Marvin Rantschl

Feasibility of UWB-based Localization for Industrial Applications

Bachelor's Thesis

to achieve the university degree of

Bachelor of Science

Bachelor's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Dipl.-Ing. Maximilian Peter Schuh
Assoc.Prof. Dr.techn. Carlo Alberto Boano

Institute of Technical Informatics
Head: Univ.-Prof. Dipl-Inform. Dr.sc.ETH Kay Uwe Römer

Graz, April 2021

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used.

Date	Signature

Abstract

Industry 4.0 and the development towards smart factories bring new technologies to the industry. Within smart factories, the focus on a decentralized production and the concept of the Internet of Things (IoT) allow more agile processes. To guarantee a seamless interaction between humans and robots, precise indoor localization has become an important feature. Therefore, this thesis investigates the feasibility of an Ultra-WideBand (UWB) technology in an industrial environment. To integrate such systems in the environment, an Android application was developed. The app processes and outputs location data provided by the UWB system. Additional features help to track the workflow of industrial productions and optimize safety aspects of working areas. Furthermore, the accuracy and the scalability of the system were observed as well. For accuracy, the locations of multiple tags were tracked. Thereby, an average accuracy of 18.3 cm were measured. Also, differences in both accuracy and precision were detected depending on the placement of the tags. With the evaluation of the system scalability, its limitations and maximum capacity were identified. Multiple measurement series were compared in order to visualize the utilization with different amounts of tags and update rates. The performance of the application was tested during a schooling in the LEAD factory at university. During the schooling, the workflow of an industrial assembly run was tracked and observed. It was possible to prove the application and the used UWB system feasible for the evaluated use case.

Contents

List of Figures

List of Tables

1 Introduction

Globalization has led Industry 4.0 grew in importance throughout recent years. It embraces the use of new communication and information technologies as well as the integration of cyber-physical systems in the manufacturing industry [14]. In addition to Industry 4.0, the term 'smart factory' has become quite prominent. These factories are described as the vision of Industry 4.0 comprising the fields of action versatility, resource efficiency, ergonomics and usability [8]. The paradigm shift from centralized to decentralized productions incorporate the concept of the Internet of Things (IoT) into smart factories. The idea is that all humans, machines, tools and products in a factory are connected to the internet and share information with each other [18]. This causes indoor localization to become an important feature towards the development of smart factories for several reasons. On the one hand, moving objects within an industrial environment allow more agile processes. Therefore, precise and real-time position tracking of all operators involved is required [18]. On the other hand, the tracking of workers and resources allows the investigation of the workflow during an industrial process, which has the potential to vastly increase the productivity.

The rise of Ultra-WideBand (UWB) technology enables real-time localization with centimeter-level precision. This promising technology brings certain benefits suitable for adapting it to industrial use cases.

1.1 Problem Statement

There already exists a variety of UWB-based location systems. However, the application aspect and feasibility of these solutions has not yet been investigated sufficiently.

Due to the complexity of smart factories, humans need to be supported to fully understand and control the complete production plant. Since these factories aim for a closer interaction between humans and robots, they need smart human machine interfaces. Their purpose is to reduce the chance of errors and help to make the right decisions [8]. The supply of such a tool is necessary to guarantee an efficient workflow. Additionally, it also influences the safety of the workplace. While agile processes may lead to an increased productivity, they also bear the risk of potentially severe injuries. The location awareness of workers and robots gives warnings in case of dangerous situations and improves the security of future productions.

1.2 Contribution

This thesis is part of a cooperation between the Institute of Technical Informatics (ITI) and the Institute of Innovation and Industrial Management (IIM). While the ITI considers the technical aspect of a UWB-based localization technology, the IIM provides access to an industrial environment and the opportunity to test the system in a real case scenario. The thesis is thereby going to establish a link between the technical perspective of UWB-based localization and its adaptation for an industrial use case. For this purpose, the used UWB localization system, which is Decawave's DRTLS, is set up. In addition, an Android application is developed as an interface to visualize localization data for an industrial end user. Furthermore, the performance of the system and the developed application is evaluated.

1.3 Outline

Chapter 2 provides a technical overview of UWB, common ranging approaches and the MQTT protocol. Moreover, Decawave's DRTLS is described. Chapter 3 lists the requirements of the application and further describes the implemented app features. In chapter 4, the accuracy and scalability of the localization system were evaluated. This chapter also presents the observations of the first test run of the application during a schooling in IIM's LEAD factory. Finally, chapter 5 summarizes the results and gives an outlook on potential future work.

2 Background

This chapter provides information about the technologies that were used for this thesis. It first covers the technical background of Ultra-WideBand. This includes a describtion of its characteristics (see Section 2.1), common ranging approaches (see Section 2.2) and Two-Way Ranging (see Section 2.3) in greater detail. Subsequently, the MQTT protocol is described in Section 2.4. Finally, this chapter introduces Decawave's localization system (DRTLS) in Section 2.5.

2.1 Ultra-WideBand (UWB)

Ultra-Wideband (UWB) describes a wireless radio technology that transfers data over short distances while using a very low power density level [20]. The Federal Communications Commission (FCC) defines a UWB device as a device which emits radio signals with a bandwidth of at least 500 MHz or a fractional bandwidth greater than 0.2 [7]. The high rate pulse (HRP) UWB physical layer (PHY) was introduced as an amendment in the IEEE 802.15.4a standard in 2007 [9]. This layer supports three frequency bands, namely the sub-gigahertz band below 1 GHz and both the low and the high band which occupy the spectrum from 3.1 to 10.6 GHz [10].

The basic concept of UWB in comparison to common narrowband technologies is the trade-off between bandwidth and transmit power. Instead of long continuous waves, UWB transmits data as pulses with a high bandwidth at a very low power spectral density [13, 20]. Due to regulations from the FCC, the power levels of UWB signals are below -41.3 dBm/MHz. Since these pulses are underneath a certain noise threshold, signals cause very little interference with other communication systems, such as GPS or WLAN [12].

In addition to the small interference, short pulses also affect accuracy when determining locations with the help of wireless signals. The length of a UWB pulse is usually in the nanosecond or picosecond order which allows accurate positioning [20]. In indoor scenarios, multipath fading often affects the propagating signals and, therefore reduces the ranging accuracy. However, the large bandwidth of over 500 MHz grants high temporal resolution, which provides a good robustness against this issue [17].

2.2 Ranging with UWB

Accurate localization with ranging techniques based on time-of-flight (ToF) measurements is a challenge for existing narrowband technologies because determining a signals time of arrival is non-trivial. With the introduction of the UWB PHY and its focus on ranging in the 802.15.4 standard, high precision localization for wireless sensor networks (WSN) became possible [17].

There are various ranging techniques that can be used with UWB. Depending on the use case, some methods are more suitable than others. A simple and common ranging approach in wireless networks are *received signal strength* (RSS) based methods. RSS is a technique to estimate the distance between a transmitter and a receiver based on the signal strength of a message [21]. UWB, however benefits from high temporal resolution which makes time-based algorithms, such as *time-of-arrival* (ToA), usually the better option for ranging. When knowing the time of transmission and reception of a message, ToA estimates the distance by calculating their time difference. Nevertheless, this requires tight clock synchronization between all network nodes [17].

Since Decawave's real-time location system (DRTLS) (see Section 2.5) uses a two-way ranging scheme, the method is described in more detail in the next section.

2.3 Two-Way Ranging (TWR)

Two-Way Ranging (TWR) is a time-based method to measure the distance between two transceiver nodes. This is done by determining the round-trip time of a signal and calculating the time-of-flight (ToF) of the packages. TWR does not require a globally synchronized clock among the network nodes, since nodes calculate distances with their transmission and reception timestamp [11].

One can differentiate between a simpler TWR approach called Single-Sided TWR (SS-TWR) and a more complex approach called Double-Sided TWR (DS-TWR). Using SS-TWR, the transceiver nodes exchange a total of two messages. The performance of this approach is more error prone than DS-TWR. The problem is the varying clock speed offset between the ranging devices, as the delay of the reply causes a proportional estimation error. DS-TWR attempts to mitigate the error by making the tag respond to the reply message of the anchor by sending a third message. This adds two new measurable values, a second round-trip and delay time, as additional information [16].

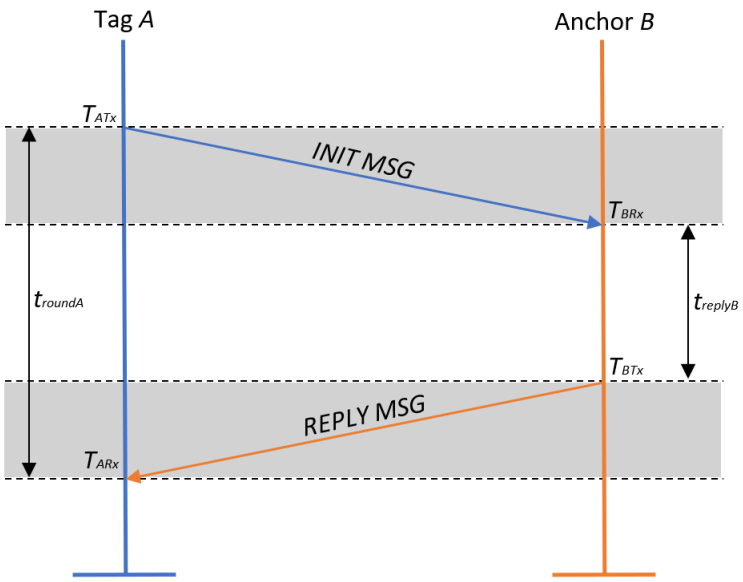


Figure 2.1: Ranging with a Single-Sided Two-Way Ranging scheme (SS-TWR).

2.3.1 Single-Sided Two-Way Ranging (SS-TWR)

SS-TWR is the ranging scheme used by Decawave's DRTLS [6] (see Section 2.5). Tag A initializes the ranging handshake by sending an $INIT$ message to anchor B. After reception, the anchor then replies with a $REPLY$ message. This procedure is shown in Figure 2.1. Anchor B measures the actual reply time t_{replyB} by subtracting the transmission timestamp T_{BTx} of the second message from the reception timestamp of the first message T_{BRx}.

$$t_{replyB} = T_{BTx} - T_{BRx} \tag{2.1}$$

Tag A measures the overall round-trip time of a signal t_{roundA} from the transmission of the first message T_{ATx} until the reception of the second message T_{ARx}. Therefore, the tag also subtracts the respective timestamps.

$$t_{roundA} = T_{ARx} - T_{ATx} \tag{2.2}$$

The *time-of-flight* ToF can be obtained by first computing the duration of the ranging procedure t_{roundA} minus the time it takes the anchor to reply t_{replyB}. Finally, the outcome is divided by 2 to achieve the time-of-flight for one message [11].

$$T_{tof} = \frac{t_{roundA} - t_{replyB}}{2} \tag{2.3}$$

2.4 Message Queue Telemetry Transport (MQTT)

MQTT is a network protocol for *machine-to-machine* (M2M) communication. The protocol uses a publish/subscribe pattern. All messages are identified over communication channels, so called topics. A device which is subscribed to a specific topic receives all published messages regarding this topic. Topics can contain the symbols '#' and '+' as a wildcard. Relevant for this thesis is the '+' symbol because it represents a certain hierarchy level in the topic tree. Subscribing to such topics means the topic is subscribed for every hierarchical instance.

Messages get published in combination with three different Quality of Service (QoS) factors 0, 1 and 2. A QoS of 0 means a message is delivered at most once without taking a message loss into account. A QoS of 1 means that a message is delivered at least once by waiting for a reply message from the receiver and a QoS of 2 guarantees that a message is delivered exactly once by using a two-stage confirmation [15]. A visualization of the QoS factors and the messages exchanged is shown in Figure 2.2.

