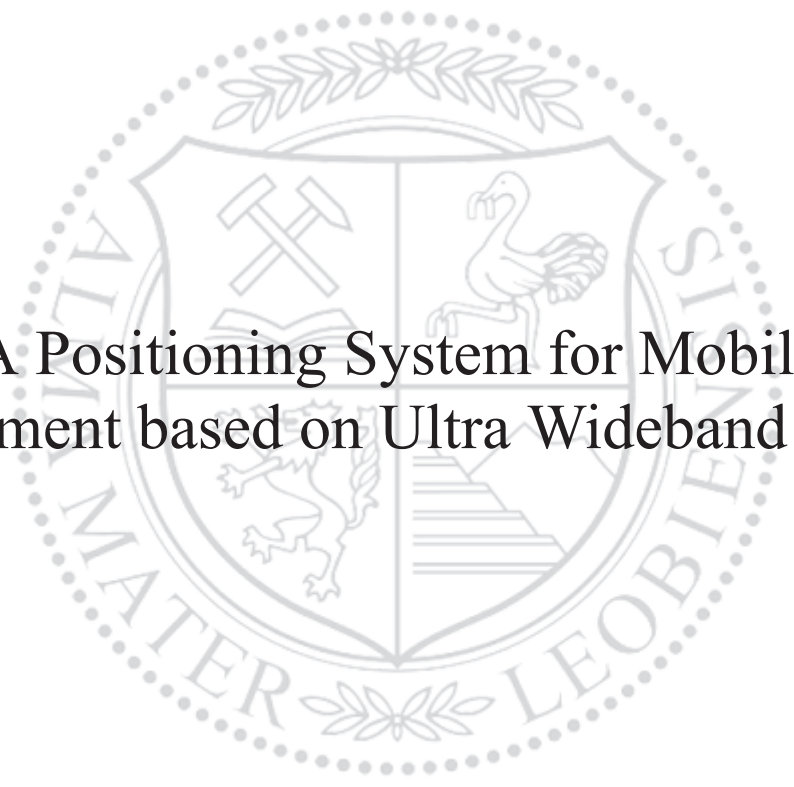




Chair of Automation

Master's Thesis



A Positioning System for Mobile
Equipment based on Ultra Wideband Radio

Lea Plessing, BSc

May 2021

Abstract

This thesis investigates the use of Ultra Wideband (UWB) technology as both position- and orientation measurement system of dynamically reconfigurable equipment. The results of implementing the UWB based system and the design of its middleware are presented. No installation of stationary infrastructure is required for the architecture that has been developed. Standard automation protocols have been selected and implemented to establish the connections from the middleware to the plant network and to eight UWB sensors, in order to communicate the range settings and measurements. An application has been programmed that manages the bidirectional data flow and executes the algorithm that yields the relative position and orientation. As part of the algorithm, a Least-Squares computation is performed on an overdetermined system of four range measurements. Furthermore, a running average computation is applied to reduce the noise of the incoming and out coming data.

The developed system has been tested, both on a test setup and on the real plant. The results have been analyzed, the causes of uncertainty determined and proposals made on how to further improve the complete system.

In five of the ten possible constellations of the mobile equipment, the measurement system achieves accuracies below 200mm. As documented by the tests, the current middleware for the UWB based positioning system, fulfills both the accuracy and real-time requirements for the positioning of mobile processing equipment in outdoor locations. This is achieved without requiring the installation of stationary infrastructure.

Index Terms

Ultra Wideband, Industrial Middleware, Plant Automation, Positioning System, Outdoor Positioning, Positioning without Stationary Infrastructure

Kurzfassung

Diese Diplomarbeit untersucht den Einsatz der Ultra-Breitband-Technologie (UWB) als Positions- und Orientierungsmesssystems von mobilen Anlagenelementen. Die Anwendungsergebnisse des UWB-basierten Systems und dessen Middleware-Konzeptionierung sind dargelegt. Für die entwickelte Konstruktion ist keine stationäre Infrastruktur notwendig.

Für die Datenverbindung der Middleware zum Anlagennetzwerk und zu acht UWB-Sensoren wurden Standard Automations-Protokolle ausgewählt und implementiert, um die Einstellungen des Positionierungssystems und die Messergebnisse zu übermitteln. Es wurde ein Programm entwickelt, welches den bidirektionalen Datenfluss realisiert und den Algorithmus ausführt, welcher Position und Orientierung berechnet. Als Teil des Algorithmus wird die Methode der kleinsten Quadrate an einem überbestimmten System von vier Messlängen angewandt. Außerdem wird eine laufende Durchschnitts-Berechnung durchgeführt, um das Rauschen der eingespeisten und ausgegebenen Daten zu reduzieren.

Das entwickelte System wurde sowohl an einem Testaufbau als auch an einer echten Anlage getestet. Die Ergebnisse wurden ausgewertet, die Ursachen für Unsicherheiten eruiert und Verbesserungsvorschläge für das komplette System vorgelegt.

In fünf der zehn möglichen Anordnungen zweier verschiebbarer Anlagenelemente erreicht das Messsystem Genauigkeiten von unter 200mm. Wie in der Arbeit gezeigt wird, erfüllt die entwickelte Middleware für das UWB-basierte Positionierungssystem sowohl die notwendige Genauigkeit, als auch Echtzeit-Fähigkeit für die Positionierung von mobilen Anlagenelementen im Freien. Dies wurde erreicht ohne die Abhängigkeit von stationären Infrastrukturen.

Schlagwörter

Ultra-Breitband, Industrielle Middleware, Anlagenautomatisierung, Positionierungssystem, Outdoor Positionierung, Positionierung ohne stationäre Infrastruktur



EIDESSTÄTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich diese Arbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt, und mich auch sonst keiner unerlaubten Hilfsmittel bedient habe.

Ich erkläre, dass ich die Richtlinien des Senats der Montanuniversität Leoben zu "Gute wissenschaftliche Praxis" gelesen, verstanden und befolgt habe.

Weiters erkläre ich, dass die elektronische und gedruckte Version der eingereichten wissenschaftlichen Abschlussarbeit formal und inhaltlich identisch sind.

Datum 13.05.2021

Unterschrift Verfasser/in
Lea Plessing

Acknowledgements

I would like to thank those who believed in me, supported me and helped me getting closer to my goal.

The one who supervised this thesis and who animated me to pursue my interest in data science is Univ. Prof. Paul O’Leary to whom I am much obliged.

I want to express great acknowledgements to Michael Habacher from eSENSEial Data Science GmbH for his support and for giving me the opportunity of developing such an exciting real-world application.

To my former mentor Assoc. Prof. Ewald Fauster I am ever thankful for the unprecedented teaching quality I received by which I was introduced to practical automation.

Furthermore, my special esteem goes to Eva whose effective and motivating instructions I will always keep in mind.

And to my family I owe great appreciation for their support and confidence.

Counting myself fortunate to have such friends, I want to thank them for their encouragement and affirmations.

My conclusive expression of gratitude goes to my fiancé Max for always believing in me and standing by my side with helpful advice.

Contents

1	Introduction	9
2	System Concept	13
2.1	Purpose of and Requirements on the System	13
2.2	Physical Environment	15
2.2.1	Influences on Ranging Signal	15
2.2.2	Geometric Setup	17
2.3	Demands on the Algorithm	18
2.3.1	Translation of the Real World Problem	19
2.4	Technical Environment	21
2.4.1	IPC	21
2.4.2	Distance Meters	22
3	UWB Technology	23
3.1	Technological Principal	23
3.2	Legal Regulation	25
3.3	Features and Characteristics	26
3.3.1	Spectral Density	26
3.3.2	Data Rate	27
3.3.3	Accuracy	27
3.3.4	NLOS Behavior	27
3.4	Recent Developments	28

3.4.1	Channel Sharing	28
3.4.2	NLOS Performance	29
3.4.3	Clock Synchronizations	30
3.5	UWB Ranging Applications	30
3.6	Critical Assessment	31
3.6.1	Chances	31
3.6.2	Limits	32
4	Choice of Bus Protocol	33
4.1	RS485 - Background	33
4.2	Physical Connection	34
4.3	Bus Comparison	35
4.4	Modbus RTU	36
4.4.1	Protocol Principal	36
4.4.2	Memory Map	38
4.4.3	Messaging	39
4.5	Interface Preparation	40
4.5.1	Register Table	40
4.5.2	Interface Configuration at the Middleware	41
5	Application Development	44
5.1	Hardware Setup	44
5.2	Object Oriented Programming	45
5.2.1	Objects and Classes	46
5.2.2	Inheritance	47
5.2.3	Thread	47
5.2.4	Lock Object	47
5.2.5	Queue	49
5.2.6	List	49

5.2.7	Dictionary	49
5.2.8	Logging	49
5.3	Implementation of the Threads	49
5.4	Data Exchange between Threads	50
5.4.1	Global Configuration	50
5.4.2	Queue of Ranges	53
5.5	Modbus Thread	54
5.5.1	MT - Initialization	54
5.5.2	MT - Loop	56
5.6	Algorithm Thread	56
5.6.1	AT - Initialization	56
5.6.2	AT - Loop	57
5.7	OPC-UA Thread	57
5.7.1	OT - Initialization	58
5.7.2	OT - Loop	58
6	Results	72
6.1	Modbus Connection	72
6.2	Algorithm Performance	76
6.2.1	Tripod Tests	76
6.2.2	Tests on Plant Units	77
6.3	OPC-UA Connection	84
6.4	Summary	85
7	Conclusion	86
A	OPC-UA Variables	91
B	Application Implementation	94

Chapter 1

Introduction

This thesis addresses the measurement of the relative position and orientation of mobile reconfigurable machines with respect to each other using the Ultra Wideband radio (UWB) technology. This is required to enable the fast and efficient reconfiguration of mobile processing units at new application sites. Additionally, there should be no need for any stationary infrastructure, since the time spent on the installation of such infrastructure would be counterproductive and require specialist staff.

While passing material between processing machines, e.g. through conveyor belts, the relative position and orientation between two succeeding units must be maintained to allow a successful material hand-over. The developed measurement system provides the necessary relative location information to align the units in the first place and to observe the relative position and orientation throughout the sorting process. Since the installation of infrastructure is not required, the same plant can be more easily moved and installed at new sites.

Considering the harsh industrial environment, the system needs to perform despite multipath propagation of the UWB signal which causes measurement offsets: Electromagnetic waves are reflected on plain metal surfaces, such as the massive steel frames of the interconnecting conveyor belts, the surrounding units or other dynamic machinery and moving parts. The reflected waves lead to time variance in the incoming signals as the reflections arrive later than the directly travelling signals, thus, causing measurement errors.

Other wireless communication standards, such as Wi-Fi and Bluetooth, are regularly used on modern industrial sites and even needed to control the plant units themselves. Therefore, their potential interference must be taken into account. Furthermore, a tight accuracy limit of 200mm poses a high demand on the positioning algorithm and the general ranging architecture.

Another challenge is the request for standardized automation protocols as the mobile equipment needs to be useable in different communication networks and allow the integration of various sensor designs. Apart from a flexible connectivity, sufficient data rate and reliability must be met, so that the system can be used as parking aid. The task is made considerably more challenging by the fact that the system should not depend on any infrastructure, e.g. a set of stationary sensors on the sides or wires in the ground, but should communicate only with the sensors of the next unit.

The literature review revealed research projects that determine only the real-time distance between two objects without requiring stationary infrastructure, such as the collision avoidance systems for autonomous cars compared by Ponte Müller [1]. However, the discussed systems do not detect the relative orientation between two cars.

Other works address the simultaneous determination of position and orientation, but depend on stationary infrastructure, such as the indoor positioning system based on passive radio frequency identification (RFID) developed by Shirehjini et al. which uses RFID Tags and stationary RFID carpets to navigate [2]. During the literature survey for this thesis, no implementation was found to determine the relative position and orientation in a real-time manner which doesn't require stationary infrastructure.

The main scientific contribution of this thesis is a new architecture which is composed of two UWB sensor pairs, see figure 1.1. One is attached to the first unit and consists of two Anchors, i.e. sensors that initiate a distance measurement, while the other pair consists of two responding Tags. Both Anchors range to both Tags respectively resulting in the measurement of four lengths while the distances between the individuals of one pair are considered as constant. The positioning algorithm is performed by an Industrial PC (IPC) which is connected to the Anchors and ingests the four measured distances to determine position and orientation without requiring external infrastructure.

Any implementation of the system belongs to one unit and is fundamentally structured with hardware elements on the one hand, i.e. UWB Anchors and Tags, the IPC and the physical connection in between, and with software elements on the other hand, i.e. the application realizing the entire bidirectional data transfer from the sensors to the plant network and the data processing including the algorithm. While the n UWB Anchor pairs feed their quartets of distance measurements to the IPC and receive settings from it, the m UWB Tag pairs only receive settings from the IPC, where n is the maximum number of relatively locatable units and m the maximum number of units that receive the location of the system carrying unit.

The necessary hardware, firmware and embedded software components to implement a functional

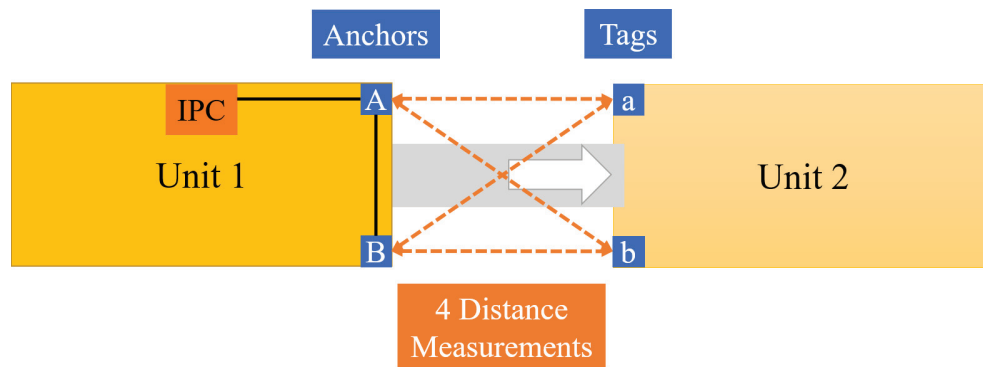


Figure 1.1: The measurement system's architecture requires an Anchor pair on one of the units and a Tag pair on the other. By ranging from each Anchor to each Tag, four measurements are obtained that yield the position and orientation after being processed on the IPC.

prototype have been integrated into the system's middleware, see figure 1.2. Hence, in addition to the new setup for an autonomous ranging system, the whole middleware linking the sensors to the plant network has been designed and established. An additional benefit of this work is the implementation of only standardized industrial protocols, i.e. Modbus RTU over RS485 and OPC-UA, for the communication from the distance meters to the plant network.

In order to evaluate the system's functionality, it has been installed on the mobile equipment

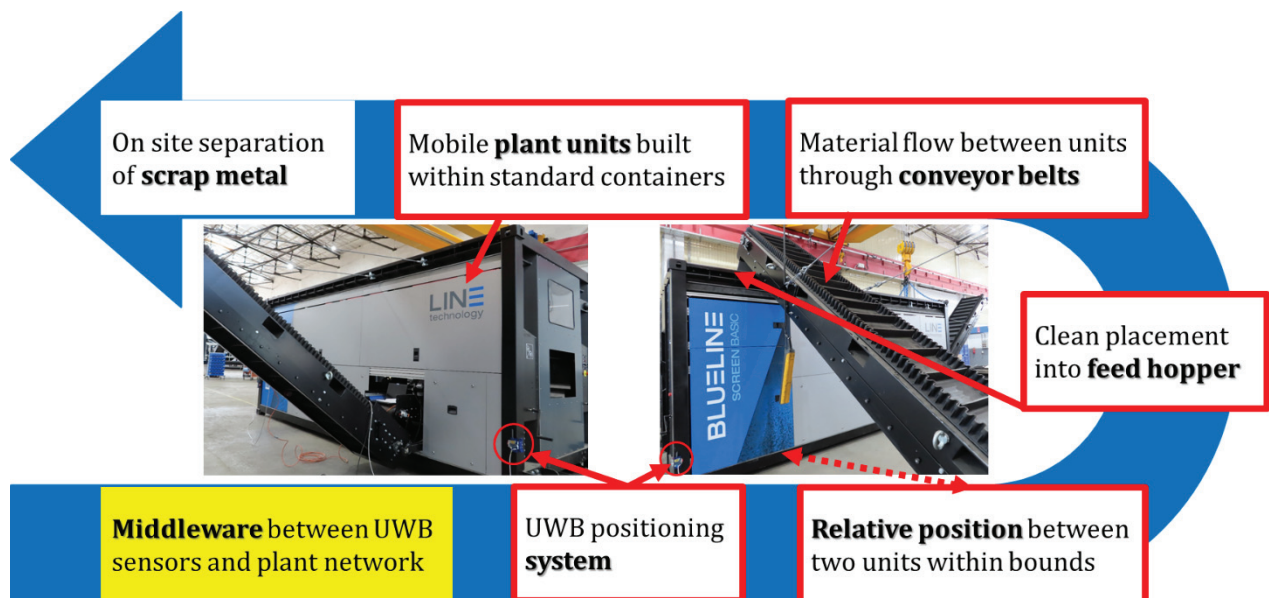


Figure 1.2: System design and test setup including conveyor belts: The positioning middleware of the autonomous positioning system links the UWB sensors mounted on the units to the plant network and performs the positioning algorithm.

of a scrap metal sorting plant. All ten possible relative arrangements of two units in the size of shipping containers have been tested while connected by conveyor belts, see figure 1.2.

The results indicate that an accuracy of 200mm is achievable when the angle of the incoming signal is not too steep.

A result of this implementation is the new possibility of positioning and orienting dynamically reconfigurable components with respect to one another with a minimum requirement of personnel.

Chapter 2

System Concept

The aim of this chapter is the definition of the physical and technical environment surrounding the location system and the layout of the middleware's architecture. Furthermore, the demands on the system's precision, responsiveness and feasibility will be specified.

2.1 Purpose of and Requirements on the System

Each unit of a process line consists of a sorting machine, that is built into the framework of a mobile shipping container. The described modularity allows flexibility w.r.t the processed materials and enhances transportation of the plant. Connected by dismountable conveyor belts, the mobile equipment forms an adaptable process line, where each conveyor belt is only attached to the feeding unit and drops the material into the collecting vessel which is fitted into the receiving unit.

By continuously measuring relative position and orientation, the misalignment can be deduced which indicates whether the material flow is interrupted or close to interruption. The two parameters are calculated out of four range measurements performed by four sensors: two on the feeding unit "A" and "B" and two on the receiving unit "a" and "b" which deliver the four ranges "A-a", "A-b", "B-a" and "B-b", see figure 2.1. To prevent an interruption of the material flow, the maximum tolerable misalignment between drop-off point of the belt and the center of the vessel are limited by 200mm in the horizontal plane.

On the one hand, the location system has to measure the relative position and orientation between the plant units throughout the whole runtime of the process. A displacement could be caused e.g. by a landslide or the impact of heavy machinery, moving along the process line. On the other hand, the positioning system functions as a parking assistant, when the units are initially

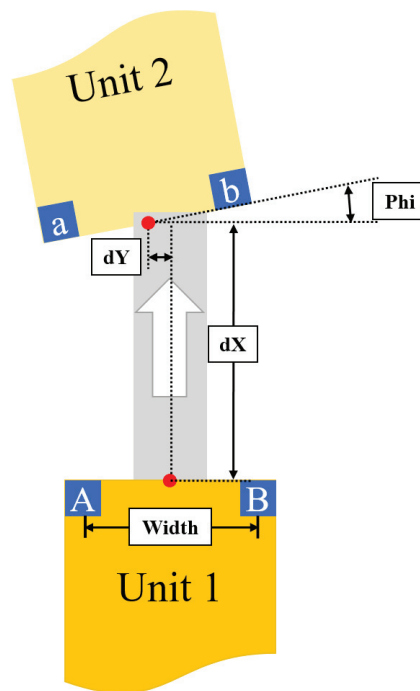


Figure 2.1: The main task of the middleware is the continuous output of the relative position between two units. In the case of two linear aligned units the sensors “A” and “B” of the first unit together with “a” and “b” from the second unit form a measurement constellation of four involved sensors.

installed. In order to ensure swift positioning, the relative location data should be updated at frequencies of approximately 1Hz.

If two units are linearly aligned, the four sensors represented in figure 2.1 are sufficient. However, there might occur the necessity of parallel or vertical positioning as well as the alignment of more than two units relatively to one another in which case a higher amount of sensors would be necessary. At most, eight sensors at once are active per unit which are divided into four sensor pairs, each pair covering one of the sides. Every unit of the process line is also equipped with an IPC, that executes the services of the middleware, see figure 2.2.

The middleware, that is developed in this work, represents the core of the positioning system, by managing the data flow between the superior network of the process line and the distance meters, as well as by calculating the relative location from the raw sensor data, see figure 2.3. All distance meters, that are mounted on one unit, are connected to the IPC with the developed application and supply it with their measurements, which are the input of the positioning algorithm. As a result of the algorithm, the relative offset and orientation are transferred to the network of the whole production line.

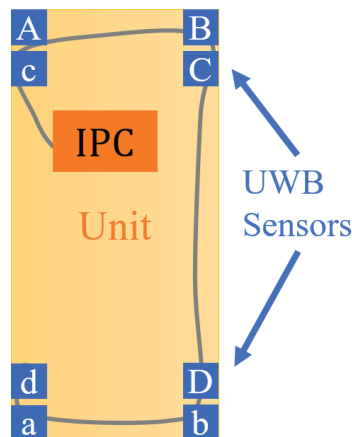


Figure 2.2: One unit equipped with eight sensors in total, all delivering range data to the middleware which is executed by one IPC per unit.

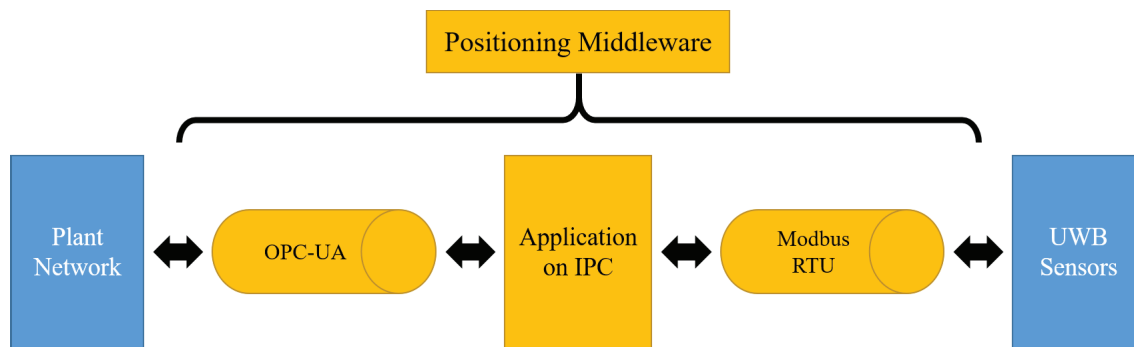


Figure 2.3: The middleware represents the core of the location system and includes the interfaces to the network and UWB sensors, as well as the computations.

2.2 Physical Environment

In most cases, the whole plant will be stationed at a metal waste disposal site, thus, outdoors in an industrial environment.

2.2.1 Influences on Ranging Signal

Within this section, the influences of that environment on ranging signals are discussed by way of multipath propagation, no-line-of-sight (NLOS) scenarios and signal interference. On referring to a line-of-sight (LOS) scenario, the shortest connection between two communicating sensors is optically unobscured, whereas NLOS means that this direct line is interrupted by some material unlike air, regardless of the dimensions of the object. Another important term is multipath propagation, where reflected and refracted signals reach the receiver [3, p. 24]. Depending on

the antenna geometry, the transmitter sends a signal that propagates in more than one direction. If there are objects close to the LOS, that cause reflection or refraction of the impulse, multiple time dispersed signals arrive at the receiver.

NLOS

It was found out that at least eight UWB sensors per unit are necessary for the system to execute reliable measurements. If only one UWB sensor is mounted per edge, some constellations of neighboring modules would require the signal to pass through the container frame. The UWB signal cannot overcome an immediate blockage to that extent, as there is no space for it to bypass the obstacle through multipath propagation, like reflecting on the ground or walls. Hence, two sensors per edge, each aligned to one of the two joining sides of the unit, are the minimal equipment. With the two distance meters per edge mounted at the least obscured vertical position, i.e. right below the container roof, the system must still deal with NLOS situations. In our specific example that scenario arises because the conveyor belts can be mounted on different locations on each plant unit. In some of the possible process line constellations the belts obscure the direct communication path, no matter at which height the distance meters are mounted on the edges. As the belt frames are mainly made of steel, they are not permeable to electromagnetic signals, thus causing NLOS communication between the distance meters.

Multipath Propagation

There will be multipath propagation of the UWB signals, brought about by reflections on nearby metal planes, like the container front at the back of each distance meter, the surface of the surrounding units and the frame of the conveyor belt – aside from surrounding scrap containers or vehicles, etc.

Potentially Interfering Signals

As the usual operational area is outdoors, the system must cope with potential electromagnetic interference, such as Wi-Fi, GPS and Bluetooth. Even on the unit itself there will be a module, which communicates with the overall network via Wi-Fi.

2.2.2 Geometric Setup

Each plant unit has the approximate dimensions of 6m in length and 2.5m in depth and height, while the distance to the next one in the process line will be approximately 4m or 6m according to the length of the feed belt in between. Orientation offsets between two modules are approximately 0° , 90° , 180° or 270° - so either in a rectangular or parallel succession in the production line. However, the corners of the units will not be aligned necessarily, as the feed belt mounts have different positions, depending on the specific unit. Considering the preset mounting positions and available lengths of feed belts, ten different relative positions of two units are possible, see figure 2.4.

The evaluation of the relative horizontal position (distance and orientation) from one unit to

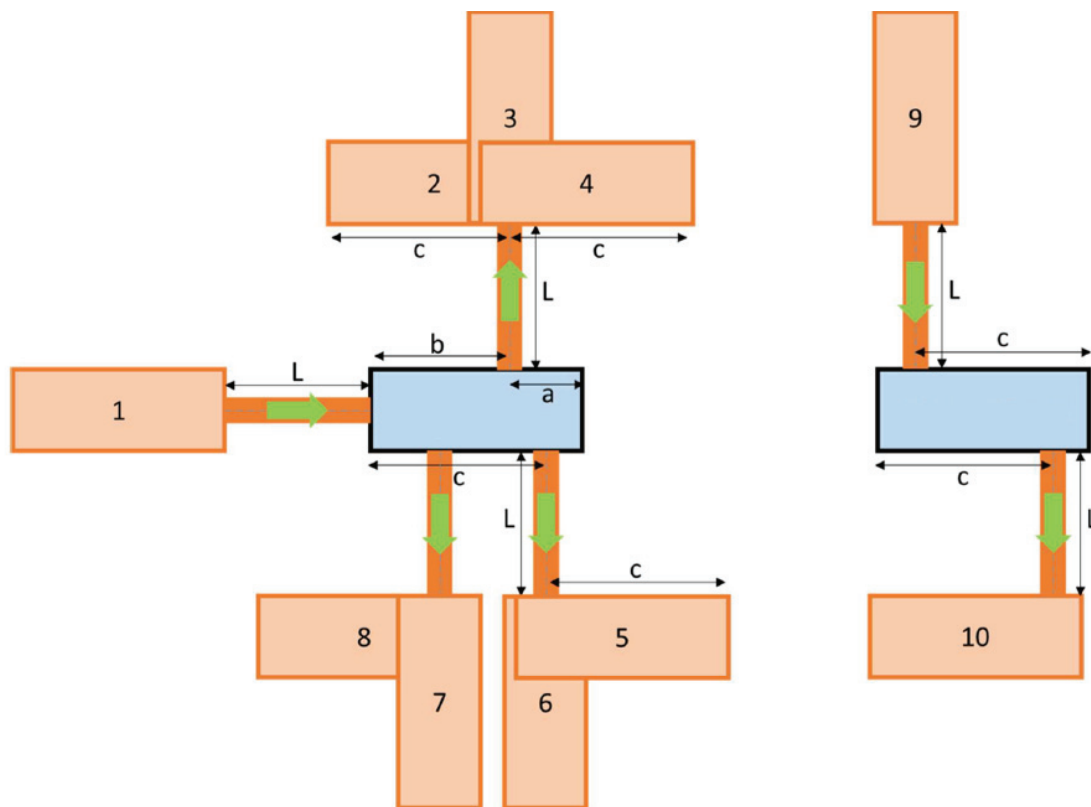


Figure 2.4: Possible alignments within the process line: Horizontal length of feed belts L is 4.1m; Mounting position offsets b and c are about 4.5m and 5.3m respectively. The green arrows on the feed belts indicate the direction of material flow.

the other involves four sensors. All edges are equipped with two perpendicularly aligned UWB sensors that are placed at the four top corners, just below the container roof. For a location task, only those sensors are used that face the direction of the other unit and are therefore mounted left and right of the connecting conveyor belt. In the case of position no. 1, the sensors "A" and "B", as shown on the left side in figure 2.5, both measure the lengths to "a" and "b", yielding

four lengths: two enclosing the feed belt and two crossing it. In a similar manner this applies to all ten possible arrangements, e.g. for position no. 5, the relevant sensors would be "C" and "D" of "Unit 1" and "c" and "d" of "Unit 2" on the right side of figure 2.5. While the minimum

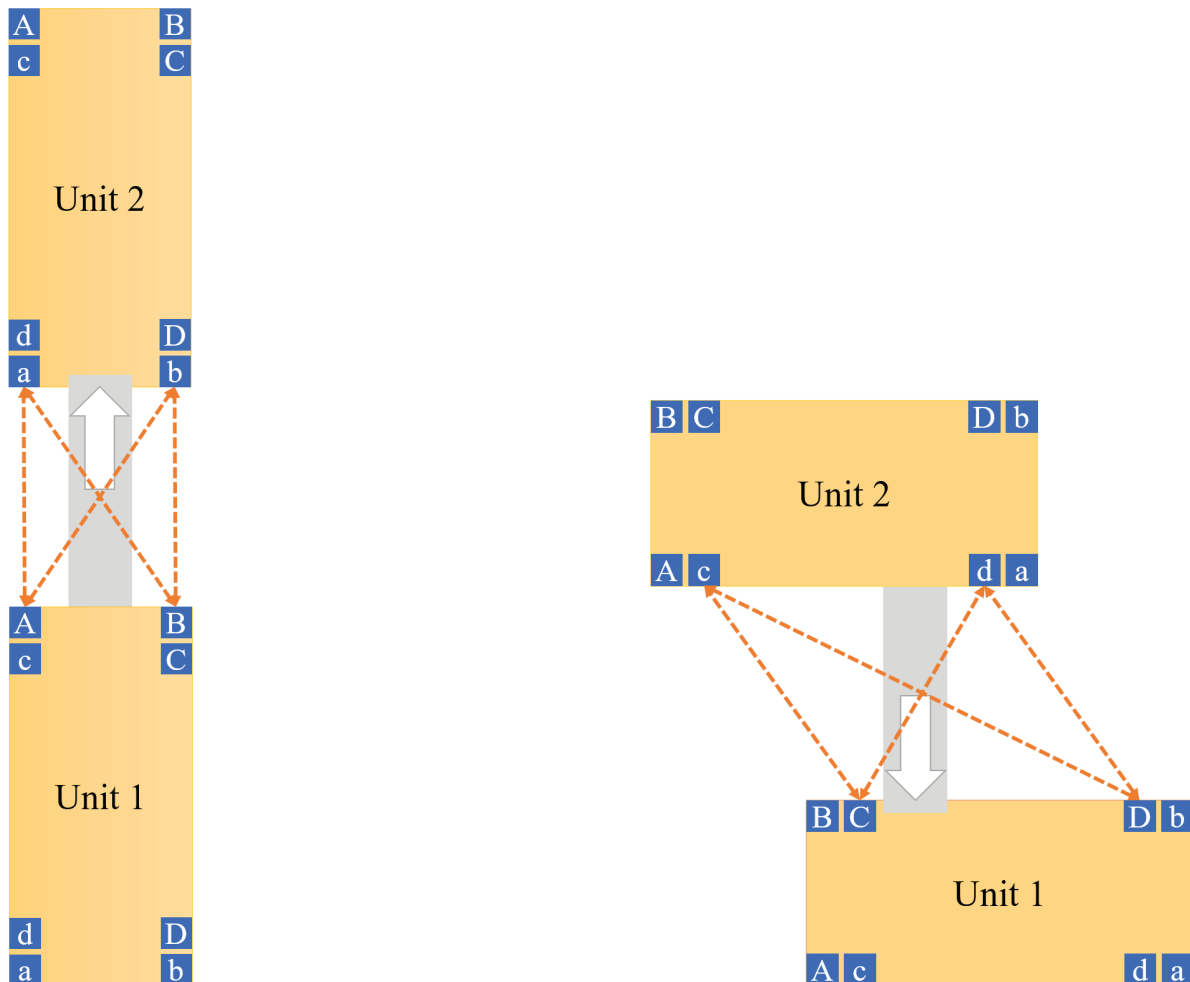


Figure 2.5: Constellation between two units no. 1 and 5. Names and positions of the UWB sensors on two succeeding units. On each corner two sensors are mounted perpendicular on the top of the edge.

offset between the units' facing sides always consists of the horizontal length of the feeding belt minus a constant overhang, the measured distance can reach up to 11m, e.g. the long diagonal measurement of position no. 5, see figure 2.4.

