embedded systems and sensors, in particular, rose. This led to falling prices and sensors becoming widespread. With the new availability of sensors, Wireless Sensor Networks (WSN) became more widespread. While not originating in during this time, the term itself was coined in 1980, and 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 where no power supply is available. The sensors, therefore, must run on limited, battery-based energy.
- Modularity: WSNs should work for various 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 networks, where routing is only established when needed, and devices can be added to or leave the network at will. Doing this takes the problem of self-organization and modularity into account. A modified version of AODV is used in Zigbee to this day [**?**].

In 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 transmit the cluster's data. New cluster heads are elected periodically to spread the increased energy drain from the data transfer to the network head. LEACH is used [**?**, **?**] and improved [**?**] to this day.

### 2.1.2 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 with which it shares its wavelength, such as WLAN or Bluetooth. Since UWB uses very short, distinct pulses over a short range, it has been found to be useful in ranging systems [1]. 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 [1], I will not discuss it further in this Thesis. Since UWB devices tend to be small and have low energy consumption, in combination with the capability of ranging and 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

2.1.3 and 2.1.4 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 10 db below the maximum power spectral density are called the lower- and upper-frequency points, see Figure 2.1. These two points must be 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.

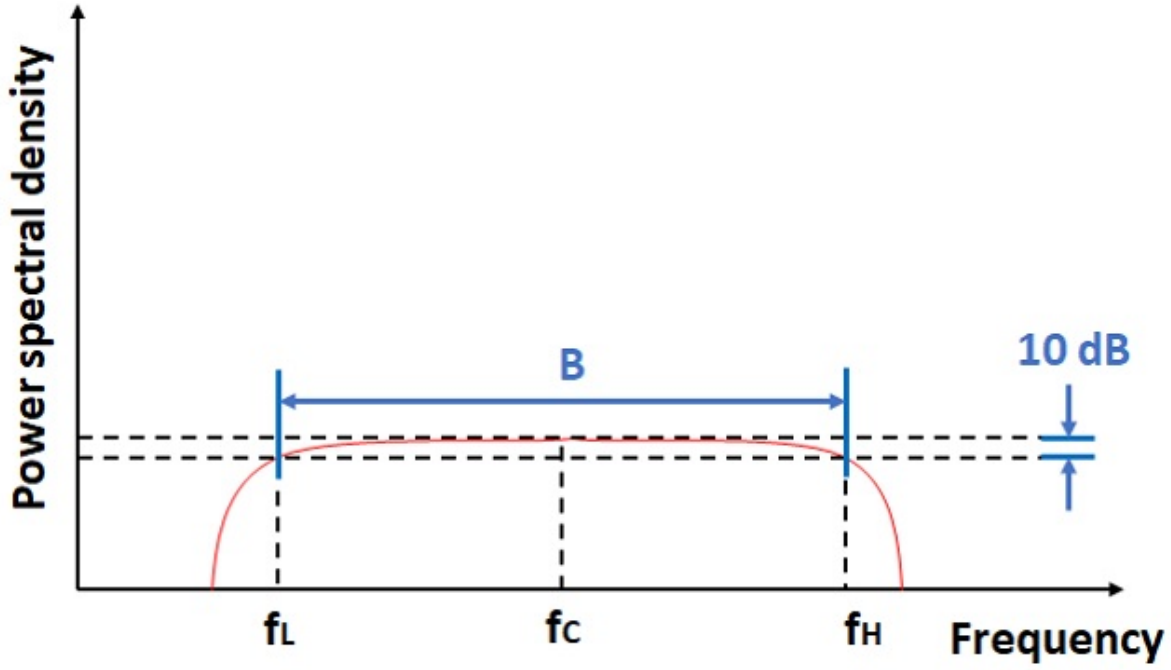



Figure 2.1: Power Spectral Density: Bandwidth B, lower-frequency $f_L$, upper-frequency $f_H$, [1]

**UWB supported Nodes**

The IEEE 802.15.4 standard distinguishes between two types of devices. Full-function devices (FFD) are capable of connecting to multiple other devices, receiving, transmitting, and coordinating. Reduced-function device (RFD), on the other hand, can only connect to one other device and act as a 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 across multiple networks while still using the short address. Each device also has an extended address that is not assigned by any coordinator and serves as a universal unique identifier (UUID).

### 2.1.3 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 frame type, frame format, security mechanism, addressing, and frame validation. The Mac Layer additionally provides rules for beacon management, and channel access.

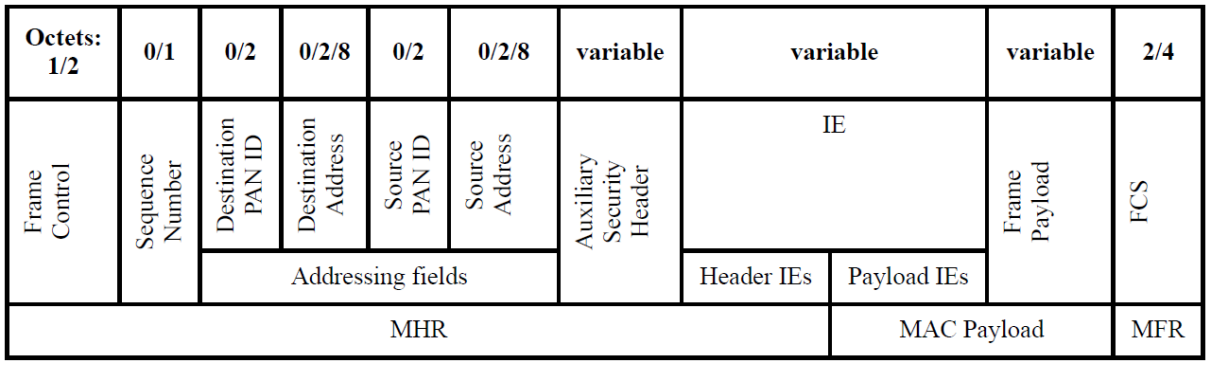



Figure 2.2: General MAC Frame Format [2]

#### 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 have arrived. The Addressing Fields carry the IDs of the sender and recipient for the frame. The Auxiliary Security Header Field only exists if specified in the control Field. It contains additional information needed for the chosen security method.

There are two parts to the information element (IE). The header IE specifies additional information about the frame, for example, data formatting information or channel 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 on the frame type and data length.

The MAC footer (MFR) marcs 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.

### 2.1.4 UWB PHY

#### PHY Chanel

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 the table (see table 2.1).

| **Channel number** | **Center frequency (MHz)** | **HRP UWB band** | **Mandatory** |
|---|---|---|---|
| 0 | 499.2 | sub-gigahertz | ✓ |
| 1 | 3494.4 | Low band | |
| 2 | 3993.6 | | |
| 3 | 4492.8 | | ✓ |
| 4 | 3993.6 | | |
| 5 | 6489.6 | High band | |
| 6 | 6988.8 | | |
| 7 | 6489.6 | | |
| 8 | 7488 | | |
| 9 | 7987.2 | | ✓ |
| 10 | 8486.4 | | |
| 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 raging results. Before

transmitting, the 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 ensured that the signal had 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 wich bursts are sent by the transmitter. The mean PRF is the average PRF while sending the payload (power switching service data unit PSDU) [1]. The higher the mean PRF, the shorter the airtime of each frame, and it allows for faster communication. HRP-ERDEV uses 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 phases of the pulses so the burst beak for one has an opposite amplitude to the other. Figure 2.4 shows the single 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 cannot 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 uses BPM and BPSK. Some HRP-ERDEV can use only BPSK, which uses a higher PRF and, therefore, reduces airtime.

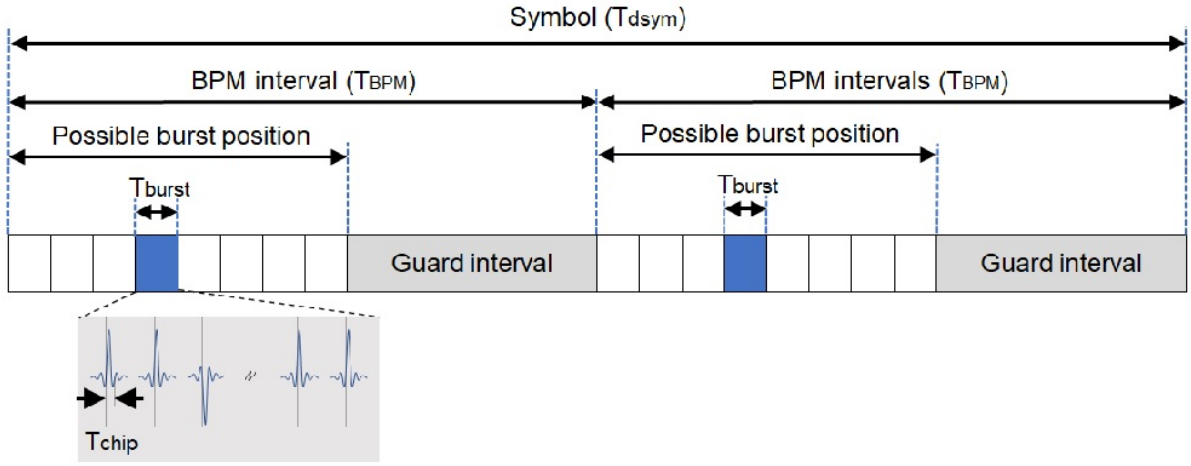



Figure 2.3: HRP UWB PHY Symbol Structure [1]

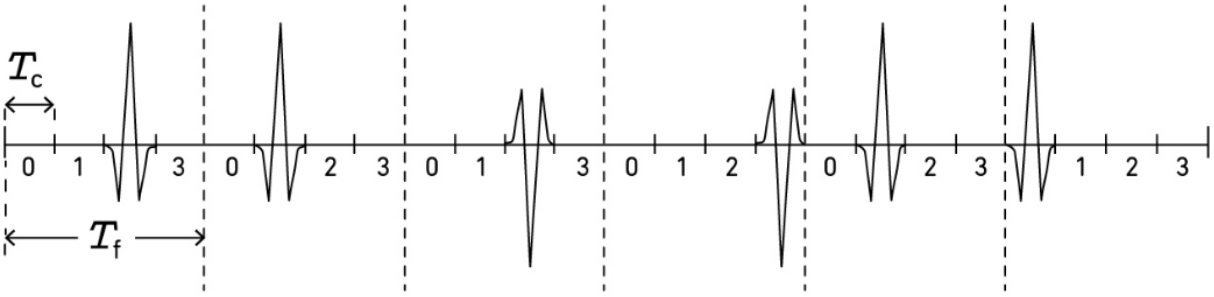



Figure 2.4: UWB signal transmission byte encoding, [4]

**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 to be sent, namly the MAC frame.

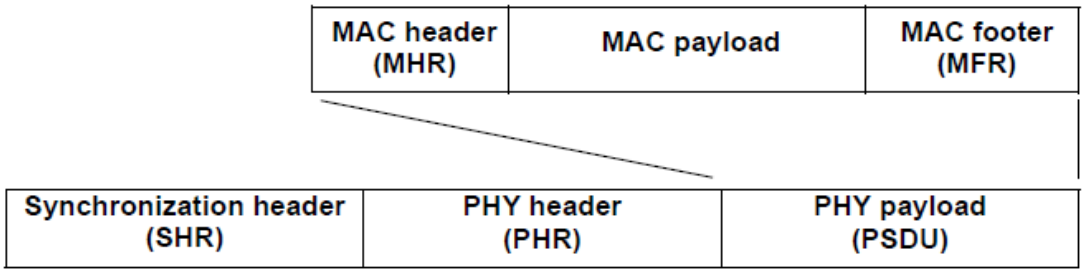



Figure 2.5: Schematic view of a PHY frame defined by IEEE 802.15.4 [2]

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 lengths are used. The sequence of pulses specifies

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 coming signals will be data, starting with the PHY header. It also contains a timestamp, which can be used for ranging using the time difference of arrival (ToA), see Section 4.1.4

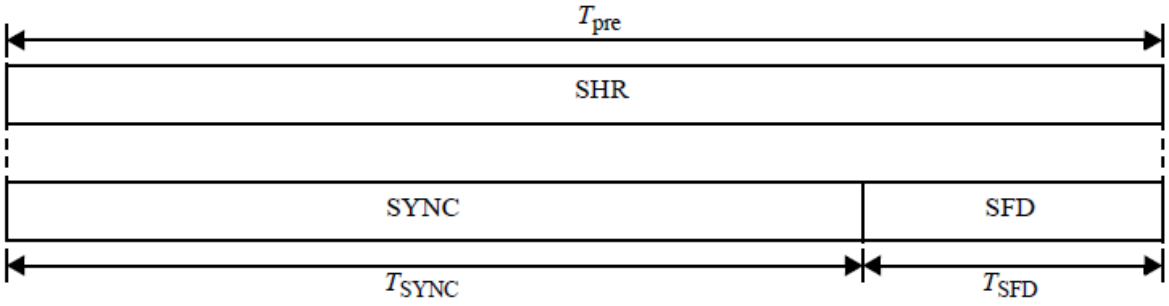



Figure 2.6: SHR Field Structure [2]

The PHY header contains all the information needed to read the PHY payload (see Figure 2.7). The first bit defines the data rate used during the payload transfer (see section **??**). The next seven bits define the length of the frame, with a frame length of a maximum of 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 detect, but not fix double-bit errors.

The last part of the PHY frame is the PHY payload (PSDU). This contains the MAC frame, as defined in section 2.1.3.

| **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 participating 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 additionally 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 three only contains the SHR and STS, without a payload.

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

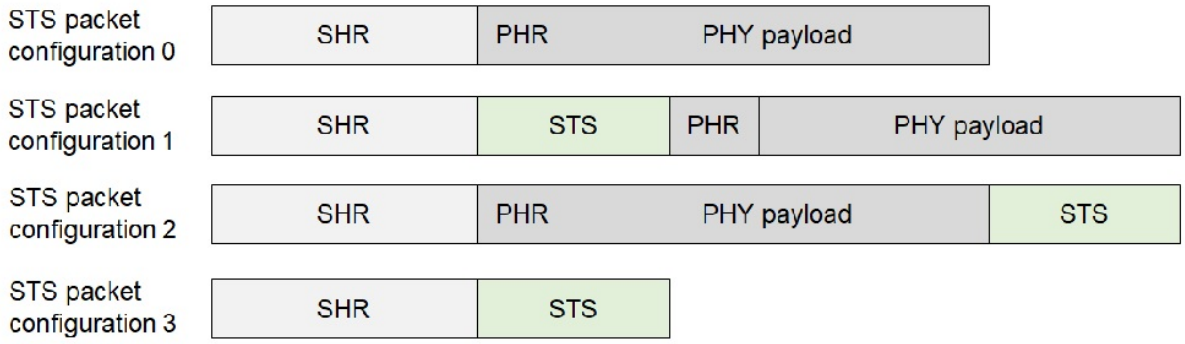



Figure 2.8: HRP-ERDEV Frame Structures [1]

| 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]

### 2.1.5 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 devices using timestamps and multiplying it by the speed of light. In this Section, both SS-TWR and DS-TWR will be discussed. Two-way ranging(TWR) refers to DS-TWR in all other parts of the Thesis.

**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 $T_{B1}$. On receiving the response, Device A records its timestamp, $T_{A1}$. The round trip time $T_{round}$ can now be calculated using the timestamps from A:

$$T_{round} = T_{A1} - T_{A0} \tag{2.1}$$

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

$$T_{reply} = T_{B1} - T_{B0} \tag{2.2}$$

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

$$distance = (\tfrac{1}{2} \cdot T_{round} - T_{reply}) \cdot c_{air} \tag{2.3}$$

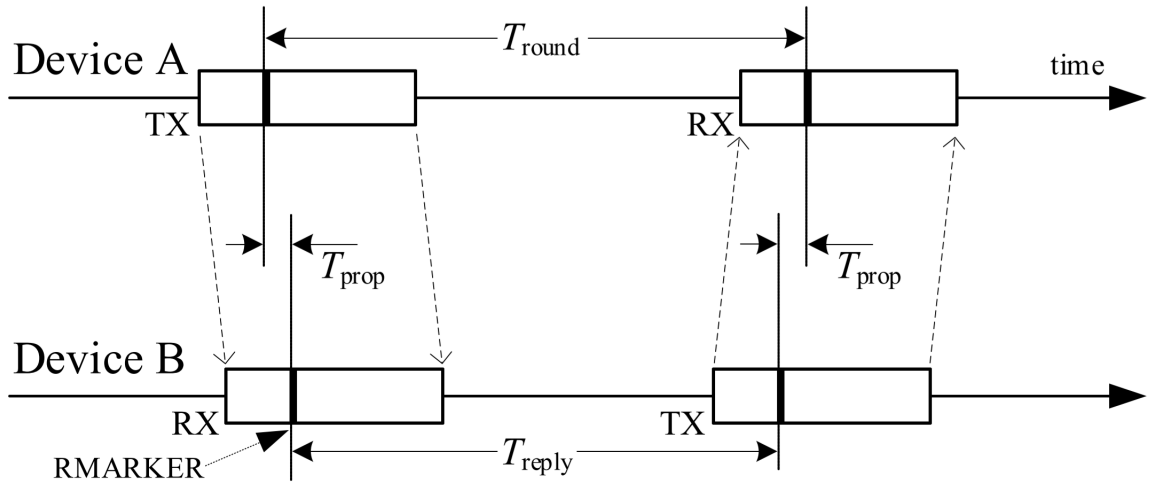



Figure 2.10: timeline of single-sided two-way ranging (SS-TWR)[3]

**Double-sided two-way ranging (DS-TWR)**: DS-TWR involves both devices A and B performing an 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:

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

$$distance = T_{prop} \cdot c_{air} \tag{2.5}$$

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.