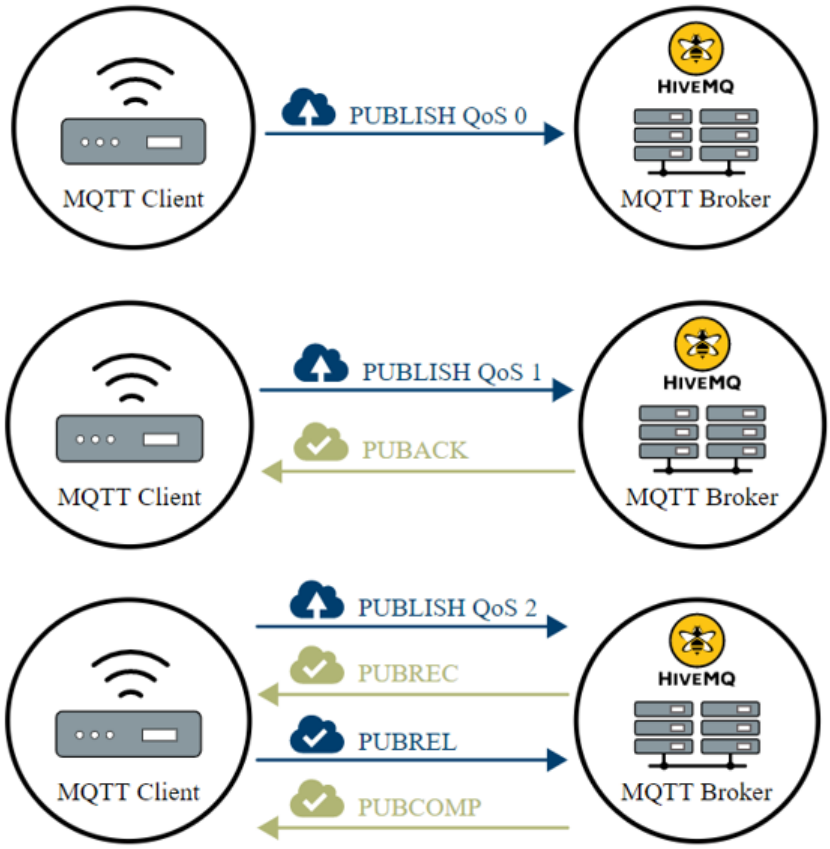


Figure 2.2: Quality of Service factors. Images taken from [19].

2.5 Decawave's Real-Time Location System (DRTLS)

The Module Development & Evaluation Kit for the Decawave DWM1001 (MDEK1001) provides hardware as well as software for a scalable UWB-based real-time location system (DRTLS) [2]. The development kit includes encased RTLS units which can be setup to operate as a network. Therefore, these units can be configured as different node types. For configuring the network and its nodes in the first place, Decawave offers an additional Android application [6].

To better understand the system, this thesis first captures all its components on both hardware and software side (see Section 2.5.1) before giving an overview of the network functionality (see Section 2.5.2). Additionally, the superframe structure (see Section 2.5.3) as well as the ranging protocol (see Section 2.5.4) are explained in greater detail.

(a) RTLS unit

(b) DWM1001-DEV Development Board

Figure 2.3: MDEK1001 Hardware. Images taken from [1].

2.5.1 Component Overview

The encased white RTLS units (see Figure 2.3a) of the development kit define the hardware of the system. Every device contains a DWM1001-DEV board with a DWM1001 (see Figure 2.3b). Each of these DWM1001 units can be configured as one of the following roles:

Anchor A node defined as an anchor has a fixed position in the network, typically mounted on walls or tripods. The system requires to configure at least four RTLS units as anchors for accuracy, where one of them is defined as the Initiator. The anchors' positions are known to all network nodes, allowing tag nodes to initate the ranging.

Tag Tag nodes are the mobile devices within the network that can perform a localization. Therefore, each tag measures the distance between itself and surrounding anchors and computes its own position. Tags automatically select their optimum four anchors for ranging [1] (see Section 2.5.4).

Bridge Node (Gateway) A unit configured as bridge node acts as a bridge between the UWB network and an additional Linux host. For this purpose, a Raspberry PI 3 Model B is used. Assembling a bridge node with a host will constitute a gateway which is a network node that can manage data traffic via MQTT messages [6]. An image of such a gateway is shown in Figure 2.4.

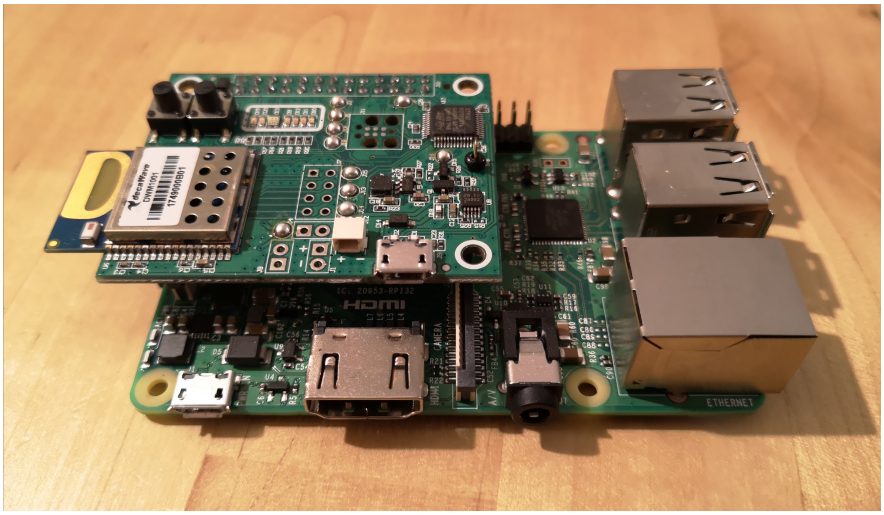


Figure 2.4: Gateway consisting of a R3B and an attached DWM1001-DEV bridge node.

The gateway's MQTT broker already provides certain predefined topics used to transfer configuration and location data. The topics relevant to this thesis and their payloads are listed in Table 2.1. Both topics contain a '+' wildcard which represents all node IDs inside the network. Messages published within Decawave's system are always published with the QoS factor 0.

Topic	Payload
dwm/node/+/uplink/config	Configuration data of a node. The content depends on the node type and differs between tag and anchor
dwm/node/+/uplink/location	xyz-coordinates, QF and the superframe number of a tag position

Table 2.1: Relevant topics and their payload predefined by Decawave [5].

In terms of software, Decawave offers an application for network setup and maintenance and a gateway software suite:

Decawave DRTLS Manager The Decawave DRTLS Manager is an open source Android application used to maintain the DRTLS network. All task, which include the creation of the network, assignment of available nodes, configuration of anchors and tags, positioning of anchors and the real-time visualization of nodes on a grid or floorplan layout are done by the manager. For communication with the devices, the application uses Bluetooth.

Decawave DRTLS Gateway Application The software suite provides a raspbian image with gateway functionalities for the Raspberry PI 3 Model B. It consists of multiple components, including an MQTT Broker or even a Gateway Web Manager. After flashing the image on the Raspberry PI and attaching a bridge node, it can route DRTLS location and sensor data traffic onto an IP based network [1]. The collected data of the gateway includes network information such as the location of tags as well as network configurations of all nodes [6].

2.5.2 Network Overview

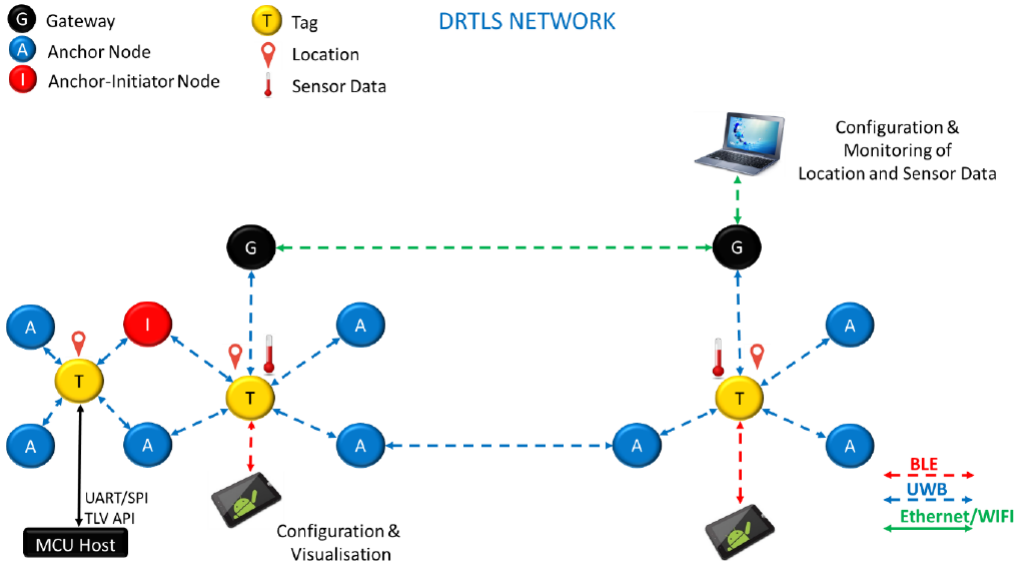


Figure 2.5: DRTLS network scheme. Image taken from [6].

A new DRTLS network is initialized with the DRTLS Manager. It discovers yet unassigned RTLS units and adds them to the network. After assigning them, at least four devices must be set as anchors, where one of them is also set as the Initiator, defining the origin of the coordinate system. The position of all other anchors must be determined either automatically with a featured auto-positioning function or manually by entering xyz-coordinates [3]. The DRTLS Manager can then be used to further

configure tags in terms of their update rates or to view their real-time positions. The adjustable update rate ranges from a minimum of every one minute (0.0167 Hz) to a maximum of 10 times per second (10 Hz).

Including an optional gateway unit allows the system to communicate with an outside network (LAN/WLAN). This enables the monitoring of network data remotely with the DRTLS Web Manager or other external applications. Alternatively, the connection between tag and a host device or a PC also enables the output of ranging information over UART [6]. A complete network scheme with all its components can be seen in Figure 2.5.

2.5.3 Superframe Structure

The Initiator's purpose is to control the timing of superframes (SF) which is a repeating structure of 100 ms duration, necessary for all network node operations. The other anchors must keep their clock synchronized with the Initiators, so that it is aligned with the SF timings. A SF consists of 30 Beacon Message (BCN) slots, two Service (SVC) slots, 15 TWR slots and additional 30 Bridge Node Beacon Message (BN-BCN) slots. There are short idle/guard times between these SF slots. This structure is shown in Figure 2.6.

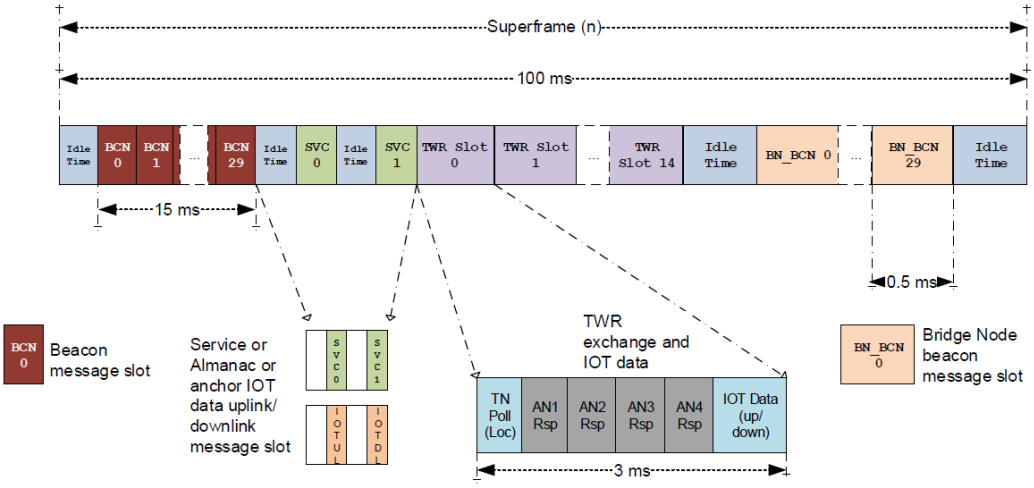


Figure 2.6: Superframe structure. Structure taken from [6].