2.3 Demands on the Algorithm

The constant output of real-time position data is the key task of the middleware. By continuously processing the raw measurements of the sensors and applying algebraic methods on them, the

horizontal offset in x and y direction, together with the angle between the units are calculated and updated fluently. The algebraic operations, through which the errors are determined and the minimum found, effect the computational complexity and therefore the frequency and promptness at which value updates are achieved, as well as on the output's accuracy.

2.3.1 Translation of the Real World Problem

In advance, a geometrical concept for an efficient algorithm has been developed, which has settled the necessary arrangement of the involved sensors. The amount of ranging paths has been elaborated, that are needed for effective pre-processing, based on trilateration. The programmatic implementation of the positioning algorithm and for that matter necessary data processing are included in this work. Additionally, the mathematical principle originated and at the Chair of Automation [coa.unileoben.ac.at] and poses an important tribute to this thesis. Four lengths are to be determined between two neighboring plant units, in order to obtain the relative horizontal position, see figure 2.6. The container fronts facing each other are equipped with two sensors,

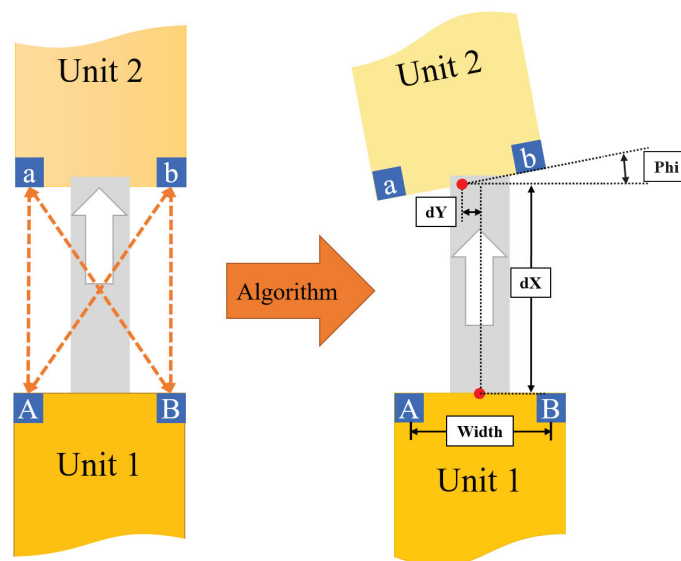


Figure 2.6: Per positioned target unit, four lengths are used to calculate the offset and angle. The running average is computed per each length separately.

each mounted on the very edge of the front. Both sensors on the same unit and measure the distances to both members of the opposite pair, resulting in four measurement paths in total: two outlining the area between the containers and two paths crossing it thus connecting the diagonal corners. With the distances given between the sensor mounts on the containers, three measured distances would theoretically yield the sought offset and angle. However, as four sensors are

also needed to calculate three lengths, the fourth length provides redundancy without any more technical effort. The overdetermined system has theoretically no exact solution and therefore results in an approximation calculation.

The following main components of the algorithm are explained within this section:

Running Average

A running average computation is the calculation of the mean value on entries of a row, list, array or another succession of data elements, like the buffer of an interface. The number of entries stays the same but the whole running average window moves along the entries. Various phenomena, like multipath propagation or noise of other signals, disturb the UWB communication and as a consequence cause scatter of the range outputs. In the case of this thesis, the running average computation is used to smoothen that data, in order to obtain steadier position calculations.

Trilateration

Two-dimensional trilateration is the calculation of the intercepting points between two ranges originating at known center points. Only if those center points are closer than the sum of the range values, can there exist intercepting points. If the center points are equal to the sum of two ranges, the solution is singular. Otherwise, there are two possible solutions for intercepting points.

Singular Value Decomposition (SVD)

Golub and Van Loan state that the importance of the SVD is hard to overestimate, especially in data analysis and approximation computations performed on matrices [4, pp. 76-81].

A SVD can be applied on square or rectangular matrices and follows the idea of decomposing a matrix of n columns into n blocks of information, where each block is composed of matrices U and V^T as well as n singular values (SVs). The higher the SV of the block is, the more impact does this block have on the solution. Consequently, the block with the highest SV represents the best fit. In other applications, like compressions or noise reduction, the SVD is applied to cut out one or more blocks correlating to the smallest SVs. [5, pp. 195-197]

The orientation calculation within the algorithm makes use of that correlation by extracting from V^T the orientation value with the highest significance, i.e. the orientation yielding the best approximation in a Least Squares sense.

2.4 Technical Environment

The technical environment includes the hardware and software related boundary conditions. As this thesis focuses on a universal and therefore brand independent approach, the interfaces to the distance meters and the superior network, will be evaluated, without going into manufacturer specific details. By adapting the specific type of interfaces, the concept of the developed middleware can be transferred to another technical environment.

2.4.1 IPC

To facilitate communication and maintenance, the hardware embedding the middleware is of the same manufacturer as already installed other nodes in the plant network. The specific model of IPC was selected for its various integrated ports and its high process power (1.2 GHz quad-core) and memory (1 GB RAM), that define the computational velocity of the positioning calculus. Integrated are one RS485, two USB-A and two RJ45 Ethernet ports, while the IPC supports various industrial automation protocols, which are prepared for user friendly implementation, including OPC-UA. The nodes in the superior network communicate over an OPC-UA network in a Server-Client topology. By implementing the protocol within the middleware and connecting the IPC via Ethernet to the plant network, it can participate as OPC-UA-TCP Server. [6]

One of the preconditions is the data supply from the middleware to the network over OPC-UA, which is a machine-to-machine communication protocol. Nowadays, OPC-UA is an established technology in industrial process control systems, to exchange real-time data. The standard allows data exchange between automation devices from different manufacturers in a Server-Client mode. [7, pp. 297-298]

In this specific work, the Server is the IPC, i.e. the developed middleware, while the Client is a control device within the plant network.

By means of a predefined namespace, the Data Excess (DA) Server assigns unique addresses to all nodes within an OPC-UA network. Each node is uniquely identified by three parameters: first, the namespace index, which defines the URI, i.e. naming authority in charge of assigning node IDs, second, the identifier type, i.e. the data type of the address, which is in the presented project a string and third, the identifier itself, e.g. "Range00a". Within an OPC-UA network, all variables are structured in tree hierarchies. [8]

While all nodes, that contain branches are referred to as object nodes, those elements which are the leafs at the end of a branch are the variable nodes. Additionally, to the node ID, that every node in the tree possesses, the variable nodes consist also of a value. [7, pp. 297-298]

To enhance OPC-UA communication with the Client, the Server needs to provide three parameters: the namespace URI, the Server URL, which is needed by a Client to access a service, and finally the Server name.

2.4.2 Distance Meters

In the following chapter, the properties and strengths of UWB as ranging devices and the reason for its application in the presented location system are discussed in detail. The specific model was decided beforehand, which offers a RS485 interface, as well as a Micro-USB connector and uses the Time-of-Arrival (ToA) principal, where the round-trip-time between two devices is measured.

One sensor initiates the ranging by addressing the opposite by its sensor ID. Those initiating sensors, that also calculate the distance between themselves and the communication partner, are called Anchors. On the other hand, the sensors, that only respond to an incoming call with their ID are called Tags. [9]

The included RS485 interface facilitates the connection to the IPC, which also provides a RS485 port [6].

Chapter 3

UWB Technology

This chapter gives an introduction to the UWB technology, which is used to obtain the raw range information for the developed positioning middleware.

3.1 Technological Principal

By definition, UWB signals are electromagnetic waves (EMW) with low transition power and a -10dB bandwidth above the lesser of the two: 500 MHz or 20% of the center frequency for center frequencies below 2.5GHz. “Ultra Wide Band” refers to the broad frequency spectrum that is emitted at the same time in short pulses, namely Gaussian pulses with amplitude distributions in the shape of a Gaussian curve, as well as its derivatives. [10, p. 369], [11, p. 2]

With only a few cycles of a radio frequency (RF) carrier, the baseband impulse consists of low duty cycles, as typically produced by impulse or step-excited antennas. As a consequent, UWB signals achieve significantly lower power spectral densities than other technologies, see figure 3.1. [12, pp. 92-93]

Their large bandwidth and low power let UWB signals appear like noise next to other wireless technologies, thus, allowing to share the frequency spectrum [13, p. 14]. A close look will be taken into UWB radio systems as ranging technology to lay the basis for the following development of a positioning middleware supplied by UWB sensors. How UWB signals are interpreted and translated into a location can be divided in to 4 major groups: Received Signal Strength Intensity (RSSI), Angle of Arrival (AoA), Time of Arrival (ToA) and Time Difference of Arrival (TDoA) [15, pp. 70-71].

- The method of RSSI relies on a path-loss model, which requires knowledge about the relation between power loss and distance [15, p. 71].

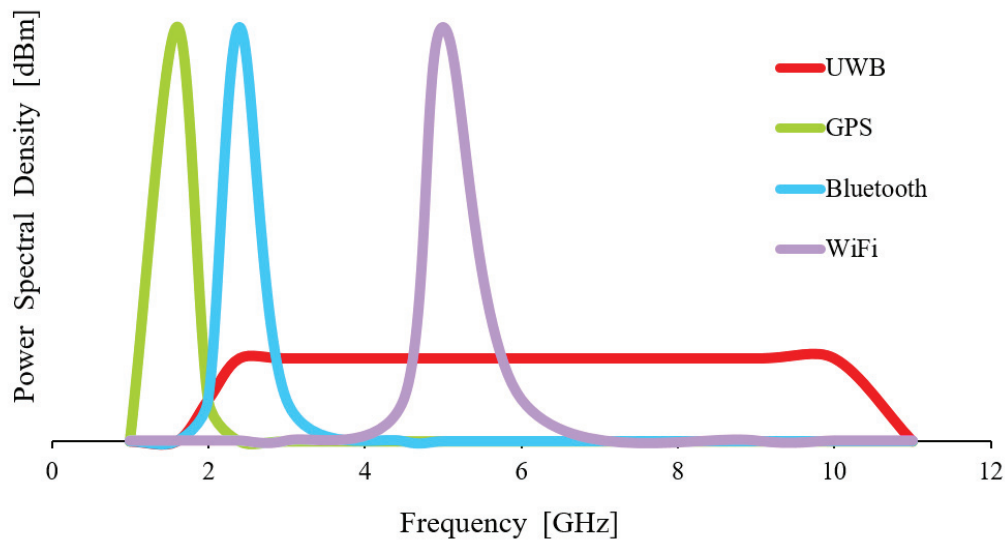


Figure 3.1: UWB frequency spectrum compared to other wireless technologies [14, pp. 403-404]: UWB sends a wide frequency band in one short pulse, but at such little power level, that it appears like noise next to other signals.

In two dimensional applications three reference nodes are necessary for determining the location through triangulation. While the receivers don't require clock synchronization, this method is highly sensitive to the characteristics of the channel. [13, p. 136], [16, p. 1]

- Another approach is the AoA technique, where only two devices provide enough location information. Here, special antenna arrays are necessary. [12, p. 90]

By measuring and comparing the arrival time of an incoming signal at different antenna elements, the angle of the source signal is estimated [15, p. 71]. The drawback of AoA are major errors in NLOS situations [12, p. 90].

- The ToA concepts measure the round-trip-time (RTT) of a signal sent by a mobile device to a stationary one – or the other way round – and back. At least three nodes must provide their RTT, which are translated into a distance and divided by two as the roundtrip includes both directions. Accurate clock synchronization between all devices required to meet high precision levels. [15, p. 71]

This technique allows to resolve different multipath signals and yields high accuracies by making use of the large bandwidth of the UWB signals [16, p. 1].

- In contrast, the TDoA method only requires clock synchronization between the base nodes, i.e. the communicating devices with known location. The mobile device measures the arrival time differences of at least three base stations thus determining its own location. [16, p. 2]

Instead of using only one of the above-mentioned signal parameters, hybrids of two, such as ToA/AoA, TDoA/AoA or ToA/RSSI, are used to obtain more accurate position information, depending on feasible processing durations, complexity constraints and the like [15, p. 71].

In general, navigation strategies are divided into either Anchor-based localization or self-localization, depending on whichever part runs the calculation. In an Anchor-based localization, the sensor on the object with unknown position only sends signals, that are received by devices with known position, named Anchors. Depending on the above explained signal parameters, two to three Anchors combine their measurements and calculate, e.g. by triangulation, the position of the object to be located. By contrast, self-localization allows the sensor on the mobile object to receive signals from the Anchors and to calculate its position itself. [15, p. 70]

Anchor-based localization is used e.g. in tracking industrial goods, while a car's navigation system is an example of a self-locating application.

3.2 Legal Regulation

Since more than 160 countries recognize the definitions of the organization IEEE, their regulations will be referenced in this thesis [17].

The IEEE 802.15.3a standard specifies short range high speed data communication from 100Mbit/s to 500Mbit/s, thus including UWB communication systems. In the meantime, the IEEE 802.15.4 standard specifies low-speed low-power signals used for accurate positioning. [18, p. 1162]

The second standard includes the definitions of the MAC layer and the physical layer of Wireless Personal Area Networks with low data rates, low power and wide ranges based on short impulses of UWB signals. The international frequency band specified in IEEE 802.15.4 can be used without license. [13, pp. 169-170]

In any case, the national radiation rules must be respected, which dictate an instantaneous large bandwidth emission. Sweeping narrowband signals one after the other over a large bandwidth is in general not allowed. For both licensed and unlicensed UWB systems, the allowed frequency bands are limited, as well as the allowed spectral power level density related to each band. However, the radiation rules depend on the specific applications and vary considerably around the world. [10, p. 370]

For indoor applications, the Federal Communications Commission (FCC) in the US allows UWB systems with a maximum emission level of -41,3 dBm/MHz and an instantaneous -10dB bandwidth above 500 MHz or 20% of a maximum 2.5GHz frequency [11, p. 2]. But in fact, the

regulations vary depending on the country or region and include different maximum emission levels, according to the different instantaneous bandwidths and the specific application [19, p. 11]. For most applications in the US, a spectral band from 3.1GHz to 10.6GHz is available for UWB systems, whereas in Europe those are limited for bands from 3.1GHz to 4.8GHz and 6.0GHz to 9.0GHz [19, pp. 13-16 23].

European LT1 systems are defined as unlicensed UWB systems operating between 6GHz and 9GHz and are intended for the general location tracking of people and objects. For LT2 systems, a frequency band from 3.1GHz to 4.8GHz is available designed for person and object tracking and industrial applications in well-defined buildings. LAES systems operate on the same frequency band but are defined as means for tracking fire fighters or other service staff during a mission. [19, pp. 23 25-26]

3.3 Features and Characteristics

UWB radio sensors are seen as a promising solution for wireless location and communication networks in the industrial and logistic sector, mainly due to their unique spectral exploitation of signal frequencies and their low signal power. Compared to other location and communication systems, UWB is a low-cost, energy-efficient and simple solution providing high ranging accuracies. [20, p. 51]

The technology enhances spectrum sharing with other licensed systems [12, p. 92]. Signal transfer in NLOS scenarios is possible, in contrast to optical ranging techniques [21, p. 179]. And depending on the specific application, the operating range can reach 10m to 50m [22, p. 9]. Even high data rates can be achieved. However, with the result of less ranging distance, as the signal power is limited [20, p. 52]. The ranging relevant properties will be discussed in detail:

3.3.1 Spectral Density

As the limited power is divided by a large spectrum of frequencies, the UWB technology is characterized by low spectral density values. Thus, UWB signals merely appear as noise for receivers that use other licensed communication or ranging systems and therefore cause no significant disturbance for those standards, even if used in the same area. [23, pp. 5-6]

Considering the fact that UWB has already found its way into smart phones which in most cases use Wireless Local Area Network (WLAN), Bluetooth or GPS simultaneously and that modern industry has no less constant need for various means of communication and sensing systems based on EMWs, the feature of seamless coexistence of UWB along with the other technologies

is required in many applications. Nevertheless, there are also complaints about the noise level that may disturb other narrowband technologies. But the general response to that opinion is, that the low power and therefore relatively low range of UWB signals, is if anything a very local issue. [23, pp. 5-6]

3.3.2 Data Rate

The IEEE 802.15.4a standard for low-rate Personal Area Network (PAN) defines data-rates for UWB usage of 50Mbit/s to 100Mbit/s [24, p. 5], while the IEEE 802.15.3a standard for high-rate PAN defines data-rates for UWB applications up to 500Mbit/s [18, p. 1162]. In comparison, Bluetooth reaches 1Mbit/s, while Wi-Fi, the commercial name for the IEEE 802.11 standard, reaches 54Mbit/s [24, p. 5].

3.3.3 Accuracy

In general, the achievable accuracy depends on the frequency response and pulse width of a signal. In the case of UWB, the response frequency ranges from 10MHz to 40MHz, while one pulse width can even go below one nanosecond resulting in a theoretical accuracy of centimeter level. [18, p. 1163]

In the work of Giretti et al., the accuracy of an impulse signal is calculated as the square root of the distance variance. Its lowest reachable value can be reduced by increasing two factors: the signal-to-noise-ratio (SNR) and the spectral bandwidth. As UWB uses only low spectral densities, it is not the SNR, but the extremely large bandwidth which is accountable for its accurate performance, in comparison to alternative ranging systems. [12, pp. 92-93]

3.3.4 NLOS Behavior

At this point, a closer look into LOS and NLOS behavior will be taken. The most accurate results are achieved in case of LOS and no multipath propagation, as the only arriving signal travels directly from transmitter to receiver without detours. In scenarios of LOS and multipath propagation, the short UWB impulse below a nanosecond is beneficial as it allows the resolution of individual multipath components. Because of the short pulse duration, a UWB system can - with the according firmware - isolate the first arriving signal from the later arriving reflections. Therefore, in multipath scenarios, UWB performs better than other technologies. [11, p. 2]

In the event of NLOS, but multipath propagation, the UWB signal reaches the receiver with a certain delay due to the longer path than direct LOS. However, the UWB communication doesn't

collapse and depending on the obstacle between the sensors, an approximation can be obtained. [3, p. 45]

In comparison, optical sensors would get no signal at all.

If there is NLOS and there are no reflecting surfaces next to the direct path, an electromagnetic signal can be detected on the other side, if the wavelength is similar to or greater than the obstacle's dimension in the direction of propagation. This phenomenon of "bending" around the object is called diffraction. [25]

An object can also be penetrated by an EMW by being magnetically permeable to it. The permeability depends on the material of the obstacle and on the signal's wavelength. Common glass e.g. is permeable to radio waves and most components of light but not to its ultraviolet component. However, that characteristic is relevant for sonar applications. [21, pp. 213-214 235]

3.4 Recent Developments

The pioneering contributions to the development of impulsive UWB technology date back to the early 1960s when the sampling oscilloscope was introduced. In the 1970s, Harmuth papers and books made the basics for UWB transmitters and receivers public, while Ross and Robbins defined UWB signals for different applications, which are communication, radar and sensing. The first ground penetrating radar was invented in the US military from the 1960s to the 1990s. Recognized in 1998 by the Federal Communication Commissions (FCC), the UWB technology was initially regulated. [13, pp. 6 11]

The implicational range has grown significantly in the past decades. Some recent achievements and developments are mentioned below to justify the motivation of using UWB technology in the presented positioning system. In the previous chapter, the preconditions and tasks of the that system are given. Those conditions in mind, the following recent developments w.r.t. the ranging accuracy, the detection of NLOS situations and the handling of multiple users within the same channel encourage the use of UWB.

3.4.1 Channel Sharing

Because of channel specific ranging behavior, all UWB sensors, that are part of the presented positioning system, communicate on the same channel. The Impulse Radio UWB (IR-UWB), which is used in this work, has the benefit of short pulse duration (below 1ns) and high pulse repetition periods, which in contrast to narrowband technologies allows multiple access by using

time hopping codes. Still, impulse interference cause communication problems. In 2013 Perez Guirao proposed impulsive interference management that is independent of the physical layer and modulation scheme and enables concurrent transmission in full power. Each signal source can adapt the pulse rate independently in order to reduce the impact of pulse collision at nearby receivers. [20, pp. 51-52]

Furthermore, the frequency exploitation has been improved in terms of data rates and energy efficiency. Until then, Orthogonal Frequency-Division Multiplexing (OFDM) based overlay systems depended on a primary user to systematically allocate unused frequencies to secondary users. In the work of Moorfeld et al., a Multiband Impulse Radio (MIR) transmitter using multiple bands and a receiver, which detects only the energy of the signals, are presented. [26, p. 45]

Only in the last few years, higher data rates and ranges have been achieved by enhancing multiple input - multiple output (MIMO) performance which mitigates multipath fading and co-channel interference. MIMO is achieved by configuring radiators orthogonally to suppress mutual coupling, by polarizing the two antenna elements differently by applying special isolation geometries or placing the elements perpendicularly. A compact portable MIMO antenna was designed only recently and covers the whole allowed bandwidth from 3.1 – 10.6GHz, which is higher than most of the former accomplishments. The radiation pattern of the antenna is quasi omni-directional and a T-shaped slot less than -15dB separates the elements to reduce coupling. [27, p. 224]

El-Hadidi et al. presented a multiple input – multiple output (MIMO) antenna synthetization that facilitates simultaneous communication between multiple transmitters and receivers without interference [28, p. 150].

3.4.2 NLOS Performance

Jimenez and Seco have defined NLOS as an open research topic and the biggest challenge for accurate positioning [11, p. 1]. Also, in this work the NLOS issues pose a major challenge, as the connecting conveyor belts obscure the LOS in some constellations. Nevertheless, important improvements concerning NLOS ranging were accomplished.

As multipath propagation caused insufficient precision, e.g. in closed environments, Kolakowski and Djaja-Josko introduced in 2016 data fusion techniques, i.e. data aggregation from different devices. In this case, the mobile sensors obtained their relative position by ranging to the stationary ones and additionally by exchanging information among themselves. Cooperative positioning using TDoA and Two-way-ranging (TWR), apart from communicating with the stations alone yield higher accuracy. [29, pp. 1 4]

Referring to the above-mentioned issues, some of the reviewed publications focus on the detection of a NLOS case. In the past, NLOS identifications were either derived from non-parametric methods or mean-excess delay calculations. In recent years, however, a classification via machine learning, namely the Least-Squares Support Vector Machine (SVM) was introduced. If the SVM operates with quadratic and polynomial kernels and the training set size is increased, LOS and NLOS scenarios are close to 100% separable. [30, p. 1]

3.4.3 Clock Synchronizations

Others devoted their work to clock synchronization schemes with higher long-term accuracy than the originally used simplified linear clock models. Clock synchronization has great influence in the ranging accuracy in ToA and TDoA calculations. The chosen sensor type in the presented positioning system operates on the ToA principle as well. By introducing a quadratic clock model, Xie et al. improved long-term precision, while maintaining the short-term precision. [31, p. 3894]

3.5 UWB Ranging Applications

Because of the above attributes, the technology plays an important role in concepts for indoor navigation and surveillance [21, p. 179], as well as the tracking of personnel, goods, tools or machinery [20, p. 51].

Dynamic position tracking was primarily based on GPS, RFID and optical systems. However, GPS is only usable outdoors, RFID signals provide low accuracy not under 1 meter and optical sensors need LOS. Other systems like Wi-Fi and Bluetooth have an extremely high-power consumption compared to UWB. [32, p. 1]

UWB technology is therefore a promising alternative within complicated and multipath rich environments, e.g. in industrial and logistics fields or indoor navigation. Even when operating without LOS, the ability to resolve multipath components allows the system to track personnel and objects, where many other technologies fail. [21, p. 179]

Giretti et al. proposed a management service for highly automated construction facilities which permits to dynamically track the position of workers and materials. There, the mobile low-power UWB device stuck to the goods can be scanned either by personnel or automatically by barriers on the site. [12, p. 109]

Another application example is the iPhone11, which includes a UWB sensor for indoor navigation and the detection of surrounding objects [33, p. 43].

3.6 Critical Assessment

3.6.1 Chances

Today, the scientific interest in UWB is positively booming. The car industry, which finds itself at the beginning of the era of autonomous driving, is already studying different methods for the employment of UWB: Ponte Müller compares already existing concepts for collision avoidance systems in cars based on UWB [1]. The achievements discussed include decimeter-level ranging precision up to 300m distance between two cars by TWR at 6.35GHz [34, pp. 11-12], while others used 2.4GHz and 5.9GHz radios for TDoA measurements with up to 90m LOS errors below 0.7m depending on the vehicles' velocity [35, pp. 1-7]. There are many more approaches including round trip delay (RTD), AoA and ToA methods, yielding accuracies within decimeter-level at LOS and multipath-rich environments. According to Kristensen et al., in the future, there might be self-driving cars in underground parking lots or autonomous robots in a warehouse applying UWB systems [30, p. 1]. As mentioned before, the NLOS issue is a very topical one and if mastered, could open many more doors in the field of UWB navigation. In the past year, Jimenez created NLOS metrics by modelling LOS and NLOS conditions with the goal of achieving NLOS mitigation techniques, which could lead to significant improvements in the future [11, p. 1].

But not only is UWB suitable for location tasks, but also for medical applications or even food quality control, as the low signal power strength of UWB is suited for harmless tissue examination. EMWs interact with substances other than air depending on the polarity of their molecules or ions. [36, p. 257], [37, p. 323]

Signals of different wavelengths react differently when hitting a certain material, i.e. the wavelength determines the distribution between reflected, refracted and attenuated components of that signal. It is therefore common practice to extract information of an unknown item or texture by pointing an electromagnetic beam with known frequency at the specimen. By measuring and analyzing the density of reflected, refracted or attenuated signal components, characteristics of the examined object can be derived. As the distribution of the three components correlate with the specimen's texture and the wavelength of the incoming signal, even more information can be extracted when studying the reaction of more than one frequency of EMWs. Hence, the advantage of UWB signals lies in the simulations testing of a probe by EMWs of various wavelengths. [10, pp. 369-370]

UWB radio waves of different frequencies can travel through different materials depending,

among other things, on the material's water content. It is possible to extract information on biological objects in a non-destructive, fast and continuous way, which is beneficial in medical applications, but also in the food industry for carrying out on-line quality control and process monitoring. [36, p. 257], [37, p. 323]

Hilger et al. introduced a powerful method to extract information about organic tissue, which exploits the electromagnetic interaction with matter. Atomic and molecular phenomena, like permittivity, permeability and conductivity allow remote or contact based microwave imaging of inner organs or remote heart monitoring. With such systems diseases, like breast cancer, can be detected. [36, p. 314]

On the other hand, Mextorf et al. proposed a food surveillance system, which makes use of the different frequency spectra absorbed by the various organic structures [37, p. 339].

Considering the demands on the presented location system and its boundary conditions, UWB radio outperforms concurrent technologies in terms of NLOS performance and multipath propagation, as well as spectrum sharing when operating parallel with other wireless communication systems.

3.6.2 Limits

Some physical constraints appear due to reflections, impermeabilities, absorptions and fading phenomena. E.g. radiofrequency interference and multipath effects cause sampling frequency offset, phase error and carrier frequency offset, which vary enormously depending on the operational area. [10, pp. 373-374]

NLOS effects can be compensated to a certain extent, but if the object between communicating devices is not permeable to UWB radiation and shadows too much of the space in between, the first arriving signal will show a significant temporal offset leading to poor ranging accuracy. Of course, there are research projects focusing on that problem, e.g. by trying to approximate the degree of deflection by comparing the arriving signal power with the power it usually has at a certain distance and in a certain area. There, however, lies the difficulty, as the free space model of the received power by Friis performs poorly in complex environments evoking multipath fading, reflections and absorption. [38, p. 149]

Due to the power restrictions of UWB, the maximum distance between communicating devices is limited by 50m. But such high distances are only feasible at the expense of the data rate in order to stay within the regulation boundaries of power emission.

Chapter 4

Choice of Bus Protocol

The implementation depends on four or eight UWB modules, that must communicate with the IPC in order to exchange settings and measurement data. Consequently, a bidirectional connection between the IPC and multiple sensors at once is required, which is robust, easy to install and capable of real-time data rates to keep the location system responsible. Considering the hazardous environment, the chosen standard has to withstand very large electromagnetic disturbances as well as movement and impact. Hence, the chosen physical line needs to tolerate noise and allow simple wiring while the protocol running on it must be able to accomplish the demanded data rate and the given number of participants.