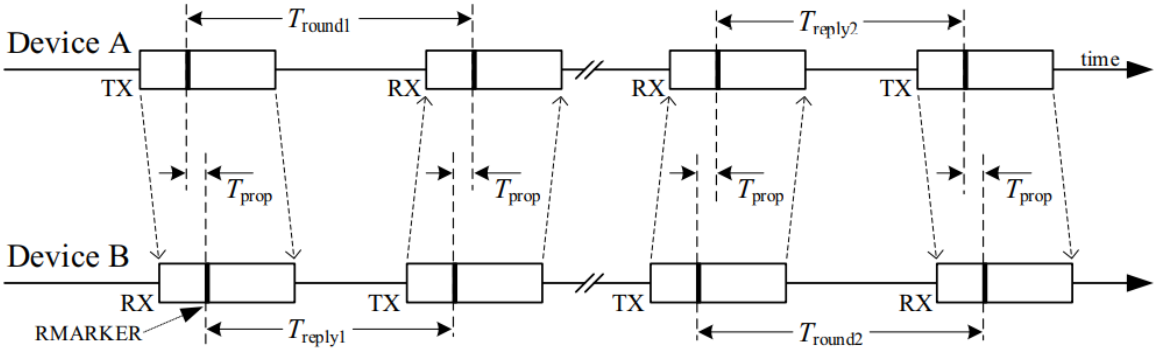



Figure 2.11: timeline of double-sided two-way ranging (DS-TWR)[3]

### 2.1.6 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

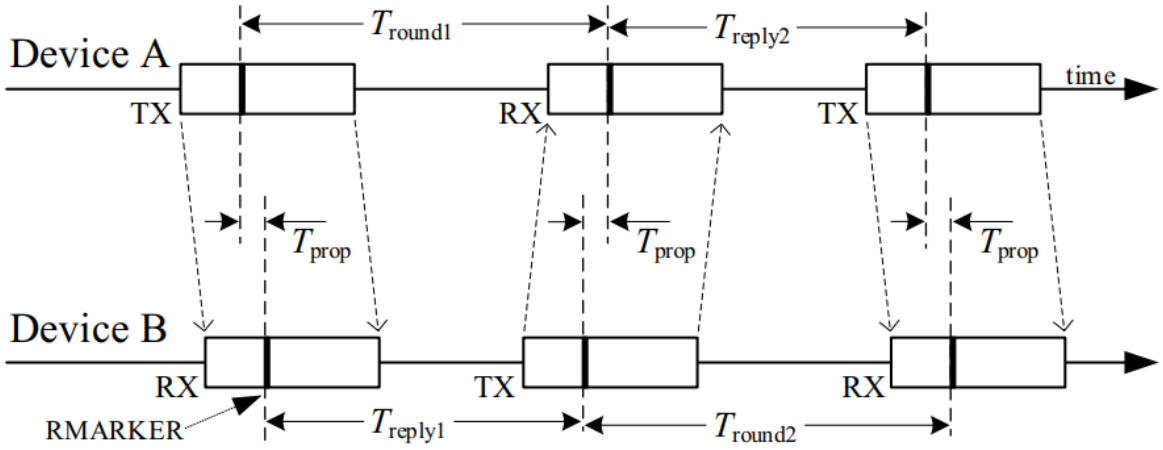



Figure 2.12: timeline of double-sided two-way ranging (SS-TWR) with three messages[3]

where it takes at least k removals 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 subgraph $(V', E', w') \in S_{(V,E)}$ 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 verteces $A, B, C \in V$, it holds that $e_{AB} + e_{BC} \geq e_{CA}$.

Finding minimally weighted k-connected subgraphs is an NP-hard problem. Kahn et al. [**?**] developed distributed approximation algorithm that finds weighted k-connected subgraphs in metric graphs. It gives an approximation ratio of $\mathcal{O}(k \cdot log(n))$. This means that the aproximated solution $w^{apr}$ and the optimal sollution $w^{opt}$ fullfill $w^{apr} < k \cdot log(n) \cdot w^{opt}$). The algorithm puts all vertices in an order, which is determined randomly, 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 a higher rank.
There are more precise approximation algorithms to find minimally weighted k-connected subgraphs [**?**, **?**]. However, they are centralized, meaning the graph has to be known in its entirety by one actor.

## 2.2 Related Work

This Section presents on overview of the literature relevant to this thesis. This includes IoT systems used in the art world, sensor networks, and wireless ranging, showing the current practices and current state of research.

### 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, an improvement on existing tools like infrared imaging, and an 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 significant 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 the 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 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 implementing or testing 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. Fort et al.'s research showes the value of low-cost sensors in detecting 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 prevalent problems arising from the development of NSWs, mainly power management, 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 the vitals of mountaineers and call for help when measurements have dangerous values. They used an Arduino Mega board equipped with a radio transceiver, using LoRa with a star topology.

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

[**?**] used an accelerometer to detect pipeline vibrations 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 involving positioning systems for industrial settings. They looked at the positioning systems in papers using RFID, BLE, UWB, Wi-Fi, and ZigBee. 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 on ranging using UWB was done by Gezici et al. [**?**, **?**]. These papers gave an overview of the different positioning systems for UWB, angle of arrival, received signal strength, time of arrival, and time difference of arrival. Time of arrival and time difference of arrival were studied further in these publications, presenting error sources and mitigation tools.

Early research focused on the augmentation of UWB-ranging methods. [**?**] proposed using integer programs for mitigating 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 were proposed to reduce the bias in time difference of arrival by applying of a well-known algebraic explicit solution for source localization [**?**]. [**?**] improved UWB ranging by eliminating random error. They did this by pre-filtering, using an 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 visualy impared 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 encountered many multipath-effects in such a complex environment. They 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. The used components are presented in the section 3.1. 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 Microcontroler

The artwork-tag's fundament is built 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) 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 10 manage the power supply for additional modules, which include 3.5 and 5 Volt supply pins. Thirty-two of the pins are installed the same way as the pins on the Arduino uno,

making it compatible with many peripherals designed with this common board in mind, such as the dwm3000. The remaining ten pins are enough to attach the sensors. The nRF52840 DK includes a USB-B port that is used for powersuply. Additionally, the USB-B port is connected to two pins and is used for UART communication and 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 and 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 many processes transparent to the user if they wish. The SDK uses the Serial Peripheral Interface (SPI) for communication between the shield and the microcontroler.

### 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 changes. This is also confirmed by the user manual [17], which states that a read-interval of less than 2s is impossible.

The humidity sensor can detect the full range from 0% to 99.9% humidity, with an advertised maximum error of 2 % pts [17]. No research confirming or denieing these claims could be found.

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, detecting headers and footers, and parsing the binary messages.

### 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 axes in degrees per second. In this project, the accelerometer data was not used, only the gyroscope.

The MPU6050 uses four pins: two for power supply and ground and two for 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 the only worker. While the nRF52 SDK does not supply an I2C API, it offers a Two Wire Interface (TWI) implementation compatible with the I2C protocol. It used to offer MPU6050-specific support in older versions of the SDK.

### 3.1.5 Tag technical plan

The microcontroller builds the base of the 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 pin's name according to the nRF52830 DK manual, but not always. This Thesis will use the names in the manual to describe the pins. Pins are the method 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 3.1a. 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 to 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. These pins can transfer data and be used for communication by using voltage modulation. The nRF52840 has an 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 two datapins. The protocol states 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 point 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 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 3.1a shows which devices use which pins. The only pins shared by multiple devices are power and ground pins. The microcontroller supplies enough power to support this.

The sensors are attached to the same power source and ground as the shield but use different data pins. The DWM3000 leaves enough pins unused that both sensors could be

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

(a) 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 | | | |

(b) Non-Adruino compatible pin assignment

Table 3.1: Pin assignments and compatibilities

attached to them. Since it is not visible which pins the shield leaves free, it was decided to use data pins that are not attached to the DWM3000 for the DHT22 and MPU6050. Table 3.1b shows how the sensors are connected to the remaining open pins.

## 3.2 Architecture

In order to discuss how the dataflow works, first, Section 3.2.1 will establish what services are implemented in each part of the system. 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 for managing its sensors. It has to do the correct setup and convert its output into an understandable form. The tag can perform ranging with all its neighbors. Additionally, the tags are responsible for searching for networks to join and reacting 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 certified 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 power management will not be provided here.

**Network:** The network is responsible for keeping 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 unavaliable. It offers the possibility for phones to connect to the network. It ensures quieries 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 devices connected to the phone.

**Phone:** The phone connects to the network via the provided method. It offers a graphical user interface (GUI) for the driver to use. 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 interval-length. The phone is responsible for querying sensor data for each tag and measuring once at 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 for alerting the driver to this fact.

The Certify project also plans 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 feasible. 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.

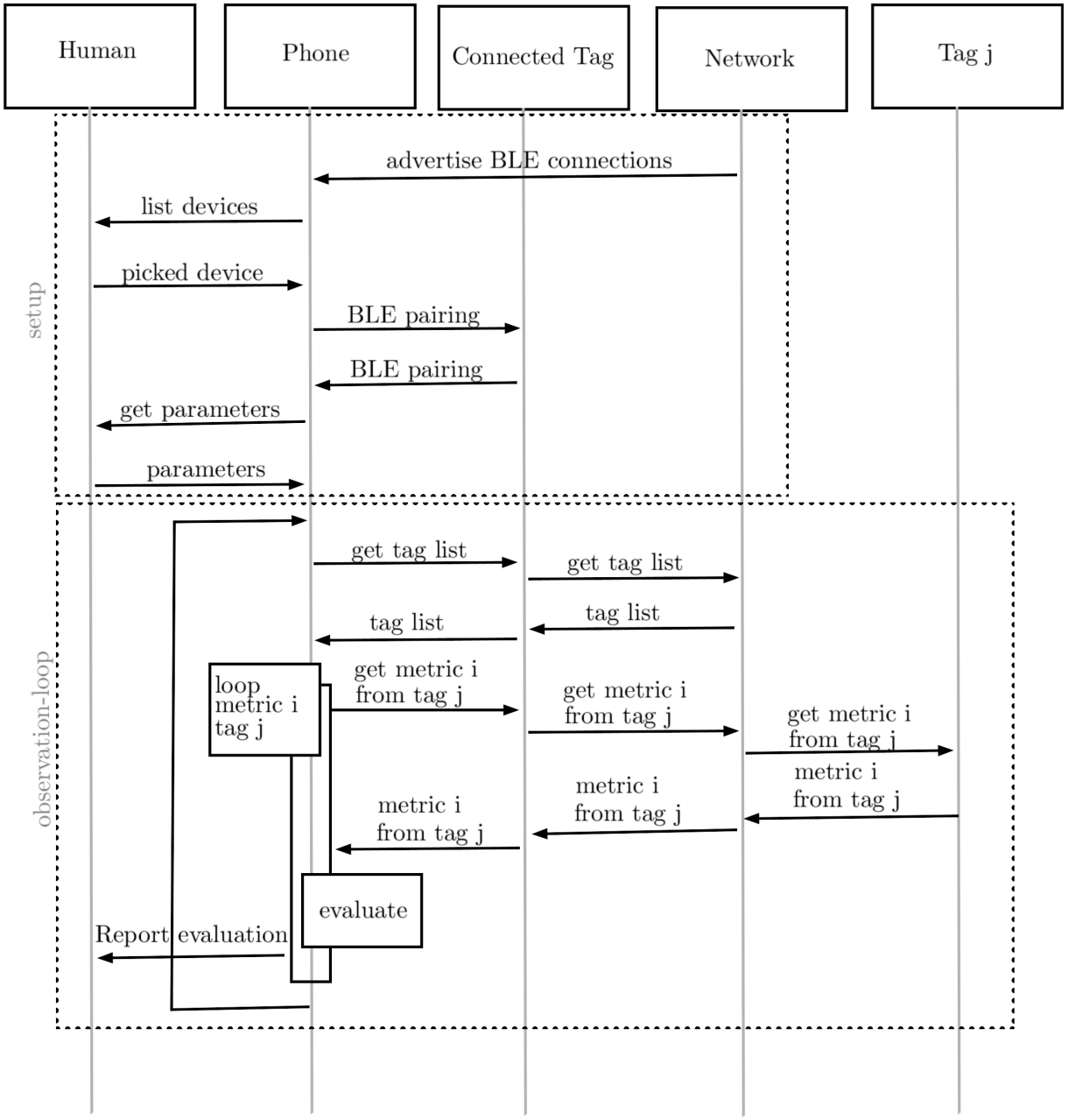



Figure 3.1: Sequence diagram of setup and observation loop. Setup is performed once, and the observation loop repeats until it is stopped.

### 3.2.2 Dataflow

Section 4.1.3 describes how a tag connects to the network. Figure 3.1 shows a sequence diagram of the setup and main observation loop of the system. At 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 used tags and the network they build.
- Connected Tag is part of Network but is listed separately. It represents the tag that is communicating to the phone
- Tag-j is also part of Network. It represents the tag that is queried during the observation loop

Phone and Human communicate by using a GUI. Phone and Connected Tag communicate using BLE. Every communication inside Network happens using UWB. This includes the communication between the Connected Tag, Network, and Tag-j.
When Phone wants to connect to Network, it looks for advertised BLE devices. It then displays the devices to Human and lets them pick one. The phone then pairs with the chosen tag, making it the Connected Tag and Phone's connection to Network. Once connected to Network, Phone will prompt 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.

Once the parameters are chosen, Human can start the observation.
Each iteration of the observation loop begins with a call to Network for a list of all tags currently in Network. Since the tag network is a dynamic sensor network, the tags in Network can theoretically change. In practice, this should only happen when artwork is loaded/unloaded or a tag becomes faulty. The request for the list is transmitted to Connected Tag over BLE, which then queries Network for all connected devices. The response is returned to Phone. Phone then starts a nested loop, iterating over the list of tags and metrics captured by the system. For each measurement and tag combination (i,j) Pphone contacts Connected Tag for the value, which in turn queries Network. Once the message arrives at Tag-j, Tag-j gets measurement i. In the 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 neighbors and will report the list of distances, together with the tag addresses they correspond to. Metric i is then transported over Network back to Connected Tag and finally to Phone. Phone must then evaluate the retrieved data.
During the evaluation process, 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 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 will be alarmed.

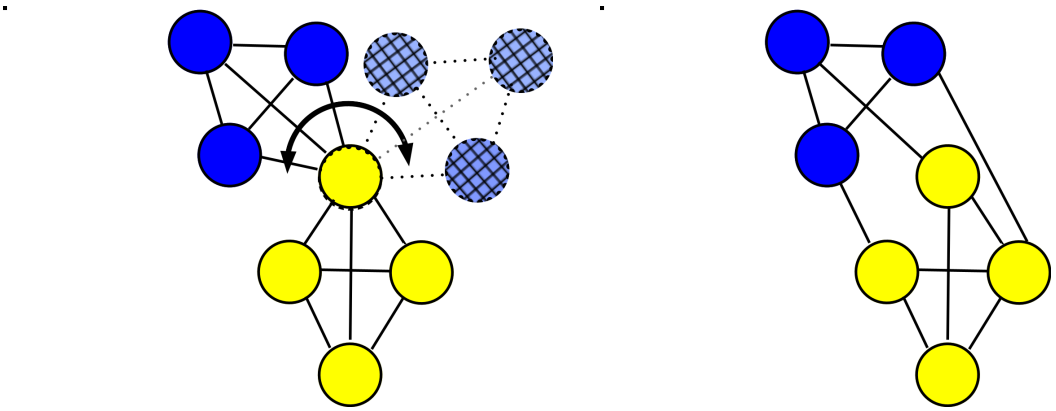
#### Distance evaluation

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, $T := \{t_1, t_2, ...\}$.
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, $t_2$ 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 consisting of the Objective function L, and no constraints.

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. The quadratic program can, therefore, be solved efficiently.
By setting the values of the tag $t_o$ to zero, the results of the quadratic function become grounded. It is not strictly necessary, but without it, the returned solution could have

values anywhere in the Euclidean space. The solution will place the other tags near that region by setting one tag to the coordinates at the origin. 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 must be known. This alone is not sufficient to ensure unique results. The left of Figure 3.2 illustrates this point in two-dimensional space. Every circle is connected to two others, but the blue circles can still 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 to 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 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 then 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 $rank(i) < rank(j)$ or $rank(j) < rank(i)$. 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 to 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.

While not connected to a phone, the tags inside the truck form a decentralized mesh using UWB for communication. Each tag keeps two lists: a list of known devices and a list of neighbors. When a new tag joins the network, it sends a joining request over

UWB, containing its UUID, using a weak signal. All tags in the network that receive this request add the new device to their list if known devices. If the new device also has a higher rank, they additionally add it to their list of neighbors. They then answer by sending their own UUID and adress 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 added tag's rank is also higher than the new tag, it will add it to the list of neighbors. If the new tag now has four neighbors, it stops. Otherwise, it will repeat the process with an increasingly stronger signal until it has either found four tags with a higher rank or reached maximum signal strength. Afterward, 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 connections. Once that happens, the tags in the tracks 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 their 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 neighbors and known devices. The coordinator records a list of all acknowledgemtns, 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, a tag take the following steps:

- It conducts a UWB two-way ranging session with each tag in the neighbor 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 neighbor list.

The first time a distance is requested, the tag will perform more ranging sessions than necessary to build a 4-connected graph. Afterward, it only performs ranging with four other tags unless a new device is added. Suppose a ranging session does not report a result because a tag left or became unavailable. In that case, 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 Euclydean space, which, when mapped to a graph, forms a metric graph. As discussed in section , a four-connected graph is needed to uniquly 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 theirfore 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 shows the implementation of the app.