Within BCN slots, anchors sent beacon messages containing information about the current SF and network slot usage of anchors and tags. SVC slots are reserved for different service messages. This includes Almanac messages containing the firmware version, anchor join requests and uplink/downlink (IoT) data to/from anchors. TWR slots are used for node TWR exchanges and for IoT data to/from tags. In the remaining 30 BN-BCN slots, bridge nodes sent messages to inform about available downlink data for tags [6].

2.5.4 DRTLS Localization

DRTLS TWR Protocol DRTLS uses a SS-TWR scheme (see Section 2.3.1). Tags range with at least three but for better accuracy optimally four surrounding anchors and calculate their own locations with respect to the anchors' positions. Therefore, they normally select their nearest anchors which span a convex hull around their positions. Figure 2.7 sketches two tags and their choice of anchors for TWR.

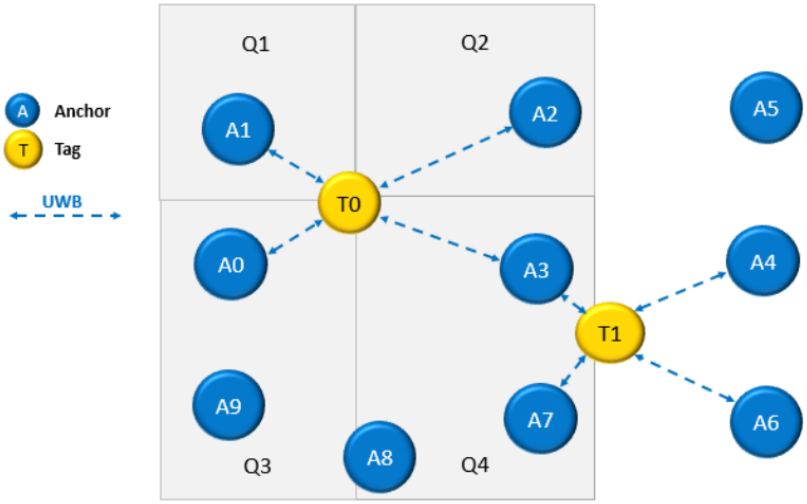


Figure 2.7: Tag T0 and T1's chosen anchors for TWR. Image taken from [6].

Tags collect ranging and data slot maps of nearby anchors through Beacon messages and combine them to find a free TWR slot in the SF. If all TWR slot are occupied, the tag tries again to obtain new data and reserve a free slot after some time. Each of the 15 ranging slots is dimensioned in a way that allows sufficient time for a tag to perform TWR with four anchors. When taking the 100 ms SF duration into consideration, this results in a maximum location rate capacity of 150 Hz.

To initiate the location attempt, the tag broadcasts a Group Poll message. This message contains necessary information such as the update period of the tag and a bitmap indicating the anchors with which it wants to range. Corresponding anchors will answer the Group Poll message by transmitting a response message which includes a determined transmit time. By sending a response, the anchor confirms the availability of a free slot in the SF in correspondence to the tag's update rate. After receiving a response from all ranging devices and following the IoT data subframe, the tag will proceed to calculate the range from all involved anchors to itself [6].

Location Engine Provided that at least three anchor ranges are valid, the final location is achieved with the internal location engine based on maximum likelihood estimation. The principle behind this location engine is to create data sets. The engine then evaluates positions by using different criteria to choose and combine this data. To calculate the final position, selected estimated positions are taken into consideration [6].

Depending on all these estimation criteria and the calculated errors, the engine reports a quality factor (QF) [6]. The official system documentation provides no details on how the QF is computed. However, in the Decawave Forum the calculation of this factor is briefly described by a moderator [4]. Ideally, the tag ranges with four anchors even if only three anchors are needed to perform a localization. This implies four different possibilities for achieving the localization. The tag calculates a position based on three anchors and then calculates the theoretical range from this estimation to the fourth anchor that was not used. The theoretical range will then be compared with the measured range of the fourth anchor. This process is performed four times where the tag position with the smallest error will be used.

The calculated error eventually describes the QF. This is a value between 0 and 100, where 100 refers to a null error with a perfect setup and 0 corresponds to an error greater than 50 cm. A special case is the QF value 50. This signifies that the tag was only able to range with three anchors so no error estimation could be performed [4].

3 Implementation

Within this thesis, the aim was to design a RTLS for an industrial use case. For this purpose, an Android application was developed. The app is an interface that visualizes real-time location data provided by Decawave's DRTLS. In addition, it should optimize workflow and safety aspects for tagged workers, vehicles or parts in an industrial environment. Throughout the development, various features were implemented to meet the specific requirements.

First, this chapter provides general information about the application and the talked-out requirements (see Section 3.1). Second, it describes the implemented features in greater detail (see Section 3.2). The end of this chapter contains a short summary of the application in relation to the requirements and implemented features (see Section 3.3).

3.1 Requirements

The application and its functionality are based on requirements defined by the Institute of Innovation and Industrial Management (IIM). The IIM institute decided to install a UWB-based indoor localization technology from Decawave into their learning (LEAD) factory to digitalize certain industrial processes.
A typical use case intended for this technology is a waste-walk. It is part of a schooling in the LEAD factory to detect wasted resources throughout an industrial assembly run. During this run, a team of workers needs to assemble TU Graz scooters over approximatly 20 minutes. To do so, they must complete certain working steps on different working areas. The schooling attendees are encouraged to observe the workflow and identify potential improvements.

Decawave's supplied DRTLS Manager already has basic RTLS functionality. However, to proper integrate this system and adapt it to fit the intended use case, an additional solution is required. An interface needs to visualize the data from Decawave's system and track the workflow of moving network nodes. Additionally, supplementary features must be specified to fully capture the needs of the use case. The corresponding application has to fulfill these requirements:

- The application will be used in the LEAD factory and should run on the institute's Android tablet. Therefore, the application and its layout should be designed to

work on a Huawei MediaPad M5 with Android version 8.0.0 and an 8.4-inch display. A picture of the device is shown in Figure 3.2a.

- Multiple devices should be able to interact with the system simultaneously. They should be able to access location data independently of a bluetooth (BLE) connection.
- The application should be generic. It should be compatible with other localization technologies and not only be limited to Decawave's system.
- It should have a position view that features the live plot of location data. In addition, the upload of a floorplan should allow a better visualization of the system in the room.
- To grant a better differentiability, the app should provide advanced configuration options for nodes and the ability to assign nodes to categories.
- On change, the latest app configurations should be saved and shared with other connected devices. Also, when the app gets restarted, configurations need to be restored.
- The supply of a safety mechanism for tracked workers and resources. Regarding this, it should be possible to restrict certain areas in the room with different alarm modes. If a node then enters a specific region, the alarm should go off and notify the app's user.
- The display of a spaghetti diagram by drawing tag paths on the floorplan. Therefore, the app should be able to continuously track the position of moving tags to showcase their covered paths.
- It should calculate the total distance of a tag path.
- For post processing of system data, the application should log relevant information in an app history that is exportable to a file. Besides, position trackings should have an option to be saved permanently on the device storage.

3.2 App Features

For the application to work as expected and meet the above-mentioned requirements, different design decisions were made. This results in the implementation of various app features. All features are intended to cover the requirements and optimize workflow, safety and usability aspects when working with a localization system in an industrial environment.

3.2.1 MQTT Communication

Data transmission between the application and the location system is solely done via MQTT. This protocol entails certain advantages: It allows the support of multiple devices, backup options for configurations and helps to make the app generic.

The major benefit of a MQTT connection is the improved usability based on the fact that multiple devices running the application can connect to the same MQTT Broker simultaneously. If one connected device then publishes a message to a certain topic, it will automatically synchronize all the other connected devices. Additionally, it also allows the application to be used with other localization systems. In order to work, other systems must use a MQTT topic structure that is similar to Decawave's system. This is because many design decisions were explicitly made to adapt the implementation to work with Decawave's system.

Figure 3.1 shows a generic scheme of how the MQTT broker links the Android app and a location system. It also shows the feature of multiple connected devices to the broker. In correspondence to the thesis, the location system represents Decawave's DRTLS, the location data is transmitted via UWB and the MQTT broker is a software component of Decawave's Gateway.

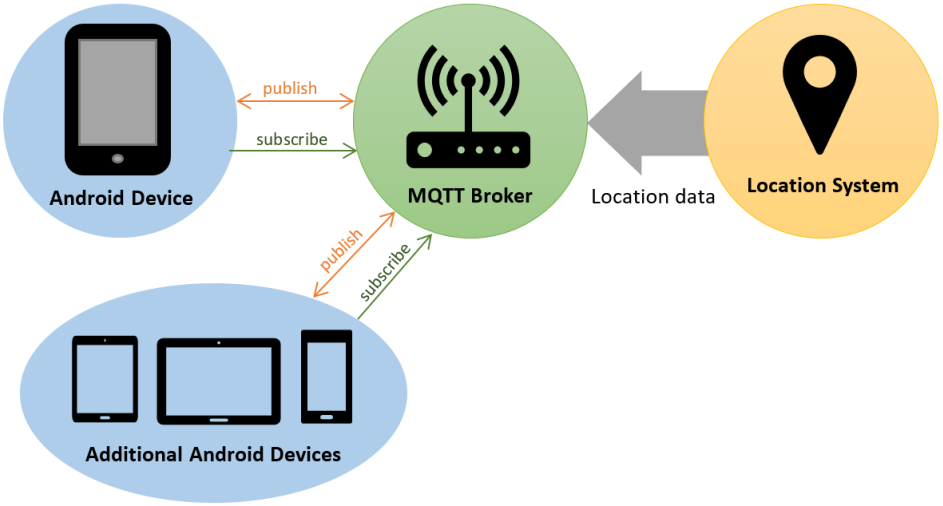


Figure 3.1: Generic connection between an Android device and a location system linked by a MQTT broker.

To connect the application to Decawave's UWB network, a node must be adapted as an interface. A picture of the used gateway node is shown in Figure 2.4. The MQTT Android client can establish a connection to the MQTT Broker of the gateway and start to transfer data.

Immediately after connecting with the gateway, the app subscribes to the predefined topics seen in Table 2.1 and starts to listen for messages. For full functionality, two additional topics are added and subscribed. They are used to save the most recent app configurations of nodes and regions (see Table 3.1). Every time either of them is modified, a message with the new information is published to its corresponding topic with the QoS factor 0.

Topic	Payload
nodeConfig	Configuration data about all nodes: ID, name, category and color
regionConfig	Configuration data about all regions: name, xy-coordinates, color and alarm mode information

Table 3.1: Configuration topics and their payload used in the application.

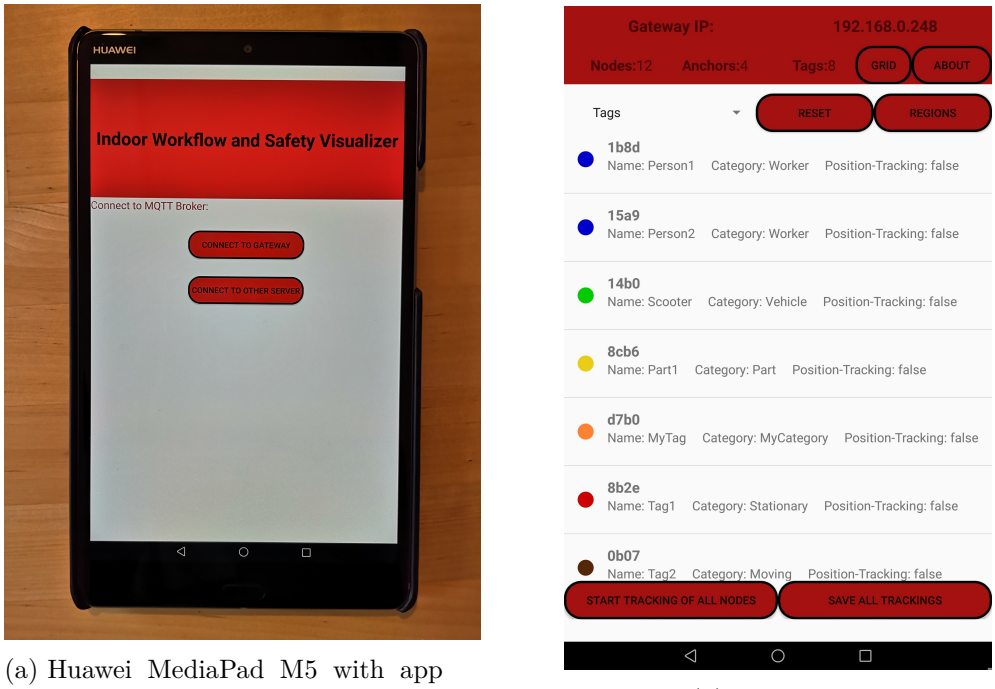
When restarting the app, saved configurations are restored from the latest published messages to these topics. This is because the MQTT broker makes the current configurations accessible to all connected devices. Even if the app crashes, certain configurations can be restored provided that the broker is active.