As the physical layer of the OSI model, a common and reliable RS485 interface has been chosen. While the IPC is by default equipped with an RS485 serial port, there are standard UWB modules available for communication over that line that are well known and tested in the environment. Due to the industrial environment, an automation protocol was selected which specifies the data link and application layer. For the above mentioned demands on the robustness, data rate and simplicity, the choice fell on the Modbus RTU protocol which is a wide-spread industrial fieldbus protocol operating on the RS485 physical line and enhancing a daisy chain topology. [39]

4.1 RS485 - Background

Originally, the RS485 interface was developed for high speed data transmission over large distances for industrial use [40, p. 97].

In case of a two-wire connection, the standard transmits data in a half-duplex manner over a twisted pair. On the other hand, a four-wire connection enables a full duplex communication,

where the Master transmits signals over one of the twisted pairs and the Slaves over the other. Either way, each node has a unique address, which allows independent communication, but only one device is able to use the line at a time, which is why the other nodes must access a high-impedance mode. While the two-wire connection needs less cables and allows the Slaves to communicate among themselves, the four-wire connection achieves higher data rates by allowing multiple simultaneous messages and avoiding turn-around delays. [40, pp. 143-145]

RS485 is used in serial communication systems and needs an additional ground line. As the data is transmitted differentially over one or two twisted pairs, the ground doesn't serve any communication purposes but is needed to tie the signals to one common ground in case of high voltage difference between the nodes, which can occur at large distances. The differential data transmission over a twisted pair offers high noise resistance, thus managing data transmissions without loss over distances greater than one kilometer. [40, pp. 97-98]

4.2 Physical Connection

As mentioned above, the physical layer of the RS485 interface is composed of one twisted pair for data transfer and one ground line. While the RS485 port of the Modbus RTU Master, which in our case is the IPC, defines one end of the physical line, the last Slave defines the other end. In between, the other Slaves are connected to the wires in daisy chain mode, as represented in figure 4.1. The difference between the two figures are the individual power supplies for each Slave versus a central power supply. While the first option saves wiring, the second reduces the amount of necessary power supplies. Whichever is preferable, depends on the additional equipment of a plant unit, i.e. whether various locations for power supplies are provided anyway.

The connection of one Slave to the main line can be achieved e.g. by using terminal blocks. For the twisted pair of data lines, a cable diameter of 0,25mm or more is required. Per specification, the power supplying wires should have a diameter of at least 0.75mm. Both guidelines are specifically true for Modbus RTU over RS485 physical line and therefore applicable in the herein developed positioning system. [41]

The serial RS485 physical line allows 32 bus participants and is not limited to a single topology. However, daisy chain is the most common one and in this particular application advantageous, because of its minimal wiring effort [42, p. 22].

Only the electrical specifications of differential receivers and transmitters in digital bus systems are defined by the RS485 standard. Additionally, the ISO standard 8482 standardizes the physical topology with a maximum length of 500m. [43, pp. 11 64]

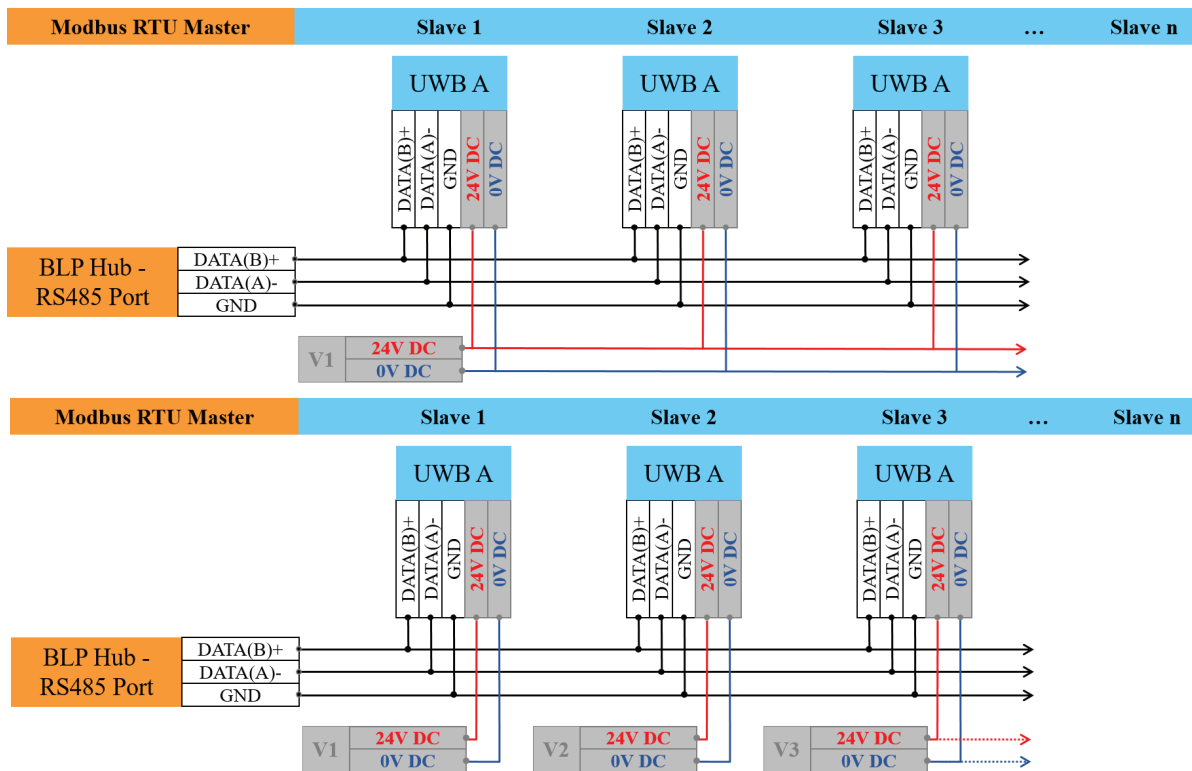


Figure 4.1: Physical connection from IPC as Modbus RTU Master and UWB modules as Modbus RTU Slaves. The power supply can be installed as central supply (top) or separately (bottom).

4.3 Bus Comparison

In order to save wiring, a bus system was considered a suitable option as it allows multiple network members to communicate over a common physical transmission line [44]. Concerning the communication protocol, two different candidates have been considered in this work: Profibus and Modbus RTU, which represent the most popular fieldbuses in industrial automation, that operate on the serial RS485 line [45]. Both interfaces are prepared on the IPC, that executes the developed service [46, p. 5]. However, the implementation of a Profibus Master requires additional hardware. Eleven aspects have been compared, rated and their importance weighed, in order to decide upon the most fitting bus protocol for the herein documented application.

In the table 4.1, the two protocols were compared and rated according to the dominance of the characteristics. The importance was weighed by the “Weight” factor from 0 to 5. For each of the two candidates, the gratification of those characteristics were graded from 0 to 5. Subsequently, the weights of all parameters were multiplied by the grade value of each candidate and the products added up separately. Those results represent the suitability of each candidate in terms

of the listed characteristics. As the two protocols were already picked as best two candidates, due to their suitability for industrial environments [45]. The protocol with the most points represents the better choice [47, pp. 7-8]. In this case, the Modbus RTU protocol outperforms the Profibus protocol foremost due to its basic architecture, which enhances easy implementation and maintenance.

4.4 Modbus RTU

Modbus RTU is an industrial fieldbus, that was founded in 1979 by Modicon (now Schneider Electric), to be used with their PLC systems [56, p. 311]. Nowadays, it is still a commonly used communication standard for interconnecting metering devices at the field level [57, p. 598]. Worldwide, control systems, such as PLCs, are communicating over the fieldbus, which can operate on various physical lines [58, p. 122]. Networks operating on Modbus are typically used for data transmission from control instrumentation to logic controllers or data archives [56, p. 311]. In many cases, those networks include a gateway, that communicates with a supervisory network, e.g. over Ethernet [59, p. 1].

The specifications for Modbus RTU are publicly available in two documents: the MODBUS Application Protocol and the MODBUS Serial Line Protocol. The first one specifies the message structure of the application layer in a Client/Server mode, while the second one provides the Master/Slave structure for the data link layer - both for the ASCII protocol and RTU – and acts as a guide for RS485 and RS232 implementations. [49, pp. 4-5]

Due to its early invention, the fieldbus protocol is short in resolving corrupted messages and has limitations to the transmission speed and the number of Slaves, as well a restriction to single Master mode [59, p. 1].

However, for its reliability as well as practical and economic advantages, the protocol is still widely used in industrial automation. The RTU variant of the standard enhances a higher transmission rate than the ASCII protocol operating under the same baud rate. Additionally, its implementation is convenient and simple in many programming languages. [57, p. 598]

4.4.1 Protocol Principal

Modbus RTU adopts a Master/ Slave communication – allowing a maximum of 247 Slaves in one network [59, p. 2]. It is positioned at OSI layer 7 and operates on serial buses or networks, e.g. on the serial interfaces RS232 or RS485 [60, p. 2].

UWB Middleware vs. Laser Reference- Constellation no. 1 to 10

Feature	Weight	Profibus over RS485	Value	WxV	Modbus RTU over RS485	Value	WxV
Market share among industrial fieldbuses	2	26% [45]	5	10	17% [45]	3	6
License free	5	Royalty for every node [48]	0	0	None, even for industrial use [49, p. 4]	5	25
Speed	5	1,2 kbps - 12 Mbps, depending on distance [50]	4	20	1,2 kbps - 12 Mbps, usually 9,6 - 19,2 kbps [49, p. 34]	3	15
Feasible Length	3	1200m [51]	5	15	1000m [49, p. 27]	5	15
Maximum Node Number	3	127 [52]	4	12	247 [49, p. 7]	5	15
Physical Layer	5	2 wires, twisted pair, connectors specified [53]	5	25	2 or 4 wires, twisted pair, connectors not specified [49, p. 5]	5	25
Noise Immunity	4	Very good [53]	5	20	Good, if proper cable determination [54]	4	16
Telegram Construction	3	Complex message layout [55]	2	6	Simple message layout [49, p. 8]	3	9
Error Report	1	Diagnostics available [53]	5	5	Exception Response by CRC [53]	3	3
Typical Applications	4	Factory and process automation [53]	5	20	Control, monitoring, smart devices [53]	5	20
Installation Convenience	5	Additional Hardware required [6]	2	10	No additional Hardware required [6]	5	25
Sum				143			174

Table 4.1: Systematic evaluation of best choice for fieldbus protocol.

The latter offers better noise immunity, as the RS485 physical data line is composed of a twisted pair spanning a voltage potential. Consequently, interfering signals cause voltage offsets of the same value, thus not affecting the potential in between. [61, p. 214]

Modbus encodes the data in the big endian binary format, where the most significant byte is sent and received first [60, p. 2].

Unlike the ASCII mode, the RTU sends and receives the least significant bit first in an 8-bit binary coding system represented in hexadecimal numbers. Next to other industrial protocols, the data presentation of Modbus RTU is simple. All participants must use the same encoding method and the same baud rate (typically 9600 or 19200), as there is no baud rate recognition. [49, p. 12]

Modbus supports only two data types: 1-bit-coils and registers of two bytes [60, p. 6]. Hence, only positive Integers from 0 to 65535 are transferrable and only data without additional parameters.

4.4.2 Memory Map

The Modbus memory map (MMM) is a subset of the device's application memory. The manufacturer or developer has to assign a specific section of memory arrays to Modbus usage. It is important to note that the addresses used by Modbus are not consistent with the hardcoded RAM or Flash Memory addresses. There is an offset between the two, depending on the hardware address, that is defined as start address for the MMM. [60, pp. 6-7]

That memory area only is then accessible by the fieldbus, which will produce an error message, in case a Modbus participant tries to access a memory address out of the dedicated area.

While the protocol allows a maximum of 65535 addresses [60, p. 6], the convention suggests to use effective addresses from 00001 to 49999. The reason behind it is that usually the first digit of the address defines the data type (coil or register) and the action (read or write) performed on a variable. Therefore, the MMM is divided into four subsets: the addresses starting with 0 (00001 to 09999) refer to coils with read/write access rights. So does the 1-prefix refer to coils with read access right only, the 3-prefix to registers with read access right only and the 4-prefix to registers with read/write access right. [62, p. 330]

There are two ways of storing the data: either in overlapping (9999 occupied registers in total) or separate blocks (19998 occupied registers and 19998 occupied coils in total), depending on the configuration of manufacturer or developer [60, pp. 6-7]. In the overlapping version, addresses starting with 3 or 4 while showing the same last four digits would point to the same memory array, but provide different access rights as described previously. If coil addresses are registered as prefix 0 or 1 and the last four digits being 0001 to 0016, they would occupy the first of the

9999 two-byte space in the map and it would not be possible to assign a register to 30001 or 40001. On the other hand, a separate block configuration stores read/write variables separately as well as coils and registers; therefore, no address blockage would occur.

Either way, neither the full range of allowed addresses nor all four data areas are usually occupied by an application, in which case the MMM can contain less elements. By way of example, only addresses from 40023 to 40122 could be assigned to Modbus usage, leading to an exception response by the Slave, if the Master tries to access an unspecified address. That way, less memory is dedicated to Modbus, leaving more resources, e.g. for other applications. Manufacturers might predefine a fixed map or allow customers to configure a map themselves, depending on the hardware's purpose. In the latter case, the extended registers, occupying addresses from 40001 to 105536, could be equally used if configured in all communicating devices, which requires the use of a 6-digit address. [60, pp. 6-7], [62, p. 330]

4.4.3 Messaging

On adopting a Master/Slave communication, the controllers of the network nodes exchange information according to a transmitted function code (FC). Initially, the Master sends a FC together with the starting memory address and number of the registers or coils to be read or written. The Slave responds with the repeated FC and the requested or changed data. [60, p. 4] According to the application protocol, each message contains four fields: Modbus Slave address (1 byte), FC (1 byte), data (1 - 252 bytes) and CRC (2 bytes) [60, p. 3]. While the address (not to confuse with the memory addresses of the memory map) contains the Modbus Slave ID reaching from 1 to 247 (while 0 is the broadcast address) [49, p. 7], the FC specifies the action (read or write registers or coils) the Slave must perform when receiving the message [60, p. 4].

In response to the Master, the Slave uses the FC field as indicator for exception (0 – 127) or flawless transaction (128 - 255). Only to the broadcast message does the Slave not reply, which is why broadcasting is hardly used. Slaves do not initiate communication, only the Master, which usually is a PLC or other supervisory computer. The data field is composed of 0-252 bytes, that may contain floating point values, tables, ASCII text, queues, and other kinds of data. Together with the FC, it forms the request-reply Protocol Data Unit (PDU), while the Application Data Unit additionally includes the preceding Slave ID and the succeeding Cyclic-Redundant Checksum (CRC). [60, pp. 3-4]

First, the CRC value is calculated at the transmission out of the address, FC and data fields. After reception the value is calculated again, which is then compared to the initial CRC value. Without matching values, the message will be disposed of without any notice. While in unicast mode, the Master can detect the loss of information after a standby time and subsequently repeat the requirement. [49, pp. 14-15]

There is no response expected in broadcast messages [49, p. 7].

4.5 Interface Preparation

4.5.1 Register Table

In order to allow data transfer over Modbus, some conditions must be configured on the middleware's side, that acts as Modbus Master. First of all, the memory space of the RAM must be reserved according to the previous section. As a documentation for the middleware's design serves the register table, in which valid register addresses are assigned to the shared variables in logical order. Other information that vary for each register or coil are listed in the register table as well, e.g. the unit or well considered default values of a variable. A boolean value defines for each variable, whether that variable is stored to the EEPROM, i.e. not deleted when a new firmware version is uploaded. Finally, there are additional access limitations, in case of a corrupted Modbus message trying to write on a read only register.

Both, Slave and Master communicate through their MMM in accordance to the documented register table. The Master only sends commands of reading or writing the content of a Slave's register or block of registers, as described in the previous chapter. If n Slaves are attached to the RS485 physical line and all of them have the same MMM of data to share with the Master, the Master's MMM must occupy n -times the memory space of a Slave's. The below register table, see figure 4.2, which is implemented on each Slave, was developed in this work.

In the two very left columns are the category and block names of the register sections, where Anchor refers to the UWB sensors initiating a measurement and Tag to the responding UWB sensor [9]. Apart from the responsibility to start a measurement, the Anchor also calculates the distance between itself and the addressed Tags. Only the Anchors need to be integrated through Modbus, to communicate the distance values to the IPC, while the Tags are connected to a voltage supply only.

The firmware version, consisting of minor, major and patch number, indicates differences in the

sensor's functionalities and is computed automatically by the firmware's version control system. As an identifier for UWB addressing serves the 8-byte long unique ID, while the anchor's Slave ID refers to the Modbus address. If an Anchor should become a Tag or the other way round, the Tag mode is changed. For surveillance purposes, the Anchor's temperature is monitored and expressed in percentage of Kelvin due to the values restriction from 0 to 65535, which corresponds to -273°C to 382°C with a resolution of two decimal values.

The following registers contain, on the one hand, important information for the Anchor to initiate the measurements, i.e. the Tags unique IDs, and on the other hand, values that are obtained through UWB communication, i.e. the ranges in mm to each Tag and the temperatures of each Tag. All Tags, needed for the position evaluation, are addressed periodically through these IDs, and their distances calculated and updated every one to two seconds.

In the last block – the debug registers – are a set of variables that can be accessed through Modbus by first entering a key, that only the maintainer has access to, as those settings define the UWB communication itself, like measurement compensation values, ranging frequency and signal power, as well as the confirmation, whether the specific sensor was calibrated. The access rights and storage to EEPROM parameter determine the possibility of manipulation through Modbus and potential recovery of settings, like the Modbus Slave ID or Tag mode, in case of power loss or firmware change.

4.5.2 Interface Configuration at the Middleware

The assignment of the names to the MMM address is done separately in the Interface Configuration File (ICF). There, the MMM is linked to the process image of the IPC, by defining a constant address offset between MMM and the RAM of the IPC. Within the ICF, also the frequency is specified, with which this MMM will be exchanged between Master and Slaves. Furthermore, the path of the used physical interface of the IPC and the communication baud rate are defined in the ICF, as well as the parity bit, the number of data bits and stop bits. Baud rate and message format are in accordance with the Slaves' configuration.

Modbus Master configuration in ICF:

- Slave ID = 0
- Baud rate = equal to Slaves'
- Stop bit = equal to Slaves'
- Implementation of Modbus Function Codes

- Memory address assigned to communicated registers
- Function Code assigned to communicated registers
- Path to RS485 interface within IPC

As argued in the previous chapter, the manufacturers of the IPC prepared the use of Modbus by defining the layout of the ICF. In an online tool from the manufacturers, the user can choose the data type, name, order and update rate of the Modbus variables. By command, those settings are automatically transferred to the ICF of the IPC, if the IPC is connected to the internet. As the online tool only allows valid settings, that are executable by the IPC, it facilitates save Modbus implementation without the need of overriding system protecting access rights. The manufacturers also provided a programming library, that gives access to the ICF and uses its information to establish Modbus communication. When transferring the user's settings from the online tool to the ICF, a specific absolute path is used for storing the ICF on the IPC. Same path is included in the library, so that all information in the ICF can be accessed when establishing the connection. For other hardware, where such convenience is not offered by the manufacturer, the ICF would need extra generation.

Block Name		REGISTERS TABLE - UWB-Sensors						
Register-Address	Description	Unit	Default Value	Access	EEPROM			
Anchor Parameters	Firmware Version	40000	Version Major	/	65535	R		
		40001	Version Minor	/	65535	R		
		40002	Version Patch	/	65535	R		
	Anchor Unique ID	40010	Device ID Byte 0 - 1	/	0	R		
		40011	Device ID Byte 2 - 3	/	0	R		
		40012	Device ID Byte 4 - 5	/	0	R		
		40013	Device ID Byte 6 - 7	/	0	R		
	Temperature	40050	Anchor Temperature	K/100	0	R		
	Settings		40100	Input Tag Mode	/	65535	R/W	*
			40101	Input Slave ID	/	65535	R/W	*
			40110	Output Tag Mode	/	65535	R	
		40111	Output Slave ID	/	65535	R		
Tag Parameters	Tag IDs	40200	Tag ID #0 Byte 0 - 1	/	0	R/W		
		40201	Tag ID #0 Byte 2 - 3	/	0	R/W		
		40202	Tag ID #0 Byte 4 - 5	/	0	R/W		
		40203	Tag ID #0 Byte 6 - 7	/	0	R/W		
		40204	Tag ID #1 Byte 0 - 1	/	0	R/W		
		40205	Tag ID #1 Byte 2 - 3	/	0	R/W		
		40206	Tag ID #1 Byte 4 - 5	/	0	R/W		
	40207	Tag ID #1 Byte 6 - 7	/	0	R/W			
	⋮	⋮	⋮	⋮	⋮	⋮		
		40292	Tag ID #23 Byte 0 - 1	/	0	R/W		
		40293	Tag ID #23 Byte 2 - 3	/	0	R/W		
		40294	Tag ID #23 Byte 4 - 5	/	0	R/W		
		40295	Tag ID #23 Byte 6 - 7	/	0	R/W		
	Tag Ranges		40300	Range #0	mm	65535	R	
			40301	Range #1	mm	65535	R	
			⋮	⋮	⋮	⋮	⋮	
			40323	Range #23	mm	65535	R	
			40350	RangeIndex #0	/	0	R	
			⋮	⋮	⋮	⋮	⋮	
			40373	RangeIndex #23	/	0	R	
	Tag Temperatures	Tag	40400	Tag Temp. #0	K/100	0	R	
			40401	Tag Temp. #1	K/100	0	R	
			⋮	⋮	⋮	⋮	⋮	
		40423	Tag Temp. #23	K/100	0	R		
Debug Registers	Debug	40900	Key	/	0	R/W		
		40901	Load from EEPROM	/	0	R*		
		40902	Save to EEPROM	/	0	R*		
		40910	Power	dB/2	48	R*		
		40911	Ranging Frequency	Hz	2	R*		
		40920	Antenna Delay RX	15.7e-12s	16439	R*		
		40921	Antenna Delay TX	15.7e-12s	16439	R*		
		40922	Bias Comp.	-dB/100	2100	R*		
		40923	Temperature Comp.	/	1	R*		
		40930	Device Calibrated	/	0	R*		

Figure 4.2: The register table is a documentation of all the registers transmitted between IPC and every single Slave. Next to the address of each registers in the MMM, the table contains the register's name and category, a short description, its unit, default value and access right as well as a boolean value indicating EEPROM storage. The Modbus Master, i.e. the IPC, stores the quantity of registers listed in the table times the number of Modbus Slaves.

Chapter 5

Application Development

In this chapter, the application of the middleware itself will be addressed. On the one hand, it controls the data flow all the way from the distance meters to the OPC-UA network of the plant. On the other hand, the program executes the most important data processing step for a location system, namely the positioning algorithm, which continuously translates the four raw distance values into the offset and orientation between two connected units.

5.1 Hardware Setup

The application in the developed middleware runs on the IPC, that will later be mounted on each unit. During the software development, the IPC is connected to the programmer's computer through Ethernet or Wi-Fi, while the four UWB sensors are mounted on the outer points of two tripods, see figure 5.1. Each tripod represented the container front of a plant unit and the sensors that are mounted on them operate as Anchor pair "A-B" or as Tag pair "a-b" during a test runs. Between two paired devices is a distance of 1m, while the position of the tripods relative to each other is variable. Before connecting all four devices to the power supply, each device was commissioned either as Anchor or as Tag. According to the final application, the UWB devices acting as Anchors have a Slave ID assigned to them and are connected through Modbus to the RS485 port of the IPC, whereby the Modbus Slave functionality is already implemented in the firmware of the sensors.

After explaining the basic elements of Object Oriented Programming (OOP), the design of the application will be looked into. First, the data flow management is realized, followed by the establishment of the Modbus communication to the UWB sensors on the middleware's end. Then, the computational part is implemented and finally, the OPC-UA interface for network

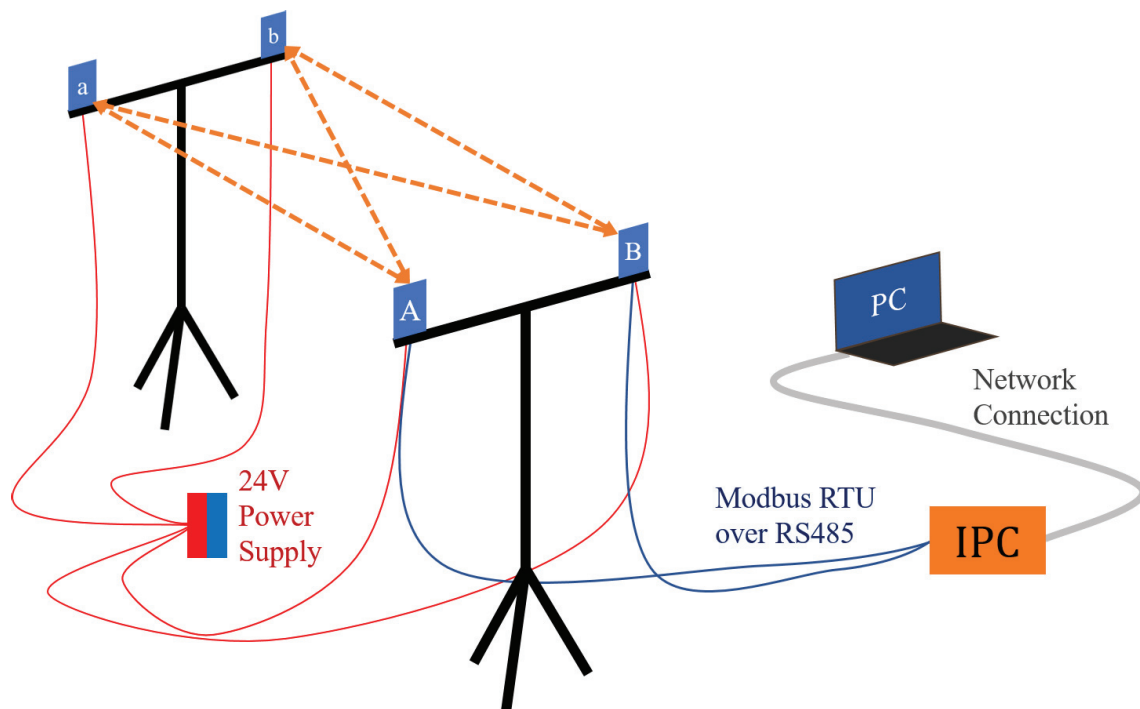


Figure 5.1: Development test setup: During the software development, two tripods imitated the fronts of two connected plant units. An Anchor pair "A-B" and a Tag pair "a-b" respectively are mounted on the two tripods to test the application. Through a Modbus RTU connection the measured data was ingested into the Middleware and processed in the application.

integration is set up.

5.2 Object Oriented Programming

Even though Python is used as programming language in this implementation, the following descriptions are kept general and include synonyms of other languages in brackets. Compared to Structured Programming (SP) where the data is processed in a series of instructions that can be combined into functions when the program grows larger, OOP combines the data, as well as the functions applied on that data within an object. The data (attribute) of an object can only be accessed, through calling a function (method) of the same object. [63, pp. 16-17]

All structures, that play an important role within the developed middleware are specified in this chapter, such as Classes, Objects, Inheritance, Queues, Dictionaries and Threads.

5.2.1 Objects and Classes

The motivation behind OOP is the representation of objects in the real world, e.g. electrical components, machines or employees, as objects in the programming sense.

Before there is an object, there must be a class, which is in fact a data type and serves as model for the instantiated object. A class contains the type of attributes and therefore determines what kind of data belongs to an object of that class. In the case of a class “Employee”, that could be “Holidays Left” or “Overtime Hours”. A class also defines the methods, i.e. functions processing those attributes, that in the above example could be “If Overtime Hours reaches 8, then increase Holidays Left by 1 and set Overtime Hours to 0”. [63, p. 18]

An instance, i.e. a derived object, of the Class “Employee” could be “John Doe” with the current state of having 17 “Holidays Left” and 3 “Overtime Hours”. During instantiation, an object may need input parameters, which in the above example could be the name and date of employment for a new “Employee” instance. Those input parameters are provided from outside only once, when the object is created. [63, p. 19]

While a class only defines an object type and its attributes without specific values or only start values (“Holidays Left” = 25 and “Overtime Hours” = 0 at the beginning of employment), a specific object has a specific state, like shown before at the instance “John Doe” of Class “Employee”. Hence, the instance of a class adds values to the attributes and enables the execution of methods. [63, p. 18]

Usually, a class is schematically illustrated by a rectangle with the class name at the very top and below a list of the attributes and the methods of the class, see figure 5.2 [64, p. 12].

On executing the initializing method (usually “init()” or similar), all attributes belonging to

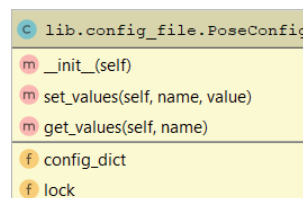


Figure 5.2: Class of the configuration object containing all attributes and methods in the below sections of the box.

the object are defined and values allocated to them. The initialization is executed without being called once the object is instantiated. Afterwards, the main part of an object is executed, which contains methods, that can be called from outside by referring to the object. [63, p. 19]

5.2.2 Inheritance

If there are classes that share most of the attributes and methods with other classes, but also have their own, one could create a class, that inherits all attributes and methods from an already existing one, and only add the rest. To stay at our example there could be a class “Weekend Worker” inheriting from “Employee” with an additional attribute “Overtime Hours on Weekends” and method “If Overtime Hours on Weekends reaches 4, then increase Holidays Left by 1 and set Overtime Hours on Weekends to 0”, in the case of double salary at weekends. Depending on the programming language, the child class contains a reference to the parent class. [63, p. 23] Within a class diagram the inheritance is symbolized with an arrow pointing from the inheriting class to the parent class, see figure 5.3 [64, p. 12].

5.2.3 Thread

Multiple threads allow an application to run various activities in a parallel manner, where every thread follows its own line of execution. E.g. if there is one thread, that is responsible for communicating data through an interface to another device, it maintains the connection throughout the process. If it awaits a response from its communication partner, another thread may perform some computations without needing the first thread to first close the connection. As multiple threads share a global memory, data can be easily exchanged between them. If the calculating thread from the above example has to wait for new input data to process, the communicating thread can send a few messages. So they are not really operating at the same time, but use the resources more efficiently [65].

In which way a thread object is created depends on the program language. However, an activity needs to be specified, e.g. by providing a method as input, when the thread object is instantiated or by adding same method to an instance of a class with thread inheritance [66]. Once, the “start()” method of a thread object (including objects with thread inheritance) is called, the allocated method is executed in a separate thread of control.