## 4.1 Tag

The tags are based on the hardware described in Section 3.1. DWT3000, DHT22 and MPU6050 were connected to the nRF52840 microcontroler. A Fresh 'N Rebel powerbank supplied the power. All components were attached to a carboard piece and attached with zip ties. Figure **??** shows a picture of a installed tag, with added labels. All four tags were build in the same maner.

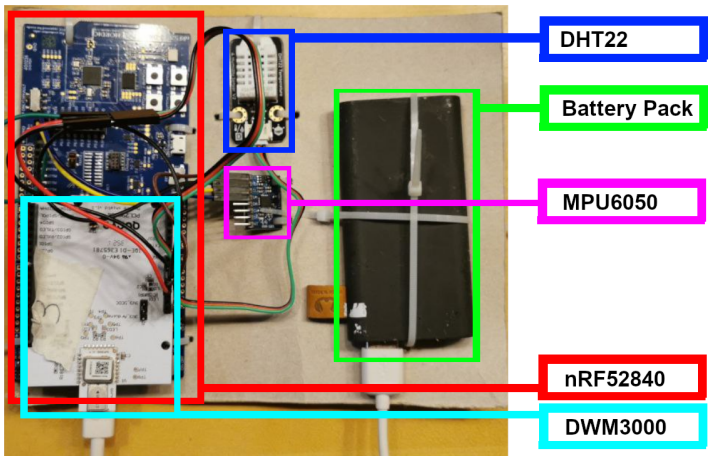



Figure 4.1: Photo of a tag, with labels.

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 and how they interact using the job handler module. The section 4.1.7 discusses the challenges of 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 for setting up the sensor during initial startup and providing the sensor's 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 builds the basis for this implementation. It was adapted for the DHT22, and functionalities needed by the job handler module were added.

Since the DHT22 is a straightforward sensor using single-bus communication, not much setup is needed. The sensor data evaluation requires that the pin's voltage be read out in pre-defined intervals. To achieve this, a clock is required. The clock resource has to be reserved and initiated at startup. No other setup 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 5ms intervals until a *logic low* is registered, signaling 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 being the start condition of the data transfer.
The data is transferred in five chunks of eight bits. Each bit is preceded 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 an 8-bit buffer, *logic high* corresponding to a 1 and *logic low* to 0.
Once all five chunks are read and the communication has ended, the module can verify the data. The first two bytes are combined to form the temperature information in Celsius, and the second and third form the humidity. Both values are multiplied by 100 and stored in a 16-bit integer. This does not lose data since the sensor only measures up to a precision of 1 after the decimal point. The data being stored in an integer helps with data transfer. It will be converted back on the phone. The fitht 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 cannot be negative and the DHT22 sensor can only detect temperatures as low as -20°C.

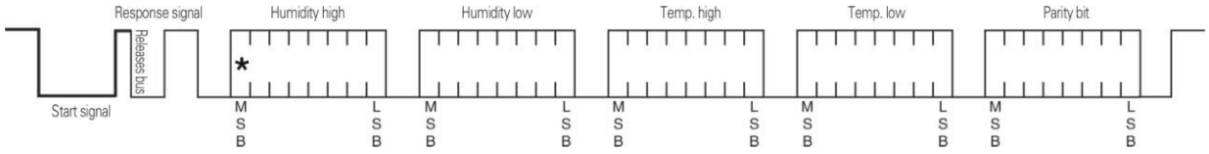



Figure 4.2: Signal of a DHT22 sensor-read as presented in the manual [17].

### 4.1.2 Gyroscope

This module manages the MPU6050 gyroscope and accelerometer. It is responsible for setting up the sensor and reporting 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 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 the 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 its resources are locked and can not be used by other services.

Afterward, the results from the sensor can be read again using the TWI service. The TWI-TX requires the address of the read device and a registry to write. The address of the sensor is the same for all MPU6050 sensors and can be found in the MPU6050 datasheet to[**?**]. It sets a flag to true once the sensor has written the data, which can then be read using the TWI-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 in what has 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*.

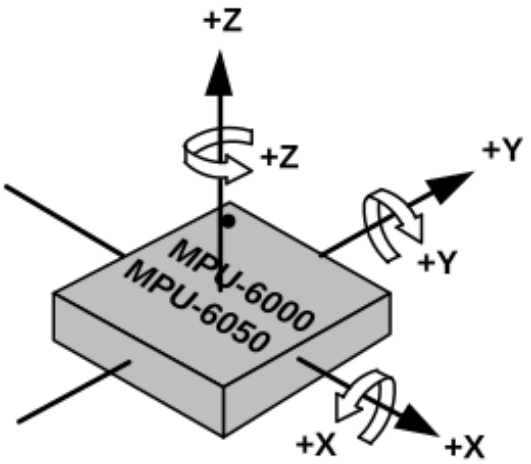



Figure 4.3: Schematic view of the MPU6050, showing the direction of the three axes: X,Y,Z.

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 around the axis at this moment and 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 similarly. 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 those of the angular velocity variables. If any angular velocity values are smaller in absolute magnitude than the corresponding read value, they are 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.

### 4.1.3 Network

The network module is responsible for managing the network. This consists of sending requests to join a network, managing requests to join a network, keeping track of its neighbors, 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 the design chapter. A 4-connected minimal graph of 4 vertices must necessarily include that all the nodes are fully connected. This leads to a simplified

network architecture. Since this implementation was built to run experiments and not be used in real-world applications, many security measures were canceled. Messages are not encrypted, and devices are not authenticated. All messages are assumed to reach their destination, and no devices are expected to become unavailable.
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 must 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 ensure no cross-effects from previous sessions are possible. Then, the board is told to initialize. After that, the used configurations are sent to the shield. This includes channel 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 use case. 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, which serves no critical service but are useful for debugging.

Figure 4.4: 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 | 9 |
| SFD type selection | 4z 8 symbol |
| Data rate | 6.8 Mbits/s |
| PHY header mode | standard PHR mode |
| PHY header rate | standard PHR rate |
| SFD timeout | 129 |
| STS mode | enabled |
| STS length | 128 bits |
| PDOA mode | off |

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 an 8-bit long address for the purposes of this implementation. Each tag also keeps a list of all known addresses, called neighbors.

The address *0x3F* was used when a tag wanted to join a network. This was chosen since none of the used devices had this device ID, and it corresponds to a question mark when using ASCI encoding. When a tag wants to join a message, it sends the address *0x3F*,

followed by the message 'findnet' and its own address. It then starts listening for answers. If the listening timed out without 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's address. If the receiver address is equivalent to the tag's own address, 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-address is "*0x3F*", indicating that a tag is looking for a network. In that case, the network module adds the tag to the list of neighbors. It then waits for a time proportional to its own address before continuing. Since addresses are unique, this ensures that no two tags response to the new tag at the same time. Afterward, it sends a new message, beginning with the address of the new tag, followed by the string 'NEW' and its own address. This way, it can be added to the neighbors of the new tag as well.
The implementation of [**?**] was modified for sending messages. It sets the DWM3000 to TX, passes an 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 neighborhood. 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 neighbors, performing two-way ranging with each of them. First, it sends a prepare-ranging request to the neighbor it wants to perform ranging with before performing the ranging. It then sends the result back over the network to the requesting tag with the following format:

$$a_r \text{DST} a_t a_n c d_{tn} \tag{4.1}$$

with

- $a_r$: The address of the requesting tag.
- DST: The string "DST" indicates the purpose of the message.
- $a_t$: The address of the tag performing the measurement.
- $a_n$: The address of the neighbor that the distance was measured too.
- $c$: A boolean. If false, this is the last neighbor measured for this query.
- $d_{tn}$: The distance to measured.

The reason for each measurement triggering its own response is the message-lenght 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 send between both participants. This would unnecessarily drain energy from the tags that are not involved. Because of that, they sleep for the expected duration of a ranging session.
If the tag is the intended receiver for the prepare-ranging message, it will enter the preparation part of the two-way ranging module. If will function as deivce 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 them immediately. When the response is received, it checks if it is the expected response. If it is, the two timestamps $T^A_{TX_1}$ and $T^A_{RX}$ are retrieved. The final transmission time $T^A_{TX_2}$ is calculated by adding a constant $c_A$ to $T^A_{RX}$:

$$T^A_{TX_2}=T^A_{RX} + c_A \tag{4.2}$$

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

The tag that performs the ranging corresponds to device B in figure 2.12. Once it has sent the prepare-ranging message to its neighbor, it will enter the receiving part of the two-way ranging module. As device A, device B will also start by settings 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 retreive the timestamp of when the message was received, $T^B_{RX_1}$. Device B will add a constant $c_B$ to this timestamp to get $T^B_{TX}$:

$$T^B_{TX}=T^B_{RX_1} + c_B \tag{4.3}$$

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

from that, the distance:

$$T_{round1} = (T^A_{RX} - T^A_{TX_1}) \tag{4.4}$$
$$T_{round2} = (T^B_{RX_2} - T^B_{TX}) \tag{4.5}$$
$$T_{reply1} = (T^B_{TX} - T^B_{RX_1}) \tag{4.6}$$
$$T_{reply2} = (T^A_{TX_2} - T^A_{RX}) \tag{4.7}$$
$$ToF^{AB} = \frac{(T_{round1} \cdot T_{round2}) - (T_{reply1} \cdot T_{reply2})}{T_{round1} + T_{round2}) + (T_{reply1} + T_{reply2}} \tag{4.8}$$
$$distance = ToF^{AB} \cdot c_{air} \tag{4.9}$$

The distance is then returned, all caches cleared, and the module continues with the next distance measure if any remain.

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 by 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.
Qorov has published a manual on calibrating 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 clock drift
- 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 the scope of this thesis, a simpler version was designed. The tags were set up in a tetrahedron, so each tag was 30 cm apart. 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 randomly and increased. If it was lower, $d_{rx}$ or $d_{tx}$ would be increased. 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, receives messages from it, and sends messages to the phone using BLE. The nRF52840 microcontroller is equipped with an antenna with BLE capabilities. The nRF52 SDK includes libraries for the management of this antenna. It also includes the *ble_app_uart* example project. The example project advertises a BLE connection and handles the pairing process. Once connected, it forwards all incoming

communication to a USB-UART module connected to a computer. Input from the computer via USB-UART is sent as a message to the paired device. The *ble_app_uart* example project was taken 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. 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 resources. Firstly, a nRF clock has to be reserved for the BLE module. Then, the power management 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) must be prepared. The information on what functions to call when the SoftDevice receives data has to be set, as well as the advertised 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 been connected to the UWB network, the BLE connection can be advertised. 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 achieve this, a query-handler function was passed to the SoftDevice during initiation. All incoming 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 being queried and transforms 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 the SoftDevice and send it to the phone.

### 4.1.6 Job Handler

The job-handler module connects all other modules. It takes job structs (see figure 4.4, interprets which module is responsible for handling them, and calls the job together with the relevant data. The job struct consists 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 4.5: Job struct

There are 14 total job types. The following list describes them, as well as how the job-handler handles them:

- **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 ID.
- **set network and address**: This job commes from the Network module. It informs thaht the network connection has been established. The job is returned to the Network module, with the received message to be added to the list of neighbors.
- **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 temperature and humidity read request 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 reques**: 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. The job is passed to the Two-Way Ranging module if the ranging session is with this tag. Otherwise, the tag will go 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 message content.
- **ble get known devices**: This job comes from the BLE module. It requests a list of all neighbors. 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. Afterward, the modules were merged into one project. The Network module was chosen as the base project, which the other projects were merged into. This choice was made since the Network module was based on [**?**], which intern was based on an example published by Qorvo. The Qorvo example uses a lot of shorthand, magic numbers, and development shortcuts that are not easily readable to developers outside the firm. The Network module was, 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 challenges that are 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, for which this example was made and is used in this thesis. The NRF52840 board does not contain the necessary pins to attack a DWM3000 board. 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 module would no longer work since those workarounds now introduced mistakes. Since fixing the Qorvo example code would have been cumbersome, it was decided to change the other modules that used pins, the Gyroscope module and the Temperature and Humidity module. The pins for those modules, pins 11, 12, and 13, were 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, and UUIDs. These tools are all enabled or disabled in the sdk_config file. Merging, in general, requires only the tools needed by the merged module.
Three modules require an nRF clock: Two-Way Ranging, Temperature, and BLE. The nRF SDK offers exactly three clock slots, so all must be enabled with the appropriate clock type. Each module has to be adapted so it uses its assigned clock slot.
The nRF52 SDK can support up to three SPI or TWI connections simultaneously, named SPI0, SPI1, SPI2, TWI1, TWI2, and TWI3. In the nRF52 SDK, SPI and TWI share their allocated memory, so SPI0 can not be used while TWI0 is used and vice-versa. Since the DWM3000 uses two SPI connections and the MPU6050 uses one TWI connection, enough resources remain for both devices to run simultaneously. 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.

Like most embedded systems, the nRF52840 requires static memory allocation during flashing. The available memory is separated into flash memory and random-access memory (RAM). Some memory segments are required by every runable system:

- FLASH, **vectors**: The interrupt vector table defines the interrupt handlers for the system, like resets and 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 all global values' initial values.
- **rodata**: This section contains the constant variables that will not change at runtime.
- RAM, **data**: The initial values for changeable global variables are copied to this section during startup. 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 runntime 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 ones 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 config module is for, but it is required for the shield to work.
Since the base project was done 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 workarounds. An example of this is the "NRFX_SPIM3_NRF52840_ANOMALY_198_WORKAROUND_ENABLED" present in the SDF configuration. These workarounds let the SPI communicate with the shield and 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 over 10 bytes to the microcontroller over SPI, it encroaches on the SoftDevice RAM. This behavior 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 Semi Conductors, the maker of the used microcontrollers, published the code to the nRF Toolbox, a simple app allowing BLE communication with their devices. It is intended to pair with the *ble_app_uart* example, published in nRF52 SDK. Since the

*ble_app_uart* example code was used as the basis for the BLE communication on the tag side, the nRF Toolbox app was adapted for this project.

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 tracking to it. The nRF Toolbox already contains different modules intended for different examples. Among them is Universal Asynchronous Receiver/Transmitter (UART) module (see 4.5), which served as the basis for a new art-tracking module since it had a lot of valuable services already implemented.

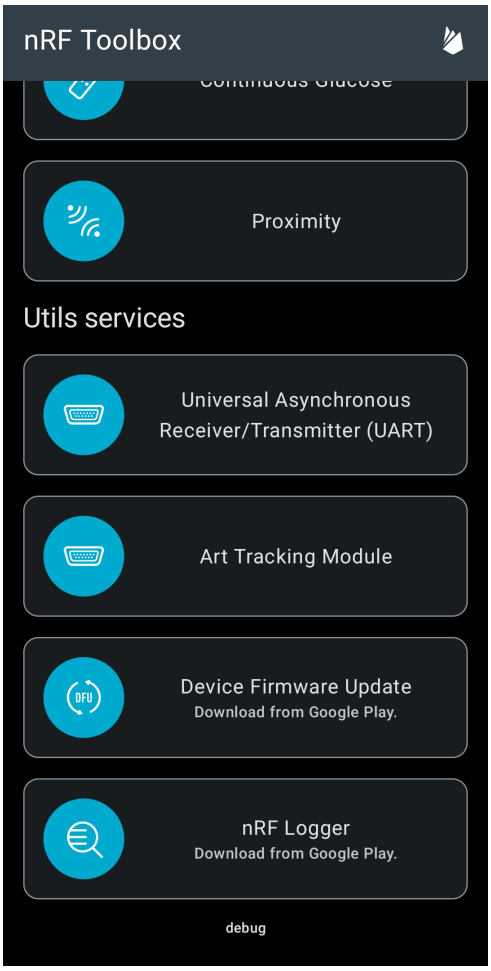
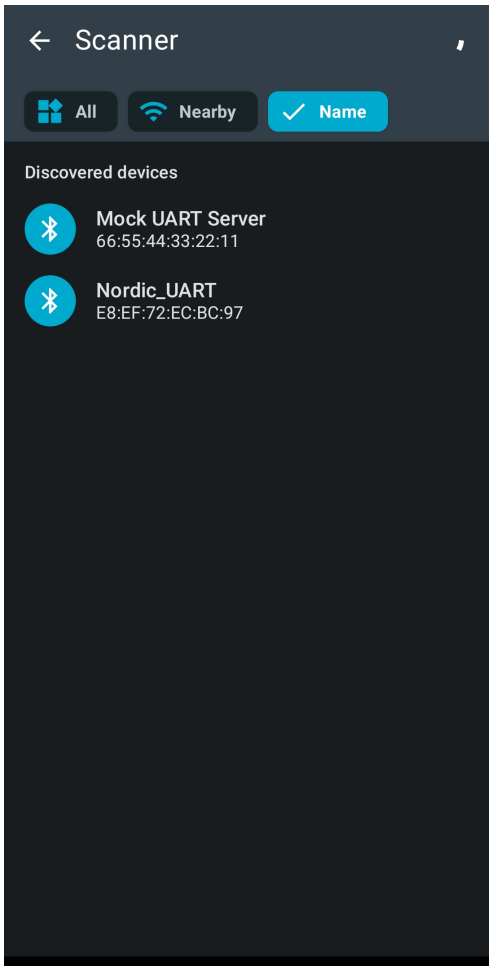


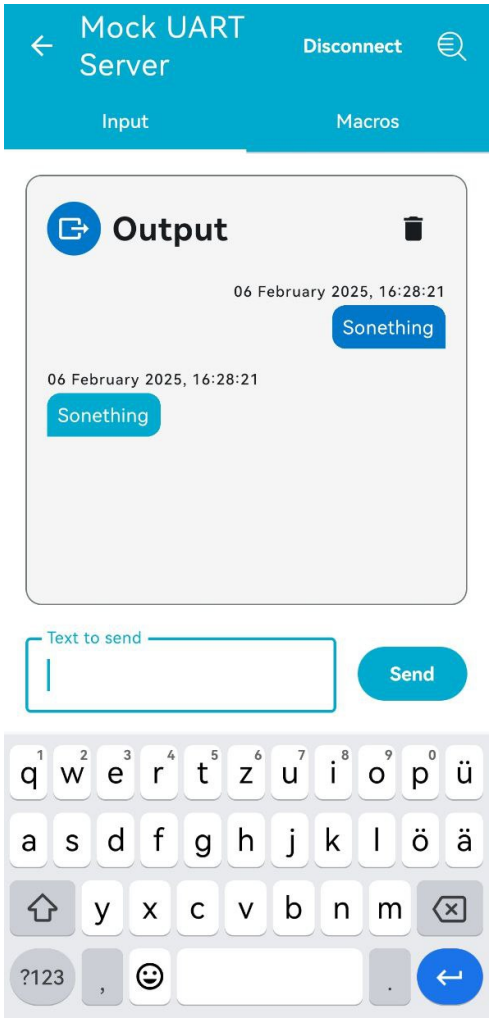

Figure 4.6: nRF Toolbox module menu, with the added Art Tracking Module

Before discussing the new implementations, a description of the original UART module is provided to facilitate understanding the modifications in the art-tracking module. When the UART module is opened, it shows all advertised BLE services and allows the user to connect to one of them 4.6a. Once connected, it opens a window similar to phone messengers. Here, it allows a keyboard input from the user and sends these messages to the connected devices (see 4.6b). This can be used to manually send queries to the tags and receive their responses unmodified in plain text.