3.2.2 Start & Main Interface

Start Screen The first view after starting the app is the connection interface to the MQTT broker. The button *'Connect to Gateway'* establishes the MQTT connection to the known broker defined in the code. It is also possible to connect to other servers without having to modify the app code. An input option for an IP address and a port for an external MQTT broker appears by pressing the button *'Connect to other server'*. Either way, after building up a connection, the view changes to the main menu. The application start screen is seen on the used tablet in Figure 3.2a.

Main Menu The main menu is the starting point for all further interaction with the system. It consists of multiple elements. General information about the system is shown at the top of the screen. This includes the IP address of the connected gateway, how many anchors/tags are currently active and the total number of network nodes.

The main focus of the menu is the centralized list view of nodes. A dropdown menu is above the list view to switch between anchor and tag nodes. Directly after entering the menu, the Android MQTT client fills the list with all network nodes according to the received messages. Each list element represents a node inside the network and contains information such as the ID, name, the assigned category, its color and if the position is currently tracked or not. Because anchor nodes are stationary, position tracking is only implemented for tags.



(a) Huawei MediaPad M5 with app start screen

(b) Main menu

Figure 3.2: The app start screen on the Android tablet and the app main menu.

The menu contains different buttons as well. All their actions are listed in Table 3.2. Figure 3.2b shows a picture of the overall interface.

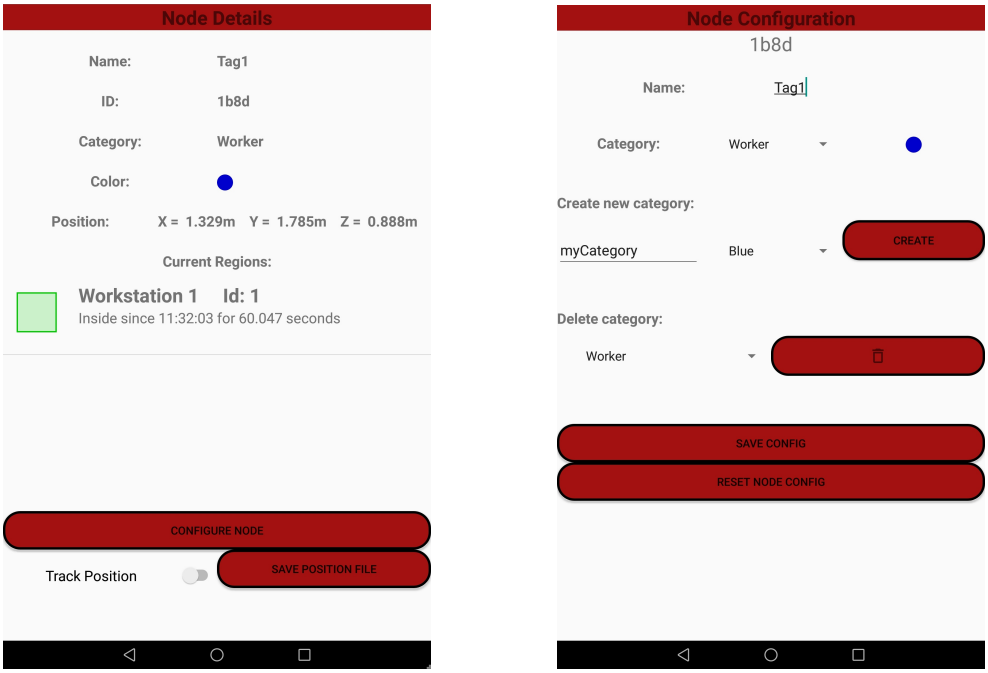
Button	Action
Grid	Opens the grid view where nodes are plotted in real-time.
About	Opens a short instruction on how to use the app.
Reset	Resets all current node configurations to default.
Regions	Opens the region menu where regions can be configured.
Start/Stop tracking of all nodes	Starts/stops the position tracking for all tags at once.
Save all Trackings	Saves all current tracking files to the device storage.

Table 3.2: Buttons of the main menu and their action.

3.2.3 Node Configuration & Position View

Node Configuration Clicking on a node within the node list shows detailed information regarding the node. Important to notice is that each node is defined through a unique ID. This ID has a length of 16-bit and is predefined by the location system.

Additionally, the name, category and color is listed. The view also shows the current xyz-position of a node and updates it regularly if a new position is available. Tag nodes provide further information as well. Selecting a tag shows the regions in which it is currently located, the exact entering time and for how long it has already been inside. Besides that tags have a switch to start/stop the tracking of their positions. A button next to the switch is for saving the current tracking to the device storage. To change the configuration, the button *'Configure Node'* must be selected. The detail view for a tag with its provided information is shown in Figure 3.3a.



(a) Node detail screen

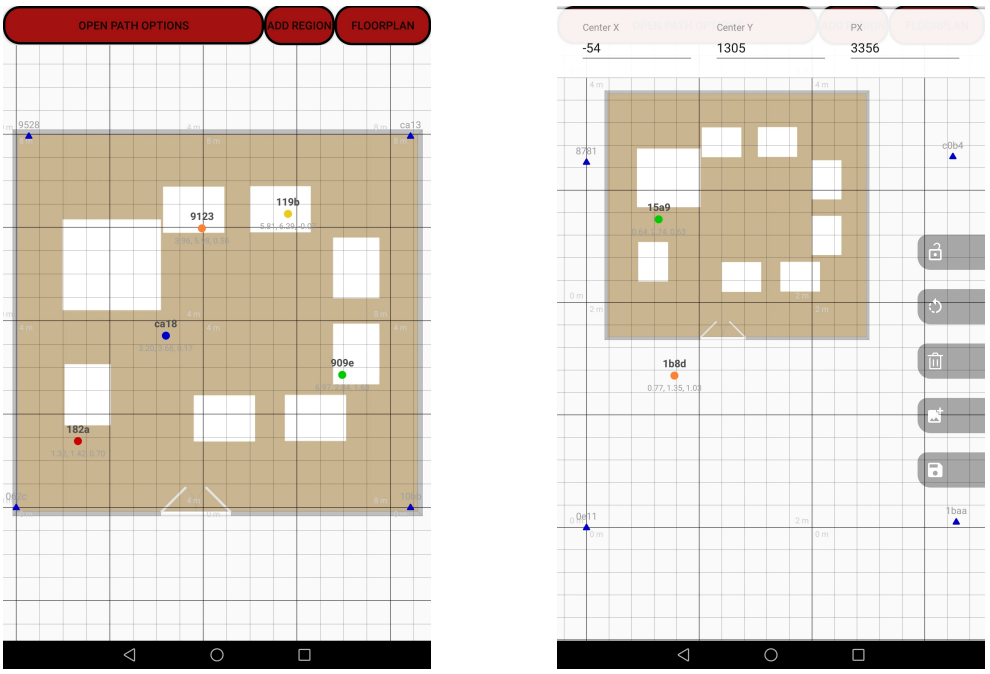
(b) Node configuration screen

Figure 3.3: Node detail and configuration screen.

The configuration screen (see Figure 3.3b) allows to change the name and the category of a node. A category can be described as a representative class that has a name and a color. Already created categories can be assigned to a node but can also be deleted if no longer used. For this purpose, the app has creation and deletion options in form of dropdown menus and text edit fields for categories. Once created, a category is saved directly to the device. This has the advantage of not having to recreate individual categories when restarting the app. The buttons *'Save Config'* and *'Reset Node Config'* are for saving the configuration or resetting it to default. The screen view returns to the detail screen afterwards.

Position View The implementation's main part is the actual position view. The current positions of all network nodes are plotted on a grid view seen in Figure 3.4a. The grid view can be zoomed in and scrolled through for better usability. Anchor nodes get plotted as triangles; tag nodes are represented as dots. Both types have their unique ID shown above them. Tags also show their position beneath them as additional information. Many other app features, such as regions and paths are visualized on this view as well. To achieve a stable app performance, the view of network nodes only gets redrawn every second, even if the actual update rate of the node positions is higher.

An image of a floorplan can be uploaded and put on the grid view to better understand where tags are located in the room. The button *'Floorplan'* opens the corresponding interface. The floorplan can be adjusted in size and rotation to be aligned with the drawn anchors and tags. Input options at the top also allow to enter floorplan dimensions. The interface and all the options can be seen in Figure 3.4b. The grid view itself and the floorplan feature are based on code of the open source DRTLS manager. This code then was adapted and rewritten to fit the application.



(a) Grid view with floorplan

(b) Floorplan configuration interface

Figure 3.4: Position view of nodes and floorplan interface.

3.2.4 Regions

The app features the ability to mark regions of interest. A region defines an area in the room where it is certainly interesting to track node activities. For that reason, a region tracks the exact time when a network node enters and leaves it and calculates the total time spent inside. The spent time within these regions provides an accurate insight into the workflow of the tracked entities in an industrial use case. Safety wise, it is possible to restrict a region with additional alarm modes. The idea is to favor safety for tracked workers. The various implemented alarm modes allow to define restrictions suitable for different use cases. Whenever a tag triggers an alarmed region, the device responds with a one-second-long vibration and a pop-up notification. Because of this additional safety mechanism, the entrance of a potentially dangerous region can be detected earlier.

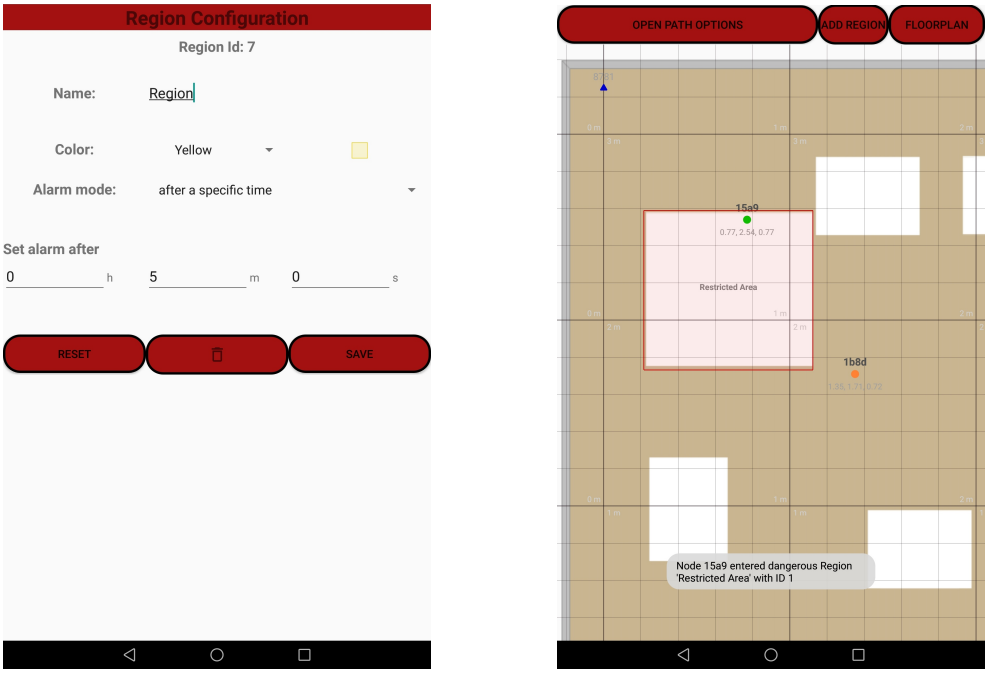
(a) Grid view with dummy region interface

(b) Region menu

Figure 3.5: Grid view with marked regions, dummy region interface and region menu.