5.2.4 Lock Object

Depending on the programming environment and language, the thread class may contain an object, that allows to force the processor to stay at that thread for a critical section of code. In that way, a series of processing steps are executed by the processor without interruptions from other threads. [67, p. 1]

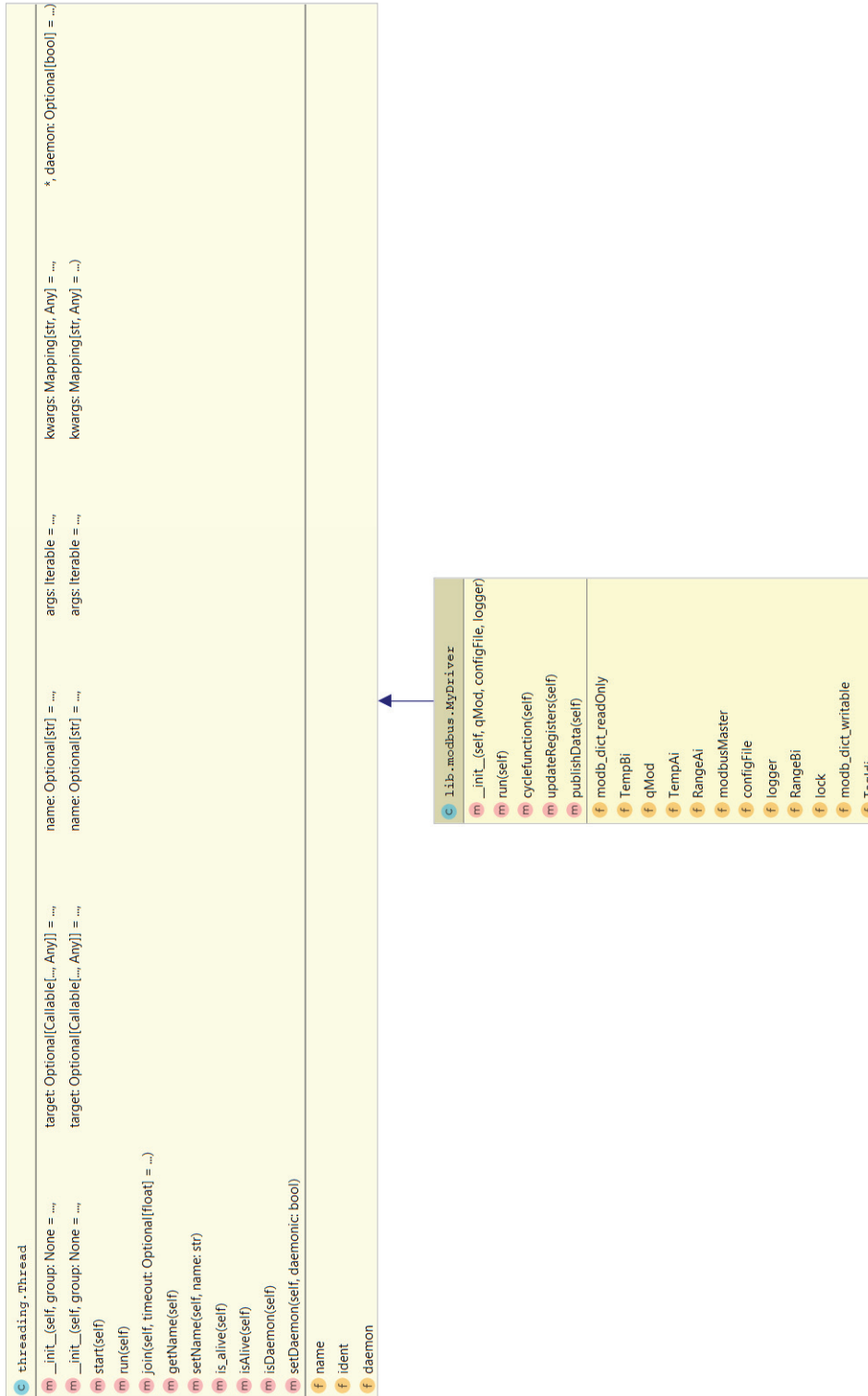


Figure 5.3: Class of the Modbus Thread containing all attributes and methods: the queues and arrays of the ranges to ingest and process them, “RangeAi” measured by Anchor “A” and “RangeBi” measured by Anchor “B”. Same applies to the temperature measurements and all Anchor specific values. Also the “ModbusMaster” and the array of the Tag IDs which are transmitted to the Anchors are among the attributes. The inheritance to the Thread Class is displayed as arrow from child to parent.

5.2.5 Queue

Attributes are either simple data types like booleans (0 or 1), integers (integer numbers) or strings (ASCII text, also called chars) or more complex elements like queues. The last one is a data storage device, that is filled and emptied - one element after the other and is equipped with a “put()” and “get()” method. [68]

Like a queue of customers, the queue structure operates on the first-in-first-out (FIFO) principal. It automatically locks the process before executing one of the two mentioned methods and releases the process after the element has been added or removed. In that manner the memory is save from data corruption, even if two threads access one queue. [69]

5.2.6 List

Lists have a static order and its elements are addressed by their index. Items of different data types can be stored in the same list and a new item would be stored at the end of the list. [70]

5.2.7 Dictionary

Another storage device is the dictionary (also called Associative Arrays or Hashes), that consist of key-value pairs. While the key, which is used as reference to obtain the value, much like the index of an array, the value can be a simple data type or a more complex object. The items in a dictionary are not ordered and the same key cannot be assigned twice. [71]

5.2.8 Logging

Additionally, a logger object is generated and written to by every thread in the application, in order to keep track of data outputs, which eventually helps to analyze the cause of problems. Filters might be applied, that let only critical errors or also warnings or even normal output data pass, depending on the filtering level. Every logger object contains formatters, that define the output structure of the log entries. [72]

5.3 Implementation of the Threads

The developed application is set up in three threads, the Modbus Thread (MT), which is in charge of the data transfer to and from the UWB devices over Modbus RTU, the OPC-UA Thread

(OT), that handles the data transfer to and from the overall network through OPC-UA, while the Algorithm Thread (AT) processes the data and continuously calculates the relative positions and orientations from the raw distance measurements. The most important code sections of the implementation are presented in Appendix B.

Within the main program, which is referred to as "manager" in this thesis, the three threads are initiated along with the configuration object ("config"), the queue of ranges ("qMod") and the logger object ("logger"), see figure 5.4. While, "config" manages the storage of global data, which is available in all threads, "qMod" is used to pass the raw measurements from the MT to the AT.

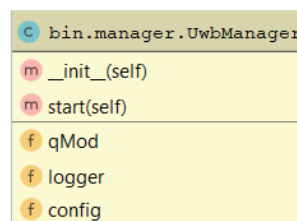


Figure 5.4: Class of the manager containing all attributes and methods. While the method "start(self)" is called only once to start the three threads, that contain the repeating loops, "init (self)" initializes the global variables "qMod", which ingests the range data from the UWB sensors, "logger" which continuously reports the applications state, and "config", which stores all variables shared between the threads.

5.4 Data Exchange between Threads

5.4.1 Global Configuration

The first attribute of "config" is a dictionary, that contains all variables that are of global interest, the second one is a lock object, see figure 5.2. As global configuration storage, "config" facilitates value transfer by its two methods: "set_values()" and "get_values()". When called, both methods receive as input a key, which addresses a key-value pair of "config"'s dictionary. While "get_values()" returns the value from the according pair, "set_value()" changes that value and therefore needs an additional input. Once any method is called by any of the three threads, "config" is locked, in order to prevent simultaneous memory access by another thread during data transfer, which could result in corrupted data, see figure 5.5.

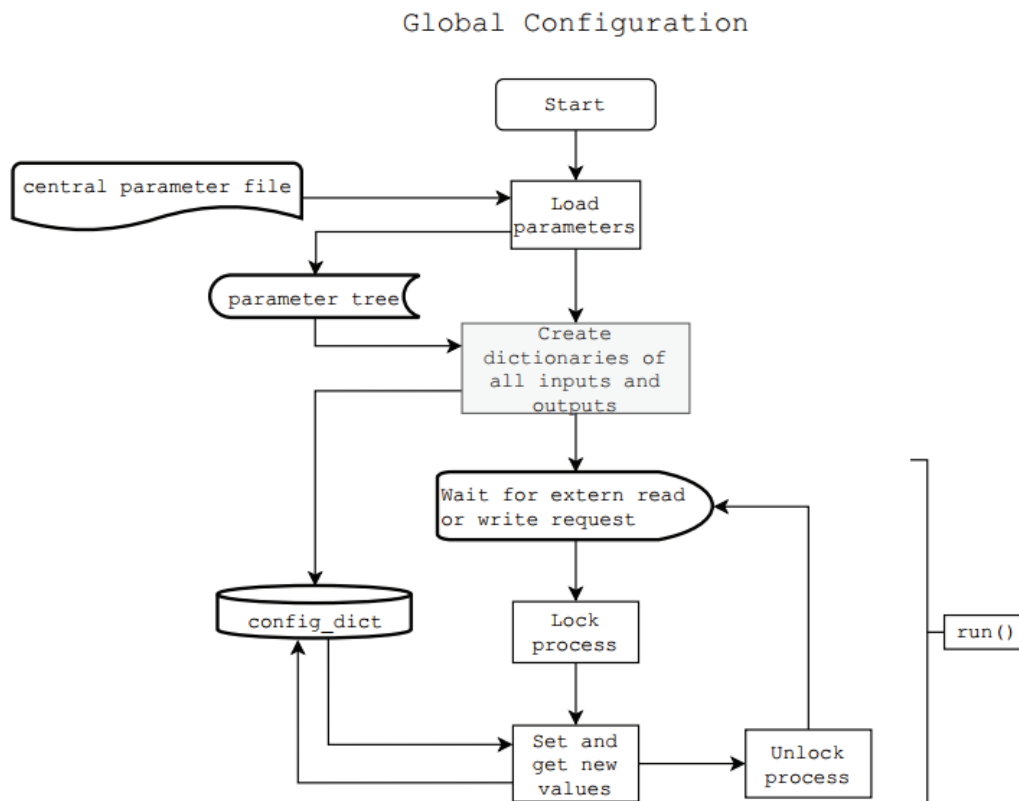


Figure 5.5: Flowchart - "config": Whole class. Once the "config" object is initialized, it is waiting for an extern read or write request. If such request occurs, the processor is locked to it until the task is finished.

Parameter File

During the initialization routine of the middleware, the central parameter file "params" is used to automatically generate a data base containing the variable names, together with their default values, dimension and access rights. According to the variable tree within, the data base is structured in sets and sub-sets to allow efficient storage and processing during the whole service routine. The file is accessible by all threads and displays the names of all global variables and their parent nodes in a tree structure, see figure 5.6.

It was prepared in the course of the software development, in order to automatically create a central database. The parameter file is written in the Yaml format, which allows the generation of a tree structure, that is interpretable by any programming language, but also conveniently readable by humans, see figure 5.6 [73]. Another format like Xml or Json would also serve the purpose [73]. The "params" file shows the superior sets ("CenterModul", "TargetModul") containing the second-level sets ("InputAlgorithm", "SlaveA" to "SlaveD" and "Modul00" to "Modul11") and sub-sets ("Input", "Output" or "Input", "OutputRaw" and "OutputCalc" respectively) at the first sub-set layer.

```

Node:
  CenterModul:
    InputAlgorithm:
      RunAvrgWindCalc: [1, 1, 10]
      RunAvrgWindRaw: [1, 1, 10]
      Width: [1, 1, 2000]

    Slaves:
      SlaveA:
        Input:
          ForceTagA: [1, 1, 0]
          SlaveIdA: [1, 1, 11]
        Output:
          DeviceIdA: [1, 0, 0]
          ForceTagModA: [1, 0, 0]
          SlaveIdModA: [1, 0, 0]
          TempA: [1, 0, 0]
          VersionA: [3, 0, 0]
      SlaveB: <2 keys>
      SlaveC: <2 keys>
      SlaveD: <2 keys>

  TargetModul:
    Modul00:
      Input:
        TagId00: [2, 1, 0]
        Width00: [1, 1, 2000]
      OutputCalc:
        DX00: [1, 0, 0]
        DY00: [1, 0, 0]
        Phi00: [1, 0, 0]
      OutputRaw:
        RangeA00: [2, 0, 0]
        RangeB00: [2, 0, 0]
        TempA00: [2, 0, 0]
        TempB00: [2, 0, 0]
    Modul01: <3 keys>
    Modul02: <3 keys>
    Modul03: <3 keys>
    Modul04: <3 keys>
    Modul05: <3 keys>
    Modul06: <3 keys>
    Modul07: <3 keys>
    Modul08: <3 keys>
    Modul09: <3 keys>
    Modul10: <3 keys>
    Modul11: <3 keys>

```

Figure 5.6: The parameter file is written in the Yaml format to be interpretable by machines and humans likewise. It contains a tree of all variables, that need to be exchanged between two threads or between the middleware and its peripherals. Only one of the four “Slave” branches and one of the twelve “Modul” branches are expanded. The rest are structured analogously.

While the “CenterModul” refers to the plant unit with the UWB Anchors ”A” and ”B”, the “TargetModul” is the unit with the Tags ”a” and ”b” attached. As there are a maximum of 12 targetted Tag pairs foreseen, they range from ”Modul00” to ”Modul11”. The ”InputAlgorithm” branch contains all algorithm specific variables, like the mounting distance between the Anchors and the running average windows. ”SlaveA” to ”SlaveD” refer to the maximum number of four UWB Anchors connected to the Modbus line. All Slave specific variables are communicated over the Modbus Interface and have therefore been discussed in the previous chapter along with the register table, see figure 4.2. Under the ”Modul” branches are also algorithm specific variables like ”Width” and the three algorithm outputs defining the relative positions ”dX”, ”dY” and the relative orientation ”Phi” per targeted Tag pair attached to one ”Modul”, i.e. one plant unit. The rest of the variables belonging to each ”Modul” branch are also transmitted over the Modbus RTU line and have already been addressed.

At the end points of this variable tree, are the variable names and an array. That array contains three entries, that define the dimension, the access right and the default value of the variable: While the first entry defines the length of an array, “1” referring to a scalar, the second value is equal to “0” for only readable or “1” for read/write access and the last entry sets the default value between 0 and 65535.

Dictionary Preparation

During the initialization routine of “config”, the dictionary is prepared by iterating through “params”, see figure 5.7. During each iteration, the dictionary is populated with a key that equals the name of the variable, i.e. the tree leaf. The value in the key-value pair has the dimension and default value of the according array entries. Within “config”, the access right value stays unused, which is not the case for the dictionary preparation in the MT and the OT, that sort the variables by access right.

5.4.2 Queue of Ranges

Thanks to the FIFO principle, queue objects have the benefit of simplifying running average computations, which is beneficial for scattered data. For another thing, especially in the case of data exchange between two threads, a queue is very convenient as it automatically locks the process during memory manipulation. Hence, no additional thread locking is necessary when filled or emptied.

As the MT continuously ingests the range outputs from the UWB sensors attached to the Modbus line, the middleware stores the incoming data in “qMod”. With its flexible length, “qMod” acts

as buffer until the first element is emptied by the AT. As the algorithm continuously computes the average of the scattered measurements, a queue is the preferable candidate for supplying the AT with range values. All other variables are communicated through dictionaries, including the algorithm output and the Modbus variables but the ranges. While the AT processes the data from “qMod”, the OT supplies the plant network with the raw distance measurements through the range arrays in “config”.

5.5 Modbus Thread

The MT is an instance of the Modbus driver class that inherited from the thread class, ergo the MT object has all functionalities of a thread object, as well as its own additional features, see figure 5.3.

5.5.1 MT - Initialization

During the initialization phase all thread specific - and Modbus interface specific attributes are initiated. The core of the Modbus interface related attributes is the “modbusMaster”, that establishes - during initialization - the connection to the Modbus network. During its initialization the “modbusMaster” attribute accesses the ICF outside the main program through the manufacturer’s Modbus library. As described in the previous chapter the ICF contains all necessary information to let the IPC participate as Modbus Master, see figure 5.8.

Among the Modbus specific attributes are also the input parameters for instantiating the Modbus driver object. When creating the object, the manager provides it with “qMod”, “config” and “logger”. Only when initialized by the manager, i.e. external to any thread, can an object or attribute be shared by two or more threads. While “config” and “logger” are shared by all threads, “qMod” is accessed by the MT and the AT only.

At this point, the storage devices for all Modbus variables are initiated, such as the range arrays measured by UWB sensor “A” and “B”: “RangeAi” and “RangeBi”. In the current version of the middleware, one Anchor pair per unit is active at a time, leading to two Modbus Slaves in conversation with the IPC as Modbus Master. While the above terminology is used, when Slaves “A” and “B” are the participants of the ranging Anchor pair, they could be replaced by “C” and “D”. Each letter refers to a different Modbus Slave ID.¹

The suffix “i” refers to the plurality of ranging partners of the Modbus Slaves “A” and “B”. In

¹The meaning of the expression “A and B” or pair “A-B” refers to the statement “A and “B” or “C” and “D” or pair “A-B” or “C-D” for better reading.

the arrangement shown in figure 5.9, the Anchor pair “C-D” from “Unit 4” measures the ranges to “c-d” of “Unit 1”, while the Anchors “C” and “D” of “Unit 1” range to the Tag pairs “a-b” of “Unit 2” and of “Unit 3” simultaneously. Consequently, Anchors “C” and “D” of “Unit 1” have two ranging partners. On the other hand, the Anchor pair “C-D” of “Unit 2” has two Tag pairs as ranging partners, which will lead to two independent position calculations, as later shown in the section AT. The maximum number of Tag pairs per Anchor pair is twelve. Hence, all attributes ending on “i” are one-dimensional arrays of length 24. While the rest of the Modbus variables is provided to the other threads by updating the entries of the “config” dictionary, the arrays “RangeAi” and “RangeBi” are updated within “config” and put into “qMod” every time the Modbus registers are refreshed.

Analogously, the temperatures measured by the partner Tags of Slave A and B are stored in the arrays “TempAi” and “TempBi”. So along with the information that is needed to calculate the distance, the Tags’ UWB signal include their own temperature measurement. As each Anchor of the Anchor pair ranges to the same set of Tags, the two temperature arrays should be identical. However, the evaluation of both offers means of surveilling the communication. If different, the arrays indicate a malfunctioning of the UWB or Modbus communication, e.g. a mismatch of UWB addresses or Modbus Slave IDs.

While the range and temperature arrays are filled by the UWB sensors with incoming data, the array of Tag IDs “TagIdi” transfers the UWB device IDs of ranging partners to both devices of the Anchor pair. Through the later documented OT, new Tag IDs can be imposed from the plant network to the middleware, that are further communicated over “config” to the MT, where they are sent to the according Slaves over the Modbus network.

Just like the dictionary preparation within the “config” object, two dictionaries are created within the MT, see figure 5.10. On the one hand, “modb_dict_readOnly” provides storage for register values that are sought from the Modbus network, such as the firmware version, Modbus address and temperature of each Slave, as well as “RangeAi” and “TempAi”. On the other hand, “modb_dict_writable” saves the data that is imposed by the Modbus Master on the Slaves, like “TagIdi”, see figure 4.2.

Unlike ranges, temperatures and Tag IDs that are arranged in arrays before being stored in “config” or transmitted over Modbus, the rest of the Modbus variables are stored directly in “config” or are fed directly to the Modbus network. Therefore, no extra attributes outside the dictionaries need preparation during initialization. Finally, the lock object is initialized that is later used to assure save data storage in the MT.

5.5.2 MT - Loop

Within the main part, in our case called "run()" method, see figure 5.8, of the MT runs a loop that executes the method "cyclefunction()", which is divided into two major methods: "updateRegisters()" and "publishData()".

The first action within the "cyclefunction()" method is the locking of the thread, followed by "updateRegisters()" method, see figure 5.11, where first the Modbus register names are related to the key-value pairs of the Modbus dictionary. That step is only necessary for variables that are stored as arrays throughout all dictionaries of the main program, because the Modbus library only allows a single register per attribute. Then, all Modbus inputs are stored in "modb_dict_readOnly" and "config", while the outputs are extracted from "config" and written to "modb_dict_writable" and further to the related Modbus registers. This is done by iterating through the keys of the according dictionary. Hence, the data exchange between Modbus network and "config" is accomplished.

After making sure that the positioning system is in use by counting the number of given Tag IDs, "RangeAi" and "RangeBi" are put into "qMod", while all Modbus outputs are written to the log file by handing them to "logger". Afterwards the MT is unlocked and then pauses for one second before executing the loop again.

5.6 Algorithm Thread

Same as the MT, the classes of the AT and the OT, too, inherited from the thread class, see figure 5.12.

5.6.1 AT - Initialization

AT and OT obtain the same input parameters from the manager during instantiation: "qMod", "config" and "logger". From "config" the following variables are extracted that are necessary for the position calculation, which are ingested over the OT as they are set from within the plant network: "runAvgWindRaw" is the running average window to smoothen the raw measurements, while "runAvgWindCalc" is the amount of smooth range sets that the output angle and offset are approximated to. Further the widths between the sensor mounts of the Anchor pair "width1" and the ones of the Tag pair "width2" are extracted and initialized, see figure 5.13.

Another queue "qRangesMean" is initialized, that stores the average values of "qMod" after applying the "runAvgWindRaw". Finally, "d2r" simply stores the translation factor to obtain

degrees from radians.

5.6.2 AT - Loop

If “qMod” is still empty, the AT pauses and checks the length of the queue again until “qMod” has been filled by the MT. If the amount of elements in the queue exceeds the “runAvgWindRaw”, the oldest entries of “qMod” are deleted. Within “runningAverage()”, the average of all “qMod” values is calculated and stored in “qRangesMean”, all done separately for each measured length. Therefore, one element of “qRangesMean” consists of the quartet of average lengths between two units.

If “qRangesMean” exceeds the “runAvgWindCalc” the oldest entries are deleted, see figure 5.14. However, if the size of “qRangesMean”, i.e. the number of quartets, on which to perform the approximation, is yet too small, the AT returns to the top of the loop after a delay until there are enough quartets to apply the algorithm. For each quartet stored in “qRangesMean”, a trilateration routine is performed, that evaluates the x and y coordinates for both Tags relative to the center of the Anchors leading to four coordinates in total per quartet.

Subsequently, those four coordinates times the “runAvgWindCalc”, i.e. the amount of quartets, are summarized in a matrix “D” and at the same time are used to calculate the overall averages “dX” and “dY”, the coordinates of the center between the Tags, see figure 5.15. Then, “dX” and “dY” are subtracted from the according entries of “D”, which yields a new matrix “DmeanFree”. After applying the singular value decomposition on “DmeanFree”, the most significant value of the matrix “V”, that contains the eigenvalues of the orientation, is extracted. The approximated angle can be easily evaluated by the arc sine of same element. The main components of the programmatic implementation of the algorithm are shown in figure B.4. Once the position data has been updated within “config” and written to the log file, the loop is repeated after a delay of half a second.

5.7 OPC-UA Thread

Finally, the OT is in charge of transmitting the algorithm output to the plant network. To the third heritage of the thread class, the manager passes on “config” and “logger” as inputs, during instantiation, see figure 5.16.

5.7.1 OT - Initialization

In order to establish a connection to the plant network, an OPC-UA object node is implemented. In our case, the IPC operates as OPC-UA Server, hence a Server object node is generated and integrated to the OPC-UA network. Same Server object is provided by a library, available in the used program language. During initialization, the “server” attribute sets name, URI and endpoint URL of the node, to enable recognition and addressing by other participants, i.e. other network nodes, see figure 5.17. The subsequently specified namespace of the object node allows the integration of new OPC-UA variable nodes into the network.

Based on the previous models, the dictionaries “opc_dict_readOnly” and “opc_dict_writable” are generated according to “params”, with the difference, that the values in the key-value pair are not simple elements but OPC-UA variable nodes, see figure 5.18. Each variable node is named after the related “params” element and that name is added to the available namespace of the OPC-UA network.

The OPC-UA variables are structured according to the “params” hierarchy, where the “server” object node corresponds to the “Node” in “params”. All underlying branches within the OPC-UA tree are named after their model in “params” and are superior to further nodes, until the OPC-UA variables terminate a branch. Not only are the access right values in “params” used to separate the dictionaries, but also defines the access right for each OPC-UA variable node.

5.7.2 OT - Loop

The data exchange between “config” and the OPC-UA variables is in accordance with the data exchange between “config” and the Modbus registers. In the OT, the above mentioned library enables easy updating of the OPC-UA variables through according read and write commands, see figure 5.19. Before rerunning the loop, the OT pauses for 0.2 seconds. As the crucial data flow starts at the MT and ends at the OT, the update frequency increases, in order to rule out data losses.

Create dictionaries of all inputs and outputs

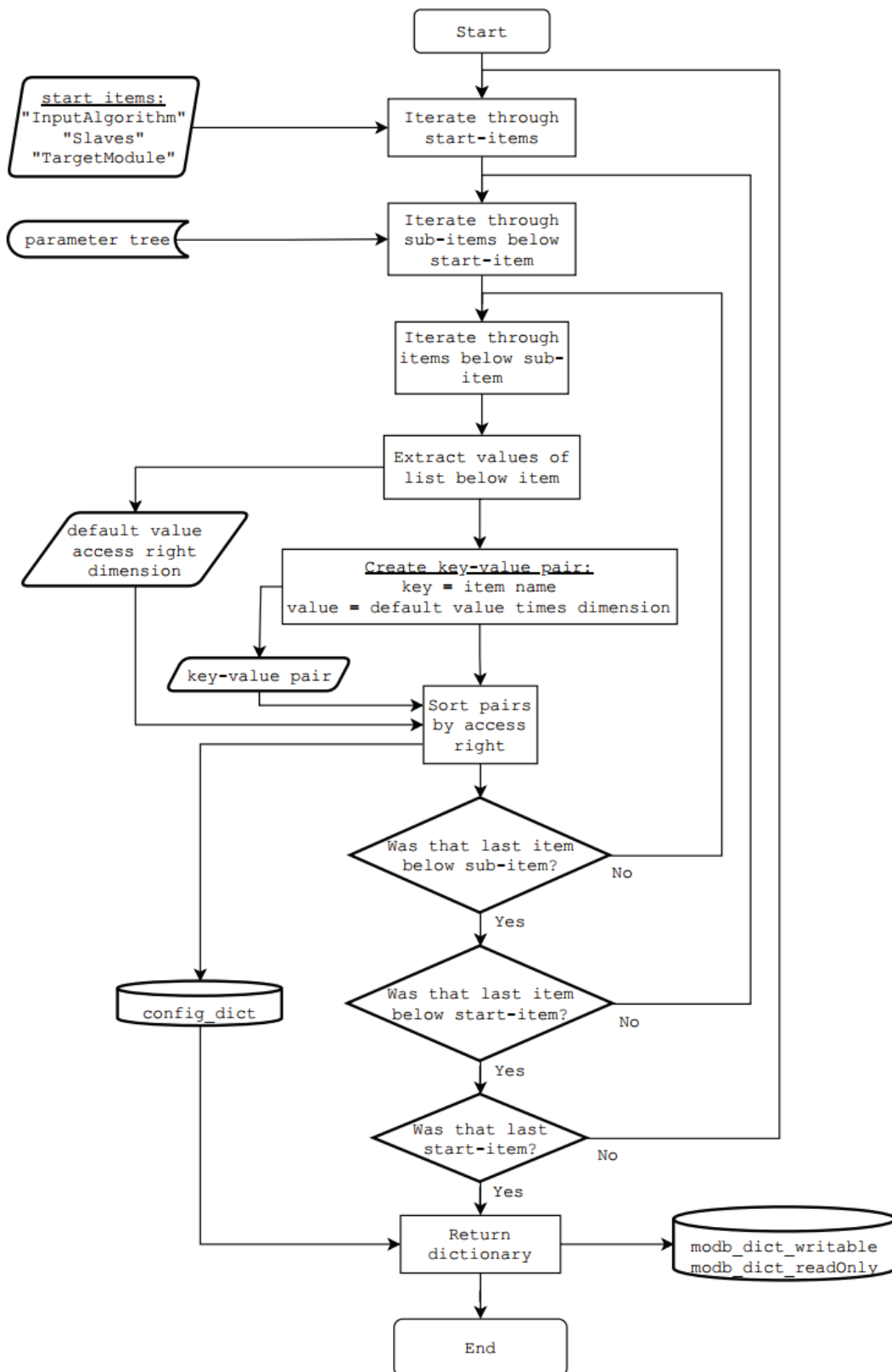


Figure 5.7: Flowchart - "config": Create dictionaries of all inputs and outputs by iterating through the Yaml document containing the parameter tree.

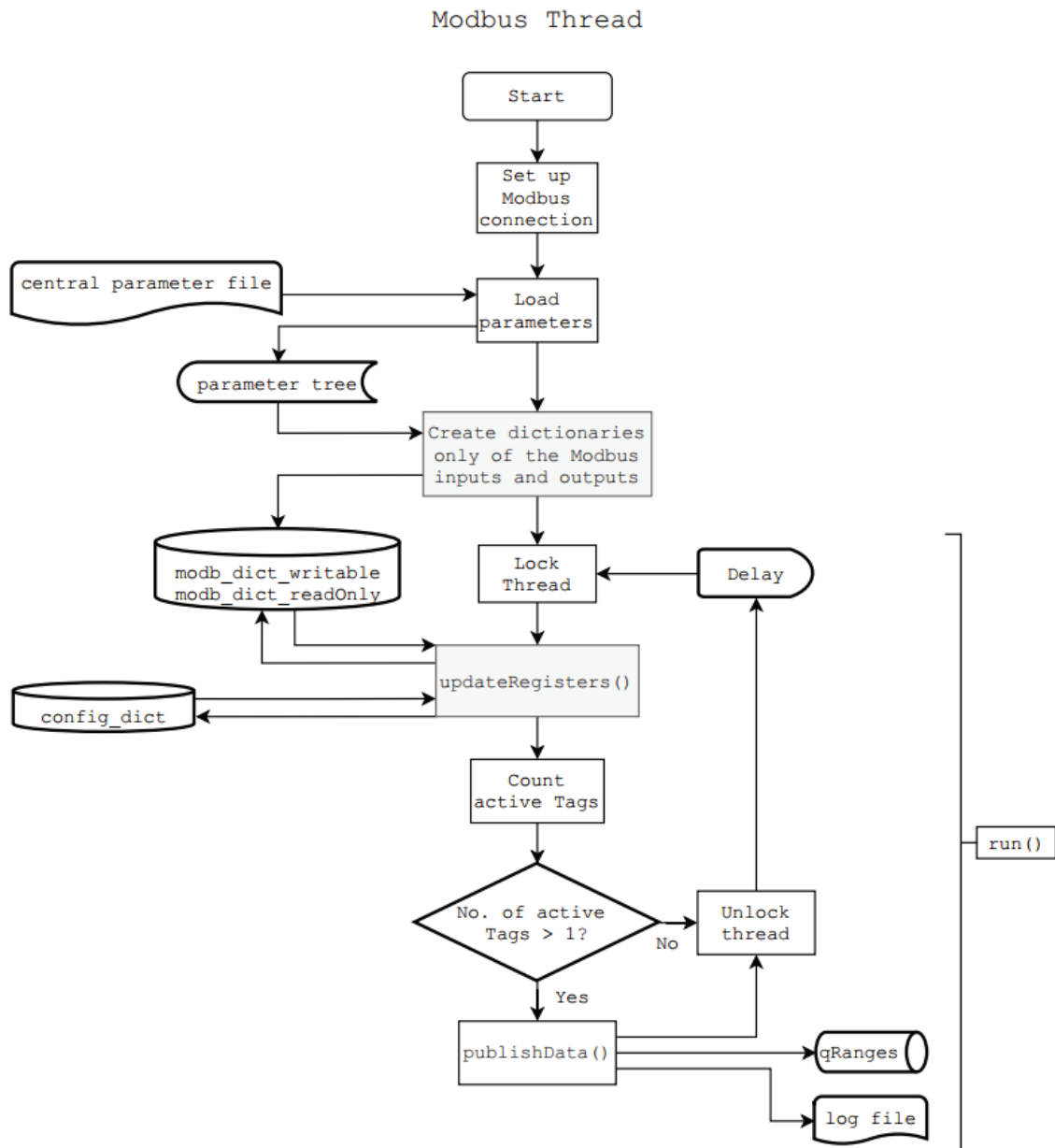


Figure 5.8: Flowchart - MT: Whole thread. After setting up the Modbus Master and updating the values on the middleware's end, two dictionaries – one for the transmitted data and one for the received data – are initialized. As long as Tag pairs are responding, the MT updates all read and written values while locking the processor to its own thread.