The main differences between the art-tracking module and the UART module are after connecting to the relevant services. So, the are-tracking module opens up the same con-




(a) nRF Toolbox shows available devices to connect to



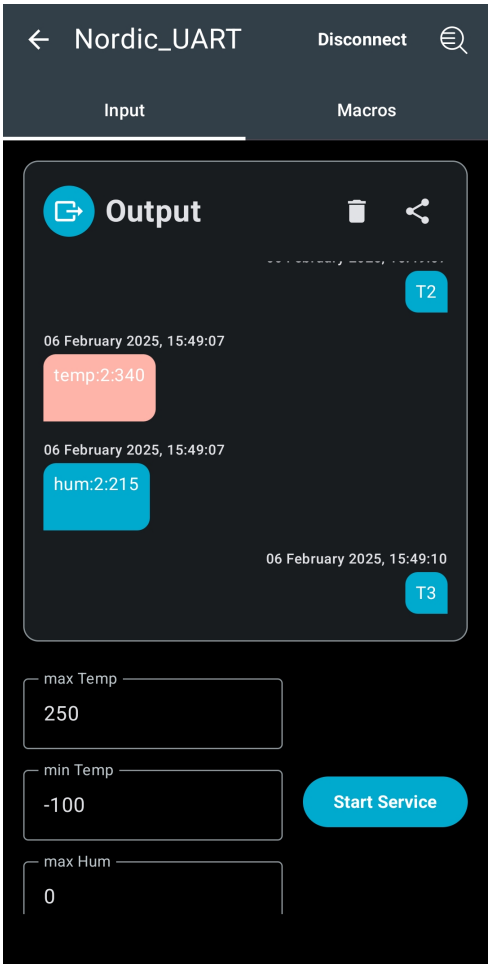

(b) nRF Toolbox UART module screen

nection page as the UART module 4.6a, in which the user is able to select the BLE service and connect to it.

Once connected, instead of the UART module, the observation screen is shown (Figure 4.7a). It also contains an output field, which will display messages. 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. The app will consider any measurement outside these parameters a dangerous value. The tolerated difference in angle compared to the previous measurement is set by *max Angle*; larger differences are considered dangerous values. Distance measurement works analogously with *max Dist* in meters. The *time* set defines the time that passes between measurements in seconds. The default time is set to 350 seconds. This means that the time passing between, for example, the temperature measurements on tag 2 is 350 seconds.

When the user presses the *Start Service* button, a program starts that periodically queries the tags for the measurements. Once this process has started, the queries will appear in the chat window on the right side of the screen (see Figure 4.7b). The current tag and


(a) Art Tracking module oberservation screen before measurements




(b) Art Tracking module, queries and responses

measurement are mentioned on the right side, and the corresponding measurements are on the left side. The message bubble will appear blue if the measurement is inside the set parameters. If the measured value is considered dangerous, the text bubble will appear red. Since the message display is programmed in an asynchronous way, it can happen that the answer to a query appears before the query itself if the queried tag is the same as the tag connected to the phone. This service can be stopped by pressing the *start service* button again or by exiting this screen in any way.

The measurement loop for the output is shown in Figure 4.8. 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 devices three measurements need tobe 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 disturb 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.

```
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