Regions are represented as square shaped 2D areas which are drawn on the grid view. The button *'Add Region'* overlays the grid view with a dummy region. The dummy region can be moved around with its upper left corner and resized on any other corner to form the needed shape. Pressing the save button then creates a region at the exact position of the dummy square. Already created regions and the dummy region interface can be seen in Figure 3.5a. Once created, a region is listed in the region menu that is reached over the *'Regions'* button of the main menu.

Region Configuration The region list in the region menu includes all created regions and shows information about them. Each region is defined through a unique ID, has a name, a color and a certain alarm mode. A list entry includes all this information as well as the nodes that are currently inside this region. The screen also has some buttons. The button *'Reset All'* is for resetting all region configurations to default. The button *'Delete All'* deletes all existing regions. Since this action cannot be undone, a pop-up window asks again if the reset or deletion should be performed in case of a miss click. Figure 3.5b shows the region menu screen.



(a) Region configuration screen (b) Alarm pop-up notification

Figure 3.6: Configuration options for regions and an alarm notification.

For the actual configuration, a region from the list needs to be selected. The configuration screen allows to change name and color of the region. The unique ID of the region is not changeable to clearly distinguish regions. However, the main option is the settable alarm mode. Besides the option to not set any alarm, there are three different alarm modes featured to have a suitable option for many use cases. They can be used to regulate how many nodes should be inside a region simultaneously, for how long a node should stay inside a region or do define a dangerous region. Table 3.3 shows all available alarm modes and describes how they are triggered.

Alarm mode	Description
none	No alarm, but the region still tracks node activities
immediately	The alarm goes off immediatly after any node enters the region
after time	The alarm goes off after any node stays in the region for longer than the set time period
after node count	The alarm goes off after there are more nodes inside the region at the same time than the set node number

Table 3.3: Region alarm modes and when they are triggered.

The buttons at the bottom of the screen are for resetting a region to default values, deleting a region and to save the current configuration. The configuration screen for a region with selected alarm mode *'after time'* can be seen in Figure 3.6a. Figure 3.6b shows the pop-up notification for a node that entered a region with alarm mode *'immediately'*.

3.2.5 Path Tracking

Path tracking is a feature that enables the drawing of a spaghetti diagram to visualize the industrial workflow. With this feature it is possible to simultaneously showcase one or more tag paths in the grid view. Paths could either be drawn live or loaded from specific files. A path consists of every tracked tag position within a certain time interval. The interval ranges from the time of starting position tracking to the time of stopping it. Both starting and stopping have to be done manually either for all tags at once or for every tag individually with their corresponding buttons or switches.

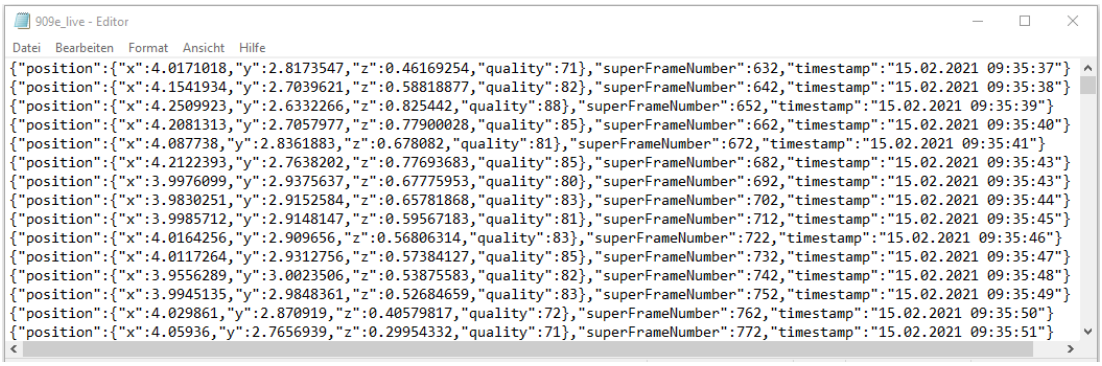
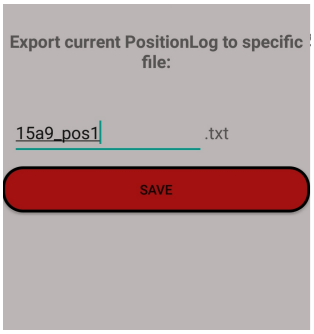


Figure 3.7: Content of a sample position file.

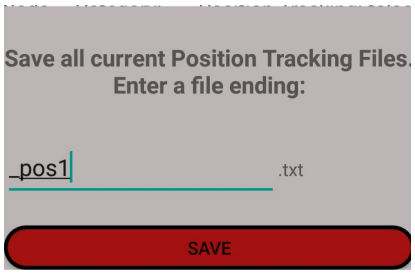
Position Tracking When position tracking is active, all new location messages coming from the location system regarding the tracked tags get processed and further written to a position file on the device storage. To preserve the data from received messages, their content will remain unmodified in this process. Because location messages from

Decawave have a JSON-format, attaching additional information is easily possible. Therefore, every message gets extended with the current time stamp. With regard to content, a message then includes the tags xyz-coordinates, the quality factor, the superframe number and the timestamp. Afterwards, the message is written in a file. Each currently tracked tag has its own position file. The file always starts with the ID followed by the file ending *'_live.txt'*. The first few lines of a sample position file are shown in Figure 3.7.

When position tracking is stopped, the app no longer processes and saves incoming messages. When the position tracking is activated again, the app first clears the content of the existing *'_live.txt'* files and starts to repeat the above procedure. However, these tracking files might be needed for analysis at a later point. To permanently store them, the app features an export to file option. For this, the *'Save All Trackings'* button of the main menu can be used. Alternatively, tag position files can be saved individually with the *'Save Position File'* button of the tag's node detail screen.



(a) Pop-up window for exporting a single position tracking to a file



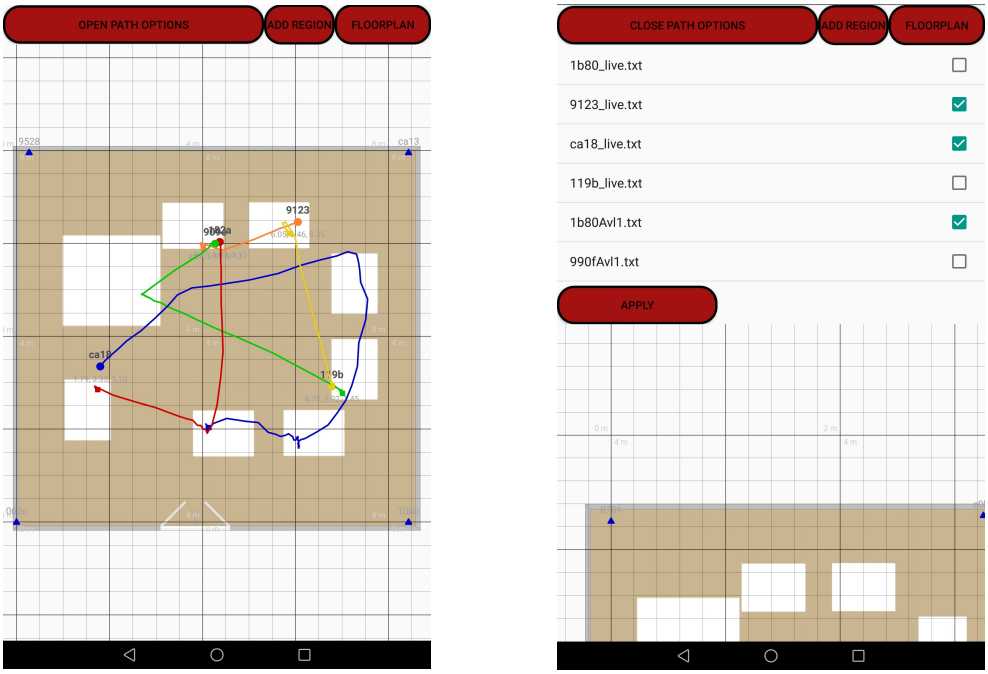
(b) Pop-up window for exporting all position trackings to files

Figure 3.8: Pop-up windows for exporting position trackings.

The pop-up window for exporting a single tag file (see Figure 3.8a) shows an input option for the intended file name. It is required to start the name with the ID of the tracked tag followed by any ending. The button *'Save'* creates a new file with the entered name and copies the content of the *'_live.txt'* file to it. In case the file name already exists, the window asks for confirmation to overwrite it or to enter a new name as a security mechanism. The pop-up menu for exporting all position files (see Figure 3.8b) works in the same way, with the exception that it only requires to enter a file ending for the affected nodes.

Spaghetti Diagram To plot paths in the grid view, the button *'Open Path Options'* must be selected. A dropdown menu then appears which includes a list of all available position files and an *'Apply'* button as shown in Figure 3.9b. An arbitrary number of them can be selected to be plotted on the grid view. Basically, there are two types of

paths. The first path type is reconstructed by exported position files. In this case, the complete path is drawn on the grid since the tracking has already finished. The other type is a live path. Currently tracked tags are able to plot position updates live during runtime on every new received location message. The color of a plotted path depends on the currently set category of its tag. If a file is selected which has no corresponding active tag, the path's color is set to blue. The gridview with some drawn paths can be seen in Figure 3.9a.



(a) Grid view with drawn paths

(b) Opened path options

Figure 3.9: Spaghetti diagram and path options.

Total Distance In combination with path tracking there is another feature implemented. When the position tracking is stopped, the app calculates the total distance a tag has covered during the interval. The total distance d is calculated as follows:

$$d = \sum_{i=1}^{\#p-1} \sqrt{(x_{i-1} - x_i)^2 + (y_{i-1} - y_i)^2} \tag{3.1}$$

In this formula, $\#p$ is the total amount of positions, x refers to the x-positions and y to the y-positions parsed out of the position file. To reduce inaccuracy, only distances greater than 10 cm between two consecutive positions are summed up. The outcome of the calculation can be seen in the app history.

3.2.6 History & Filesystem

History The application uses a history to log information about app activities. Its aim is to provide an overview of events that occured during runtime. These events include various node and region activities as well as the starting and stopping of the position trackings.

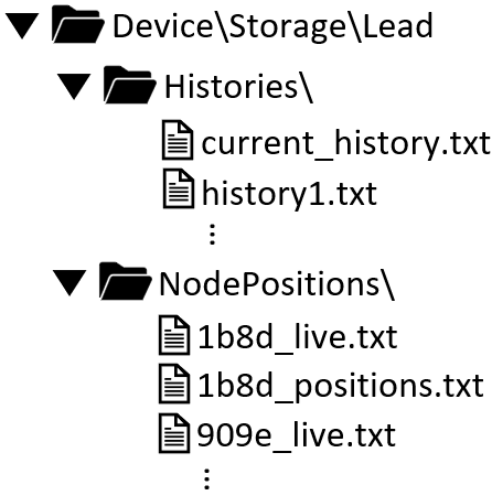
The app history is reached through the top right *'History'* button of the region menu. In the center of the history view is a list of all logged events appearing in chronological order. Each entry starts with a time stamp when the event happened followed by a flag indicating the type of event and lastly the description of the event. The history is saved in the *current_history.txt* file on the device storage. Every time an event happens, the history log on screen is extended and the history file is updated. All events that are considered in the history as well as their flags and their log text can be seen in the following table:

Flag	Event	Log
CONFIG	Create region	Created Region ... with ID ...
	Rename region	Renamed Region ... with ID ... from ... to ...
	Change region color	Recolored Region ... with ID ... from ... to ...
	Change region alarm mode	Changed alarm mode for Region ... with ID ... from ... to ...
	Reset region	Reset Region ... with ID ... to default values
	Delete region	Deleted Region ... with ID ...
INFO	Start position tracking	Started to track Position for Node ...
	Stop position tracking	Stopped to track Position for Node ... The node covered a distance of ... meters
	Node enters region	Node ... entered Region ... with ID ...
	Node leaves region	Node ... left Region ... with ID ... after ...
ALARM	Alarm mode 'immediately' is triggered	Node ... entered dangerous Region ... with ID ...
	Alarm mode 'after time' is triggered	Node ... stayed in Region ... with ID ... for more than ...
	Alarm mode 'after node count' is triggered	There are more than ... nodes in Region ... with ID ...