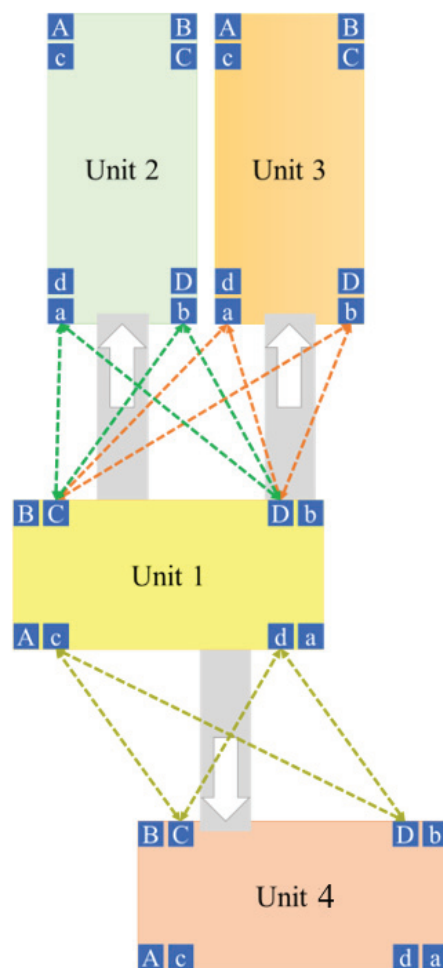


Figure 5.9: Each unit is equipped with four Anchors ("A" to "D") and four Tags ("a" to "d"). Hence, the Middleware of "Unit 1" is in charge of positioning "Unit 2", as it is connected to the Anchors of the ranging constellation, and not the other way round.

Create dictionaries only of the Modbus inputs and outputs

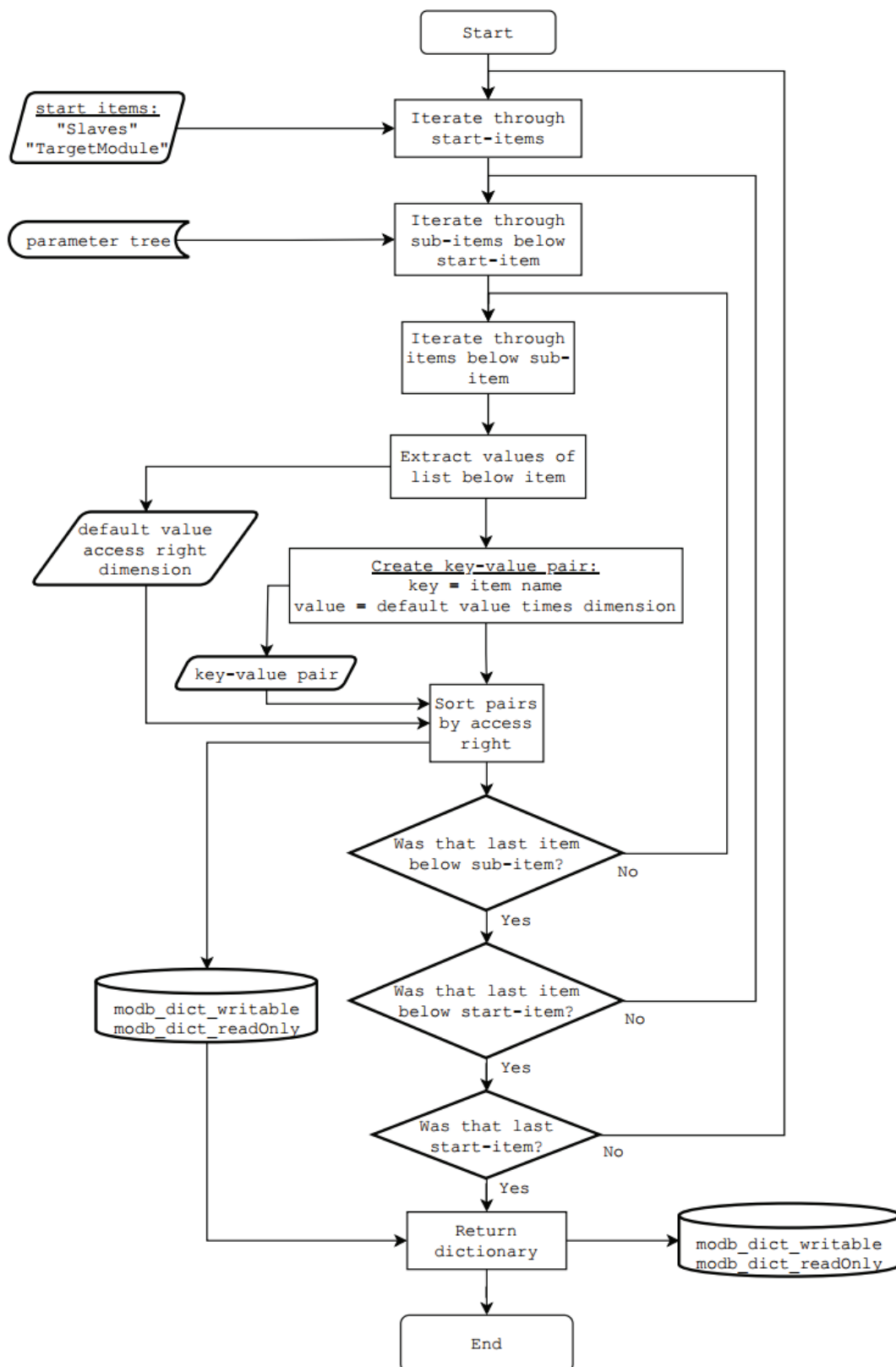


Figure 5.10: Flowchart - MT: Create dictionaries only of the Modbus inputs and outputs by iterating through the parameter tree. The two separate dictionaries facilitate the efficient execution of read and write tasks.

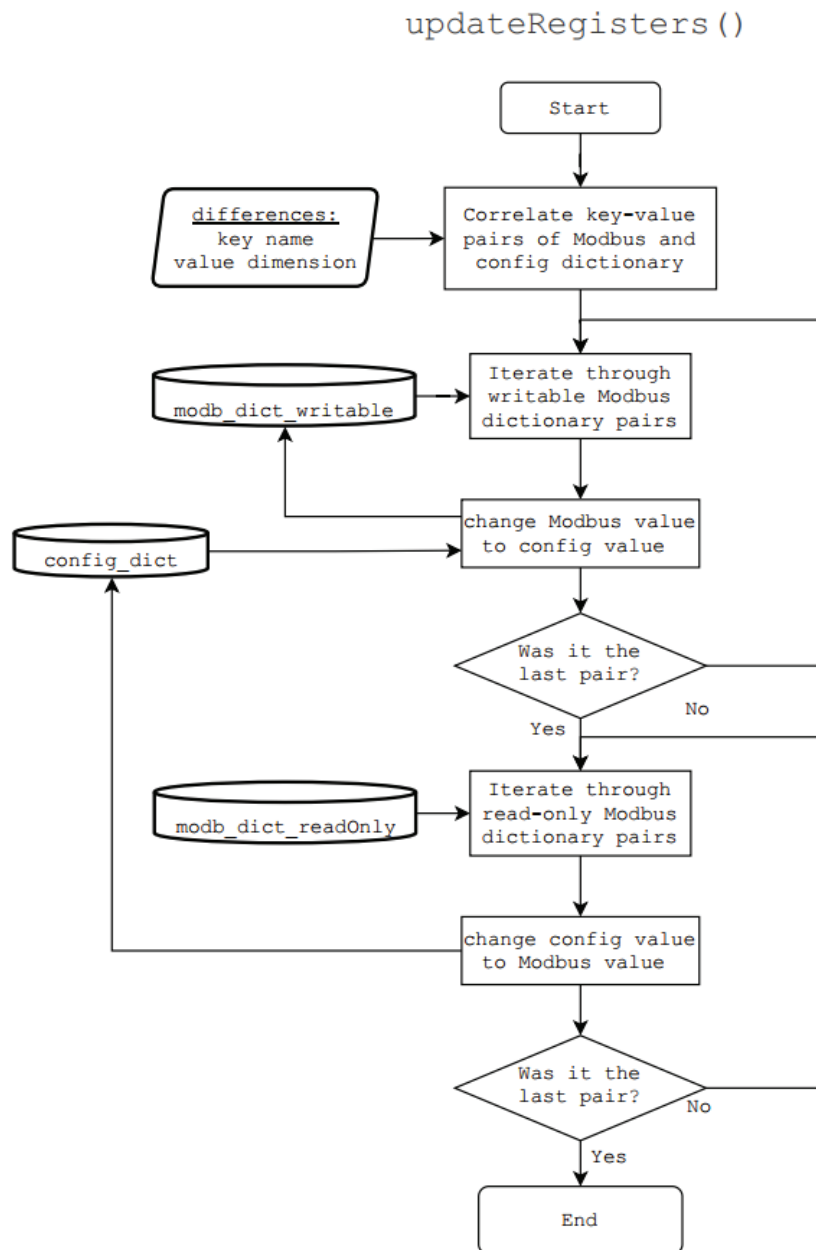


Figure 5.11: Flowchart - MT: Set and get new values in the “updateRegisters()” method. The MT iterates separately through the read- and write dictionary until the entire dictionaries have been checked for updates.

```
lib.algorithm_new.RpoAlgorithr
m __init__(self, qMod, configFile, logger)
m do_algorithm(self)
m runningAverage(self, qMod, la)
m trilaterateRPO(self, width1, dataSet)
m circirc(self, x1, y1, r1, x2, y2, r2)
m run(self)
f qRangesMean
f qMod
f d2r
f configFile
f logger
f width2
f width1
f RunAvgWindRaw
f RunAvgWindCalc
```

Figure 5.12: Class of the AT containing all attributes and methods of the thread child. Additionally to the thread specific attributes and methods, the AT contains all methods that process the data until the relative position and orientation are computed in each iteration every one to two seconds. Furthermore, the raw measurements are among its attributes, “qMod”, together with other parameters from the network, like the running average windows or the mounting width between two paired sensors, “runAvgWindRaw”, “runAvgWindCalc”, “width1” and “width2”.

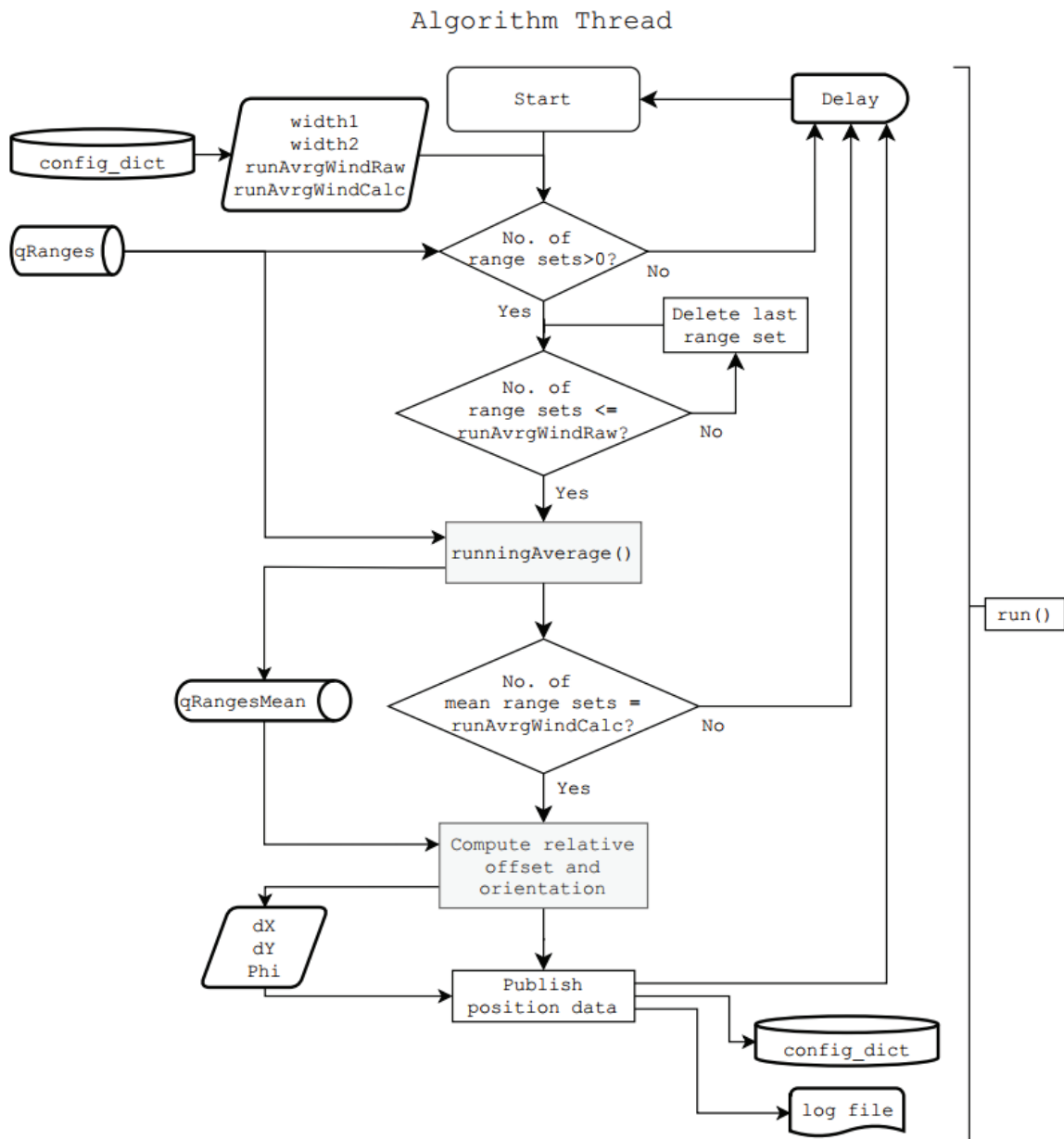


Figure 5.13: Flowchart - AT: Whole thread. After reading the network specified geometry inputs and running average windows, the AT checks if there are enough entries to perform the computations. If so, the number of data sets is reduced to the first running average window to smoothen the ranges separately. If the amount of smooth sets is enough, the relative offset and orientation themselves are computed and published.

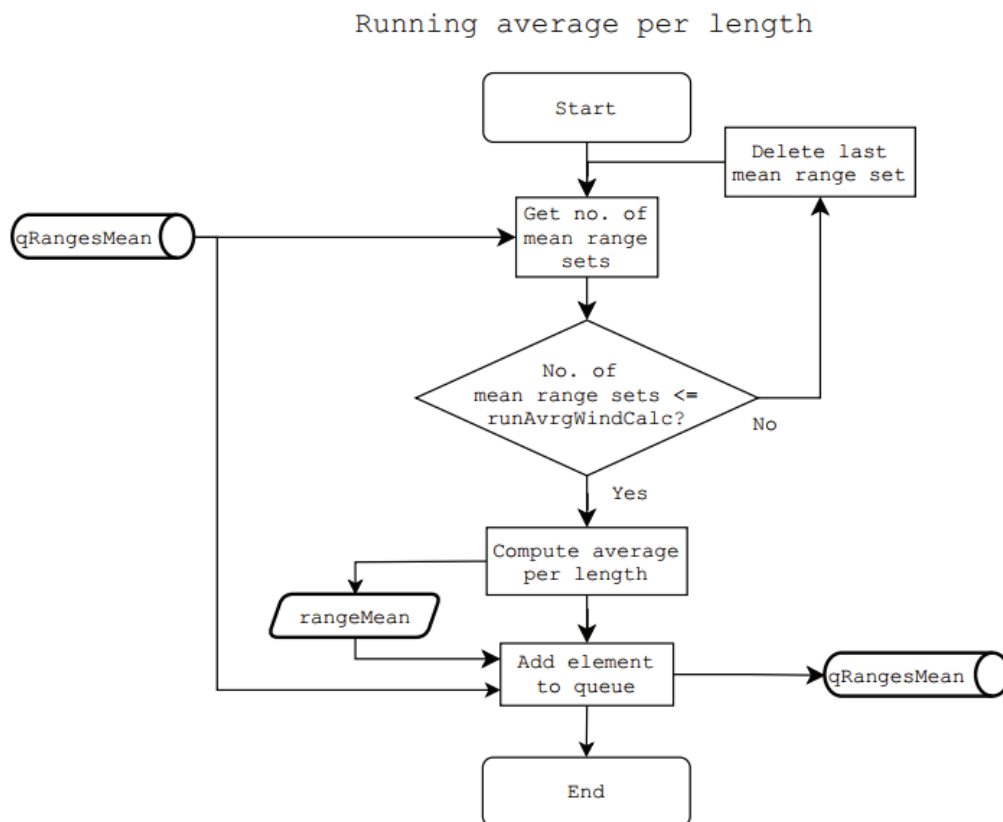


Figure 5.14: Flowchart - AT: Run average per length. Within the AT another queue, “qRangeMean” is initialized to store the results of the first data smoothing step and feed the actual position and orientation calculating equation.

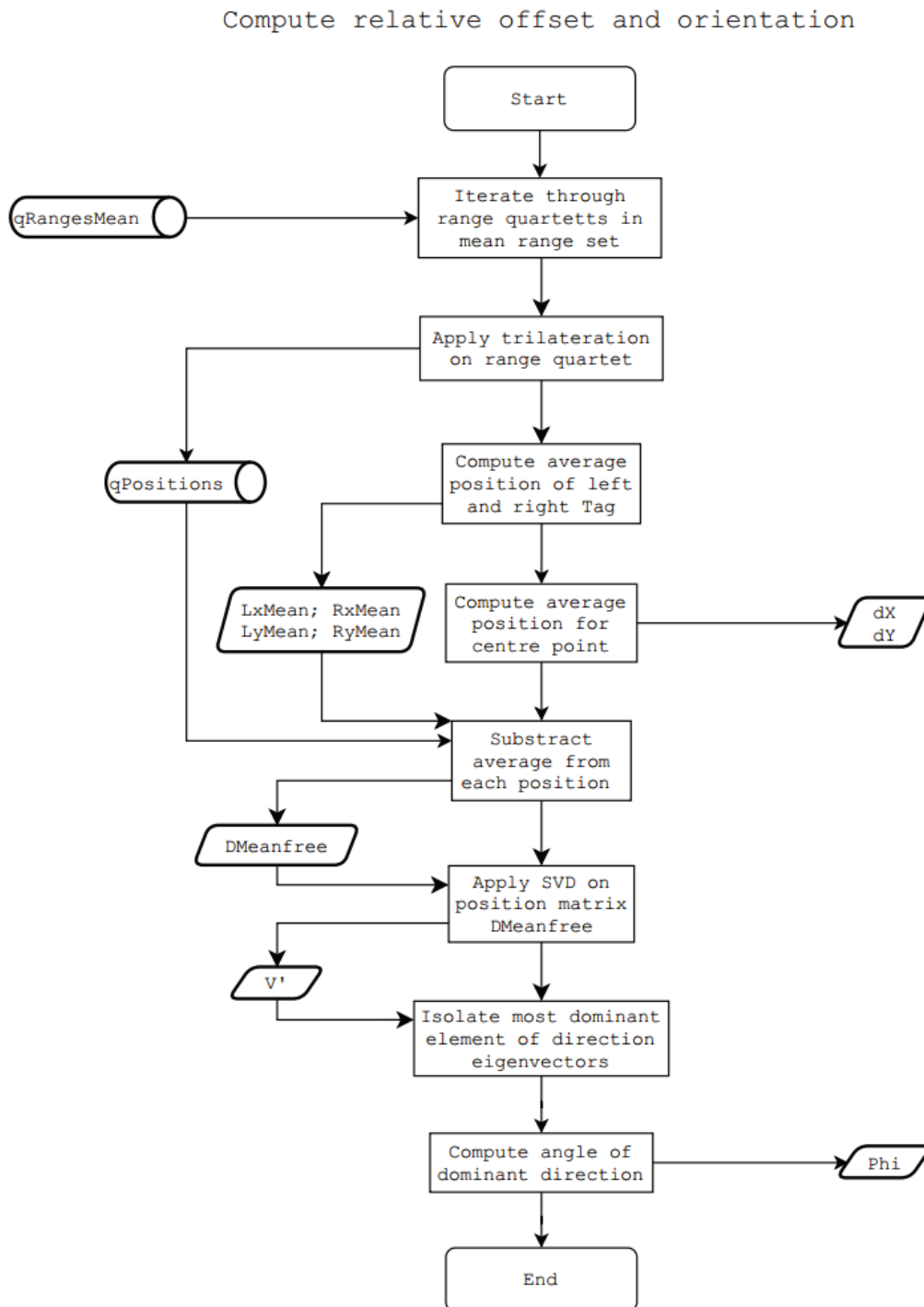


Figure 5.15: Flowchart - AT: Compute relative offset and orientation. Through trilateration, which is the calculation of intercepting points of two cycles, the evaluation of mean distances and an SVD, the best fitting approximation of the relative position and orientation are extracted in every iteration.

```
lib.opcua_server.OpcuaWorke
m __init__(self, qMod, configFile, logger
m run(self)
m setConfig(self)
m publishData(self)
m search_dict(self, ns, var, text)
f server
f opc_dict_writable
f qMod
f var
f logger
f configFile
f goOn
f text
f opc_dict_readOnly
f SERVER_NAME
f SERVER_URI
```

Figure 5.16: Class of the OT containing all attributes and methods. Apart from dictionaries that are used for data transfer, the attributes: SERVER NAME and SERVER URI are defined to successfully establish an OPC-UA connection.

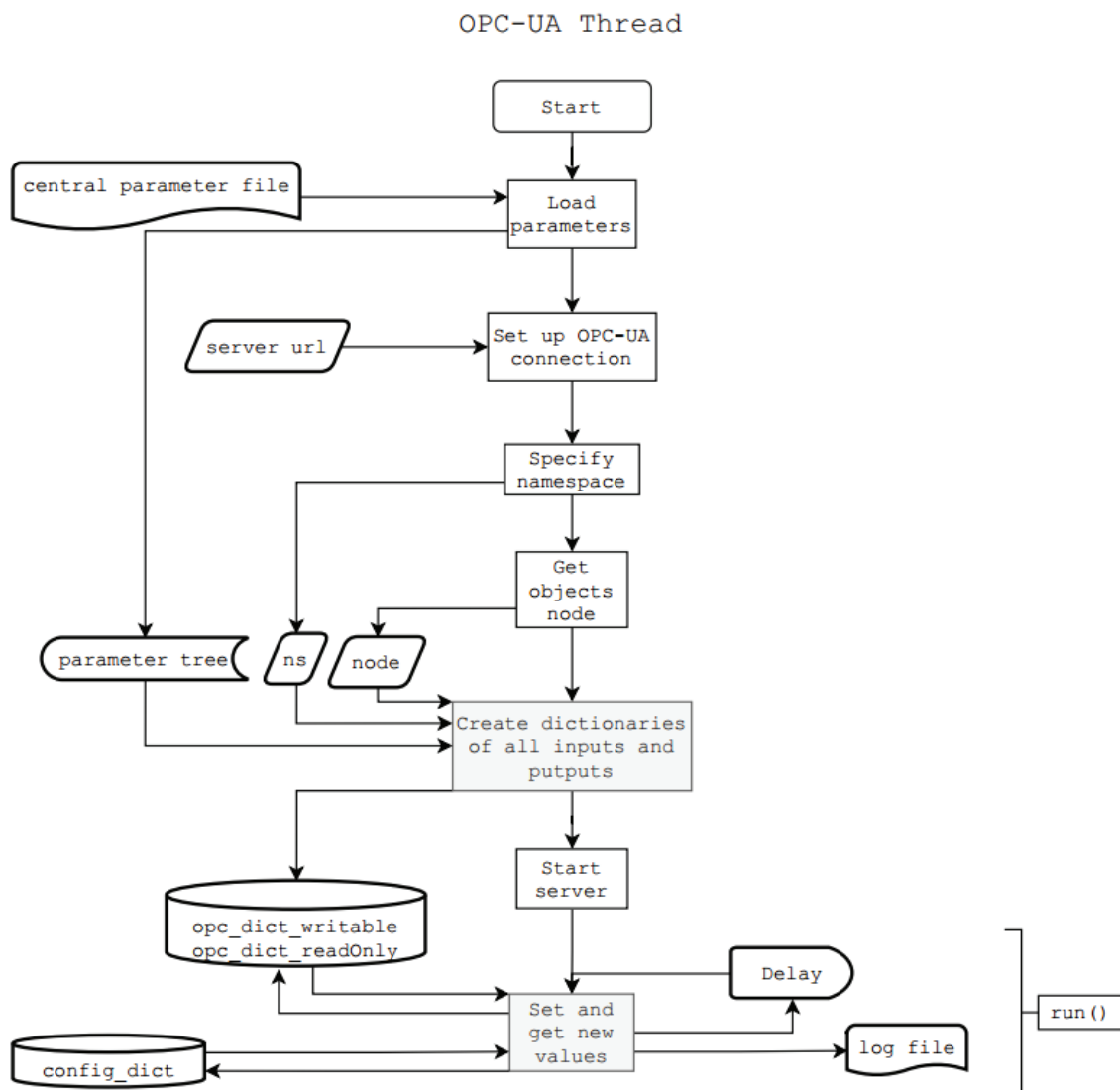


Figure 5.17: Flowchart - OT: Whole thread. After setting up the OPC-UA Server, the namespace is specified and used to generate OPC-UA object nodes, that are addressable by the network's OPC-UA Client. When the dictionaries are initialized, the OT is ready for data updates.

Create dictionaries of all inputs and outputs

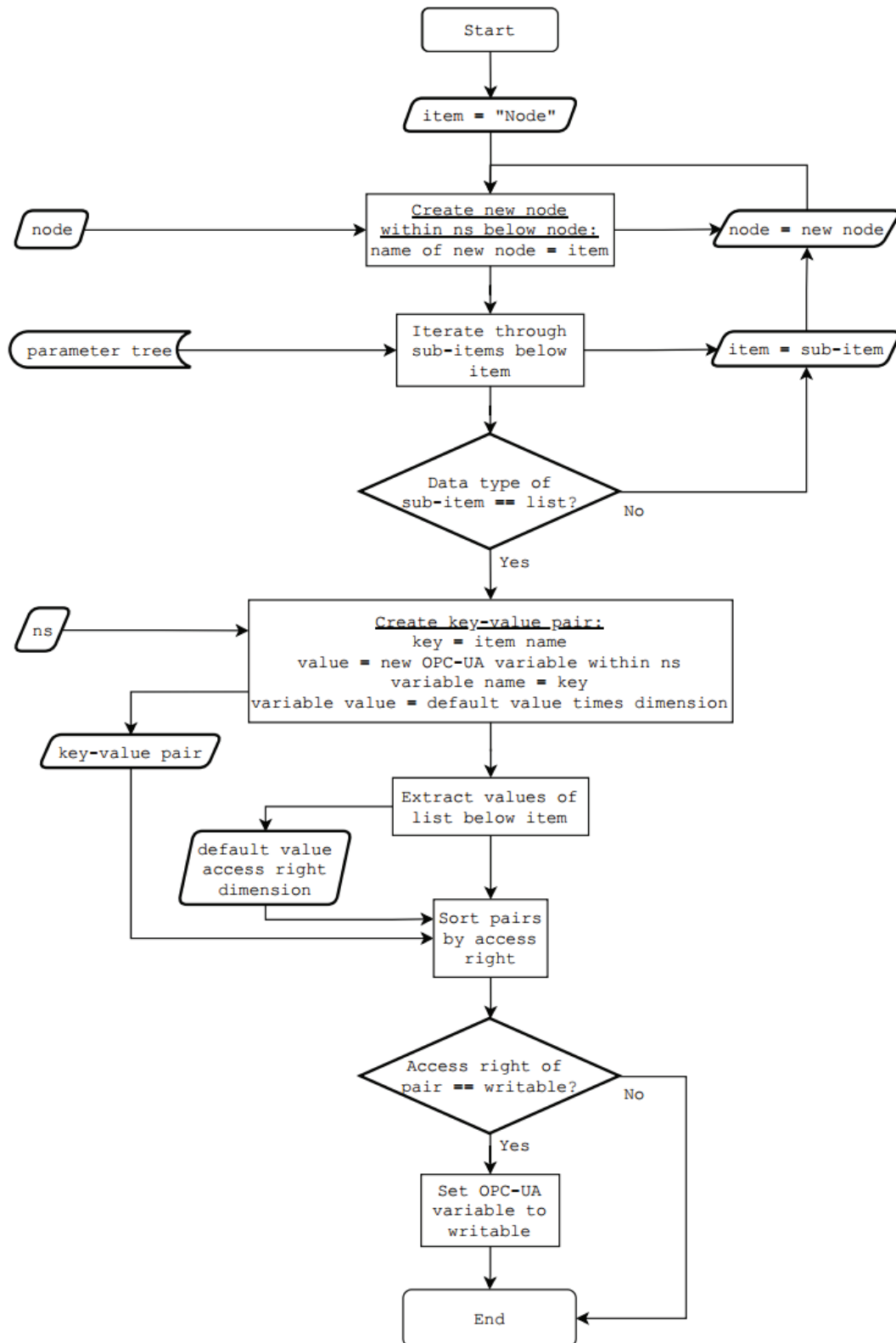


Figure 5.18: Flowchart - OT: Create dictionaries of all inputs and outputs. During the initialization of the OPC-UA read and write dictionaries, OPC-UA object nodes are created that adopt the read and write access rights from the parameter file in addition to the key-value pair.