```
[06.02.25 15:28] dist:2-4.0:[1.214]:true
[06.02.25 15:28] dist:3-1.0:[0.942]:true
[06.02.25 15:29] dist:3-2.0:[0.356]:true
[06.02.25 15:29] dist:3-4.0:[1.078]:true
[06.02.25 15:30] dist:4-1.0:[0.633198]:true
[06.02.25 15:30] dist:4-2.0:[1.252325]:true
[06.02.25 15:30] dist:4-3.0:[1.088163]:true
[06.02.25 15:31] temp:1:[225.0]:true
[06.02.25 15:31] hum:1:[352.0]:true
[06.02.25 15:31] temp:2:[278.0]:false
[06.02.25 15:31] hum:2:[299.0]:true
[06.02.25 15:31] temp:3:[225.0]:true
```

Figure 4.10: Excerpt of a file saved by the Art Tracking Module

The query answers are appended to a file that is saved in the app storage. The information appended consists of: the queried tag, the returned values, a timestamp, and if the value was unproblematic. Figure 4.9 shows an excerpt of such a file. This functionality is intended for experimental evaluation. In a real-world 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 4.7b, it will open the Android native share functionality to share the file over mail, an installed messenger, save it to one drive or send

it over Bluetooth. For this Thesis, 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 sessions.

# 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 eperiments 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 confrim 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 refered 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.

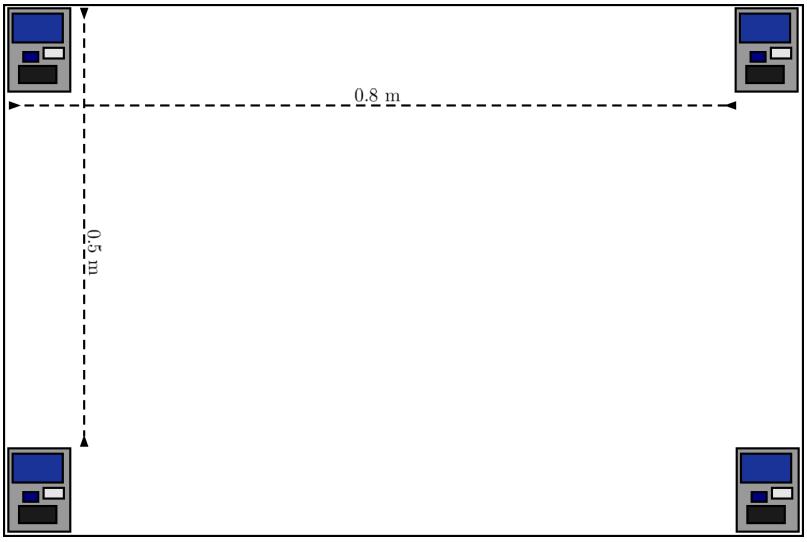



Figure 5.1: Schema of the setup of experiment 1.

### 5.1.1 Results

In eperiment 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 distinguische the lines, the Y axis only displayes the relevant section, rather then 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 varaince 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% .

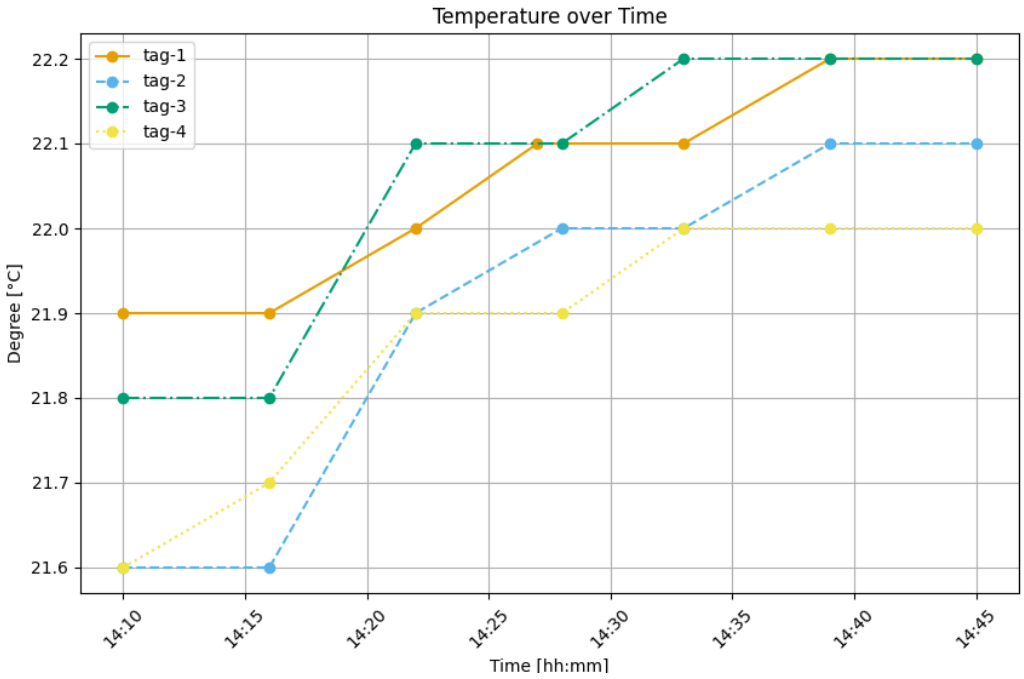



Figure 5.2: Experiment 1, temperature over time.

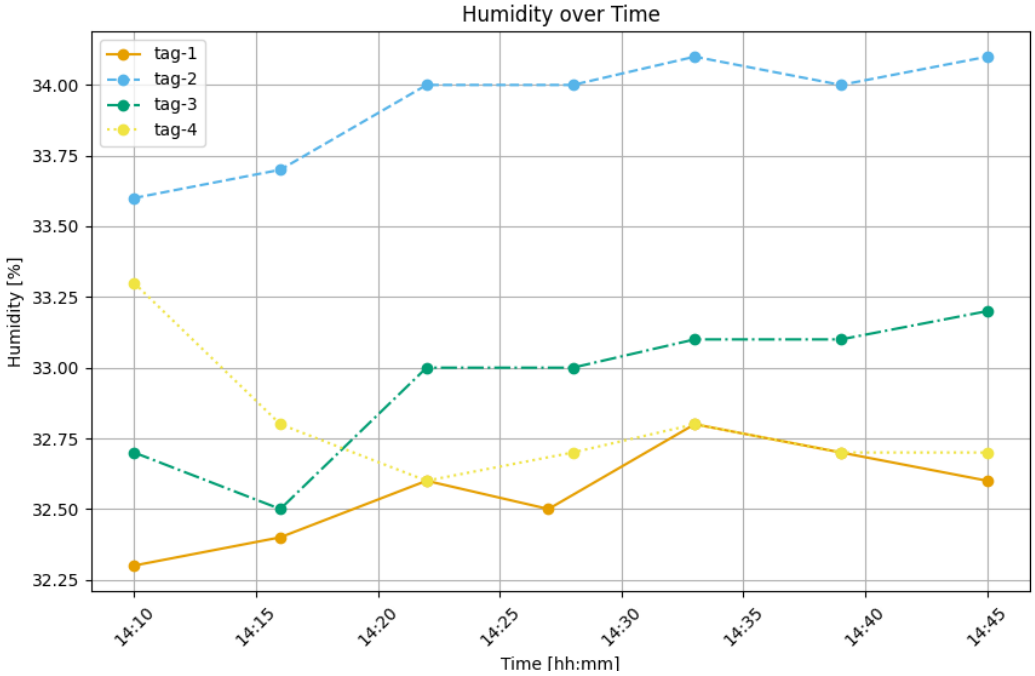



Figure 5.3: Experiment 1, humidty over time.

Te 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.

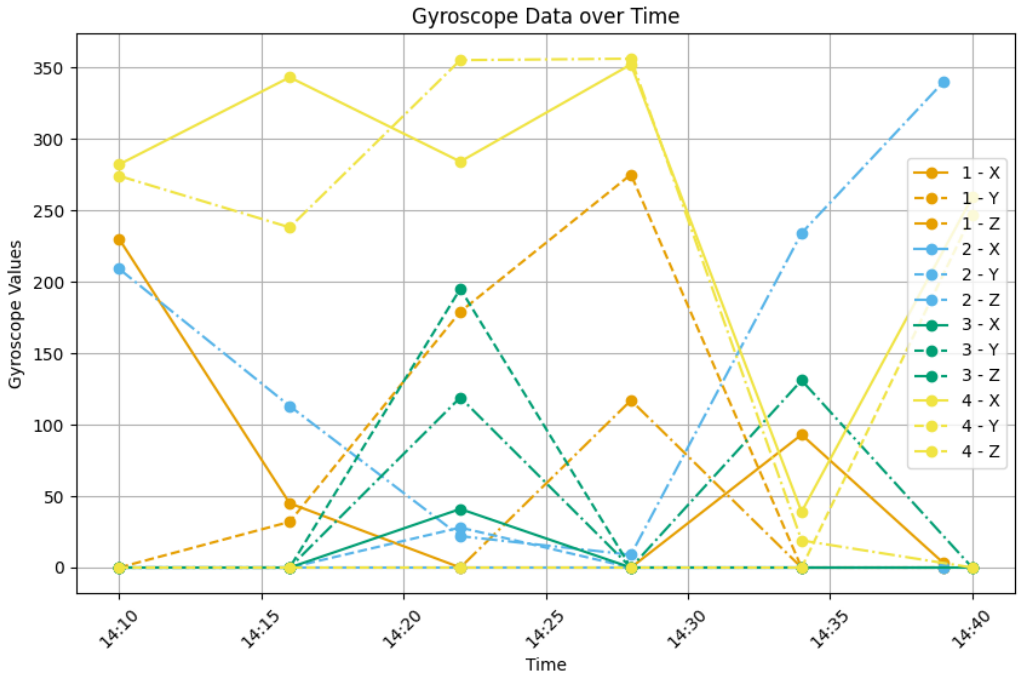



Figure 5.4: Registered value of the gyroscope over time during experiment 1.

Since all tags were stationary during the experiment, the gyro sensor was expected to be unchanging. This is not what occured. The graph 5.4 shows the values of the Gyroscope during experiment 1. Orientational read was used, so the values shoud 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 diplayed 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 tragetory. 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 "wrappes around" at 360°, means can only meanigfully be taken if the angles are in a small range. Since this is not the case for most tags, the only mean that is meaningfull is tag 2 axis-x, which has a mean of 0 and a variance 0.

Table 5.2, shows the mean, excpected 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 diagonaly oposed to each other, one can see that the measured distaances is the the same, indipendent 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 negibly small, beeing bellow 0.005m. This shows that the measured distances are constant and stable, escept for the measurement from Tag-3 to Tag-1. The distances measured do not correspond

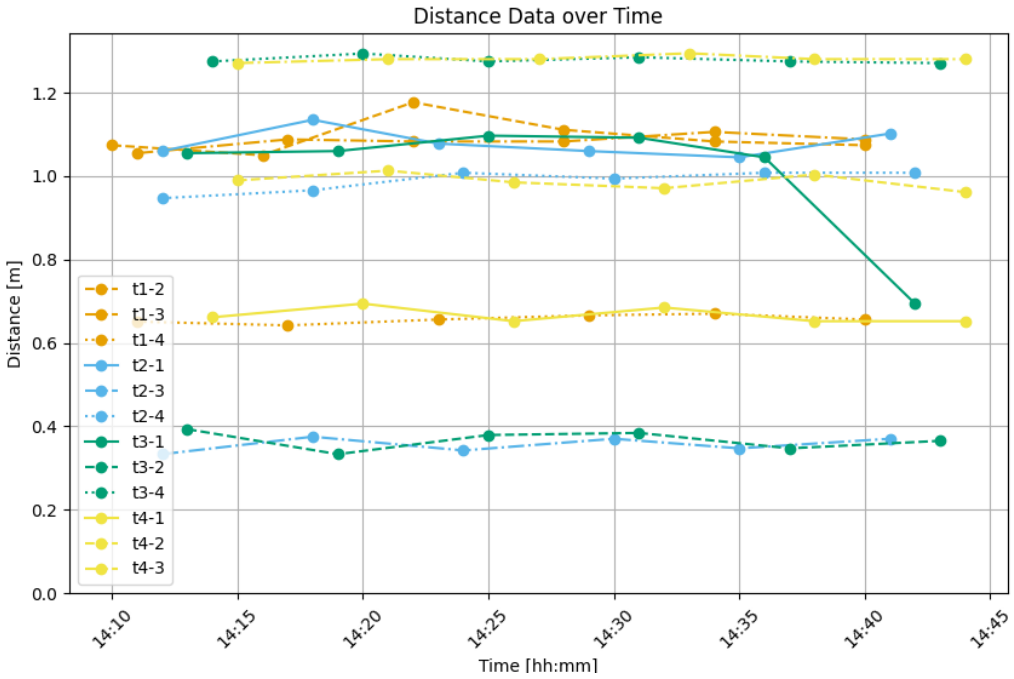



Figure 5.5: Experiment 1, distance over time.

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 meausrements 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 untl 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 anre 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, excpected values and variance of distant measurements, experiment 1.

| | **Tag-1** | **Tag-2** | **Tag-3** | **Tag-4** |
|---|---|---|---|---|
| **Tag-1** | 0.0 | 1.094 | 1.084 | 0.657 |
| **Tag-2** | 1.080 | 0.0 | 0.356 | 0.989 |
| **Tag-3** | 1.007 | 0.367 | 0.0 | 1.279 |
| **Tag-4** | 0.666 | 0.987 | 1.281 | 0.0 |

Mean

| | **Tag-1** | **Tag-2** | **Tag-3** | **Tag-4** |
|---|---|---|---|---|
| **Tag-1** | 0.0 | 0.8 | 0.94 | 0.5 |
| **Tag-2** | 0.8 | 0.0 | 0.5 | 0.94 |
| **Tag-3** | 0.94 | 0.5 | 0.0 | 0.8 |
| **Tag-4** | 0.5 | 0.94 | 0.8 | 0.0 |

Excpected 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 reseiving 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-1** | 0.112 | 0.265 | 0.177 |
| **Tag-2** | | 0.368 | 0.824 |
| **Tag-3** | | | 0.385 |

Mean

| | **Tag-2** | **Tag-3** | **Tag-4** |
|---|---|---|---|
| **Tag-1** | 0.001 | 0.013 | 0.000 |
| **Tag-2** | | 0.000 | 0.000 |
| **Tag-3** | | | 0.000 |

Variance

### 5.1.2 Discussion

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

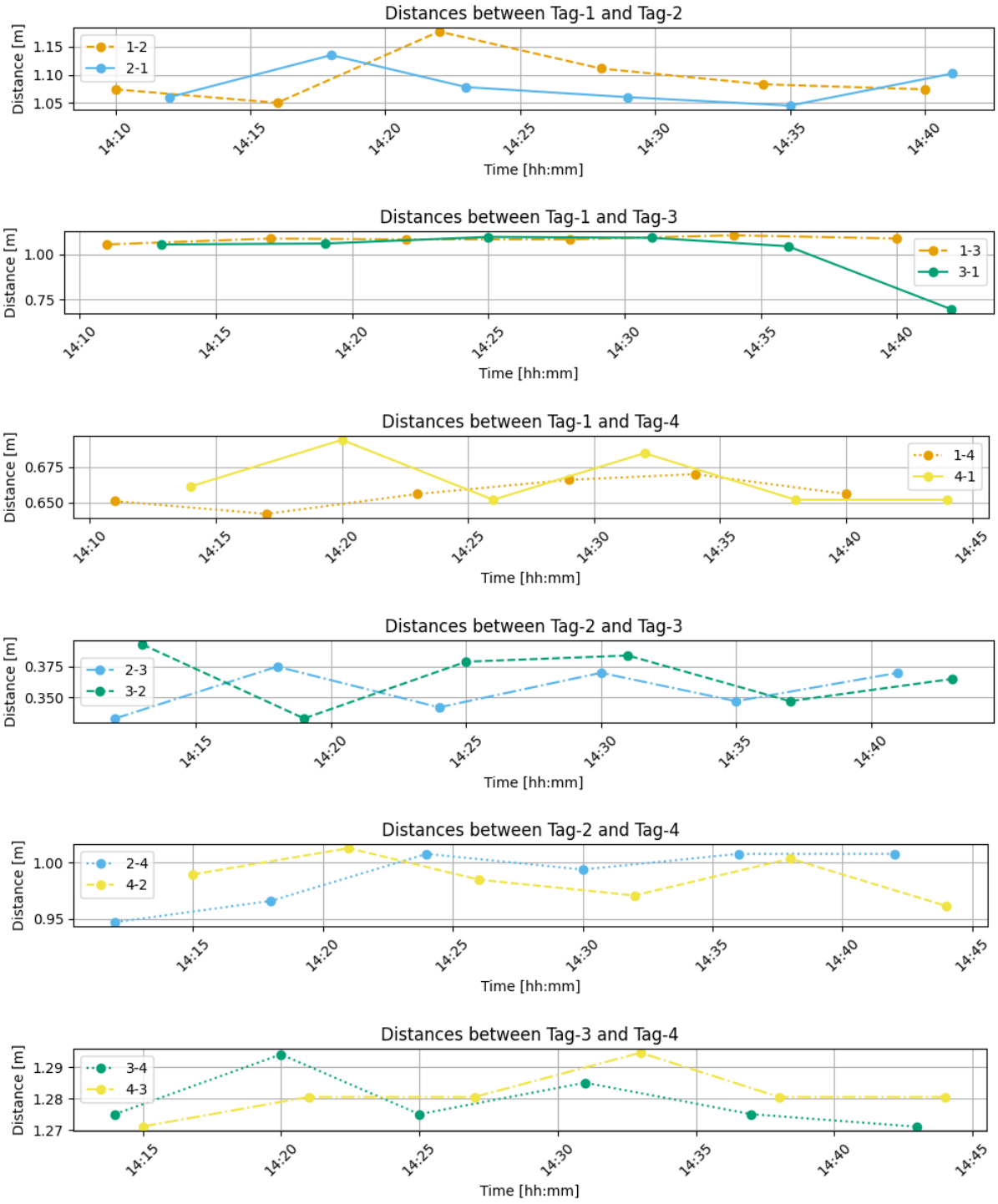



Figure 5.6: Experiment 1, distance over time, for all pairs i-j.

temperature measurement.
Two possible explenations were found for the increase in temperature during the experiment. One possible explenation 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 explenation is, that the microprocessors proucde heat that was detected. The fact that the temperature decreased

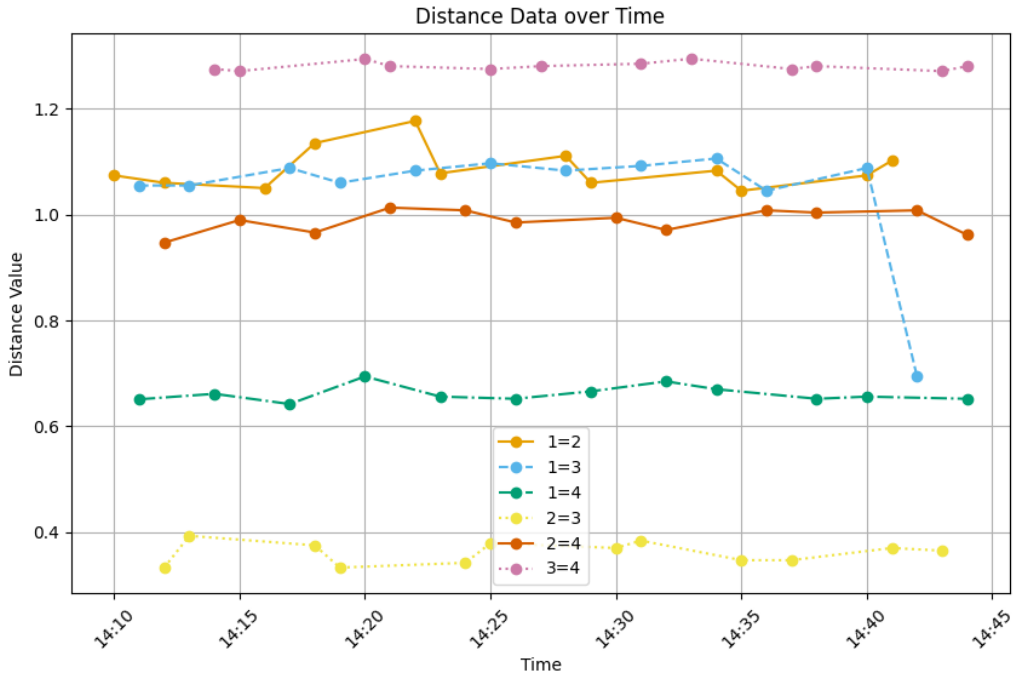



Figure 5.7: Experiment 1, distance over time, for comined pairs i=j.

during subsequent experiments, favors explenation one, since their would be no reason for the microcontrolers 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 dicrease in temperature was smaller, this difference in closness 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, beeing bellow 0.05The slight increase in humidity can again be explained by this beeing the first experiment of the day, and the person performing the experiment having wet hair from the rain. This again weakenes the microcontoler heat theory, since rising temperature without adding moisture would only decrease the humidity.
The dicrease in humidity in subsequent experiments lacks a clear explenation. It is a very weak trend, so factors global factors could explain the difference. Changing weather conditions could account for the difference. Another proposed explenation 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 clamy hands, that feasably could lead the sensors to detect additional humidity at the beginning of the experiment. Without additional data, no one explenation can be favored over the other. Since the decrease in temperature was small, this is not considered a issue for this system.

The Gyroscoping sensor data does not produce any meaningfull result. The measured orientation of the tags varied widely, while the physical tags stood still. A possible explenation 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 acumulate over time. The time between measurements was 5 minutes and

30 seconds. A bias of only $\frac{12}{11}\frac{°}{s}$ would correspond to an accumulated error of 360°over this timespan. Since rotational position in inheritly circular, rapping arround at 360 °, unless there was no variance next to the bias, the values would end sudo randomly scattered over the range of $[0°, 360°]$. The MPU6050 outpus only integers, so any bias at all would have this effect.
Be bias hypothoses is additionaly strengthend, by the existance 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 section5.5, Tag-2 axis X did produce meaningfull results. While this doesn't disproove, 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 acount for no more than $\frac{1}{240}\frac{°}{s}$ around the X or Y axis, if standing on the equator, where the effect is strongest.
The bias explenation seems to be a reasonable and explaines the measured results. As a consequence, the orientational read has to be considered usless.

The distance measurements have mixed results. The fact that the tag pairs produce the same same reults 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 occasionaly 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 envolving multi-path effect based on position can be rejected. The possible explenation envolves a fault on the nRF52840 microcontroler or the DMW3000 shield. Since the final calculation relies on the timestamps recorded during the ranging, a possible explenation 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 themself 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 a 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

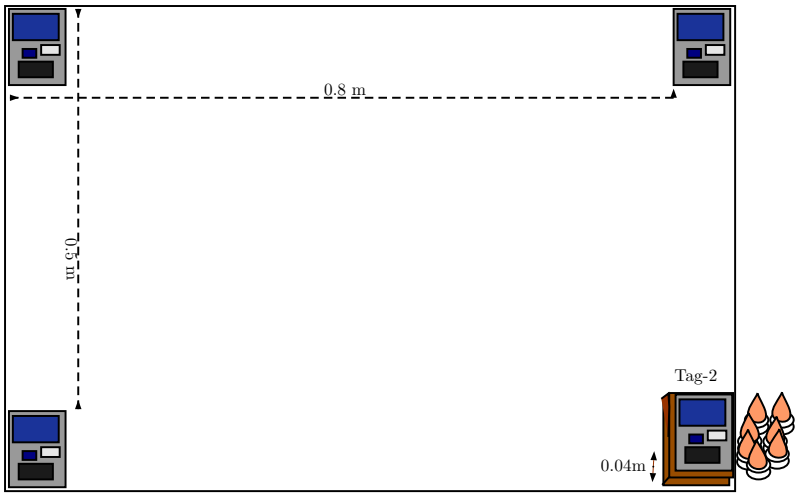



Figure 5.8: Schema of the setup of experiment 2.

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 comparesment. 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 measioned, one can assume it behaved the same as for experiment 1 (see section 5.1.1).

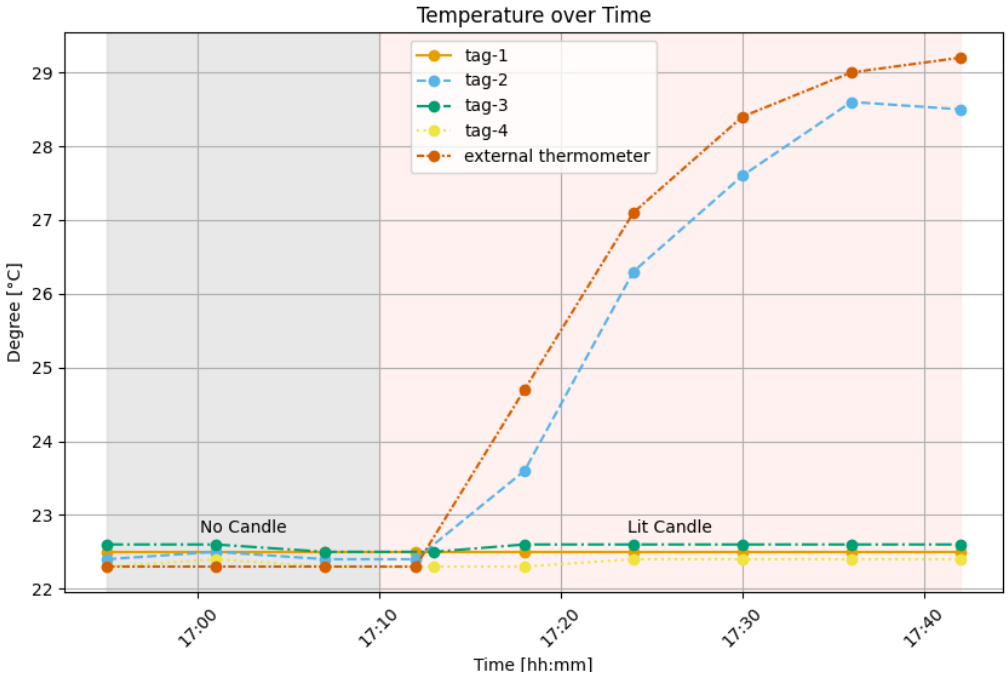



Figure 5.10: Experiment 3, temperature over time, mith external measurement added.

The progression of the external thermoeter 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 thermometor 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 thermoeter 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 klimbing 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 thermoeter nether exeeds 1 °C and gets smaller towards the end of the experiment, ending with a difference of 0.7 °C.

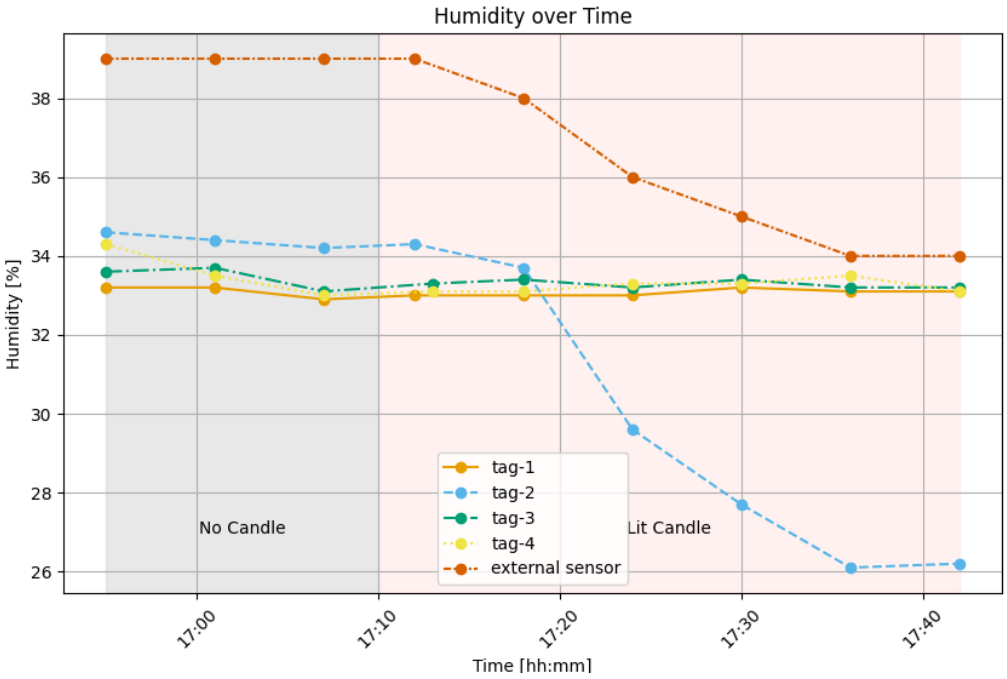



Figure 5.11: Experiment 3, humidity over time, with external measurement added.

Experiment 3 was intended to test the temperature and not the humidity. Luckily, the external thermoeter also included a humidity sensor, that could retroactivly 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 wasa added to the graph and is shown with a red dashed and dotted line. Since the external humidity sensor was initialy not intended to be used, it is not perticulalry precise and does not display any digits after the decimal point.
Its values were again manual 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 havin variances bellow 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 dicrease to 33.7 % , followed by a large drop to 29.6 %, then 22.7 % and finally platoing at 26.2 % .
The humidity sensor consitently 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 than first notes a small dicline of 1% pt, followed by a larger decline of 2% pt to 36 % and then decends with 1% pt at a

time until it platoes at 34% .
The humdity registered by Tag-2 and the external sensor forms a similar line. This two lines have a similar tragectory, but are not parallel. While the difference in registerd humidity originaly is around 4.6% pts, when both platoe, the difference has risten 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 Discussion