Table 3.4: Events that are logged in the app history.

At the top of the history screen there are two buttons. The *'Clear History'* button is to delete the current history and the *'Export to File'* button is for exporting the current history to a file. This allows to save the current log to a separate file for post reviewing the gathered information. The export window looks the same as the pop-up window for exporting all position trackings (see Figure 3.8b), but for saving the current history to a file. Figure 3.10a shows an image of the history screen with its log.

(a) Sample history log



(b) File hierarchy in the device storage with sample files

Figure 3.10: History log and file hierarchy in the device storage.

Filesystem The application saves all tracked tag positions as well as history files as *.txt* files. Whenever any file is exported, it is added to a certain folder hierarchy in the storage of the used Android device. The main folder '*Lead*' has two subfolders. The first folder '*Histories*' contains the current history file and all other exported history files. The second folder '*NodePositions*' includes the live position files of currently tracked tags and all exported position files. A scheme of the file hierarchy with sample files is shown in Figure 3.10b.

3.3 Summary

For this thesis, a solution had to be found to adapt Decawave's DRTLS to work within a certain use case at the IIM's LEAD factory. Many requirements were defined to tailor the solution for an industrial purpose. The outcome is an Android application that is

feature rich in different terms. It optimizes industrial workflow and safety aspects while having a user-friendly interface with many configuration options.

The application was required to work in the LEAD factory on the institute's Android device and is used to track workers or parts during a waste-walk. Workflow features include live path tracking, the upload of a floorplan and the tracking of node activities in an app history. In terms of safety, it is possible to mark regions with different alarm modes to define dangerous or restricted areas. For usability, the app offers various configuration options for nodes and regions and an export to file option for the history and the tracked tag positions. Because of MQTT data traffic between app and system, the app is not only bound to Decawave's DRTLS but can be used with other location technologies too. This allows it to be a suitable option for various other location-based use cases.
To conclude, the application and its wide spectrum of implemented features cover the defined requirements and fulfill the aimed aspects of workflow, safety and usability.

4 Evaluation

This chapter contains an evaluation of the performance of Decawave's DRTLS. This includes accuracy measurements of tags (see Section 4.1) and scalability measurements to determine the system capacity (see Section 4.2). In addition, the developed application was tested in a real case scenario during a schooling in the LEAD factory. The observations which were made during this test run are presented in Section 4.3.

4.1 Accuracy

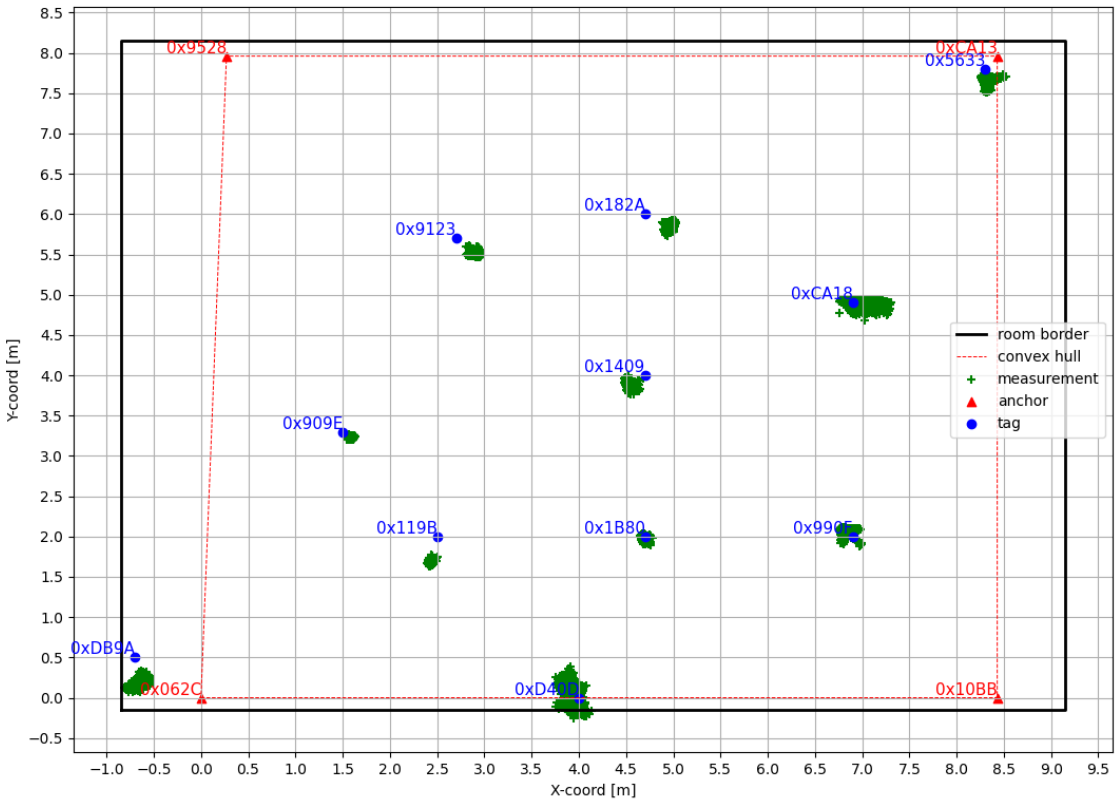


Figure 4.1: Overview of the experimental setup. The plot includes all real node positions and the actual measurements of the tags.

NodeType	NodeID	X [m]	Y [m]	Z [m]
Anchor	0x062C	0	0	2
	0x10BB	8.43	0	2
	0xCA13	8.43	7.96	2
	0x9528	0.27	7.96	2
Tag (LOS)	0x9123	2.7	5.7	1.1
	0x909E	1.5	3.3	0.9
	0x119B	2.5	2	0.9
	0x182A	4.7	6	0.9
	0x1409	4.7	4	0.9
	0x1B80	4.7	2	0.9
	0xCA18	6.9	4.9	0.9
	0x990F	6.9	2	0.9
Tag (edge-case)	0xDB9A	-0.7	0.5	1
	0xD40D	4	0	1.12
	0x5633	8.3	7.8	1.48

Table 4.1: Exact positions of anchor and tag nodes.

4.1.1 Experimental Setup

The accuracy measurements of Decawave's system were taken in a room of about 8 x 8.5 meters in size. Near each corner of the room, an anchor was mounted to the wall at a height of two meters. The positions of eleven different tags were measured where each tag position was tracked over 11000 times. Eight of these tags were placed around the center of the room within line of sight (LOS) to the anchors. The remaining three tags were placed near, on or outside the convex hull of the anchors which represent measurements with edge-case conditions. The exact node positions are presented in Table 4.1. Figure 4.1 should give an overview of all node positions within the experimental setup as well as the measured tag positions.

4.1.2 Results & Observations

It can be observed that the overall accuracy of Decawave's system ranges between a minimum of 1.9 cm to a maximum of 31.4 cm with an average of 18.3 cm. The measurements are very precise with a minimum standard deviation σ of 0.6 cm and a maximum of 5.7 cm. Considering all measurements, the average σ is 2.1 cm. All values are presented in Table 4.2. It shows the measured accuracy, the standard deviation σ, the minimum, maximum and average QF as well as how often the QF 50 was measured for each tag. A QF of 50 should be noted, since it means a tag has only been able to range with three of the anchors (see Section 2.5.4). In addition, Table 4.3 compares the observed average values of LOS and edge-case tags.

TagID	accuracy [m]	σ [m]	QF avg	QF min	QF max	QF 50 [#]
0x9123	0.305	0.011	71.555	61	80	0
0x909E	0.111	0.006	59.024	56	81	0
0x119B	0.313	0.008	51.89	47	97	866
0x182A	0.314	0.014	61.218	57	83	0
0x1409	0.201	0.014	89.65	76	98	0
0x1B80	0.019	0.01	77.646	69	82	0
0xCA18	0.104	0.057	60.967	51	87	0
0x990F	0.124	0.018	49.435	45	67	3579
0xDB9A	0.306	0.023	60.251	48	77	10
0xD40D	0.086	0.051	58.859	49	96	3
0x5633	0.131	0.015	67.497	33	99	64

Table 4.2: Observed values from all measured tags.

Tag condition	avg accuracy [m]	avg σ [m]	QF avg
LOS	0.186	0.017	65.173
edge-case	0.174	0.03	62.202
LOS & edge-case	0.183	0.021	64,363

Table 4.3: Comparison of LOS and edge-case tags in terms of average values.

By comparing LOS and edge-case tags in terms of accuracy and QF, no remarkable differences were found. However, there is a large difference in the average standard deviation σ. Edge-case measurements are spread more widely. Except for the outlier tag 0xCA18, edge-case tags have the highest σ values of all tags. With an average σ of 3 cm it is almost twice as high as the average for LOS tags.

The most accurate measured tag was 0x1B80 with a σ value of 1 cm. It has an average quality of about 78 and a measured minimum of 69, both of which were the second highest measured values of their kind. The tag with the worst accuracy was 0x182A. Still, its quality and precision measurements were not worse, but rather in the middle compared to the other tags.

Figure 4.2 shows point clouds for every tag. A plot includes the tag's real position and the measured average position beside the actual position measurements. The measurements are colored based on the measured QF. Therefore, the scale on the right should be considered as a reference.

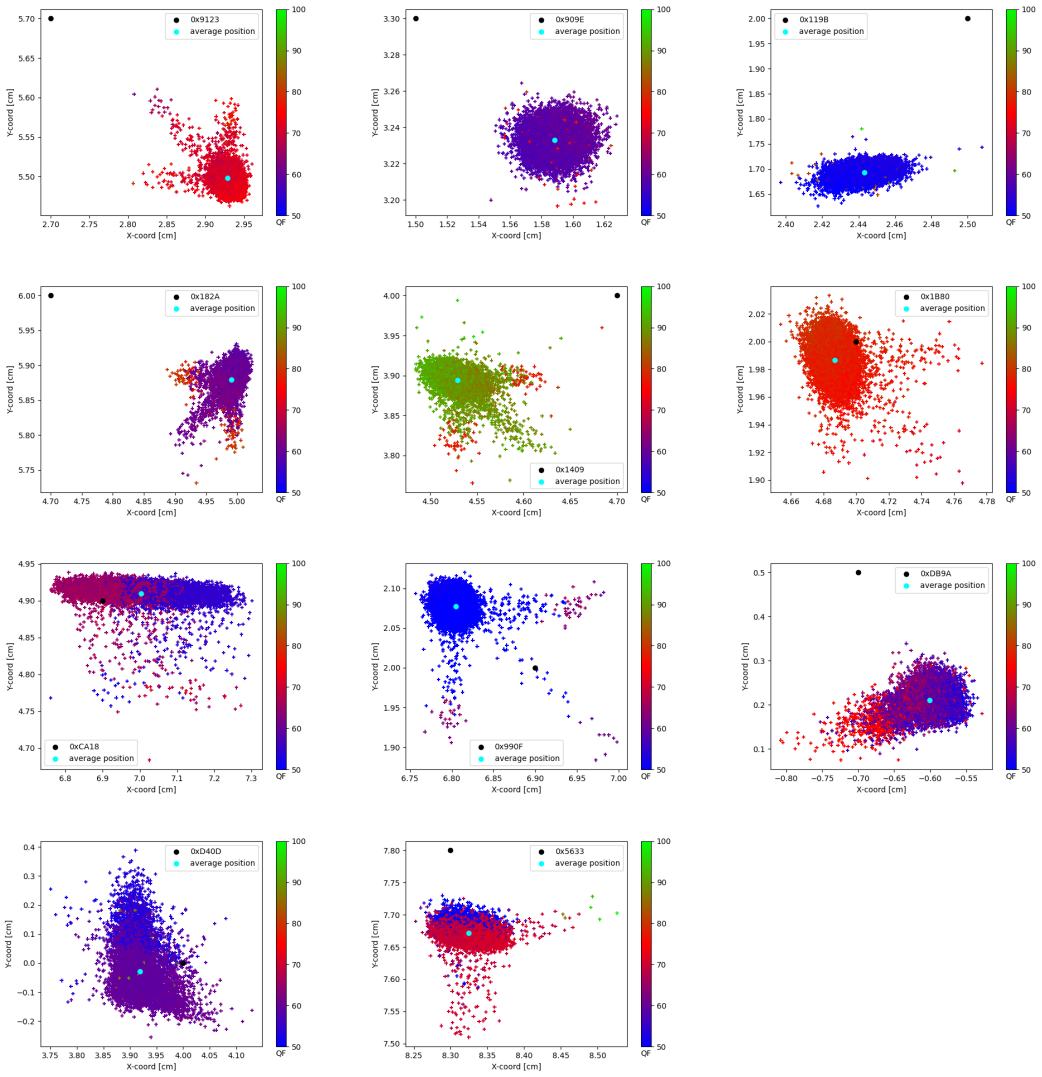


Figure 4.2: Colored point clouds of the evaluated tags depending on the measured QF.