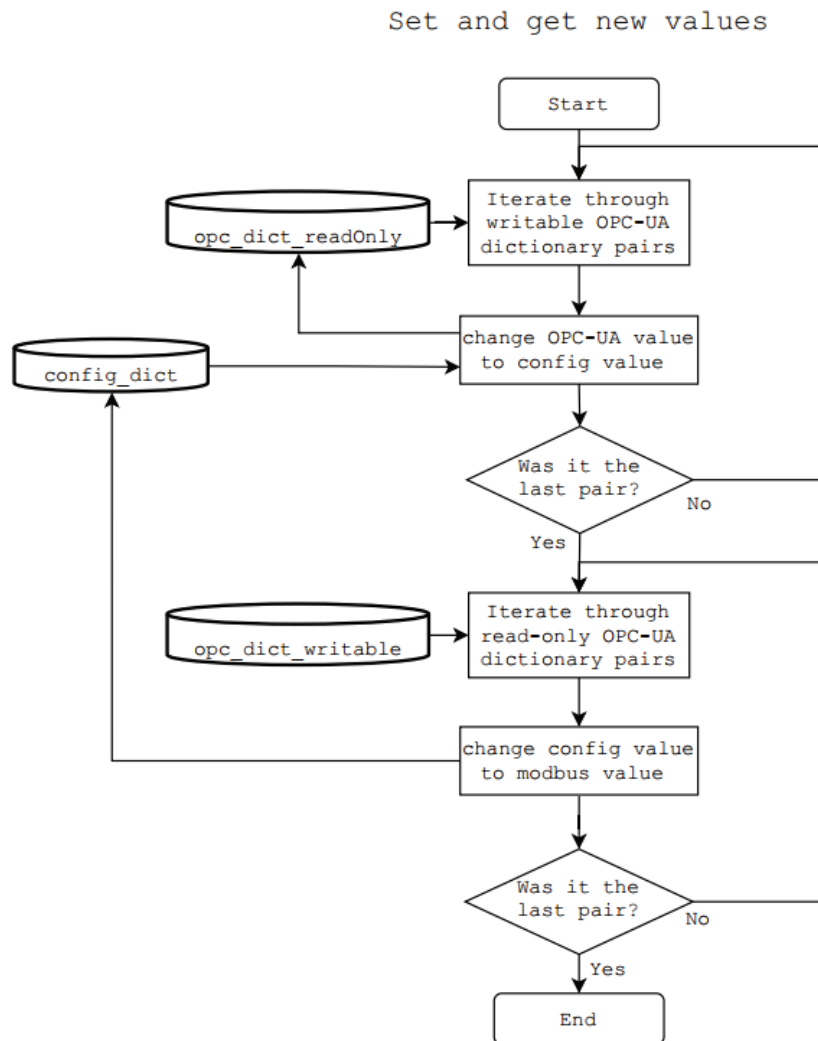


Figure 5.19: Flowchart - OT: Set and get new values. When updating the OPC-UA values, the OT directly reads from or writes to the object nodes integrated into the OPC-UA network of the plant.

Chapter 6

Results

Within this chapter, the developed middleware is evaluated w.r.t the Modbus transmission, the algorithm implementation and the OPC-UA connection. In the end, an assessment of the whole middleware will be given. Most of the tests were performed with the same setup as the one during software development, in which tripods act as plant units, see figure 5.1.

Before connected to the IPC, the UWB sensors received a firmware, that includes the Modbus interface and were adapted in such a manner, that a measurement initialization is addressed specifically to one or more Tags through their IDs, but not broadcasted to any Tag within reach. Therefore, the relevant UWB Tag IDs are required at the Anchor's end and must be provided by the main application over the Modbus connection. Hence, the data transfer from the IPC over Modbus to the Anchors is necessary to even initialize a UWB measurement.

6.1 Modbus Connection

The first tests determine if all registers are transmitted and received by the IPC and if the data rate is accomplished. In the MT, the register values were written to the log file, which is at the same time the source of the graphs. The following graph 6.1 shows a test measurement containing range values in detail. There, the distance values of Anchor "A" and Anchor "B" ranging to Tag "a" and Tag "b" are displayed, which are read in through Modbus at 1Hz. Simultaneously, the UWB Tag IDs were transmitted from the IPC to the Modbus Slaves, i.e. Anchors "A" and "B".

As the UWB Anchors require the UWB Tag IDs, in order to initiate a measurement in the first place, the transfer of the Tag IDs from the IPC to the sensors over Modbus was successful. The Modbus connection also refreshes the values at every new register reading once a second, which can be gathered from the small jumps between two data points. Consequently, the data exchange

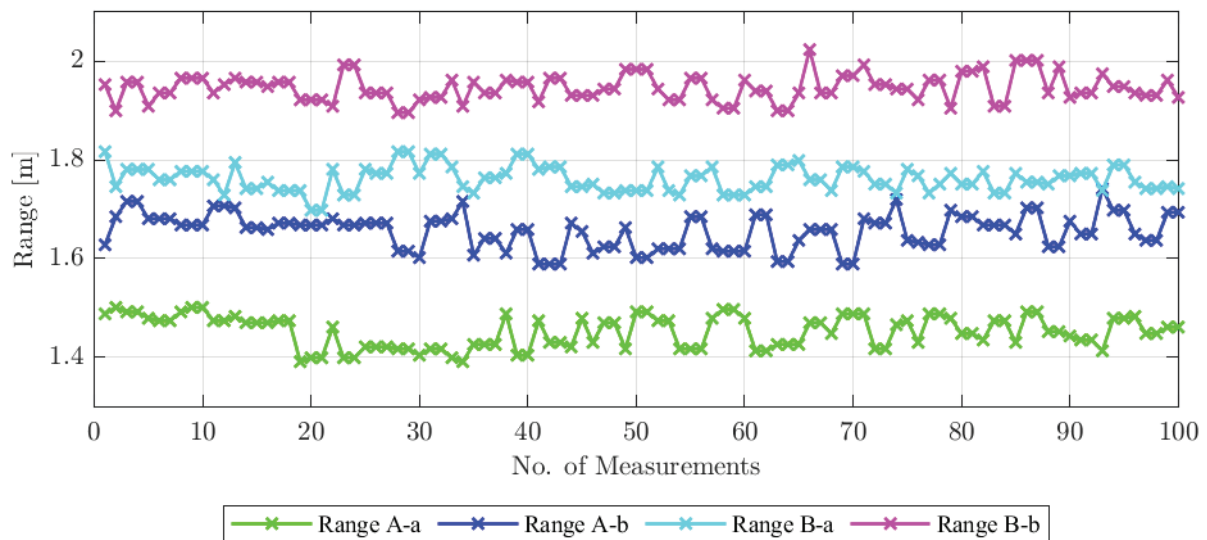


Figure 6.1: UWB range measurements read from the middleware: All four UWB range measurements are displayed, that are obtained by two UWB Anchors "A" and "B" measuring the distance to two UWB Tags "a" and "b", see figure 5.1. While the specific Tag IDs are sent through the Middleware and Modbus to the Anchors, i.e. the Modbus Slaves, the Anchors feed the Middleware over Modbus with their measured values, if the communication with the specified Tags was successful.

over Modbus works in both directions.

Graph 6.2 also includes the output of the temperature readings. While the temperatures of the Anchors are transmitted once per Anchor, those of the Tags are transmitted for redundancy by both Anchors per each Tag. Hence, the temperature of each Tags is ingested two times by each Anchor respectively which results in two identical values represented as two overlapping lines in the graph in case of a successful data connection.

Unlike the range values, that jump almost at every new measurement, the temperature values stay the same for 2 to 35 measurements. The origin lies most probably in the resolution of the temperature within the UWB sensor's software. The assumption was made, because the jumps and the stable values of the temperature outputs are constant. In figure 6.3 all temperatures are displayed, that were updated by the same Anchor and Tag pairs during 500 measurements.

The temperature values only occupy a maximum of three steady values. The gap between two steady values is constant throughout the 500 test points. Furthermore, the graph shows, that both Anchors receive the same temperature values from both Tags. "Temp a (by A)" is identical to "Temp a (by B)", as is the case with the measurements from Tag "b".

On the other hand, the data points of the range measurements jump up and down from one output to the next within most sections, which implies, that the Modbus interface transmits each data value correctly. However, during other sections, like data points no. 7 to 12 in graph 6.4, the

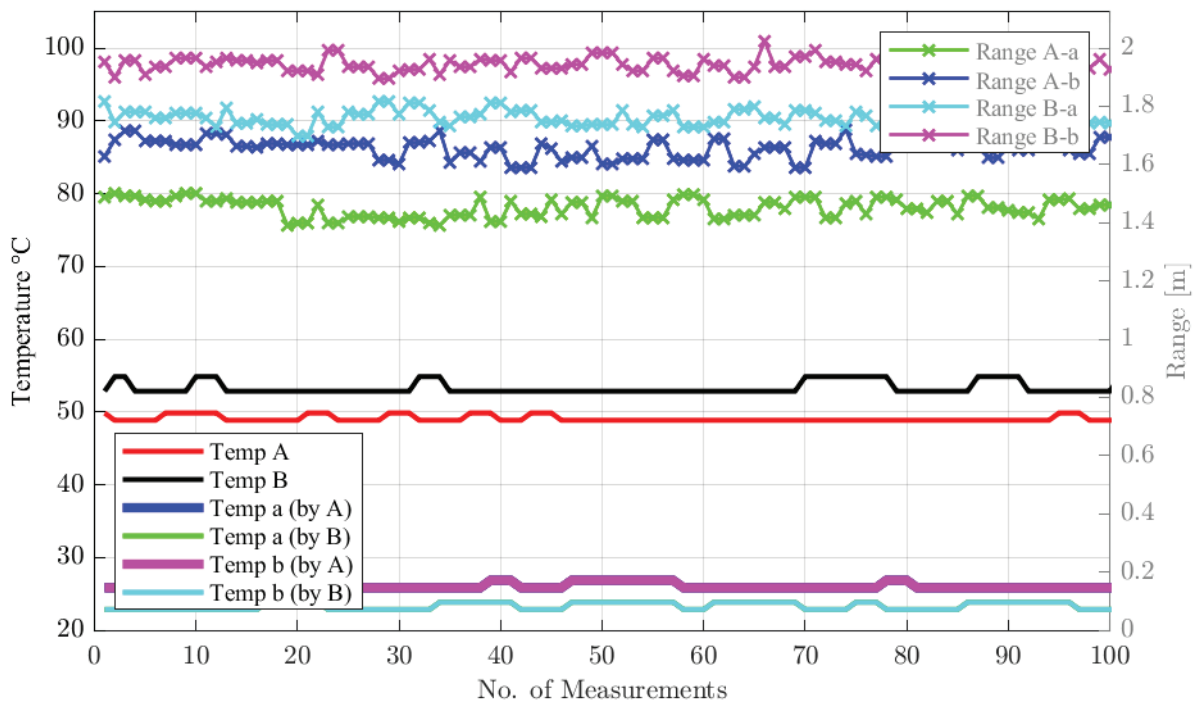


Figure 6.2: UWB range and temperature measurements from the middleware: Next to the four range measurements "Range A-a", "Range A-b", "Range B-a", "Range B-b" the Temperature measurements are shown. While "Temp A" and "Temp B" are simply the temperature values of the Anchors themselves, "Temp a (by A)" is the temperature measurement from Tag "a" that is communicated over UWB to Anchor "A" and from there transmitted to the IPC over the Modbus interface. In case of a successful transmission, the plots of "Temp a (by A)" and Temp b (by B)" are overlapping as shown by the thin lines on top of the bold lines. Same applies for Tag "b".

incoming range value of the "A-a" pairing is frozen.

Those stuck values could suggest, that a transmission rate of 1Hz eventually causes unsuccessful requests or responses between Master and Slaves, when the demanded amount of registers is communicated over Modbus. In this application, many registers needed to be communicated, due to the implementation of longer unique IDs per Tag, that each occupy four registers instead of one in the original setup. On top of that, 20 Tag IDs need to be permanently reserved and continuously transmitted over Modbus to each of the four Slaves. The reason for the extensive register transmission is that the online tool provided by the manufacturers of the IPC, only permits the generation of a static MMM, which means, that the maximum amount of registers, that might be occupied during the running of the sorting plant, must be reserved from the beginning.

With all transmitted range and temperature registers per reserved Tag per connected Anchor, together with the 320 Tag ID registers, as well as the Anchor specific ones, a total of 816 registers are transmitted at the same time. 816 registers transmitted per second by the Master correspond to 13,056 kbps, while the Modbus protocol usually operates between 9,6 and 19,2 kbps [74, p. 1]. As the Slaves also need to respond to the Master's request, the assumption is made, that

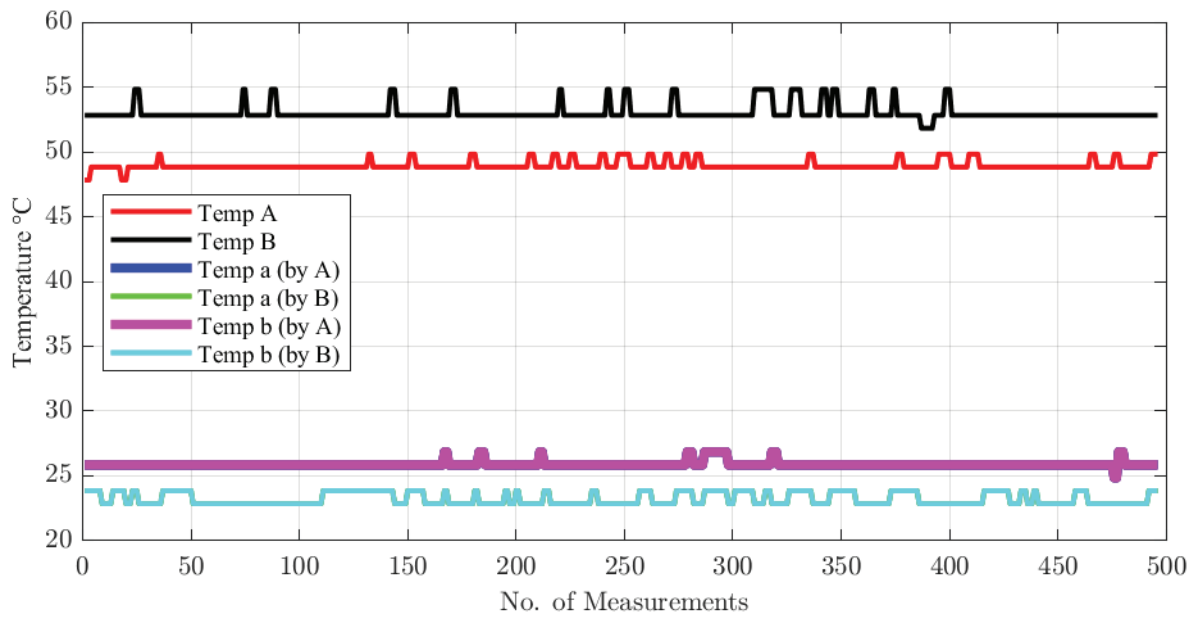


Figure 6.3: Temperature measurements: Only the temperature measurements of the UWB sensors are shown for a total of 500 measurements. Each overlapping data pair represent the temperature measurement of one Tag which is communicated over UWB to both Anchors and indicates that the temperatures are being reliably measured. They are stable during the measurement and the resolution of the temperatures is visible in the discrete step nature of the data.

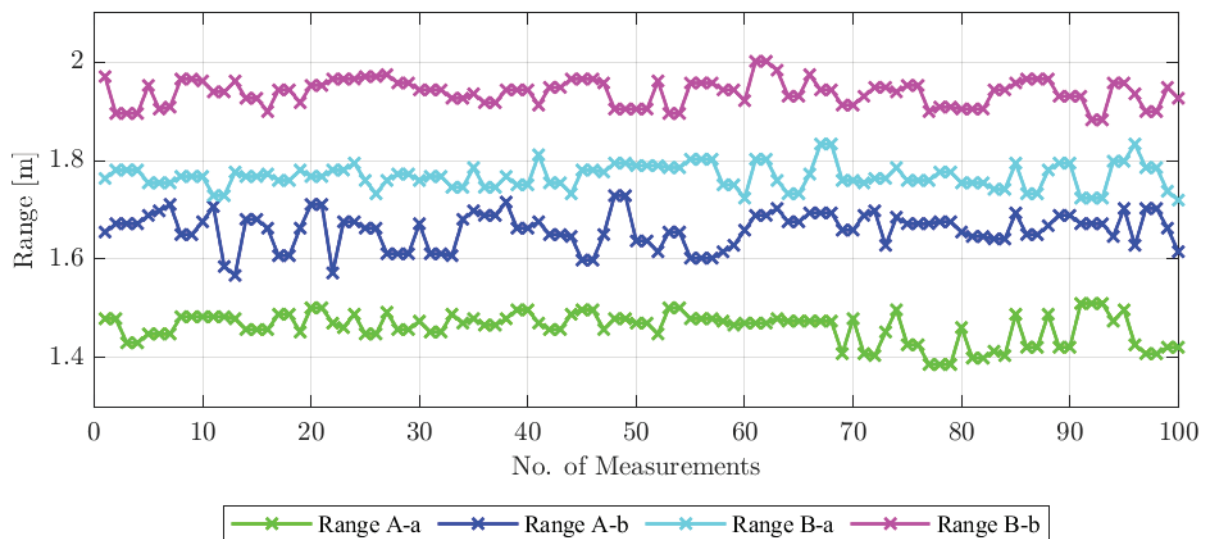


Figure 6.4: UWB range measurements with frozen values read from the middleware: All four UWB range measurements are displayed, that are obtained by two UWB Anchors "A" and "B" measuring the distance to two UWB Tags "a" and "b", see figure 5.1. Occasionally the otherwise jumping range values, that are ingested over Modbus RTU remain constant throughout a few seconds.

the high data rate might cause the occasional freezing phenomena.

6.2 Algorithm Performance

6.2.1 Tripod Tests

At first, the algorithm implementation was tested w.r.t the approximate relative offset and angle between the tripods, which was then compared to the algorithm outputs. However, the tripod tests only provided significant information concerning the data flow and smoothening of the data. Even though the setup served its purpose during software development, it wasn't fit for judging the accuracy, as the space around the tripods was limited and therefore only small ranges could be evaluated.

Graph 6.5 shows the algorithm outputs, whereas the angular offset "Phi" is plotted to the secondary y-axis on the right side and expressed in degrees. During those measurements, the tripod containing the Tag pair was turned, while the centers of the tripods remained at fixed locations. The turning of the tripod can be observed in the graph when looking at the shape of

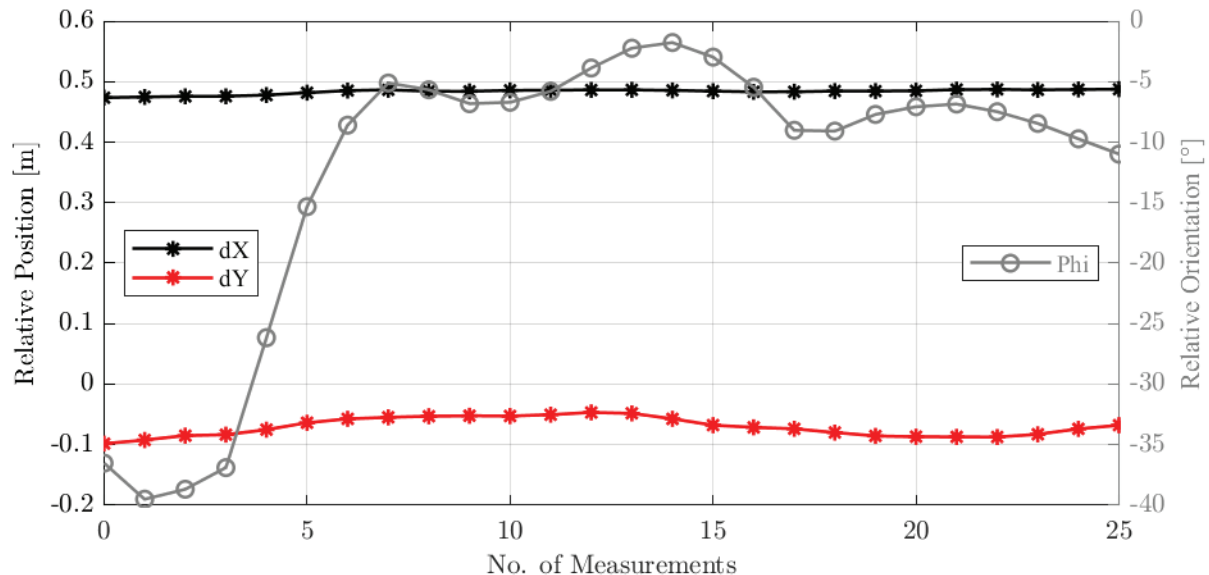


Figure 6.5: Relative position and orientation output: After the UWB range measurements are smoothened and processed by the algorithm, the relative position in x- and y-direction "dX" and "dY", as well as the relative orientation "Phi" between two succeeding units are obtained.

the "Phi" curve. Starting at -40° it reaches at the end -10° . Also the fixed location of the tripods is presented in the graph.

In order to better observe the smoothened outputs of the algorithm compared to the original

UWB values, graph 6.6 shows all range values between the Anchor and Tag pairs, as well as the resulting algorithm output “dX”. Compared to the jumps of the raw UWB measurements,

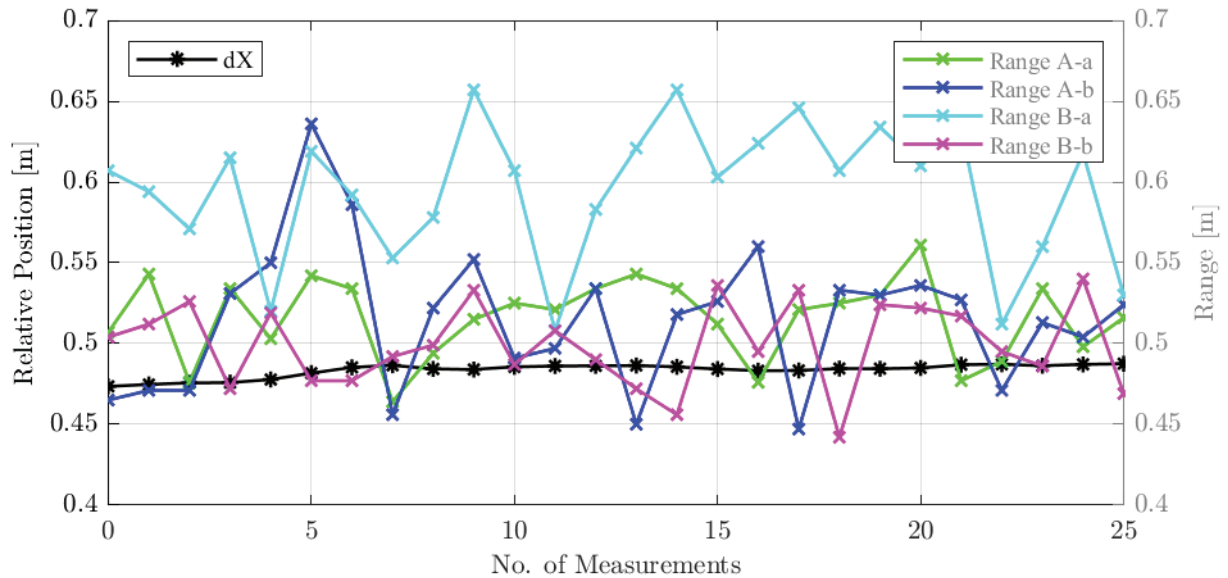


Figure 6.6: UWB range measurements compared to the calculated relative position: “dX” is by far steadier compared to the unsmoothed raw inputs and smaller than the average range measurement due to the diagonal UWB communication where at least one must be greater than the relative position.

the output of the offset in x-direction “dX” is considerably steadier, which is the result of the running average computations. As the locations of the tripods and consequently their relative offsets “dX” and “dY” also remained practically constant, the computations with smoothed data is in this case a better image of the reality, than without.

6.2.2 Tests on Plant Units

The detailed evaluation was done by mounting the UWB sensors on real plant units, see figure 6.7. Within those test series, the implementation of the algorithm is examined w.r.t. to its accuracy. All ten constellations, presented in chapter “Preconditions”, shown in figure 6.8, were tested in a big hall. On all containers, that were now connected through conveyor belts, the UWB sensors were mounted at the very top of the edges, see figure 6.9. This mounting position was found out to yield the least disturbed signals w.r.t. the NLOS situations caused by the belts.

The technical setup remained as before: While the Tags were connected to the power supply, the Anchors were additionally connected to the IPC over the Modbus line. By means of a laser sensor, the distances between the UWB devices were measured and documented as a reference, before the UWB ranging took place.

The plot 6.10 shows the output of the positioning middleware of the first constellation, see figure



Figure 6.7: First field test constellation: Two plant units were arranged in a predefined relative position in an industrial environment, not yet connected through a conveyor belt.

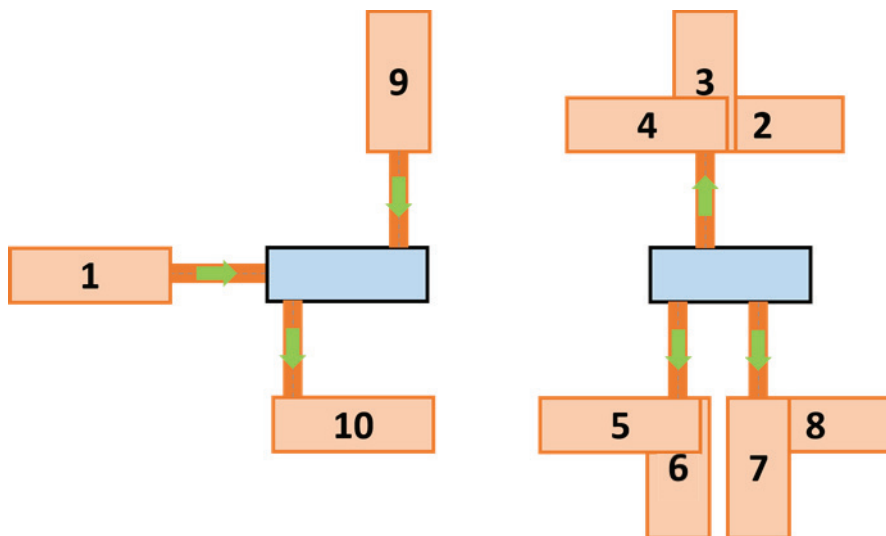


Figure 6.8: The ten different possible arrangements of two plant units: Considering the actual succeeding sorting processes, as well as the given variety and designs of the sorting machines, a total of ten relative location arrangements are test cases for the developed system.

6.11. While the blue circles represent the median and quantiles of the UWB measurements from Anchor “A” to both Tags, the red ones originate at Anchor “B” ranging to both Tags “a” and “b” respectively. The green dashed line and the y-axis of the plot enclose the computed angular offset “Phi” between the units. Same green line intersects the yellow line at the point with the coordinates “dX” and “dY” of the middleware’s computations. Hence, the yellow line connects the centers of the point clouds, that are generated in the middleware by the continuous trilateration of the UWB measurements, see figure 6.12. The black dots result from the trilateration of the laser measurements. The circles with no boarding dotted lines represent the true distance between the two Tags from the center of the equally colored point cloud.

Arrangement no. 1 of two units resulted in position outputs within the demanded accuracy



Figure 6.9: Second constellation of field tests: For the final system verification the developed system is installed on plant units in an assembly hall, now connected through conveyor belts. The sensors are mounted at the top of each edge - marked by red rectangles - and connected to the power supply. Only the Anchors are additionally connected through Modbus RTU to the IPC as well.

limits of 200mm. Plot 6.10 shows that the mean values of “dX”, “dY” and “Phi”, that were calculated by the middleware, are 20mm, 24mm and $0,37^\circ$ off those values based on the laser measurements. The system’s output is very close to the reference measurement in the first constellation. Therefore, the cross sections of the red and blue circles representing the middleware’s calculation of the Tags’ position obscure the grey spot marking the Tags’ position evaluated by the reference measurements. As shown in table 6.1, the relative position outputs of the algorithm are within the required accuracy in arrangement no. 1,2,6,9 and 10 and outside in the other arrangements. Arrangement no. 5, see figure 6.13, yields the position outputs with the greatest errors of an average “dY” value, that is of 270mm off the reference, see figure 6.14.

However the source of the errors can be found in the box plots of the UWB measurements, see figure 6.15. Two of the UWB measurement errors in constellation no. 5 exceed their reference by more than 200mm. In comparison, all errors in constellation no. 1 are smaller than 100mm, three of them even smaller than 25mm. Between Anchor “A” and Tag “b”, the UWB ranging path crosses two feed belt, that tend to have an impact on electromagnetic waves, as their frames

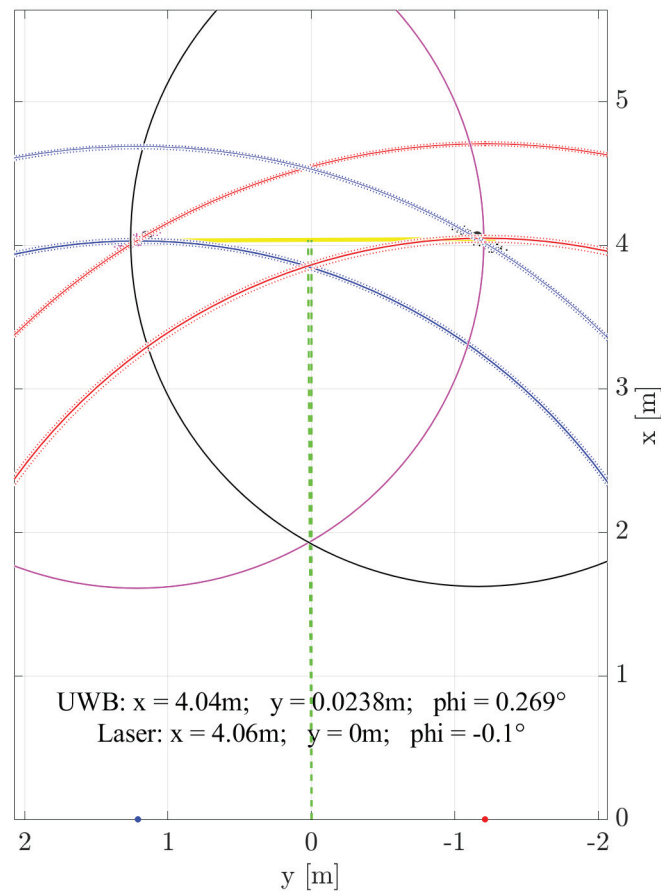


Figure 6.10: Graphic representation of the determination of the relative position and orientation for constellation no. 1: Shown are the UWB range measurements as blue and red triple circles, the Tag positions calculated by the middleware as black and pink point clouds and the by the reference measurements as gray spots (almost obscured for constellation no. 1). The black and pink single circles define the real distance between the Tags once spanned from the center of the black point cloud and once from the pink to show the relation to the calculated Tag distance. Same centers of the point clouds are connected with a yellow line which serves as perpendicular basis to construct the green broken line indicating the relative orientation which is computed in the middleware's algorithm as angle.

are made of metal¹. That and the fact, that the incoming angle at the sensors is the steepest one, leads to the conclusion, that reflections and multipath propagation are causing disturbance in the UWB signal in the more obscured arrangements.