The fact that the three sensors that are far away from the sensor 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 nor rise, the energy added to the system would be added with an efficiency of $\mathbf{O}(d^{\frac{1}{3}})$, where d is the distance.
The tag that is close to the candle does notice the candle with a similar speed as the external thermoetor. 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 explenations. The first explenation is, that the external thermoetor and the DHT22 sensor are both not high precision instruments and have a natural inertia. The second explenation 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 obervation that the external thermoetor registeres a higher heat than the internal sensor has two possible explenations. 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, sine it was placed closer to the edge of the cardboard piece. Since initialy both sensor have very similar values, the second explenation is mire 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 startes 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 explenation. Since the humidity function is not the main purpose of the external thermometor, it is possible that it was implemented poorly, thus leading to the difference. Further research is required to analize the precision of the DHT22 sensors. Nethertheless, the sensor on Tag-2 shows clearly a happening thenomenon, indicating that the gerneal implementation and setup is soudn.

## 5.3 Experiment 3: Gyroscope

Again all for 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 orirentational read was originaly the only implementation for the gyroscope. After experiments one to four were evaluated, the lack of usefull results from the gyroscope readings prompted a redesign of the sensor. This resulted in the development and implementaiton 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 $\frac{deg}{s}$. The results for the orientational and the angular velocity read will be presented seperatly. The conclusion will talk about them both.

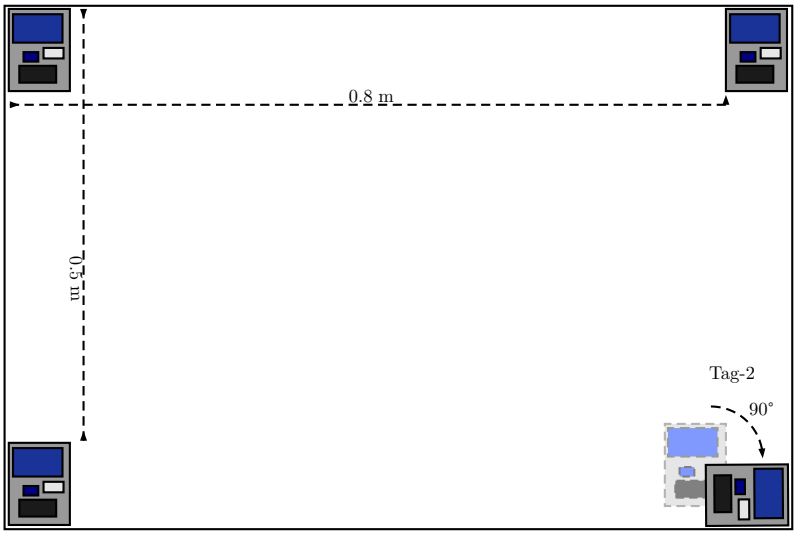



Figure 5.12: Schema of the setup of experiment 3.

### 5.3.1 Reults 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°at 22.25 around the Z axis. The gray section marks the park of the experiment beofre 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 disernable 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°respecivly This is escpecially unescpected, since Tag-2 was the one who was turned around axis Z.
Axis z of Tag-1 forms zig-tag 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.

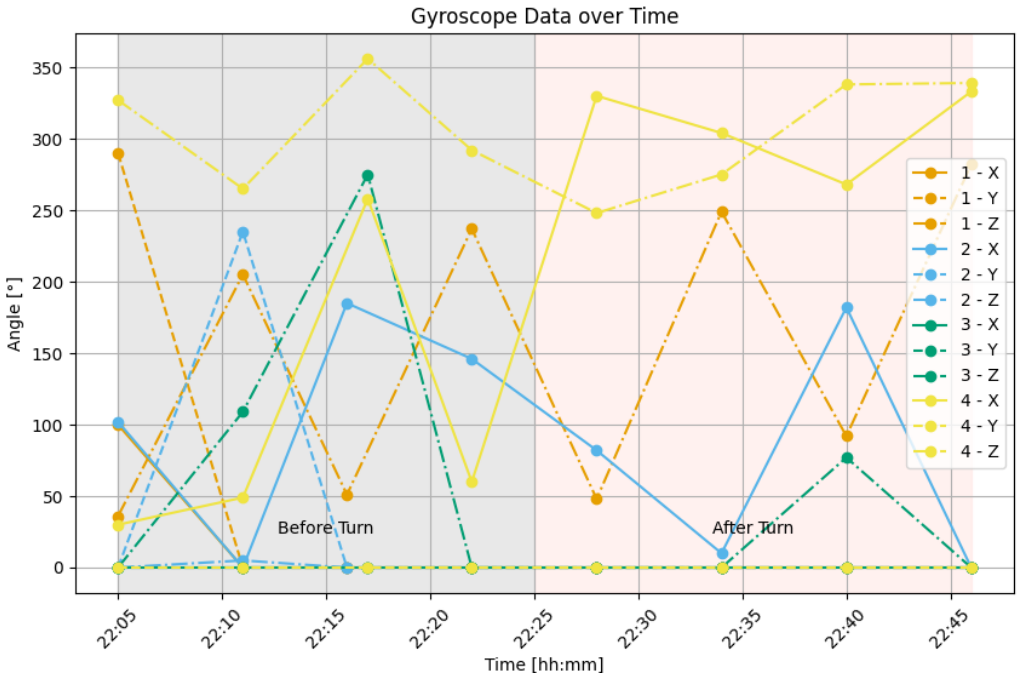



Figure 5.13: Experiment 4, gyroscope over time.

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 dicreases 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 originaly 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 siginificantly during this time, having variances of 0.001, 0.004 and 0.000 respectifly.

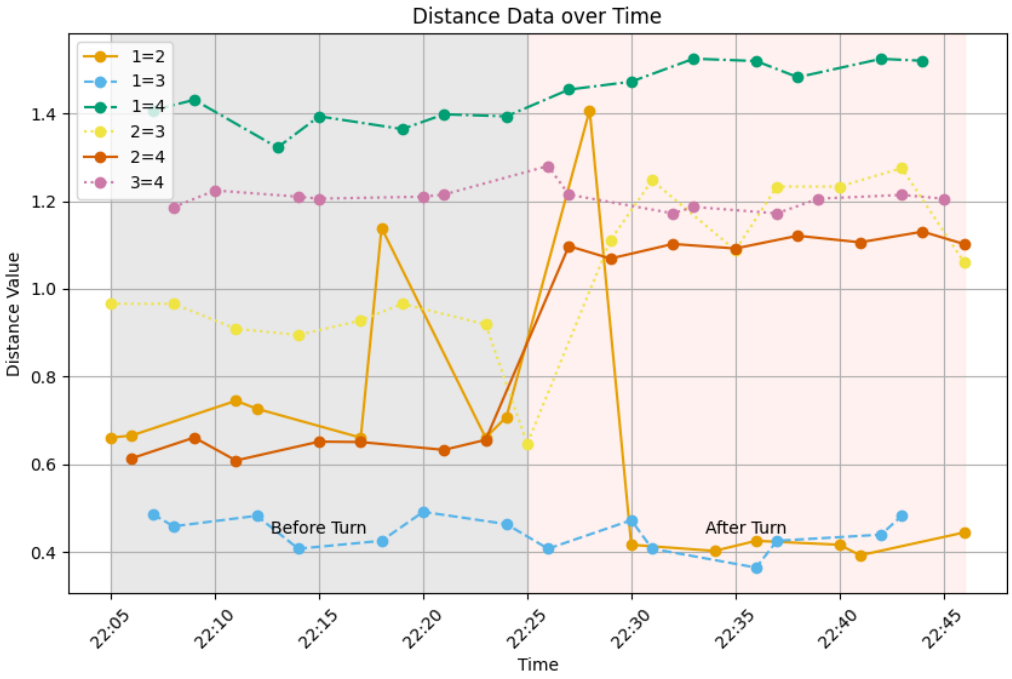



Figure 5.14: Experiment 3, gyroscope over time.

### 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 angualr velocity of all tags during the experiment. The angular velocity of tag i around axis v will be labeled as $a_i^v$. All axises of Tag-2 are shown in blue, with the angular velocity around the X axis, $a_1^X$ ,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 backgroudn, 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

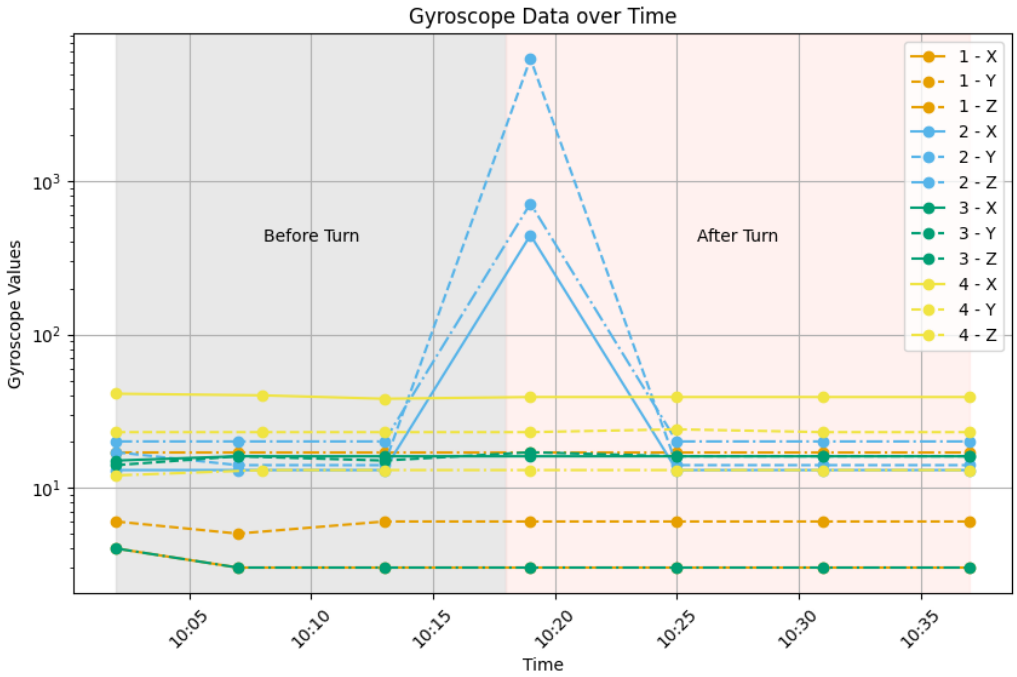



Figure 5.15: Experiment 3, gyroscope over time, using the angular velocity read.

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 ditances over time of experiment 3 using angualr 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 milimeter. 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 comperatavly 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.

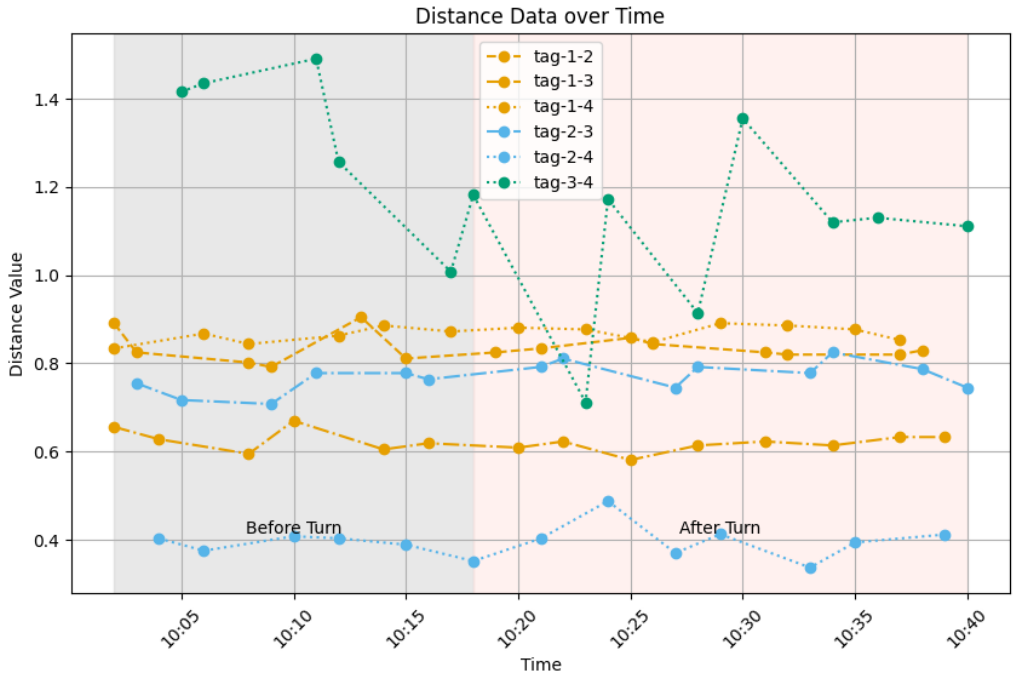



Figure 5.16: Experiment 3, combined distance over time, using angular velocity read.

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

| | 2 | 3 | 4 |
|---|---|---|---|
| 1 | 0.0.834 | 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

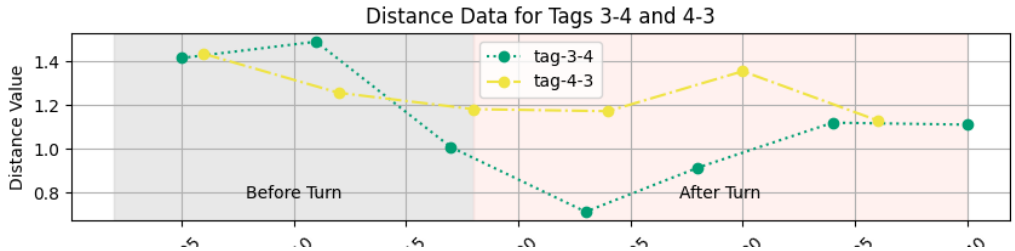



Figure 5.17: Experiment 3, split distance measurement between Tag-3 and Tag-4 over time.

### 5.3.3 Discussion

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 continously. 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, escpecialy for it to happen four times in one experiment. Additionaly, the one orientation that should change, because it was physically turned, remains at zero before and after the turn. The most plausible explenation for this behaviour is, that their is an error in the implementation of the gyrosscope 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 usefull results. The turn of tag 2 is very visible, while the other tags remain unaffected. The angular velocity aaround 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. Additionaly since the turn was performed by hand, it is reasonable to assume, that it was not performed perfectly around the X-axis. These to 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 atempt beeing 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 caputes something that actually happened in during the experiment.
Since the results of the oritentational read were already known during the angular velocity read experiment, a descision 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 baord. Additionaly the turn was centered around the DWM3000 shield instead of the MPU6050. This results in the turn not beeing visible in the distance reads from the experiment.
It is unclear why the distance measurements between Tag-3 and Tag-4 were inconsisten, when originating with Tag-3. Both tags were not part of the experiment. If a repeated multipath-effect would appear, it should affect both mesurments euqaly, 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 sequentilay, 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 schemativ 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.

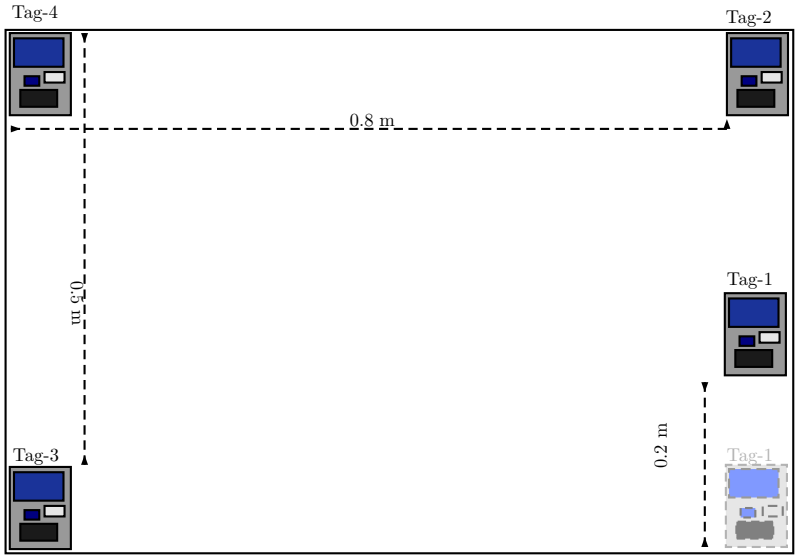



Figure 5.18: Schema of the setup of experiment 4.

### 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 dicreases. 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

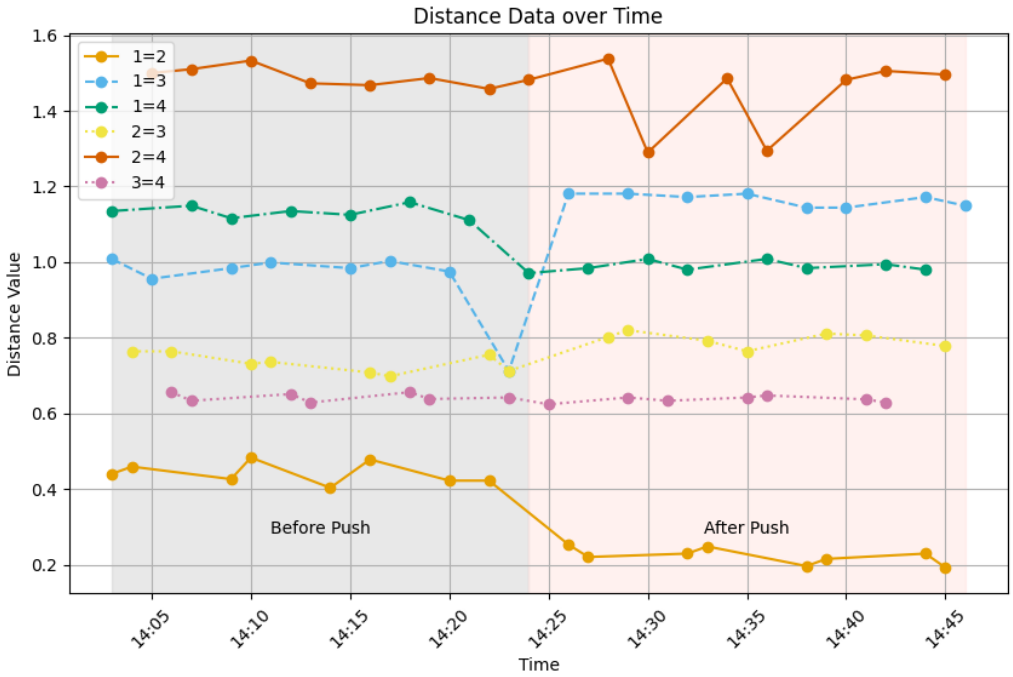



Figure 5.19: Experiment 4, pairs of distance over time.

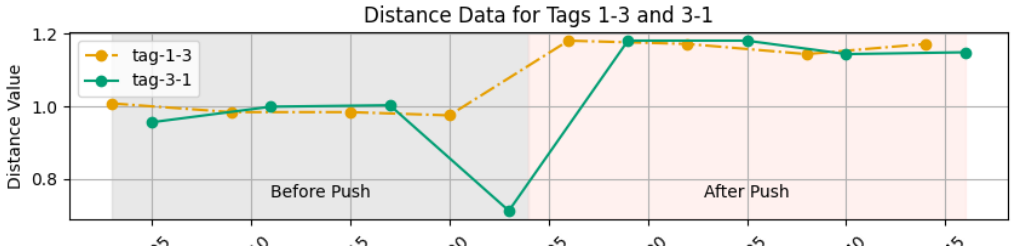



Figure 5.20: Experiment 4, distance between Tag-1 and Tag-3

experiment 1 and 3, the reported distances do not correspond to the what is phisically happending. 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-1** | 0.442 | 0.953 | 1.133 |
| **Tag-2** | | 0.734 | 1.490 |
| **Tag-3** | | | 0.654 |

Mean before move

| | **Tag-2** | **Tag-3** | **Tag-4** |
|---|---|---|---|
| **Tag-1** | 0.223 | 1.166 | 0.989 |
| **Tag-2** | | 0.796 | 1.447 |
| **Tag-3** | | | 0.635 |

Mean afer move

| | **Tag-2** | **Tag-3** | **Tag-4** |
|---|---|---|---|
| **Tag-1** | 0.000 | 0.010 | 0.000 |
| **Tag-2** | | 0.001 | 0.001 |
| **Tag-3** | | | 0.000 |

Variance before move

| | **Tag-2** | **Tag-3** | **Tag-4** |
|---|---|---|---|
| **Tag-1** | 0.000 | 0.000 | 0.000 |
| **Tag-2** | | 0.000 | 0.010 |
| **Tag-3** | | | 0.000 |

Variance afer move

### 5.4.2 Discussion

As was shown in Experiment 1, the distances do not match with the phsiscal distances. However, the differences in distance materialize correctly. The correct distances increase and dicrease 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 exeeds 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 a diamater of 0.12m 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 Binzmmühlstrasse 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 trinride in the S6 will have a red background. The time in Zurich Main Station is blue and the final trainride to Lenzburg green.

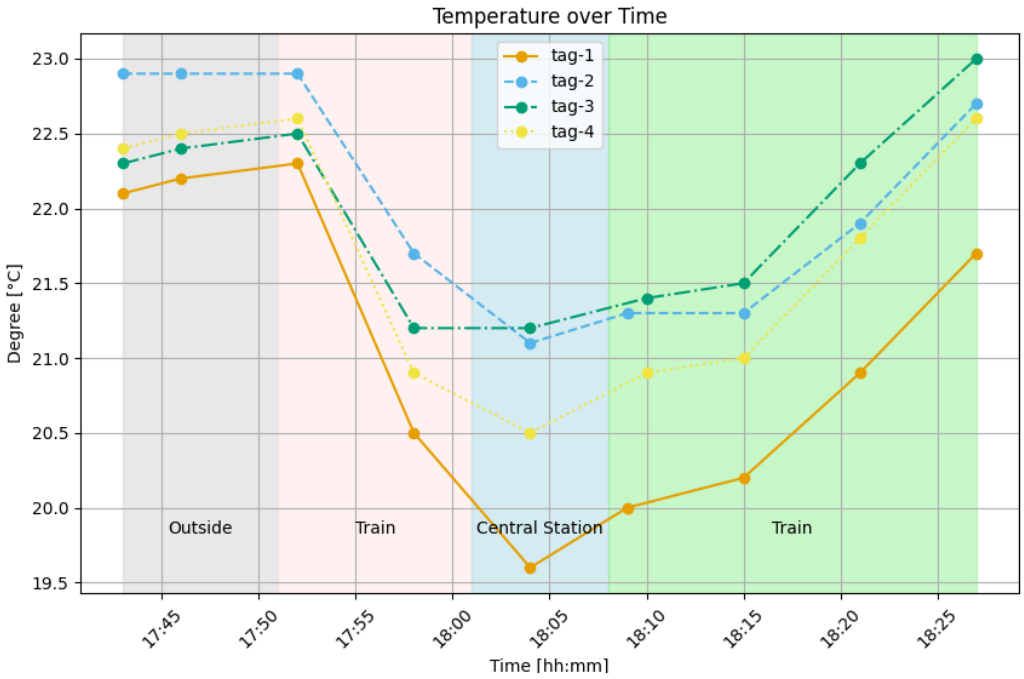



Figure 5.21: Temperature over time during experiment 5.

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, were the temperatures start drop during the second measurement in the train to by 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 lowset 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 stadely increasing temperature.
Tag-2 starts with the 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 alwys below it.

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

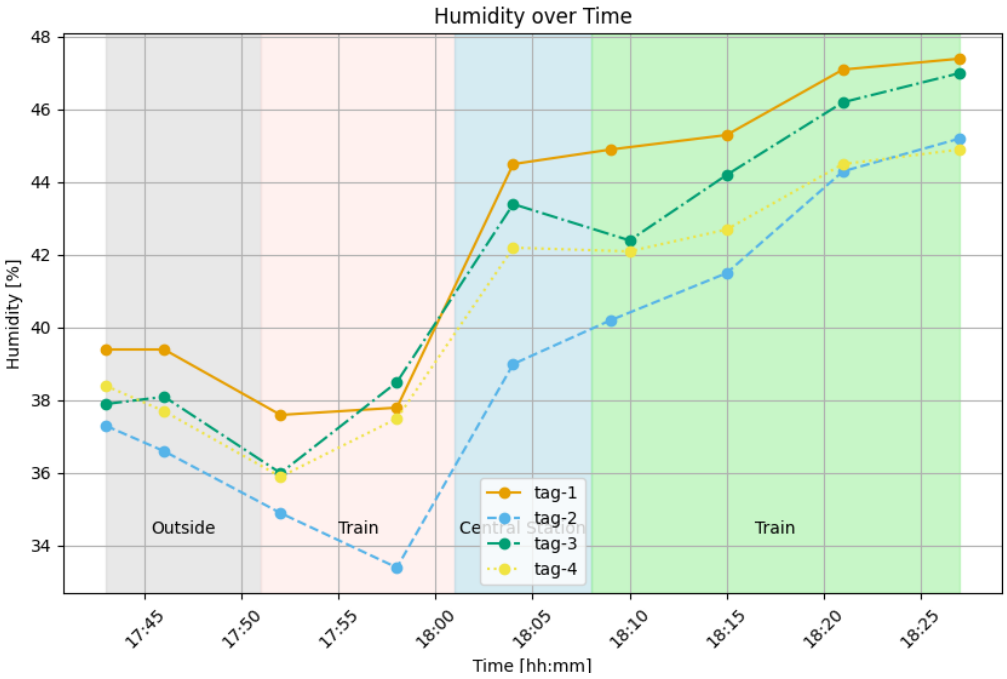



Figure 5.22: Humidity over time during experiment 5.

Figure 5.22 shows the humidity over time during the experiment. All tags start with a humidity measurement between and 37.0% and 39.5 %. The humdity then drops until the trainride and starts increasing again at the main station. On average it keeps increasing until the Lenzburg station is reached, except for Tag-3, which has a small drop during the first measurement in the train.
Tag-2 consistently reported the lowest humidity, beein at one point 4% lower than every other tag. Tag-1 for the most part has the highest Humidity, only once during the train ride recording a lower value then Tag-3. Tag-3 and Tag-4 have very similar values, until the second trin is reached. From there, Tag-3 starts recording higher humidity than Tag-4, ending with a difference of 2.1 % .

Figures 5.23, 5.25, 5.26 shows the measurement from the gyro module for the axes X, Y and Z. Angular velocoty reading was used during ths experiment. A log scale is used, to account for the wide range of measurements. Figure **??** shows a schemativ view of the direction of the Axes. Axis Z was pointed in the walking direction, Axis Y upward and Axis X towards the side.
Tag-2 has a missing gyro read. Tag-4 is also missing a value for the Z-Axis during the last measurement. These are the only example in all experiments, were values are missing.
On all three axes Tag-1 has a comparativly very low starting measurement and then starts, before having similar values to the other tags. On all Axes the starting measurements are the highest of the whole graph, if one takes the second measurement for Tag-1.
While the walk continues, the values on axis X and Y begin to decline, reaching a local minimum during the second measurement of the train-ride. The measurements taken in

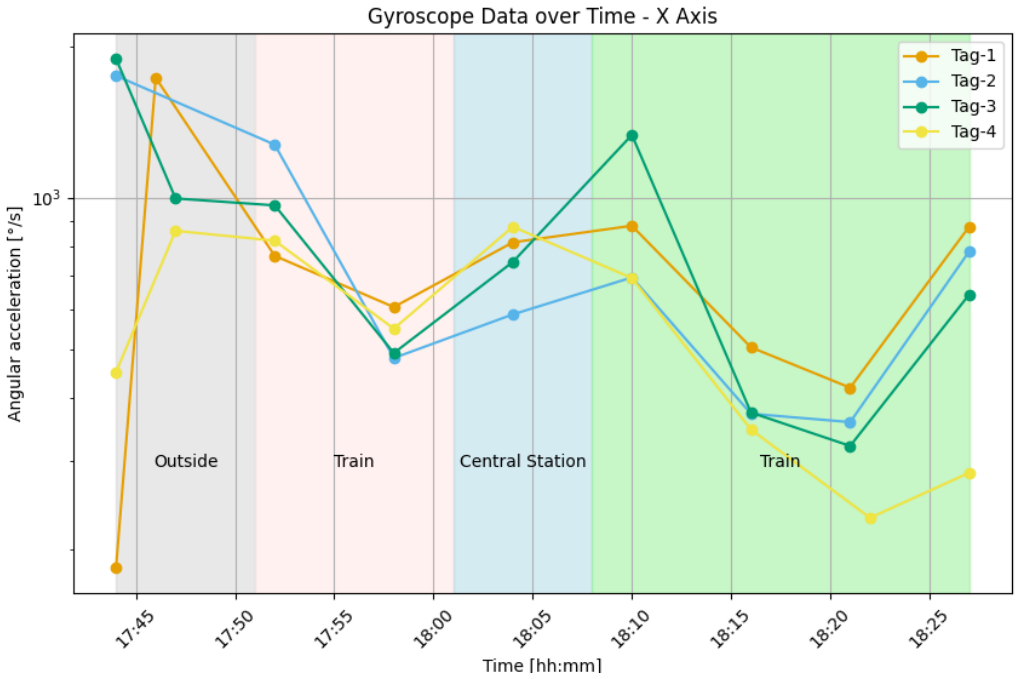



Figure 5.23: Gyroscope results using angular velocity reading over time during experiment 5.

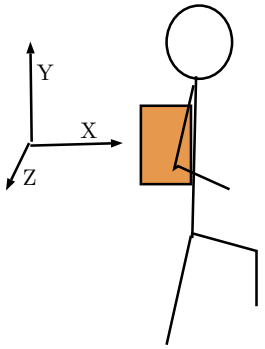



Figure 5.24: Direction of the axes of the gyroscope during experiment 5.