When observing the plots in general, it can be seen that in some of them the quality of the measurements around the average position is equal or higher than more distant measurements. Within these cases, a position with a high QF tends to represent the measurement average. Good examples for this observation are the plots for the tags 0xD40D or 0x5633. Another observation are the shifted measurements. Most evaluated tags have a small deviation with an evenly spread point cloud, but an offset when looking at their real positions. The average y-coordinate is almost always smaller than it should be. Therefore, the height difference between anchors and tags should be taken into consideration. The anchors are placed approximately 1 meter higher than the tags. The larger distance increases the transmission time of the UWB signals and may have influenced the localization.

The tag 0xCA18 is also interesting because of its high σ. It has a deviation of 5.7 cm but still has an accuracy below average. Observing the plot of this tag shows that even with its high deviation, the average position is close to the real position. In addition, the real position is within the cluster of the best measured QFs. This observation also applies to the tags 0x1B80 and 0xD40D.

Another observation regards the three edge-case tags. All of them measured the QF 50 a few times. Since all of them were placed in a non-optimal way, this outcome was expected. However, the two LOS tags 0x990F and 0x119B were also not able to range with all anchors continuously. Tag 0x119B has 866 QF 50 measurements while tag 0x990F measured this factor a total of 3579 times, which is approximately one third of the time. The corresponding plot has an arrow shape where no clear cluster of measured QF 50 positions can be located, since these positions are spread across the whole shape. Interestingly, the arms of the arrow include positions with the best measured quality. The point clouds for the tags 0x9123, 0x182A and 0x1409 have a similar shape.

There are some potential sources of interference in the upper left area of the room which could have caused this and, hence, are responsible for localization inaccuracies. On the wall to the right of the anchor 0x9528 is a metal frame as a storage for industrial parts. In front of the storage is a workstation equipped with a PC, a 3D printer and a CNC milling machine. The whole machinery and additional electronics are placed on a desk with a metal frame. Both the objects with their metal structure and the electronics may interfere with the anchor and cause certain ranging attempts to fail or distort the calculated position. A possible solution to overcome this disadvantage is the introduction of additional anchors. Then, a tag could range with better visible anchors depending on its position in the room.

4.2 Scalability

To evaluate the scalability and overall capacity of Decawave's DRTLS, a total of twelve separate measurements were taken. The aim was to investigate the SF numbers to see their behaviour with different amounts of active tags ranging with different update rates. Thereby, each 5, 10, 15 and 20 simultaneously active tags were tracked with an update rate of 0.1, 0.2 and 1 second over a duration of five minutes.

The results of all measurement series are visualized in Table 4.4. The plots show the capacity utilization of the SFs depending on the amount of currently active tags and set update rate. For better visualization, the bars within the plots are colored based on four different groups where each group consists of five tags. The y-coordinate indicates the number of tags ranging within a SF with an index between 0 and 15. This is because Decawave's SF structure defines a total of 15 TWR slots for ranging data (see Section 2.5.3). The x-coordinate represents the observed SF number. For simplicity, only the pattern for 30 continuous SFs were plotted. This still provides full insight into the measured series since the pattern constantly repeats itself.

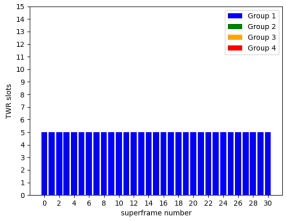
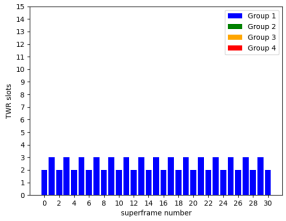
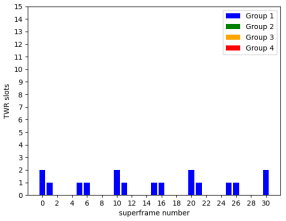
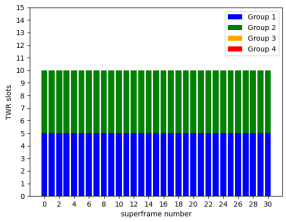
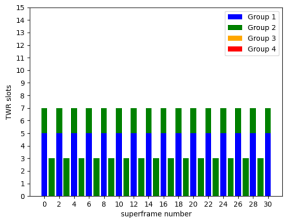
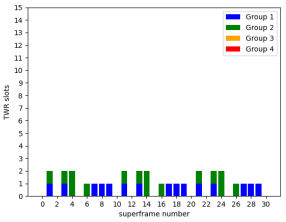
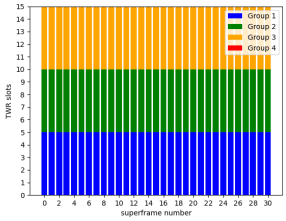
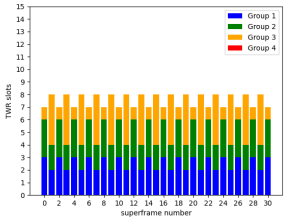
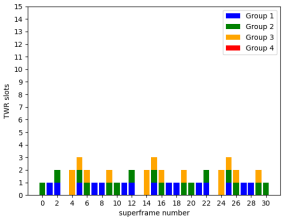
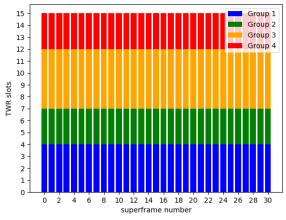
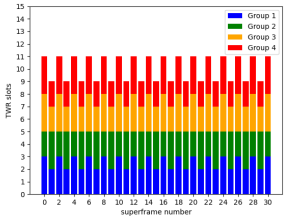
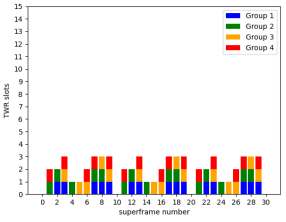
Tags	Update rate		
	0.1 s/(10 Hz)	**0.2 s/(5 Hz)**	**1 s/(1 Hz)**
5			
10			
15			
20			

Table 4.4: TWR slot utilization per superframe for a different amount of tags and update rates.

With an update rate of 10 Hz, a tag ranges in each of the SF numbers if it has a TWR slot occupied. With an update rate of 5 Hz, only every second SF is used for ranging and with an update rate of 1 Hz, only every tenth frame. The plots for 5 active tags within the first row of the table accurately sketch this behavior. The first plot of this row shows that each tag of group 1 has occupied a TWR slot in every SF number. In the second plot, two tags range in every even SF number and three tags within every odd SF number. The third plot shows a very low utilization of the TWR slots since most of the tags range on different SF numbers.

When looking at the plots in general, an even distribution of the colored bars can be observed. Plots with an update rate of 5 Hz or 1 Hz in particular show such distribution, because not every tag needs to occupy a slot within every SF. A relief of the slot occupation on a single SF number allows tags to better distribute and use free slots on SF numbers where the capacity is still low. Observing the plot with 15 tags and 5 Hz update rate demonstrates this approach. Instead of using all 15 slots on every second SF simultaneously, tags divide evenly on even and odd SF numbers. Here, seven and eight slots are occupied alternately on every two frames. This helps Decawave's system to stay on a constant capacity level and prevent it from reaching its maximum. The bottom right plot shows the most advanced example of this. 20 tags range within ten SFs where at no time more than three slots are occupied at once.

The main observation in terms of system capacity regards the plots with 15 and 20 tags on the highest update rate. As seen in the corresponding plot for 15 tags, every ranging slot within each SF is constantly occupied. Decawave states that the system is only capable of 15 tags at maximum location rate, but up to 9000 tags when using the minimum location rate of one measurement per minute [6]. Therefore, this scenario has reached the maximum capacity, but further aims at ranging with 20 tags. This measurement is visualized in the plot with 20 tags and a 10 Hz update rate. It can be seen that only four tags of group 1 have been able to reserve a free slot in the SF. Additionally, both groups 2 and 4 miss out slots for two of their tags resulting in a total of five non rangeable tags per SF. Investigating this specific measurement series shows that the system tags do keep their reserved slots occupied if possible and do not exchange them regularly. Four tags were not able to achieve a single localization while two other tags swapped out a ranging slot only once after approximately one fifth of the time. This led to one tag locating itself for 80 percent of the time and the other for the remaining 20 percent.

4.3 Waste-Walk Analysis

The application was tested during a schooling for an external company at the IIM institute, which included a waste-walk conducted as a practical exercise. Participants had to identify flaws in both workflow and resource management. See Section 3.1 for a recap of this use case. Figure 4.3 shows pictures of the setup room.

Figure 4.3: Use case setup in the Seminarraum of the IIM institute.

15 test candidates participated in the schooling. Eight of them assembled scooters as workers while the remaining seven investigated the process from an outside perspective. This was due to the fact that the assembly process was spread over a total of eight different workstations. One worker was assigned to each station. Three of the workers were stationary and did not change their working area during the process. The other five workers had to move between workstations, depending on given work instructions. For the learning aspect, the layout of the workstations and their workload was non-optimal on purpose. A scheme of the production cycle is shown in Figure 4.4.

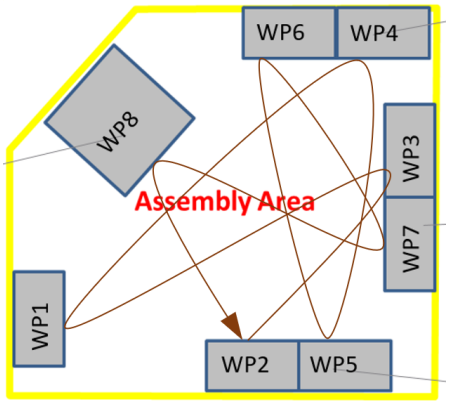
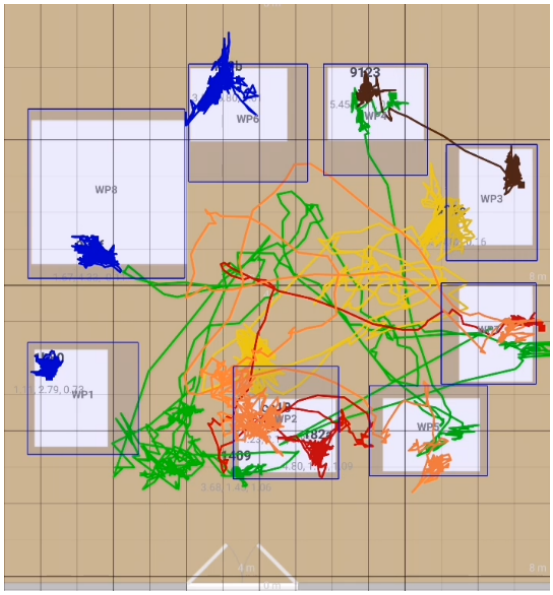


Figure 4.4: Workflow scheme of the production cycle.