An important observation can be made when inspecting the plots of 6.16. They show, that the implemented algorithm performs as it should. The calculus yields the center point coordinates and relative angle of the opposite unit by connecting the corner points (black and pink point

¹Tag "a" and "b" refer to any Tag pair, the Anchors of the system are ranging to, as the algorithm doesn't differ between Tag and Anchor positions "A-B" or "C-D" and "a-b" or "c-d"

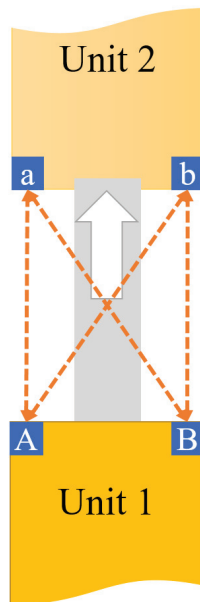


Figure 6.11: Constellation no. 1 of two neighboring plant units.

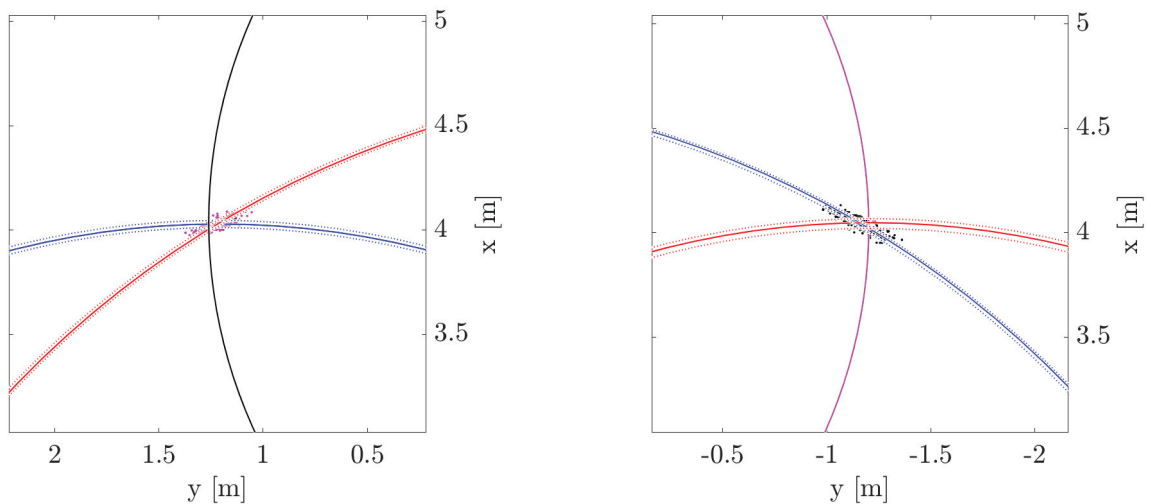


Figure 6.12: UWB range measurements from the middleware compared to the laser reference measurements for constellation no. 1: The red and blue circles represent the median bordered by the dotted 25%- and 75% quantiles of the UWB range measurements. While the inner circles correspond to the measurements "A-a" or "B-b", the outer ones correspond to the diagonal ranges "A-b" and "B-a". Measurements initiated by Anchor "A" are in blue, by Anchor "B" in red. The intersection between the blue inner circle ("A-a") and the red outer circle ("B-a") represents the middleware's calculation for the position of Tag "a" which analogously applies for the position of Tag "b".

cloud), that were found through trilateration of the raw UWB measurements (blue and red circles). However, the corner points, that were obtained by the laser measurements (black dots) are distant to the intersections of the red and blue circles.

UWB Middleware vs. Laser Reference- Constellation no. 1 to 10

No.	dX	dY	Phi
1	-20mm	24mm	0,37°
2	-70mm	-103mm	-0,21°
3	0mm	220mm	-0,90°
4	-120mm	250mm	2,60°
5	-220mm	-270mm	-6,15°
6	-80mm	-180mm	-0,55°
7	20mm	213mm	1,88°
8	-140mm	230mm	4,25°
9	-10mm	-110mm	-2,56°
10	-70mm	181mm	0,19°

Table 6.1: Errors of algorithm outputs "dX", "dY" and "Phi" compared to reference measurements for all ten possible constellations between two connected plant units.

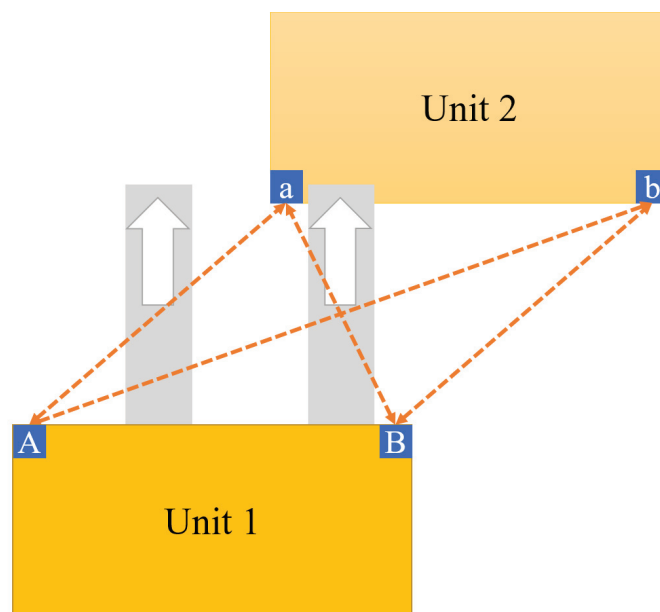


Figure 6.13: Constellation no. 5 of two neighboring plant units.

The black circle has the center of the black point cloud at its center point and the width between the mounting positions of the Tags as its radius. Analogically, the pink circle was generated. The distance between the pink point cloud to the closest part of the black circle is 400mm, which means that the UWB measurements would yield a greater distance between the Tags, than there is.

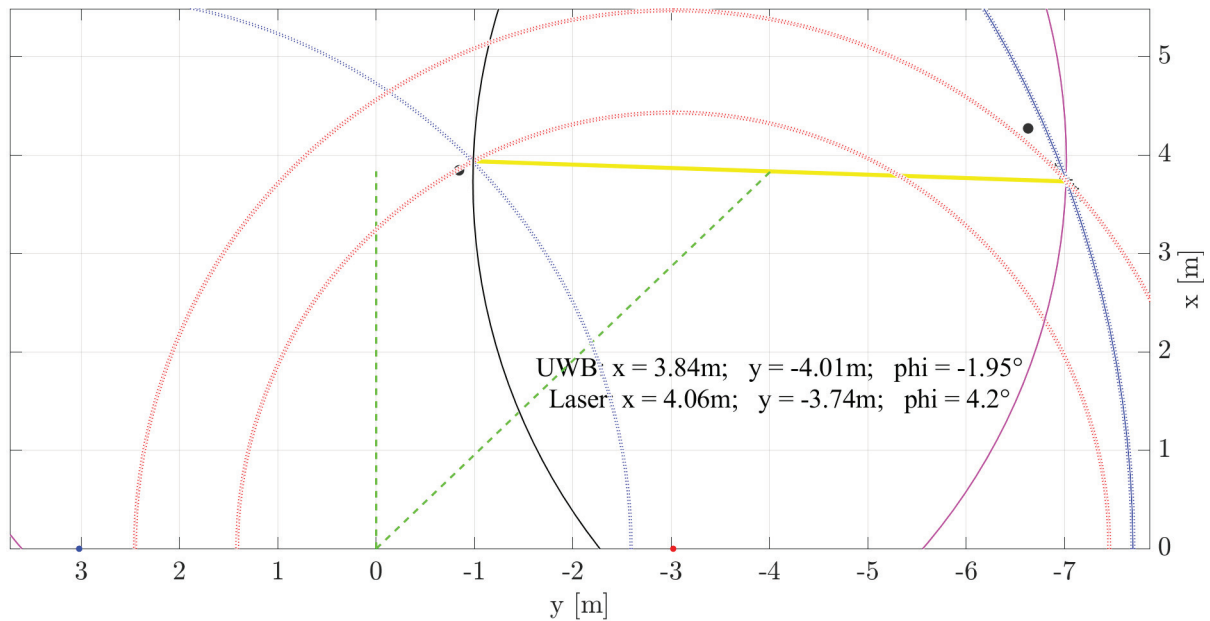


Figure 6.14: Graphic representation of the determination of the relative position and orientation for constellation no. 5: Shown are the UWB range measurements as blue and red triple circles, the Tag positions calculated by the middleware as black and pink point clouds and the reference measurements as gray spots. The black and pink single circle define the real distance between the Tags once spanned from the center of the black point cloud and once from the pink to show the relation between to the calculated Tag distance. Same centers of the point clouds are connected with a yellow line which serves as perpendicular basis to construct the green broken line indicating the relative orientation which is computed in the middleware's algorithm as angle.

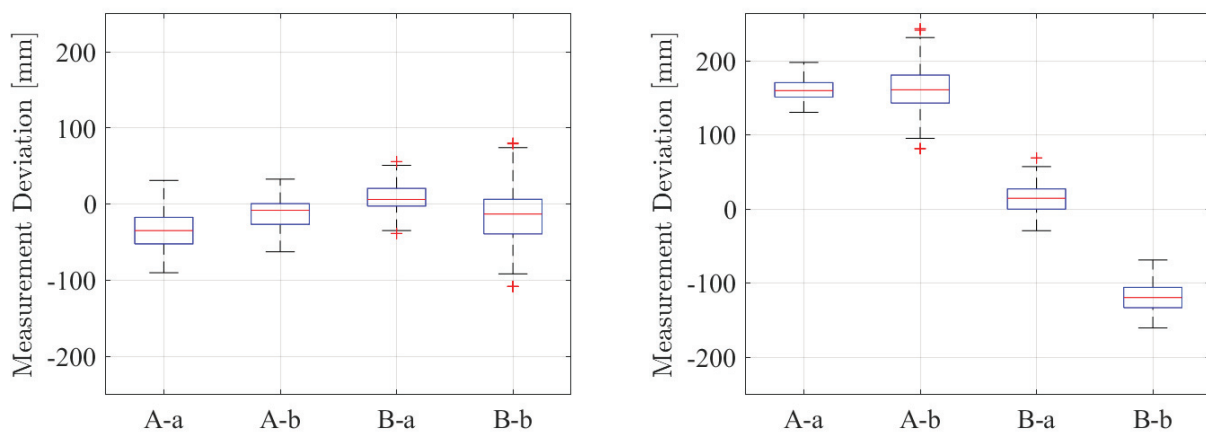


Figure 6.15: Boxplots of all four UWB range measurements relative to the laser reference measurements, for constellation no. 1 and 5 respectively: Per constellation the median, the 25%- and 75%-quantile as well as the whiskers to the smallest and biggest values not considered outliers are presented, separately for each Anchor-Tag combination. Left - constellation no. 1: All four measurements yield a mean value close to the reference. Only one mean value is more than 25mm off. Right - constellation no. 5: Three of the four lengths exceed the reference, two of them by more than 200mm.

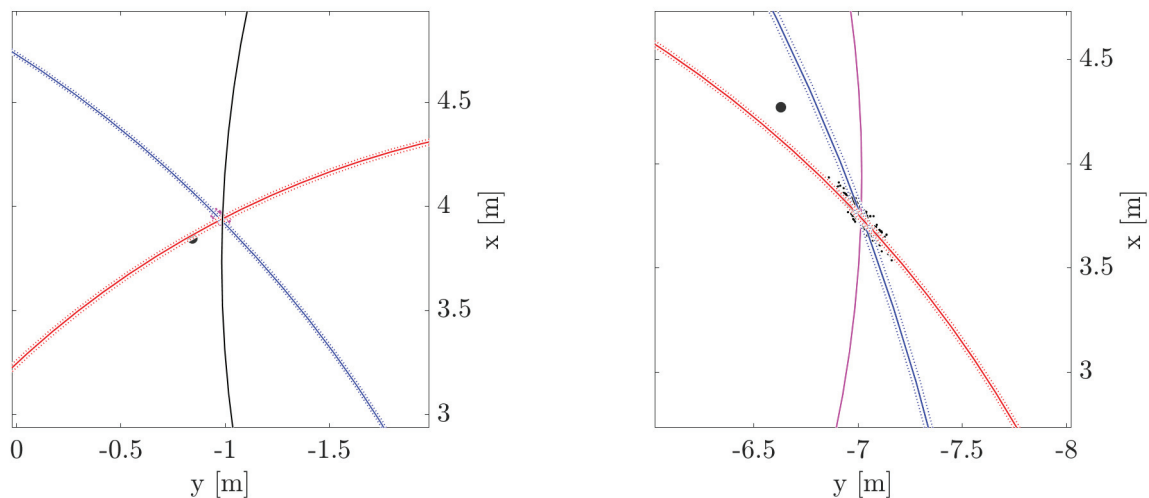


Figure 6.16: UWB range measurements from the middleware compared to the laser reference measurements for constellation no. 5: The red and blue circles represent the median bordered by the dotted 25%- and 75% quantiles of the UWB range measurements. While the inner circles correspond to the measurements "A-a" or "B-b", the outer ones correspond to the diagonal ranges "A-b" and "B-a". Measurements initiated by Anchor "A" are in blue, by Anchor "B" in red. The intersection between the blue inner circle ("A-a") and the red outer circle ("B-a") represents the middleware's calculation for the position of Tag "a" which analogously applies for the position of Tag "b".

6.3 OPC-UA Connection

The last major task of the middleware is the communication with the superior plant network over OPC-UA. Therefore, the final inspection tends to the OPC-UA connection, including the reliability of data exchange with the network and the automatic tree generation of OPC-UA variable nodes. For that purpose, a tool was installed, that simulates an OPC-UA Client if running on a PC, that is connected to the same network as the IPC.

After setting the security configurations for save data transfer, the OPC-UA Client establishes a connection with OPC-UA Server, i.e. the IPC, by addressing it with its URL ID "10.XX.XXX.XX:4840", see figure 6.17, which was specified within the OT of the main application. In the lower right corner of figure 6.17, the IPC appears as Server object node in the variables tree. On pressing the "+" sign next to it, the Server expands to the variables tree, see figure A.1, which was generated in the OT. The tree of OPC-UA variable nodes shows the same topology as the parameter tree in "params". In that manner, any Yaml document can serve as model for the automatic generation of an OPC-UA variable tree.

When clicking on a variable node, all its information is on display, see figure A.2. In that manner any variable can be checked for functional transmission.

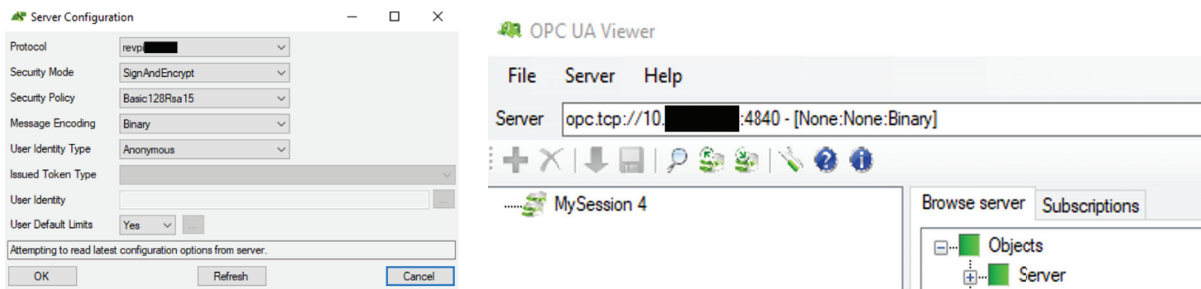


Figure 6.17: OPC-UA Viewer: Within the OPC-UA Client tool, the OPC-UA Server must first have an URL assigned to it and then be configured w.r.t. security parameters, before appearing as Server object node in the OPC-UA network. Left - security configurations. Right – URL ID and Server node.

Upon pressing the “Refresh” button on the lower left corner of the variable window, the value of the variable is updated. In a similar manner, will the OPC-UA Client of the plant network ingest the position outputs from the developed middleware of the positioning system. However, the OPC-UA connection has never been tested when communicating with the intended OPC-UA Client.

6.4 Summary

In five of the ten possible constellations, the field tests show that the positioning system meets the required accuracy, i.e. a maximum positioning error of 200mm. In the other half of the constellations, that do not manage the required limits, the UWB range measurements are unreliable as shown by the representative boxplots of constellation no. 5, see figure 6.15. It can be observed that those constellations causing the most acute angles of the longer diagonal which are no. 4, 5 and 8, see figure 6.8, also show the highest errors in all three parameters “dX”, “dY” and “Phi”, see table 6.1. Constellations no. 3 and 7, the other two that cause insufficient accuracy, have the high end of the conveyor belts in close proximity of the Tags, hence, NLOS or multipath phenomena are likely the cause of the measurement errors. However, more investigations are necessary before being able to determine definite geometric limits for the current system. Thanks to a redundancy check of the Tags’ temperature measurements the reliability of the Modbus RTU connection is verified. Only at rare occasions do the received values freeze which is likely caused by the high data rate. Also the OPC-UA connection’s functionality on the middleware’s end is successful when tested by an OPC-UA Client simulator.

Chapter 7

Conclusion

The results of this thesis show that the concept and the measurement of the relative position and orientation without the necessity for infrastructure are accomplished. Nevertheless, there are still issues that need further attention concerning the accuracy of the UWB range measurements. Due to the difficulties of multipath propagation for acute angles, further tests in real applications are suggested. Two factors need further investigation: the surroundings causing reflections and obstruction and the angular influence of the incoming UWB signals. Hence, the constellations no. 3 and 7 presented in section 6.2.2, which cause critical multipath influence, should be compared to no. 6, which unlike no. 3 and 7 permits the required accuracy even though all three layouts are comparable in their geometry, see figure 6.8. Also constellations no. 4, 5 and 8 should be studied to quantify the influence of the acuteness of the incoming angle on the measurement deviation. In the future, the impact of the incoming angle and falsification of the outputs by reflecting surfaces should be attended to, either by adjusting the signal calculation or the hardware of the sensors.

While representing a simple and common industrial automation solution, the choice of Modbus RTU accomplishes the reliable communication when transmitting less than 13kbps. However, at data rates above 13kbps, occasional short freezing of the data occurs, which should be addressed in the future, e.g. by reducing the update frequency or no. of transmitted registers. Meanwhile, the implemented OPC-UA connection shows stable performance, when inspected through the OPC-UA Client simulator. The developed main application fulfills every requested task: the data transmission in both directions is stable and the algorithm yields precise results. That allows the middleware to deliver accurate position data with high precision and promptness, also in case of widely scattered primary data as long as the mean UWB range measurements are moderately accurate.

The most significant contribution of this work is the development of a positioning system which measures relative position and orientation and doesn't require stationary infrastructure. The system is capable of achieving the given accuracy limit of 200mm in five of ten constellations. In the other five constellations, which are characterized by signal incoming angles above 45° , the highest error is 270mm. Hence, the results are very accurate considering the harsh environment, due to the measurement architecture in which the overdetermined system provides redundancy. Same redundancy, in combination with a Least Squares approximation, proof more error tolerant and robust than single measurements.

Appendices

Appendix A

OPC-UA Variables



Figure A.1: The whole OPC-UA variable tree is composed of object nodes. The first one is the Server object node, while the leaf nodes are the variable nodes.

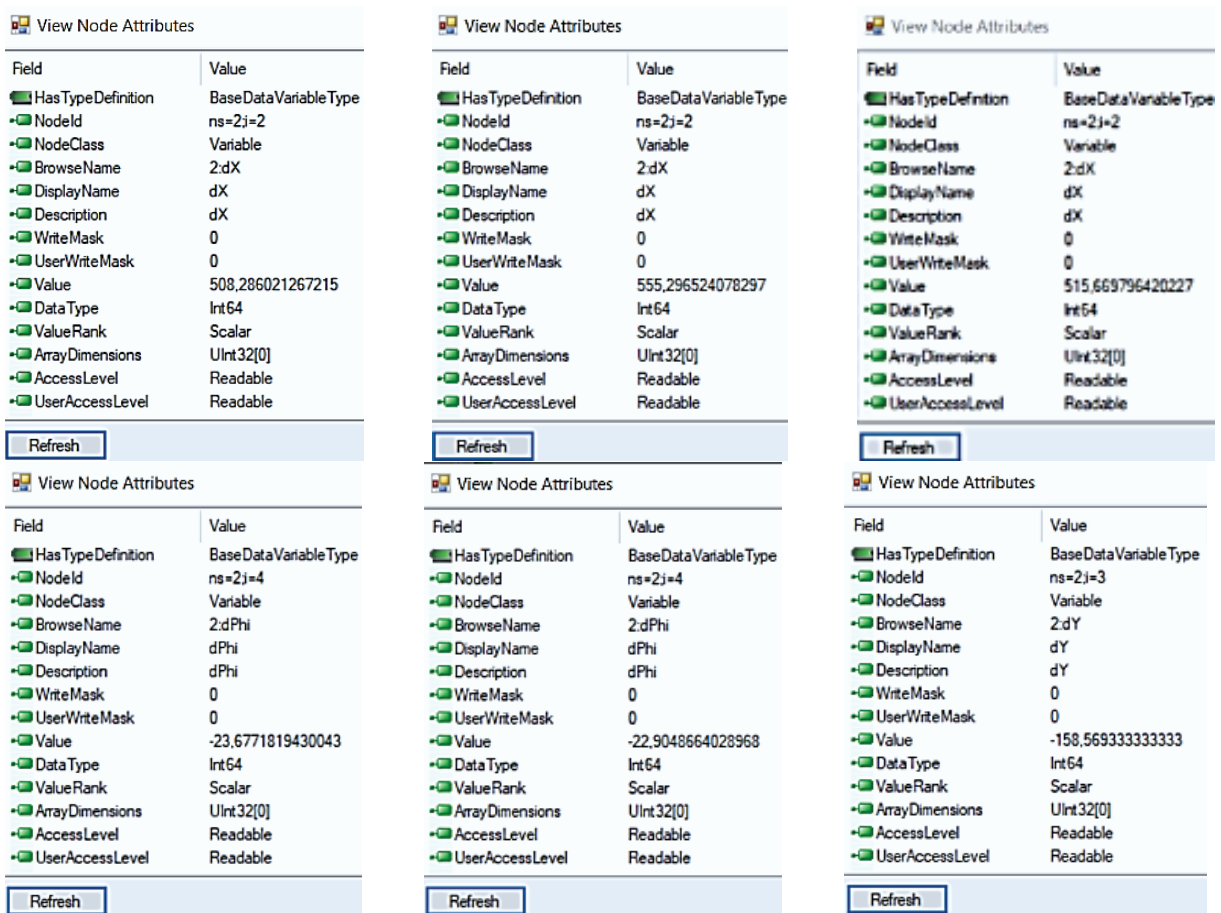


Figure A.2: Value updates of OPC-UA variables. By pressing the "Refresh"-button in the lower left corner, the OPC-UA Client simulator sends a read request to the OPC-UA Server, i.e. the IPC, which returns the current variable value of the middleware.

Appendix B

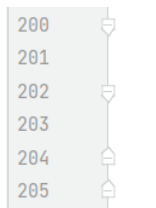
Application Implementation

```
43 class UwbManager:
44     def __init__(self):
45
46         self.logger = lib.json_text_file.WriteToJson()
47
48         self.qMod = queue.Queue()
49         self.config = lib.config_file.PoseConfig()
50
51     def start(self):
52         opc = lib.opcua_server.OpcuaWorker(self.qMod, self.config, self.logger)
53         opc.start()
54         alg = lib.algorithm_new.RpoAlgorithm(self.qMod, self.config, self.logger)
55         alg.start()
56         mb = lib.modbus.MyDriver(self.qMod, self.config, self.logger)
57         mb.start()
58
59
60 if __name__ == "__main__":
61     manager = UwbManager()
62     manager.start()
```

Figure B.1: Manager: The “manager” object prepares the global variables and starts the processes of the three threads.

```
12 class MyDriver(threading.Thread):
13     def __init__(self, qMod, configFile, logger):
14         super().__init__()
15         self.modbusMaster = revpimodio2.RevPiModIO(autorefresh=True)
16         self.modbusMaster.handleSignalend()
```

Figure B.2: MT: The “revpimodio2” library is publicly available and reads the necessary information from the ICF on the IPC to automatically implement the Modbus RTU Master.



```
for SlaveNo in range(4):
    letter = chr(65 + SlaveNo)
    version = list([self.modbusMaster.io["Major" + letter].value,
                  self.modbusMaster.io["Minor" + letter].value,
                  self.modbusMaster.io["Patch" + letter].value])
    self.configFile.set_values("Version" + letter, version)
```

Figure B.3: MT: The middleware intern variable names must be adjusted to match the name format of the “revpimodio2” library to permit transmission over Modbus RTU.

```

29 def do_algorithm(self):
30     while True:
31         time.sleep(0.5)
32         if self.qMod.qsize() > 0:
33             self.width1 = self.configFile.config_dict['Width'][0]
34             self.width2 = self.configFile.config_dict['Width'][0]
35             raw = self.configFile.config_dict['RunAvgWindRaw'][0]
36             calc = self.configFile.config_dict['RunAvgWindCalc'][0]
37             self.RunAvgWindCalc = int(raw)
38             self.RunAvgWindRaw = int(calc)
39             while self.qMod.qsize() > self.RunAvgWindCalc:
40                 self.qMod.get()
41             self.runningAverage(self.qMod, self.RunAvgWindRaw)
42
43         q_length = self.qRangesMean.qsize()
44         if q_length == self.RunAvgWindRaw:
45             modulQsize = np.array(self.qRangesMean.queue).reshape(12, q_length, 4)
46             result = []
47             for modul_i in range(12):
48                 measData = modulQsize[modul_i]
49                 (xL, yL, xR, yR) = self.trilaterateRPO(self.width1, measData)
50
51                 xLR = np.array([xL, xR]).reshape(2*self.RunAvgWindRaw, 1)
52                 xPC = np.mean(xLR)
53                 yLR = np.array([yL, yR]).reshape(2*self.RunAvgWindRaw, 1)
54                 yPC = np.mean(yLR)
55                 D = np.array([xLR, yLR]).reshape(2, 2*self.RunAvgWindRaw)
56                 D = np.transpose(D)
57
58                 PC = np.array([xPC, yPC])
59                 DmFree = np.array(D) - PC
60                 [u, s, vh] = np.linalg.svd(DmFree, False)
61                 input = np.abs(vh[1, 1])
62                 phi = asind(input)
63                 meanXL = np.mean(xL)
64                 meanXR = np.mean(xR)
65                 meanYL = np.mean(yL)
66                 meanYR = np.mean(yR)
67
68                 if meanXL > meanXR:
69                     phi = - phi
70                 dX = (meanXL + meanXR)/2
71                 dY = (meanYL + meanYR)/2

```

Figure B.4: AT: The algorithm includes running average operations, trilateration between the Tags' distances from each Anchor and a SVD in order to yield the relative position and orientation between two units.


```
9 class OpcuaWorker(threading.Thread):
10     SERVER_NAME = 'UWB OPC UA server'
11     SERVER_URI = 'urn:esenseial:uwb:server'
12
13     def __init__(self, qMod, configFile, logger):
14         super().__init__()
15         self.server = Server()
16         self.server.set_server_name(self.SERVER_NAME)
17         self.server.set_application_uri(self.SERVER_URI)
18         url = "opc.tcp://10.XX.XXX.XX:4840"
19         self.server.set_endpoint(url)
20         self.logger = logger
21
22         self.qMod = qMod
23         self.configFile = configFile
24
25         self.opc_dict_readOnly = {}
26         self.opc_dict_writable = {}
27         self.goOn = 1
28
29         ns = self.server.register_namespace("Reg-Namespace")
30         self.var = self.server.get_objects_node()
```

Figure B.5: OT: The OPC-UA object receives the server name, server URI and the URL to be addressable by the OPC-UA Client and add its own OPC-UA objects to the namespace of the network.