the central station are increasing again, reaching a local maxmima with the first reading during the second train-ride. The measurements two and three in the second trainride are lower than all other measurements for all tags. Afterwards the measurement increase again for Tag-1, Tag-2 and Tag-3. Tag-4 has notably low rotational values at the end of the experiment, the last measrement around the Y axis even beeing the second lowest value to be recorded during the hole experiment.

On the Z axis, Tag-2 has a second, even higher measured rotational velocity then the first one as its second measurement. The measurents of Tag-1, Tag-2 and Tag-3 for the Z-axis drop a lot faster then for the X and Y axis. For Tag-3 and Tag-4, the roatational velocoties also staed comparably steady between the entry of the end of the walk, the trainride and the main station and the beginning of the second train tide. The measurements take another noticable dip during the second and third measurement in the train, and the

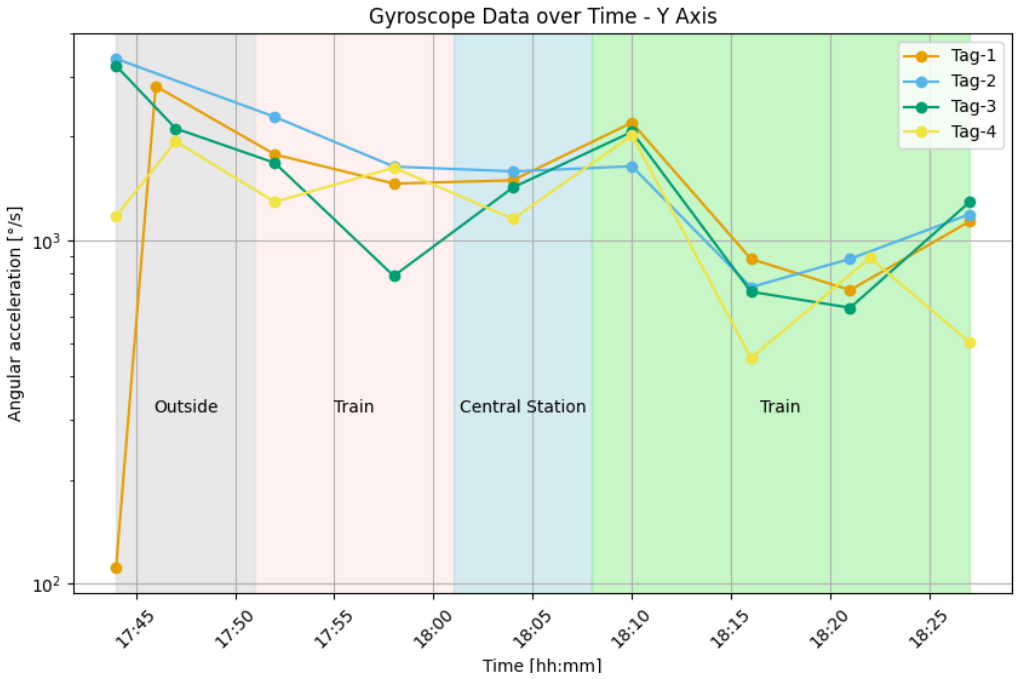



Figure 5.25: Gyroscope results using angular velocity reading over time during experiment 5.

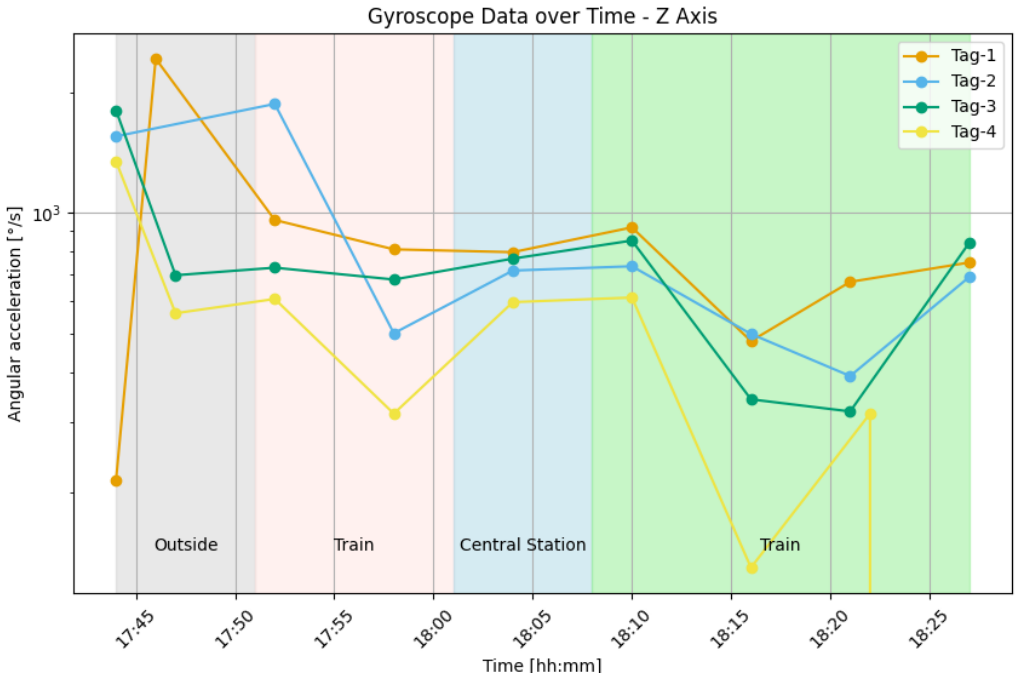



Figure 5.26: Gyroscope results using angular velocity reading over time during experiment 5.

going back up with the last measurement.

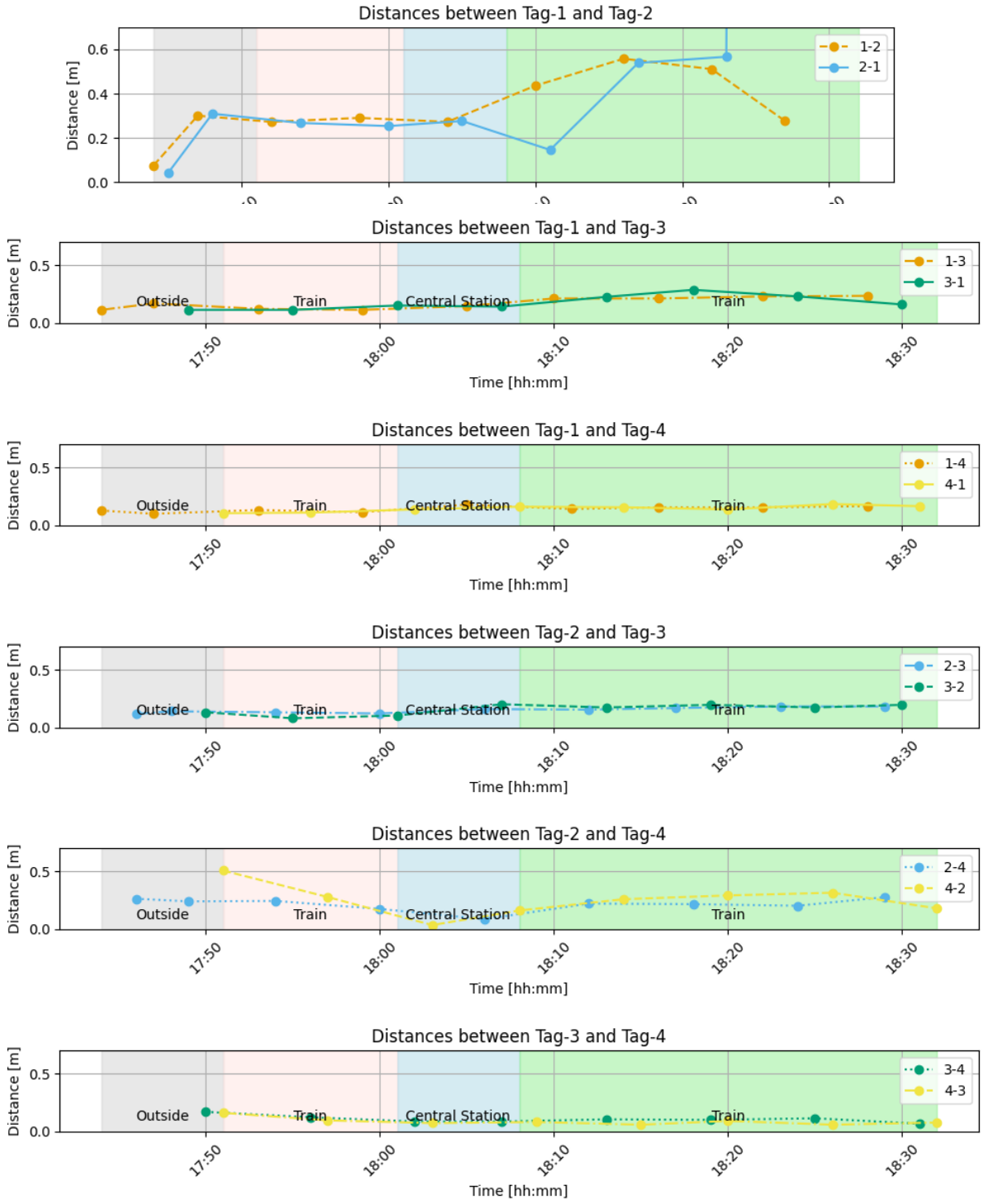



Figure 5.27: Experiment 5, distance over time, for all pairs i=j.

Figure 5.27 shows the pair of distance measurements that descibe the same distance. As in the other experiments, the measurement pairs measure the same values. An expetion is the last measurements of Tag-1 measuring the distance to Tag-2. it is 65.5m, which is more than 100 times the value of the next highest measurement. Table 5.7 shows the mean values and variance for the distance pairs. The variance of 1=2 is 235. This is mostl because of the extreme last measrement, but without it the variance would still be

0.025m. All other variances are very low, with distance 2=3 having the highest variance with 0.010m.
The mean value of distance 1=2 is 0.317m, if the large value at the end is ignored. This makes it still the largest, the others all beeing in the range of 0.09m and 0.18m. The lowest possible distance inside the rucksack would be two buckets were standing next to each other. This would lead to a distance of 0.12m. The highest possible distance is the distance between tow buckets diagonaly from each other, which would be 0.18m. Distances 1=3, 1=4 and 2=3 are possible, the others descibe impossible distances in this scenario.
As shown by the low variances, most distances remain static during the whole experiment. The only exception beeing 1=2. The first two measurements between Tag-1 and Tag-2 are very small, 0.075m and 0.042m. it then increases to 0.3m, where it stayes until the second train. During the second train it increases again to around 0.55m. The last two measurement are the impossibly high one of 2-1 and a drop back down to 3m by 1-2.

Table 5.7: Statistics of the combined distance measurements between tags for experiment 5

| | **Tag-2** | **Tag-3** | **Tag-4** |
|---|---|---|---|
| **Tag-1** | 3.93 | 0.174 | 0.141 |
| **Tag-2** | | 0.153 | 0.231 |
| **Tag-3** | | | 0.094 |

Mean

| | **Tag-2** | **Tag-3** | **Tag-4** |
|---|---|---|---|
| **Tag-1** | 235 | 0.003 | 0.001 |
| **Tag-2** | | 0.001 | 0.010 |
| **Tag-3** | | | 0.001 |

Variance

### 5.5.2 Discussion

Experiment 6 shows the acumulation of results that were seen before. Most results can be explained by chronologically going frew the events of the journey.

After the experiment was setup, the rucksack was shoulder and a building was left, walking towards the Zurich Oerlikon train station.
The original warm temperature of the building can be seen in the results of the temperature measurements. Sice warm air was caputed in the bag and the back of the person carying the bag also warmed the baf, it took a while for the air in the rucksack to cool down. The same goes for the higher humidity of the building, that set a higherstarting humidity than what was present outside.
The shouldering of the rucksack can be seen in the high rotational speeds, captured by the gyroscope. A rotation arround axis Z would imply movement similar to a sommersault. Since the person performning the experiment was avoiding such acrobatic feats, the rotation around the Z axis never reached high values again.
The initial increase in distance between Tag-1 and Tag-2 might also be explained by a initial movment of the tags while shouldering the bag. This is unlikely, since all other distances were unaffected and it is unlikely that a tag changed position in such a way, that it only affected one distance, though theoretically possible.
Tag-1 is the first tag to be quieried for its gyroscopic values. The fact that the rotational

speed of Tag-1 is so low at the beginning is probably because the measurement was taken before the rucksack was shouldered. This would also explain the high rotational speed captured in the next measurement, this now capturing the movment of putting on the rucksack.

The walk to the train-station involved turing a lot of corners. These would turn up as roatations around the Y axis, which we can see in Figure 5.25. The gyroscope captures the maximal rotational velocity since the last measurement. As a consequence, the first measurement in the train is still high, from the walk to the trainstation. The second value then is lower, since standing in a train involves less rotation. Since the person who performed the experiment stood in the door section of the train and had to move out of the way of some people, there is stiall a higher gyroscope measurement captured in the train.
The results around axis X capture movement that corresponds to leaning. This also happens while walking, so the results mirror the results from axis Y, but in a less pronounced way.

The initial warmth from the building was starting to wear of during the train-ride. It is unclear if this is because of the fresh air entering frew the door at stations, or because the train was not heated a lot. While the temperature droped, there was not much exchange with the air around the bag. The falling temperature increased the humidity in the bag, in a reverse effect of what was seen in experiment 2 5.2. This continued during the time in the main-station, which is also not heated.

At the main station, the person taking the experiment originaly took it slow and then suddendly had to rush to catch the next train. This explaines why the rotational speeds for X and Y are steady, but not to extreme and then peak on the first measurement in the train. This measurement again caputring the second part of the stay at the Central Station, which envoled rushing to the train.

Since the experiment was done during roush hour, the train was quite full and no seat was found. The carrier of the rucksack had to stand togehter with many other people until reaching Lenzburg. This was different than the S6 to the main station, which was surplisingly not that full.
This is seen in the temperature graph. Many people standing close together will produce a lot of heat, that was enventualy increasing the temperature inside the bag.
One can also see the lack of movement during this time in the gyroscope data. The rotational velocity is the lowest it has been during all of the experiment, since this is the first occurens ot the rucksack carrier actually standing still.

When reaching the final destination, the train was left. This again involved a lot of turning and climbing down stairs, which can be seen in the increased roational speed, captured by the gyroscope. The experiment was ended shortly after leaving the train, not giving enough time for the temperature to drop again.

Why Tag-4 captured lower temperature values is unclear. It is possible that the air around it was colder, due to an opening. But then one would also expect the temperature to ride faster in the crowded train at the end. Another explenation is a problem with the DHT22 sensor. The sensor did not have any problems since, so this explenation also seems unlikly.

The difference between the other tags can be explained more easely. could have a higher inital heat for a varity of reasons. The building were the experiment was prepared had a bunch of runnign computers. If the bucket containing Tag-2 was near one of them, it would have been warmer than the other buckets. Tag-3was probably at a more exposed part of the bag. it react faster to the change in temperature than the other tags.

It is unclear, why the humidity kept rising, while in the second train. Since the temperature was increasing, the expectation would have been for the humidity to drop. It is possible, that the humidity produced by many people standing close to each other would also affect the inside of the bag. How humidity behaves is not as obvious as temperature, so it is complicated to tell, if this is a problem for this experiment.
The humidity during the static experiment was spread around a range of 2 % pts, so the fact that a similar spread exists here seems to indicate, that this is how precice DHT22 is in sensing humidity.

The missing values probably indicates, that a query has not reached the Tag. This was to be expected to happen sometimes and it is reasuring that it only happend two times over 15 experiments. It is also a good sign that the system can handle a missing measurement, without further issues.

The constant and low values of the distance measurements show a realistic depicition of what happened inside the backpack, since the tags could not move much in there. The values again do not correspond to realistiv values, this probably beaning a consequence of the flawd calibration method used.
There is no working theory of the measurements behaviour of the measurements 1=2. They do involve Tag-1 again, which is proune to faulty behaviour. But a mistake of 10000% exeeds previous errors by a lot. The higher captured distances during the second train ride can also not be exaplined by beeing outliers, they are to consistent. A slight rearangement, inside the rucksack leading to an unfortuned multipath effect could explain this.

Overall, experiment 5 shows, that the implemented system is sufficient for tracking during a real-world use. It is however noticable that it was designed for a system with fewer changing parameter, and fails to capture the samll details of a journey with changes that occur in a smaller than five minute time frame.