4.3.1 Observations

To test the application, each worker received a tag. They were asked to take the tag with them while moving around the assembly area. During the use case, the paths of all tags were tracked to visualize the workers' movement. In addition, every workstation was marked as a region to track relevant timestamps for all production steps.



WorkerID	TagID	Start station
1	0x1409	WP4
2	0x9123	WP3
3	0x909E	WP2
4	0x182A	WP7
5	0xCA18	WP5
6	0x119B	WP6
7	0x990F	WP8
8	0x1B80	WP1

Figure 4.5: Spaghetti diagram and additional information about the workers.

The outcome of the app's spaghetti diagram can be seen in Figure 4.5. The table to the right provides information about the assignment of tags to workers as well as each worker's starting workstation. For simplicity, the thesis further mentions the worker IDs instead of their assigned tag IDs.

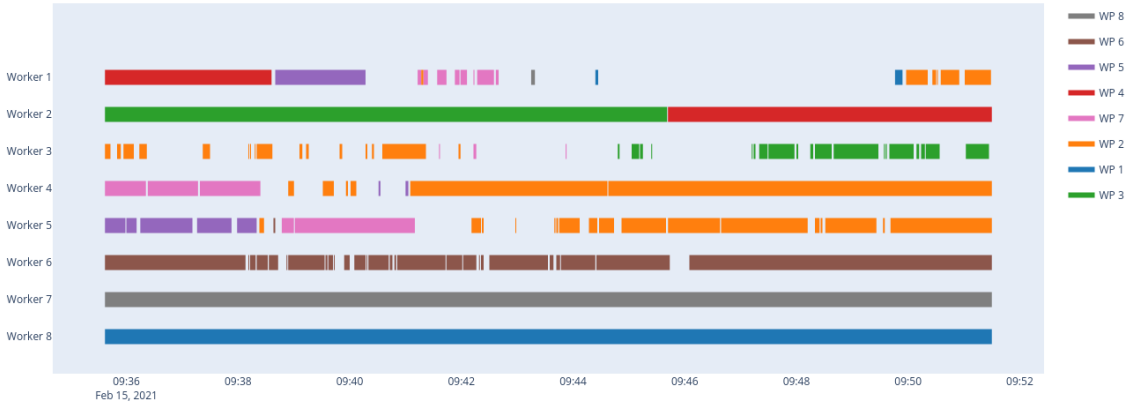


Figure 4.6: Visualization of the production process in chornological order for each worker over the tracked time span.

Figure 4.6 shows a Gantt chart of the whole production process. It visualizes the chronological sequence of production steps for each worker throughout the assembly run. In this context, each bar defines the exact time a tag was registered inside a marked region from entering until leaving. The coloring is used to better follow the production process. All relevant information for creating this chart was gathered from the app's exported history file.

The chart provides relevant insights into the wasted resources on both workload and workflow of the production process. Due to the non-optimal setup, the complexity in terms of workplace instructions and general workload differs between the stations. The chart shows that the workstations WP5 and WP7 were completed the fastest. On average a worker spends 2:10 minutes on WP5 and 2:25 minutes on WP7. Some production steps needed more time to be completed. WP3 took 7:95 minutes and WP2 8:05 minutes on average, which is almost four times longer than WP5.
The empty spaces between the colored bars further represent inefficiencies within the workflow. The different amount of workflow at each station caused several waiting times for workers. During these periods, workers were unable to do work because they needed to wait for the next station to become available. Worker 1 had to wait over seven minutes after leaving WP7 until he could enter WP2. Regarding workers 4 and 5, the chart shows that both were tracked at WP2 at the same time. However, this is false, because worker 4 entered the region before worker 5, but worker 5 had already placed his tag on the desk while waiting. Therefore, worker 5 also had an overall waiting time of more than ten minutes.

When looking at the three stationary tags in the spaghetti diagram, it can be seen that the tag at WP6 had a significantly larger deviation compared to the other two. This most likely had two reasons. At this workplace, many big parts reduced the available space on the table. This caused the worker to move the tag around from time to time. Furthermore, a compressed air screwdriver was used which produced strong vibrations that moved the objects on the table. When observing the last three timelines on the Gantt chart, the corresponding tag often left and re-entered the region while the other two stationary tags stayed inside their assigned region until the end.

There was made an interesting observation with regard to workers 2 and 3. Worker 2 never moved the tag around at the table except for changing the workstation. This results in very accurate measurements where at no time the tag was detected beside its expected region. The opposite extreme was worker 3 who put the tag in his back pocket. Because of this, the tag inside the pocket had NLOS conditions. This is visible in both the spaghetti diagram and the Gantt chart. The Gantt chart in particular shows that the tag was often tracked incorrectly which resulted in a large proportion of time where it was detected outside the expected region.

WorkerID	total distance [m]	
	within application	with threshold of 20 cm
1	79.684	48.088
2	5.661	2.957
3	69.25	30.943
4	16.946	6.273
5	45.587	25.683
6	20.388	4.098
7	10.856	0.895
8	2.688	0.747

Table 4.5: Comparison of the total calculated distances of the app and when using a higher threshold for filtering measurements.

At the end of the tracking, the app calculated the total distance of each worker based on all the tracked measurements. The app already filters consecutive measurements smaller than 10 cm for accuracy. However, within this use case the fluctuation in measurements was very high. Table 4.5 shows the total distances for each worker calculated by the app in comparison to the calculated distances when using a threshold of 20 cm. It can be seen that within the use case environment, a higher threshold would have been more suitable because it outputs more realistic distances.

To conclude the observation, the provided application data helped to identify wasted resources within the production cycle. Therefore, the app either visualized data live or pre-processed data for post-analysis.

5 Conclusion

This thesis investigated the feasibility of a UWB-based indoor localization system for an industrial purpose. The used UWB system was Decawave's scalable RTLS solution (DRTLS). Since the rise of Industry 4.0 and the shift to a decentralized production allow more agile processes, a feasible location system needs to be both precise and scalable for a large amount of tracked entities.

To evaluate the feasibility, the system first needed to be integrated within an industrial environment. For this purpose, an Android application was developed as a human machine interface to visualize location data provided by Decawave's DRTLS. The aim of the application was to improve the workflow and the safety in an industrial production. To do so, the application's various features process received data to simplify post-analysis for an end user.

To observe the accuracy of the system, the positions of eleven tags were measured over 11000 times each. This evaluation showed an average accuracy of around 18 cm. The best measured tag had an accuracy of under 2 cm which is very accurate. Additionally, the observed average standard deviation of a little over 2 cm could be considered low when keeping the big amount of measurements in mind. However, the measurements highly depend on the LOS conditions between tags and anchors. Modern industrial production plants often have many and sometimes moving obsticles spread throughout the area. Multiple anchors need to cover the area to guarantee that tags are able to range with their optimal four surrounded anchors at all times.

In terms of scalability, a total of twelve measurement series were taken to observe the system's behavior with different amounts of tags and update rates. While doing these measurements, it was possible to reach the system's maximum capacity. On the one hand, the scalability is limited to the available ranging slots per superframe (SF). This means that if all slots are occupied within one SF, the system cannot range more tags during this SF. On the other hand, the scalability depends on the set update rate. The higher the update rate of the tags, the higher is the ranging slot utilization per SF. If there are no more ranging slots available over continuous SFs, the system simply ignores the tags with no occupied slots. Hence, using the system in real industrial settings requires adjusting the update rate according to Decawave's limitations.

The UWB system in combination with the developed app was tested in a real industrial environment during a schooling at the IIM institute. The use case was a waste-walk to observe wasted resources of a production cycle. In this context, the app tracked the paths of workers during the run. Additionally, the spent time of each worker on the working areas was tracked. It was observed that the best results were achieved when the tag had good LOS conditions to the anchors. This has a significant impact on the quality of the observed paths. Moreover, a more stable position causes the tags to leave a marked region less often by accident. For continuous use of the app, it should be noted that for best results, tags should always be placed within sight to the anchors (e.g. placing them on the desk while working instead of putting them in the pocket). Also, regions should be marked approximately 30 cm larger in each direction to compensate with outlier measurements.

To summarize, the requirements of the app were met since wasted resources on workflow and workload were detected. Also, the evaluated UWB-based localization technology was feasible for the intended use case. However, for a large-scale integration of UWB-based systems in industries with an agile environment, further work is necessary.

Bibliography

[1] Decawave. MDEK1001 Kit User Manual. Technical Report Version 1.2, 2017.

[2] Decawave. MDEK1001 Product Brief. Technical Report Version 1.2, 2017.

[3] Decawave. MDEK1001 Quick Start Guide. Technical Report Version 1.2, 2017.

[4] Decawave. Decaforum: Anchor Selection, Quality Factor. `https://decaforum.decawave.com/t/anchor-selection-quality-factor/2495`, June 2018. Accessed: 2021-01-28.

[5] Decawave. DWM1001 Gateway Quick Deployment Guide. Technical Report Version 1.0, 2018.

[6] Decawave. DWM1001 System Overview And Performance. Technical Report Version 2.0, 2018.

[7] Federal Communications Commission (FCC). Revision of part 15 of the commission's rules regarding ultra-wideband transmission systems, 2002.

[8] H. Flatt, N. Koch, C. Röcker, A. Günter, and J. Jasperneite. A context-aware assistance system for maintenance applications in smart factories based on augmented reality and indoor localization. In *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–4, 2015.

[9] IEEE. IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs): Amendment 1: Add Alternate PHYs. Standard, 2007.

[10] IEEE. IEEE standard for low-rate wireless networks. Standard, 2020.

[11] C. Lian Sang, M. Adams, T. Hörmann, M. Hesse, and M. Porrmann. Numerical and experimental evaluation of error estimation for two-way ranging methods. *Sensors*, 19:616, 02 2019.

[12] Liuqing Yang and G. B. Giannakis. Ultra-wideband communications: an idea whose time has come. *IEEE Signal Processing Magazine*, 21(6):26–54, 2004.

[13] A. Muqaibel, A. Safaai-Jazi, and S. Riad. Ultra wideband vs . narrowband communications. Feb. 2004.

[14] C. Prinz, F. Morlock, S. Freith, N. Kreggenfeld, D. Kreimeier, and B. Kuhlenkötter. Learning factory modules for smart factories in industrie 4.0. *Procedia CIRP*, 54:113–118, 2016. 6th CIRP Conference on Learning Factories.

[15] F. Raschbichler. MQTT - Leitfaden zum Protokoll für das Internet der Dinge. `https://www.informatik-aktuell.de/betrieb/netzwerke/mqtt-leitfaden-zum-protokoll-fuer-das-internet-der-dinge.html`, June 2017. Accessed: 2021-02-21.

[16] S. Shah and T. Demeechai. Multiple simultaneous ranging in IR-UWB networks. *Sensors*, 19(24):5415, Dec. 2019.

[17] B. Silva, Z. Pang, J. Åkerberg, J. Neander, and G. Hancke. Experimental study of uwb-based high precision localization for industrial applications. In *2014 IEEE International Conference on Ultra-WideBand (ICUWB)*, pages 280–285, 2014.

[18] A. Syberfeldt, M. Ayani, M. Holm, L. Wang, and R. Lindgren-Brewster. Localizing operators in the smart factory: A review of existing techniques and systems. In *2016 International Symposium on Flexible Automation (ISFA)*, pages 179–185, 2016.

[19] H. Team. Quality of Service 0,1 & 2 - MQTT Essentials: Part 6. `https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/`, February 2015. Accessed: 2021-02-24.

[20] J. Vishwesh and R. P. Ultra wide-band (UWB): Characteristics and applications. *International Journal of Recent Trends in Engineering and Research*, 4(6):45–52, June 2018.

[21] F. Zafari, A. Gkelias, and K. K. Leung. A survey of indoor localization systems and technologies. *IEEE Communications Surveys Tutorials*, 21(3):2568–2599, 2019.