List of Figures

1.1	The measurement system’s architecture requires an Anchor pair on one of the units and a Tag pair on the other. By ranging from each Anchor to each Tag, four measurements are obtained that yield the position and orientation after being processed on the IPC.	11
1.2	System design and test setup including conveyor belts: The positioning middleware of the autonomous positioning system links the UWB sensors mounted on the units to the plant network and performs the positioning algorithm.	11
2.1	The main task of the middleware is the continuous output of the relative position between two units. In the case of two linear aligned units the sensors “A” and “B” of the first unit together with “a” and “b” from the second unit form a measurement constellation of four involved sensors.	14
2.2	One unit equipped with eight sensors in total, all delivering range data to the middleware which is executed by one IPC per unit.	15
2.3	The middleware represents the core of the location system and includes the interfaces to the network and UWB sensors, as well as the computations.	15
2.4	Possible alignments within the process line: Horizontal length of feed belts L is 4.1m; Mounting position offsets b and c are about 4.5m and 5.3m respectively. The green arrows on the feed belts indicate the direction of material flow.	17
2.5	Constellation between two units no. 1 and 5. Names and positions of the UWB sensors on two succeeding units. On each corner two sensors are mounted perpendicular on the top of the edge.	18
2.6	Per positioned target unit, four lengths are used to calculate the offset and angle. The running average is computed per each length separately.	19

3.1	UWB frequency spectrum compared to other wireless technologies [14, pp. 403-404]: UWB sends a wide frequency band in one short pulse, but at such little power level, that it appears like noise next to other signals.	24
4.1	Physical connection from IPC as Modbus RTU Master and UWB modules as Modbus RTU Slaves. The power supply can be installed as central supply (top) or separately (bottom).	35
4.2	The register table is a documentation of all the registers transmitted between IPC and every single Slave. Next to the address of each registers in the MMM, the table contains the register's name and category, a short description, its unit, default value and access right as well as a boolean value indicating EEPROM storage. The Modbus Master, i.e. the IPC, stores the quantity of registers listed in the table times the number of Modbus Slaves.	43
5.1	Development test setup: During the software development, two tripods imitated the fronts of two connected plant units. An Anchor pair "A-B" and a Tag pair "a-b" respectively are mounted on the two tripods to test the application. Through a Modbus RTU connection the measured data was ingested into the Middleware and processed in the application.	45
5.2	Class of the configuration object containing all attributes and methods in the below sections of the box.	46
5.3	Class of the Modbus Thread containing all attributes and methods: the queues and arrays of the ranges to ingest and process them, "RangeAi" measured by Anchor "A" and "RangeBi" measured by Anchor "B". Same applies to the temperature measurements and all Anchor specific values. Also the "ModbusMaster" and the array of the Tag IDs which are transmitted to the Anchors are among the attributes. The inheritance to the Thread Class is displayed as arrow from child to parent.	48
5.4	Class of the manager containing all attributes and methods. While the method "start(self)" is called only once to start the three threads, that contain the repeating loops, "init (self)" initializes the global variables "qMod", which ingests the range data from the UWB sensors, "logger" which continuously reports the applications state, and "config", which stores all variables shared between the threads.	50

-
- 5.5 Flowchart - "config": Whole class. Once the "config" object is initialized, it is waiting for an extern read or write request. If such request occurs, the processor is locked to it until the task is finished. 51
- 5.6 The parameter file is written in the Yaml format to be interpretable by machines and humans likewise. It contains a tree of all variables, that need to be exchanged between two threads or between the middleware and its peripherals. Only one of the four "Slave" branches and one of the twelve "Modul" branches are expanded. The rest are structured analogously. 52
- 5.7 Flowchart - "config": Create dictionaries of all inputs and outputs by iterating through the Yaml document containing the parameter tree. 59
- 5.8 Flowchart - MT: Whole thread. After setting up the Modbus Master and updating the values on the middleware's end, two dictionaries – one for the transmitted data and one for the received data – are initialized. As long as Tag pairs are responding, the MT updates all read and written values while locking the processor to its own thread. 60
- 5.9 Each unit is equipped with four Anchors ("A" to "D") and four Tags ("a" to "d"). Hence, the Middleware of "Unit 1" is in charge of positioning "Unit 2", as it is connected to the Anchors of the ranging constellation, and not the other way round. 61
- 5.10 Flowchart - MT: Create dictionaries only of the Modbus inputs and outputs by iterating through the parameter tree. The two separate dictionaries facilitate the efficient execution of read and write tasks. 62
- 5.11 Flowchart - MT: Set and get new values in the "updateRegisters()" method. The MT iterates separately through the read- and write dictionary until the entire dictionaries have been checked for updates. 63
- 5.12 Class of the AT containing all attributes and methods of the thread child. Additionally to the thread specific attributes and methods, the AT contains all methods that process the data until the relative position and orientation are computed in each iteration every one to two seconds. Furthermore, the raw measurements are among its attributes, "qMod", together with other parameters from the network, like the running average windows or the mounting width between two paired sensors, "runAvgWindRaw", "runAvgWindCalc", "width1" and "width2". . . 64

-
- 5.13 Flowchart - AT: Whole thread. After reading the network specified geometry inputs and running average windows, the AT checks if there are enough entries to perform the computations. If so, the number of data sets is reduced to the first running average window to smoothen the ranges separately. If the amount of smooth sets is enough, the relative offset and orientation themselves are computed and published. 65
- 5.14 Flowchart - AT: Run average per length. Within the AT another queue, “qRange-Mean” is initialized to store the results of the first data smoothing step and feed the actual position and orientation calculating equation. 66
- 5.15 Flowchart - AT: Compute relative offset and orientation. Through trilateration, which is the calculation of intercepting points of two cycles, the evaluation of mean distances and an SVD, the best fitting approximation of the relative position and orientation are extracted in every iteration. 67
- 5.16 Class of the OT containing all attributes and methods. Apart from dictionaries that are used for data transfer, the attributes: SERVER NAME and SERVER URI are defined to successfully establish an OPC-UA connection. 68
- 5.17 Flowchart - OT: Whole thread. After setting up the OPC-UA Server, the namespace is specified and used to generate OPC-UA object nodes, that are addressable by the network’s OPC-UA Client. When the dictionaries are initialized, the OT is ready for data updates. 69
- 5.18 Flowchart - OT: Create dictionaries of all inputs and outputs. During the initialization of the OPC-UA read and write dictionaries, OPC-UA object nodes are created that adopt the read and write access rights from the parameter file in addition to the key-value pair. 70
- 5.19 Flowchart - OT: Set and get new values. When updating the OPC-UA values, the OT directly reads from or writes to the object nodes integrated into the OPC-UA network of the plant. 71
- 6.1 UWB range measurements read from the middleware: All four UWB range measurements are displayed, that are obtained by two UWB Anchors ”A” and ”B” measuring the distance to two UWB Tags ”a” and ”b”, see figure 5.1. While the specific Tag IDs are sent through the Middleware and Modbus to the Anchors, i.e. the Modbus Slaves, the Anchors feed the Middleware over Modbus with their measured values, if the communication with the specified Tags was successful. 73

- 6.2 UWB range and temperature measurements from the middleware: Next to the four range measurements "Range A-a", "Range A-b", "Range B-a", "Range B-b" the Temperature measurements are shown. While "Temp A" and "Temp B" are simply the temperature values of the Anchors themselves, "Temp a (by A)" is the temperature measurement from Tag "a" that is communicated over UWB to Anchor "A" and from there transmitted to the IPC over the Modbus interface. In case of a successful transmission, the plots of "Temp a (by A)" and Temp b (by B)" are overlapping as shown by the thin lines on top of the bold lines. Same applies for Tag "b". 74
- 6.3 Temperature measurements: Only the temperature measurements of the UWB sensors are shown for a total of 500 measurements. Each overlapping data pair represent the temperature measurement of one Tag which is communicated over UWB to both Anchors and indicates that the temperatures are being reliably measured. They are stable during the measurement and the resolution of the temperatures is visible in the discrete step nature of the data. 75
- 6.4 UWB range measurements with frozen values read from the middleware: All four UWB range measurements are displayed, that are obtained by two UWB Anchors "A" and "B" measuring the distance to two UWB Tags "a" and "b", see figure 5.1. Occasionally the otherwise jumping range values, that are ingested over Modbus RTU remain constant throughout a few seconds. 75
- 6.5 Relative position and orientation output: After the UWB range measurements are smoothed and processed by the algorithm, the relative position in x- and y-direction "dX" and "dY", as well as the relative orientation "Phi" between two succeeding units are obtained. 76
- 6.6 UWB range measurements compared to the calculated relative position: "dX" is by far steadier compared to the unsmoothed raw inputs and smaller than the average range measurement due to the diagonal UWB communication where at least one must be greater than the relative position. 77
- 6.7 First field test constellation: Two plant units were arranged in a predefined relative position in an industrial environment, not yet connected through a conveyor belt. 78

-
- 6.8 The ten different possible arrangements of two plant units: Considering the actual succeeding sorting processes, as well as the given variety and designs of the sorting machines, a total of ten relative location arrangements are test cases for the developed system. 78
- 6.9 Second constellation of field tests: For the final system verification the developed system is installed on plant units in an assembly hall, now connected through conveyor belts. The sensors are mounted at the top of each edge - marked by red rectangles - and connected to the power supply. Only the Anchors are additionally connected through Modbus RTU to the IPC as well. 79
- 6.10 Graphic representation of the determination of the relative position and orientation for constellation no. 1: Shown are the UWB range measurements as blue and red triple circles, the Tag positions calculated by the middleware as black and pink point clouds and the by the reference measurements as gray spots (almost obscured for constellation no. 1). The black and pink single circles define the real distance between the Tags once spanned from the center of the black point cloud and once from the pink to show the relation to the calculated Tag distance. Same centers of the point clouds are connected with a yellow line which serves as perpendicular basis to construct the green broken line indicating the relative orientation which is computed in the middleware's algorithm as angle. 80
- 6.11 Constellation no. 1 of two neighboring plant units. 81
- 6.12 UWB range measurements from the middleware compared to the laser reference measurements for constellation no. 1: The red and blue circles represent the median bordered by the dotted 25%- and 75% quantiles of the UWB range measurements. While the inner circles correspond to the measurements "A-a" or "B-b", the outer ones correspond to the diagonal ranges "A-b" and "B-a". Measurements initiated by Anchor "A" are in blue, by Anchor "B" in red. The intersection between the blue inner circle ("A-a") and the red outer circle ("B-a") represents the middleware's calculation for the position of Tag "a" which analogously applies for the position of Tag "b". 81
- 6.13 Constellation no. 5 of two neighboring plant units. 82

- 6.14 Graphic representation of the determination of the relative position and orientation for constellation no. 5: Shown are the UWB range measurements as blue and red triple circles, the Tag positions calculated by the middleware as black and pink point clouds and the reference measurements as gray spots. The black and pink single circle define the real distance between the Tags once spanned from the center of the black point cloud and once from the pink to show the relation between to the calculated Tag distance. Same centers of the point clouds are connected with a yellow line which serves as perpendicular basis to construct the green broken line indicating the relative orientation which is computed in the middleware's algorithm as angle. 83
- 6.15 Boxplots of all four UWB range measurements relative to the laser reference measurements, for constellation no. 1 and 5 respectively: Per constellation the median, the 25%- and 75%-quantile as well as the whiskers to the smallest and biggest values not considered outliers are presented, separately for each Anchor-Tag combination. Left - constellation no. 1: All four measurements yield a mean value close to the reference. Only one mean value is more than 25mm off. Right - constellation no. 5: Three of the four lengths exceed the reference, two of them by more than 200mm. 83
- 6.16 UWB range measurements from the middleware compared to the laser reference measurements for constellation no. 5: The red and blue circles represent the median bordered by the dotted 25%- and 75% quantiles of the UWB range measurements. While the inner circles correspond to the measurements "A-a" or "B-b", the outer ones correspond to the diagonal ranges "A-b" and "B-a". Measurements initiated by Anchor "A" are in blue, by Anchor "B" in red. The intersection between the blue inner circle ("A-a") and the red outer circle ("B-a") represents the middleware's calculation for the position of Tag "a" which analogously applies for the position of Tag "b". 84
- 6.17 OPC-UA Viewer: Within the OPC-UA Client tool, the OPC-UA Server must first have an URL assigned to it and then be configured w.r.t. security parameters, before appearing as Server object node in the OPC-UA network. Left - security configurations. Right – URL ID and Server node. 85
- A.1 The whole OPC-UA variable tree is composed of object nodes. The first one is the Server object node, while the leaf nodes are the variable nodes. 92

A.2	Value updates of OPC-UA variables. By pressing the "Refresh"-button in the lower left corner, the OPC-UA Client simulator sends a read request to the OPC-UA Server, i.e. the IPC, which returns the current variable value of the middleware.	93
B.1	Manager: The "manager" object prepares the global variables and starts the processes of the three threads.	94
B.2	MT: The "revpimodio2" library is publicly available and reads the necessary information from the ICF on the IPC to automatically implement the Modbus RTU Master.	94
B.3	MT: The middleware intern variable names must be adjusted to match the name format of the "revpimodio2" library to permit transmission over Modbus RTU.	95
B.4	AT: The algorithm includes running average operations, trilateration between the Tags' distances from each Anchor and a SVD in order to yield the relative position and orientation between two units.	96
B.5	OT: The OPC-UA object receives the server name, server URI and the URL to be addressable by the OPC-UA Client and add its own OPC-UA objects to the namespace of the network.	97

List of Tables

- 4.1 Systematic evaluation of best choice for fieldbus protocol. 37

- 6.1 Errors of algorithm outputs "dX", "dY" and "Phi" compared to reference measurements for all ten possible constellations between two connected plant units. 82

Bibliography

- [1] Fabian de Ponte Müller. “Survey on Ranging Sensors and Cooperative Techniques for Relative Positioning of Vehicles: Journal Article”. In: *Sensors (Basel, Switzerland)* 17.2 (2017). DOI: 10.3390/s17020271.
- [2] Ali Asghar Nazari Shirehjini, Abdulsalam Yassine, and Shervin Shirmohammadi. “An RFID-Based Position and Orientation Measurement System for Mobile Objects in Intelligent Environments”. In: *IEEE Transactions on Instrumentation and Measurement* 61.6 (2012), pp. 1664–1675. ISSN: 0018-9456. DOI: 10.1109/TIM.2011.2181912.
- [3] Pascal Pagani et al. *Ultra-wideband radio propagation channels. A practical approach*. London and Hoboken, NJ: ISTE and Wiley, 2008. ISBN: 9781848210844.
- [4] Gene H. Golub and Charles F. van Loan. *Matrix computations*. 4. ed. Johns Hopkins studies in mathematical sciences. Baltimore, Md.: Johns Hopkins Univ. Press, 2013. ISBN: 9781421407944.
- [5] Paul Gerrard and Radia M. Johnson. *Mastering scientific computing with R*. Birmingham, England and Sebastapol, California: Packt Publishing and Safari Books Online, 2015. ISBN: 9781783555260.
- [6] Kunbus. *RevPi Connect+: Datasheet*. Ed. by KUNBUS GmbH. 2021. URL: <https://revolution.kunbus.com/revpi-connect/> (visited on 05/16/2021).
- [7] Robert Henßen and Miriam Schleipen. “Interoperability between OPC UA and AutomationML”. In: *Procedia CIRP* 25 (2014), pp. 297–304. ISSN: 22128271. DOI: 10.1016/j.procir.2014.10.042.
- [8] Unified Automation. *OPC UA NodeId Concepts*. 2021. URL: https://documentation.unified-automation.com/uasdkhp/1.4.1/html/_12_ua_node_ids.html (visited on 05/16/2021).
- [9] IC. *DW1000 Anchor: Datasheet*. Ed. by In-Circuit GmbH. 2021. URL: http://wiki.in-circuit.de/images/6/6a/305000107A_DW1000_Anchor.pdf (visited on 05/16/2021).

- [10] Stefan Heinen et al. “HaLoS – Integrated RF-Hardware Components for Ultra-Wideband Localization and Sensing”. In: *Ultra-Wideband Radio Technologies for Communications, Localization and Sensor Applications*. Ed. by Reiner Thom. InTech, 2013, pp. 369–438. ISBN: 978-953-51-0936-5. DOI: 10.5772/54987.
- [11] A. R. Jiménez and F. Seco. “Comparing Decawave and Bespoon UWB location systems: indoor/outdoor performance analysis: 4-7 October 2016, Alcalá de Henares, Madrid, Spain”. In: *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN) (2016)*. DOI: 10.1109/IPIN.2016.7743686.
- [12] Alberto Giretti, Alessandro Carbonari, and Massimo Vaccarini. “Ultra Wide Band Positioning Systems for Advanced Construction Site Management”. In: *New Approach of Indoor and Outdoor Localization Systems*. Ed. by Fouzia Elbahhar. InTech, 2012, pp. 89–112. ISBN: 978-953-51-0775-0. DOI: 10.5772/48260.
- [13] Homayoun Nikookar and Ramjee Prasad. *Introduction to Ultra Wideband for Wireless Communications*. Signals and Communication Technology. Dordrecht: Springer Netherlands, 2009. ISBN: 9781402066337. DOI: 10.1007/978-1-4020-6633-7.
- [14] Mohammed Al-Husseini et al. “Cognitive Radio: UWB Integration and Related Antenna Design”. In: *New Trends in Technologies: Control, Management, Computational Intelligence and Network Systems*. Ed. by Meng Joo. Sciyo, 2010. ISBN: 978-953-307-213-5. DOI: 10.5772/10405.
- [15] F. Elbahhar et al. “Indoor Positioning System Based on the Ultra Wide Band for Transport Applications”. In: *New Approach of Indoor and Outdoor Localization Systems*. Ed. by Fouzia Elbahhar. InTech, 2012, pp. 69–88. ISBN: 978-953-51-0775-0. DOI: 10.5772/50017.
- [16] Syed Naveen Altaf Ahmed and Yonghong Zeng. *UWB Positioning Accuracy and Enhancements: 5-8 Nov. 2017*. Piscataway, NJ: IEEE, 2017. ISBN: 9781509011346.
- [17] Institute of Electrical and Electronics Engineers. *About IEEE*. 2021. URL: <https://www.ieee.org/about/index.html> (visited on 05/16/2021).
- [18] Kun Zhang et al. “Research on Similarity Metric Distance Algorithm for Indoor and Outdoor Firefighting Personnel Precision Wireless Location System Based on Vague Set on UWB: October 27-30, 2017, Chengdu, China”. In: *2017 17th IEEE International Conference on Communication Technology (ICCT 2017) (2017)*. DOI: 10.1109/ICCT.2017.8359817.

- [19] European Telecommunications Standards Institute. *Short Range Devices (SRD) using Ultra Wide Band (UWB): Part 3: Worldwide UWB regulations between 3,1 and 10,6 GHz*. 2019. URL: https://www.etsi.org/deliver/etsi_TR/103100_103199/10318103/02.01.01_60/tr_10318103v020101p.pdf (visited on 05/16/2021).
- [20] Maria Dolores Perez Guirao. “Pulse Rate Control for Low Power and Low Data Rate Ultra Wideband Networks”. In: *Ultra-Wideband Radio Technologies for Communications, Localization and Sensor Applications*. Ed. by Reiner Thom. InTech, 2013, pp. 51–74. ISBN: 978-953-51-0936-5. DOI: 10.5772/52497.
- [21] Rudolf Zetik et al. “Cooperative Localization and Object Recognition in Autonomous UWB Sensor Networks”. In: *Ultra-Wideband Radio Technologies for Communications, Localization and Sensor Applications*. Ed. by Reiner Thom. InTech, 2013, pp. 179–240. ISBN: 978-953-51-0936-5. DOI: 10.5772/55077.
- [22] Lorenz M. Hilty et al. *Lokalisiert und identifiziert: Wie Ortungstechnologien unser Leben verändern*. Vol. 57. TA-SWISS. Zürich: vdf, 2012. ISBN: 978-3-7281-3477-6. DOI: 10.3218/3477-6.
- [23] Michael Eisenacher. “Optimierung von Ultra-Wideband-Signalen (UWB)”. dissertation. Karlsruhe: Techn. Univ. Karlsruhe, 2006. URL: <https://d-nb.info/1003466907/34>.
- [24] Federal Communications Commission. *A Technology Comparison: Adopting Ultra-Wideband for Memsen’s file sharing and wireless marketing platform*. 2004. URL: <https://www.fcc.gov/2004-wireless-broadband-forum-comments-received> (visited on 05/16/2021).
- [25] University of Cambridge. *Diffraction*. 2021. URL: https://isaacphysics.org/concepts/cp_diffraction (visited on 05/16/2021).
- [26] Rainer Moorfeld et al. “MIRA – Physical Layer Optimisation for the Multiband Impulse Radio UWB Architecture”. In: *Ultra-Wideband Radio Technologies for Communications, Localization and Sensor Applications*. Ed. by Reiner Thom. InTech, 2013, pp. 1–50. ISBN: 978-953-51-0936-5. DOI: 10.5772/55076.
- [27] Deval V. Jansari and Reza K. Amineh. “A two-element antenna array for compact portable MIMO-UWB communication systems”. In: *AIMS Electronics and Electrical Engineering* 3.2 (2019), pp. 224–232. ISSN: 2578-1588. DOI: 10.3934/ElectrEng.2019.3.224.

- [28] Mohamed El-Hadidy et al. “Interference Alignment for UWB-MIMO Communication Systems”. In: *Ultra-Wideband Radio Technologies for Communications, Localization and Sensor Applications*. Ed. by Reiner Thom. InTech, 2013, pp. 133–152. ISBN: 978-953-51-0936-5. DOI: 10.5772/55083.
- [29] Marcin Kolakowski and Vitomir Djaja-Josko. “TDOA-TWR based positioning algorithm for UWB localization system”. In: *2016 21st International Conference on Microwave, Radar and Wireless Communications (MIKON)*. Ed. by Artur Rydosz. Piscataway, NJ: IEEE, 2016. ISBN: 9781509022144.
- [30] Jeppe Bro Kristensen et al. “Non-Line-of-Sight Identification for UWB Indoor Positioning Systems using Support Vector Machines”. In: *2019 IEEE MTT-S International Wireless Symposium (IWS)*. IEEE, 19.05.2019 - 22.05.2019, pp. 1–3. ISBN: 978-1-7281-0716-5. DOI: 10.1109/IEEE-IWS.2019.8804072.
- [31] Yan Xie, Gerard J. M. Janssen, and Alle-Jan van der Veen. “A practical clock synchronization algorithm for UWB positioning systems”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 20.03.2016 - 25.03.2016, pp. 3891–3895. ISBN: 978-1-4799-9988-0. DOI: 10.1109/ICASSP.2016.7472406.
- [32] Clarinox Technologies Pty Ltd. *Real Time Location Systems*. 2009. URL: http://www.clarinox.com/docs/whitepapers/RealTime_main.pdf (visited on 05/16/2021).
- [33] Glenn Fleishman. *Take Control of iOS & iPadOS Privacy and Security*. 1st edition. [Erscheinungsort nicht ermittelbar] and Boston, MA: Take Control Books and Safari, 2020. ISBN: 9781947282643.
- [34] Mark G. Petovello et al. “Demonstration of Inter-Vehicle UWB Ranging to Augment DGPS for Improved Relative Positioning”. In: *Journal of Global Positioning Systems* 11.1 (2012), pp. 11–21. ISSN: 14463156. DOI: 10.5081/jgps.11.1.11.
- [35] Yasser Morgan. “Accurate positioning using Short-Range Communications”. In: *2009 International Conference on Ultra Modern Telecommunications & Workshops*. IEEE, 12.10.2009 - 14.10.2009, pp. 1–7. ISBN: 978-1-4244-3942-3. DOI: 10.1109/ICUMT.2009.5345553.
- [36] Ingrid Hilger et al. “ultraMEDIS – Ultra-Wideband Sensing in Medicine”. In: *Ultra-Wideband Radio Technologies for Communications, Localization and Sensor Applications*. Ed. by Reiner Thom. InTech, 2013, pp. 257–322. ISBN: 978-953-51-0936-5. DOI: 10.5772/55081.

- [37] Henning Mextorf et al. “ISOPerm: Non-Contacting Measurement of Dielectric Properties of Irregular Shaped Objects”. In: *Ultra-Wideband Radio Technologies for Communications, Localization and Sensor Applications*. Ed. by Reiner Thom. InTech, 2013, pp. 323–342. ISBN: 978-953-51-0936-5. DOI: 10.5772/55079.
- [38] Stefan Knauth. “Study and Evaluation of Selected RSSI-Based Positioning Algorithms”. In: *Geographical and Fingerprinting Data to Create Systems for Indoor Positioning and Indoor/Outdoor Navigation*. Elsevier, 2019, pp. 147–167. ISBN: 9780128131893. DOI: 10.1016/B978-0-12-813189-3.00006-X.
- [39] Pornchai Pongpipatpakdee et al. “Integration of wireless HART network system into SCADA software for operation & management”. In: *2016 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. IEEE, 2016, pp. 549–554. ISBN: 978-4-907764-50-0.
- [40] Deon Reynders, Steve Mackay, and E. Wright. *Practical industrial data communications: Best practice techniques*. Practical professional books from Elsevier. Oxford, Amsterdam, and Boston: Newnes and Elsevier, 2005. ISBN: 0750663952.
- [41] Schneider Electric. *Modbus Cable Characteristics*. 2019. URL: https://product-help.schneider-electric.com/ED/ES_Power/ULP_System_IEC_Guide/EDMS/DOCA0093EN/DOCA0093xx/ULP-Chapter_Appendix_ULP_IFM_TRV00210/ULP-Chapter_Appendix_ULP_IFM_TRV00210-6.htm (visited on 05/16/2021).
- [42] John Douglas McDonald. *Electric power substations engineering*. 2nd ed. The Electric Power Engineering Hbk, Second Edition. Boca Raton: Taylor & Francis, 2007. ISBN: 9780849373831.
- [43] Gerhard Schnell and Bernhard Wiedemann. *Bussysteme in der Automatisierungs- und Prozesstechnik*. Wiesbaden: Springer Fachmedien Wiesbaden, 2019. ISBN: 978-3-658-23687-8. DOI: 10.1007/978-3-658-23688-5.
- [44] IEC. *Electropedia: Bus*. Ed. by International Electrotechnical Commission. 2006. URL: <http://www.electropedia.org/iev/iev.nsf/display?openform&ievref=351-56-10> (visited on 05/16/2021).
- [45] Electronicsworld. *Research finds Industrial Ethernet increases its market share yet Fieldbus decline continues*. Ed. by Electronics World Magazine. 2020. URL: <https://www.electronicsworld.co.uk/research-finds-industrial-ethernet-increases-its-market-share-yet-fieldbus-decline-continues/24994/> (visited on 05/16/2021).

- [46] Kunbus. *RevPi Connect+: Flyer: IIoT GATEWAY*. Ed. by KUNBUS GmbH. 2021. URL: <https://revolution.kunbus.com/revpi-connect/> (visited on 05/16/2021).
- [47] Humiras Hardi Purba et al. "Product Development of Chocolate with Quality Function Deployment Approach: A Case Study in SMEs Chocolate Industry in Indonesia". In: *IOP Conference Series: Earth and Environmental Science* 209 (2018), p. 012011. DOI: 10.1088/1755-1315/209/1/012011.
- [48] Carl Henning. *Profibus - Profinet: 4 MORE PROFINET MYTHS AND THE MUCH BETTER REALITIES*. 2014. URL: <https://us.profinet.com/4-profinet-myths-much-better-realities/> (visited on 05/16/2021).
- [49] Modbus. *MODBUS over Serial Line: Specification and Implementation Guide: V1.02*. Ed. by Modbus Organization. 2006. URL: https://modbus.org/docs/Modbus_over_serial_line_V1_02.pdf (visited on 05/16/2021).
- [50] Procentec. *PROFIBUS cable length*. 2021. URL: <https://procentec.com/content/profibus-cable-length/> (visited on 05/16/2021).
- [51] Emotron. *Fieldbus Option*. 2018. URL: https://www.emotron.com/globalassets/downloads/products/softstarters/emotron-tsa/option---fieldbus-2.0-instruction-manual-for-ac-drives-and-tsa/option_fieldbus_instruction_for-ac-drives-and-tsa_01-3698-01r11.en.pdf (visited on 05/16/2021).
- [52] PROFIBUS. *PROFIBUS System Description: Technology and Application*. 2021. URL: <https://www.profibus.com/index.php?eID=dumpFile&t=f&f=52380&token=4868812e468cd5e71d2a07c7b3da955b47a8e10d> (visited on 05/16/2021).
- [53] James Powell. *Profibus and Modbus: a comparison*. 2013. URL: <https://www.automation.com/en-us/articles/2013-2/profibus-and-modbus-a-comparison> (visited on 05/16/2021).
- [54] DECK Monitoring. *RS485 / Modbus RTU Wiring Standards*. 2021. URL: <https://deckmonitoring.zendesk.com/hc/en-us/articles/222853008-RS485-Modbus-RTU-Wiring-Standards> (visited on 05/16/2021).
- [55] Acromag. *INTRODUCTION TO PROFIBUS DP*. 2002. URL: <http://www.diit.unict.it/users/scava/dispense/II/Profibus.pdf> (visited on 05/16/2021).
- [56] Barrie A. Sosinsky. *Networking bible*. Vol. v.567. Bible. Indianapolis, IN: Wiley, 2009. ISBN: 9780470431313.

- [57] Wang Yongliang et al. “Design of Environmental Control System Based on Embedded Modbus”. In: *Proceedings of the 31st Chinese Control and Decision Conference (2019 CCDC)*. Piscataway, NJ: IEEE, 2019, pp. 598–601. ISBN: 978-1-7281-0106-4.
- [58] Byoung-Koo Kim et al. “Detecting Abnormal Behavior in SCADA Networks Using Normal Traffic Pattern Learning”. In: *Computer Science and its Applications*. Ed. by James J. Park et al. Vol. 330. Lecture Notes in Electrical Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 121–126. ISBN: 978-3-662-45401-5. DOI: 10.1007/978-3-662-45402-2.
- [59] Claudio Urrea, Claudio Morales, and John Kern. “Implementation of error detection and correction in the Modbus-RTU serial protocol”. In: *International Journal of Critical Infrastructure Protection* 15 (2016), pp. 27–37. ISSN: 18745482. DOI: 10.1016/j.ijcip.2016.07.001.
- [60] Modbus. *MODBUS APPLICATION PROTOCOL SPECIFICATION: V1.1b3*. Ed. by Modbus Organization. 2012. URL: https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf (visited on 05/16/2021).
- [61] Joe Cieszynski. *Closed circuit television*. Second edition. Oxford England, Burlington, MA: Newnes, 2004. ISBN: 9780080545738.
- [62] B. R. Mehta. *Industrial process automation systems: Design and implementation*. Waltham, MA: Butterworth-Heinemann, 2015. ISBN: 9780128010983.
- [63] Rajkumar Buyya, S. Thamarai Selvi, and Xingchen Chu. *Object-oriented programming with Java: Essentials and applications*. New Delhi and Singapore: Tata McGraw-Hill, 2009. ISBN: 9780070669086.
- [64] J. M. Almendros Jimenez and L. Iribarne. “UML Modeling of User and Database Interaction”. In: *The Computer Journal* 52.3 (2009), pp. 348–367. ISSN: 0010-4620. DOI: 10.1093/comjnl/bxn028.
- [65] Mark Lutz. *Programming Python*. 2nd ed. Beijing and Sebastopol, CA: O’Reilly, 2001. ISBN: 9780596000851.
- [66] Python. *threading — Thread-based parallelism: Python 3.9.2rc1 documentation*. Ed. by Python Software Foundation. 2021. URL: <https://docs.python.org/3/library/threading.html#thread-objects> (visited on 05/16/2021).
- [67] James H. Anderson, Yong-Jik Kim, and Ted Herman. “Shared-memory mutual exclusion: major research trends since 1986”. In: *Distributed Computing* 16.2-3 (2003), pp. 75–110. ISSN: 0178-2770. DOI: 10.1007/s00446-003-0088-6.

- [68] Robert Lafore. *Object-oriented programming in C++*. 4th ed. Indianapolis, Ind. and Hemel Hempstead: Sams and Prentice Hall [distributor], 2001. ISBN: 9780132714297.
- [69] Python. *queue — A synchronized queue class: Python 3.9.2rc1 documentation*. Ed. by Python Software Foundation. 2021. URL: <https://docs.python.org/3/library/queue.html> (visited on 05/16/2021).
- [70] Python. *Built-in Types: Python 3.9.2rc1 documentation: Lists*. Ed. by Python Software Foundation. 2021. URL: <https://docs.python.org/3.9/library/stdtypes.html#lists> (visited on 05/16/2021).
- [71] Python. *Data Structures: Python 3.9.2rc1 documentation: Dictionaries*. Ed. by Python Software Foundation. 2021. URL: <https://docs.python.org/3.9/tutorial/datastructures.html#dictionaries> (visited on 05/16/2021).
- [72] Python. *logging — Logging facility for Python: Python 3.9.2rc1 documentation: Dictionaries*. Ed. by Python Software Foundation. 2021. URL: <https://docs.python.org/3.9/library/logging.html#module-logging> (visited on 05/16/2021).
- [73] Yaml. *YAML Ain't Markup Language (YAML™): Version 1.2: 3rd Edition*. 2009. URL: <https://yaml.org/spec/1.2/spec.html#id2759572> (visited on 05/16/2021).
- [74] PR electronics. *4511 MODBUS RTU: Configuration Manual*. 2021. URL: https://www.prelectronics.com/filearkiv/PDF/9100%20series/9106/Config.%20Manual/9106_MCM_101.pdf (visited on 05/16/2021).