## 5.6 Challagnes and Limitations

The experiments showed that the two-way ranging implementation used does not provide precice results. A probable reason for this is the lack op propper calibration, using the method published by Qorvo [**?**]. This would envolve establishing a synchronized clock between the tags and building a star-topology network using UART USB connections with a central computer. Buildings this calibration systeam was out of scope for this thesis, but would have presumably improved distance measurement accuracy.

Tag-1 frequently provided outliers in while ranging. In sensor networks it has to be expected that some evaluations are not wrong. The design presented in this thesis does

not contain means to correct these readings, but rather relies on the driver to interpret them correctly. An alternative design could detect outlier readings on the phone by comparing to previous results. By using the data reported by the tags surrounding the tag with the outlier, the phone could determine the feasaility of the result and optionaly discard it based on the evaluation. Such a system would be enhanced if the positioning of the tags works, since it would allow for clearer results. It would also have to be calibrated quite carefully, since discarding a outlier that is reporting an actual problem would defeat the purpose of the system. But it would take a task away from the driver, who should not be overwhelmed with data while driving.

The avaliability of only four tags for experiment limited the parts of the system that could be tested. Specifically the building of a working model of the position positions using a quadratic program was not possible using only four tags. For this reason this design aspect stayed theoretical for now. The implementation would require the network descibed in section 3 to be used, including its network joining system. Additionaly, the app would need to be extended, to build and solve the quadratic program. The results would need to be graphicaly visualized and displayed to the user.

No-one involved in this thesis owns a truck or is allowed to drive it in switzerland. The system was only tested in laboratory environments, walking and on a train, but never in a truck. It would be interesting to see how a experiment in a truck would differ from the experiments performed on the train.

The experiment involving the train showed, that the gyroscope can sense small movements, when using the rotational velocity read. By including an accelerometer, a vibration detector could be developed, similar to the work done by [**?**]. This could lead to an additional measurement beeing captured by the system.

# Chapter 6

# Final Considerations

This chapter provides a summary of this thesis 6.1, a conclusion of what was achieved 6.2 and it lays out future work that could be build on this thesis 6.3. The goal of this is to provide a wrap-up of this thesis.

## 6.1 Summary

This thesis introduced a design for a wireless sensor network (WSN) to monitor artworks during transportation and inform the driver about the status of his load. This involved the tag beeing equiped with temperature and humidity sensors and a gyroscope, as well as peripherals for UWB and BLE communications. The tags would build a peer to peer network using UWB, that a phone could connect to using BLE. The phone would run an app that queries the WSN for measurements and presents the result to the driver. Based on parameters provided by the driver, the app would warn the driver about measurements that indicate a problem. Next to sensor data, the provided system also included distance data between the tags, gathered by using UWB two-way ranging. The App builds a working model of the positions of the tags based on this data.

In a first step, theoretical knowledge about relevant field was gathered. This thesis provides a sumary on the relevant parts in the fields of: Wireless sensory Networks, Ultra wideband, Two-Way ranging and k-connected graphs (TODO: Liste erweitern wenn mer dazu kommt!). Additionaly the current state of research in related fields was presented, consisting of: Artwork Tracking, sensor networks and wireless ranging.

The system design includes a description of the hardware used, including a breakdown of the tag components. It descirbes the way all hardware components communicate with each other and how to connect them correctly. It then provides the architecture of the system, describing the different modules used and what theor individual services and responsibilities are. A detailed description of how data flows through the system was provided. Lastly the network architecture was provided, describing how tag join and communicate. The thsis showed how this implementation ensures, that a unique model

of the tag positions in the system can be calculated, and how this model is calculated efficiently.

The implementation followed the outlined system design, using a nRF52840 microcontroler, the DHT22 temperature and humidity sensor, the MPU6050 gyroscope, DWT3000 UWB shield and an Adnroid phone. The implementation descibes how each sensor was initiated and operated. It describes how the UWB network was build, how UWB two-way ranging was implemented and how a BLE connection is established. It shows how all parts of the system are coordinated and the steps needed to run them all on the same system. Lastly the process of developing the app and how it is operated by the user is described.

The system was evaluated, by performing five experiments and evaluating the gathered results. The first experiment tested a static system, with no changes to the tags. The second introduced heat to one of the tags, testing if the workings of the heat and humidity sensors. The third tested the functionality of the gyroscope and involved turning one the the tags around itself after a period of rest. The fourth experiment tested the ranging capabilities of the tag and consistet of moving the tag around. The fith ecperiment checked how the system would behave outisde of the laboratory environment. The tags were taken on a journey that envolved walking and taking trains.

## 6.2 Conclusions

A comprehensive exploration of the current standing of sensory networks for the monitoring of artworks has been conducted, giving insight into the use of IoT devices and wireless sensory networks in this field. A design was presented that would allow for the tracking of artworks during transportation in a truck, using different sensors and distance measurements. The provided design was sucesfully implemented and tested.

The testing showed that the developed system is capable of querying data from the tags and reporting it. The system can run for an extended amount of time unsuperwised and reports problematic data correctly to the user via phone.
The results from temperature and humidity measurements showed, that these measurements are working in the systems as intended. The temperature data showed readings that were consistent with other devices and were consisten betwen the tags with up to one degree. The humidity readings were mostly consosten between the tags but differed from external sensors by a larger margin.
The reading of the gyroscope had to be implemented two times. An atempt to track the current orientation of the tag using the gyroscope readings showed inconsistent data. Tracking the maximal angular velocity captured since the last measurement was fruitfull and provided usefull insights into the movement of the tag.
The distance measurements showed readings that did not correspond to the distances in the physical world. Nethertheless, the readings were consistent and the difference in distance between event mirrored what happend en the event. The distance measurements are currently not precise enough to build a tag positioning systems, but they can be used to detect unwanted movement between the tags.

Overall the design was sucesfull and shows that a artwork tracking system using these fundamentals could be used. The selection of the sensors that are most sutable for the system still needs to be developed. The distance measurement needs to be improved. These are both potental research topics for future work.

## 6.3 Future Work

The implementation used in this thesis was intended as a proof of concept and therefore does not implement a fully functional art-tracking system. A focus was put on different types of sensors and how they would interact and report to the system. Therefore the focus was not on choosing the optimal sensors for an art-tracking system. Future work could focus on the types of sensors used and assure they are optimaly chosen to capture all aspects that are relevant during art-preservation. This research could include, but is not limited to:

- The inclusion of a light-tracking sensor
- Compare different humidity and temperature sensors.
- Use gyroscopic data and ascelerometers to detect problematic vibration during travel, similar to [11]

The distance measrement and evaluation is not fully actulaized in this implementation. Future research could improve on the implementation by developing a better calibration method, then used in this implementation. This would presumably involve building the calibration system proposed by Qorovo.
With better distance measurement, future researchers could implement the positioning model proposed in the design section of this thesis. This would include small ajustment to the network building of the tags, and adding the functionality to the phone to build and solve the quadratic program.
A better distance measurement would also allow for a better handeling of outliers and currupted data. As described in the limitations section of the evaluation 5.6, the measurements from nearby nodes could be used to handle this kind of behaviour.

The experiments were performed with only four tags and without access to an actual truck. It would be interesting to test the implementation on a larger scale, envolving more tags and differing modes of transportation. Research like that could investigate, if the design is well suited for real world use.

Finaly the system could be addapted for systems other than trucks. The general design of the wireless sensor network could remain the same, even when transporting on a train or airplane. The communication with the phone would need to be adapted. BLE does not have infinite range and the presence of solid barriers shortens its distance even more, especialy metal ones. So a new design to communicate between the respoinsible person and the sensor network would be required.

# 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

# List of Tables

# Listings

# Appendix A

# Contents of the Repository

The code repository contains the following content:

**Installation**

**Operation**