

Bachelor Thesis



**Czech
Technical
University
in Prague**

F3

**Faculty of Electrical Engineering
Department of Radioelectronics**

SW for Indoor Visible Light Positioning Testbed

Martin Suda

**Supervisor: Ing. Stanislav Vítek, Ph.D.
May 2020**

I. Personal and study details

Student's name: **Suda Martin**

Personal ID number: **474251**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Radioelectronics**

Study program: **Electronics and Communications**

II. Bachelor's thesis details

Bachelor's thesis title in English:

SW for Indoor Visible Light Positioning Testbed

Bachelor's thesis title in Czech:

SW řešení systému pro navigaci uvnitř budov s využitím komunikace ve viditelném světle

Guidelines:

The aim of this work is to design a platform for testing algorithms for indoor navigation using visible light communication. Follow these guidelines:

- 1) Introduce yourself with the algorithms of navigation using visible light communication
- 2) Testbed consists of nodes communicating with the central unit
- 3) Design a protocol for setting up individual nodes of the testbed
- 4) Design and implement a program able to use testbed in different work modes and configurations
- 5) Perform basic measurements to verify the functionality of the entire system under laboratory conditions

Bibliography / sources:

- [1] CHAUDHARY, Neha; ALVES, Luis Nero; GHASSEMBLOOY, Zabih. Current Trends on Visible Light Positioning Techniques. In: 2019 2nd West Asian Colloquium on Optical Wireless Communications (WACOWC). IEEE, 2019. p. 100-105.
- [2] GONG, Chen. Visible Light Communication and Positioning: Present and Future. 2019.
- [3] XU, Jiaojiao; GONG, Chen; XU, Zhengyuan. Experimental indoor visible light positioning systems with centimeter accuracy based on a commercial smartphone camera. IEEE Photonics Journal, 2018, 10.6: 1-17.

Name and workplace of bachelor's thesis supervisor:

Ing. Stanislav Vitek, Ph.D., Department of Radioelectronics, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **03.02.2020** Deadline for bachelor thesis submission: _____

Assignment valid until: **30.09.2021**

Ing. Stanislav Vitek, Ph.D.
Supervisor's signature

doc. Ing. Josef Dobeš, CSc.
Head of department's signature

prof. Mgr. Petr Páta, Ph.D.
Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student's signature

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Ing. Stanislav Vítek, Ph.D. for his continuous support during the preparation of my Bachelor thesis, for his enthusiasm, motivation, immense knowledge, and his time he devoted to me.

Besides my supervisor, I would like to thank my family, for the patience and for their support throughout my study.

Finally, I would like to thank my colleague Štěpán Bosák for a great team collaboration on this project.

Declaration

I declare that I have written submitted thesis by myself and that I have listed all information sources in accordance with Methodical Guideline on Compliance with Ethical Principles.

In Prague, 22 May 2020

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 22. května 2020

Abstract

The aim of this work is to propose a communication design for a VLC indoor positioning system. In the beginning, the thesis is dedicated to the classification of indoor positioning systems, their mathematical interpretation, and to the algorithms using VLC. The next part continues with an introduction to the technologies utilized in the implemented system. The following technologies are discussed: LoRa™, LoRaWAN™ protocol, MQTT protocol, Ionic and Angular framework. With the help of these technologies, a system is presented in the second half of this work. The system is formed by end-devices (LEDs controlled by a microcontroller), a gateway and mobile application. The thesis discusses, in detail, the architecture and implementation of the gateway, and mobile application. The documentation of end-devices is available in Reference [1].

Keywords: VLC, visible light positioning, LoRa™, LoRaWAN™, gateway, Ionic, Angular, mobile application

Supervisor: Ing. Stanislav Vítek, Ph.D.

Abstrakt

Cílem této práce je návrh implementace komunikačního systému pro navigaci uvnitř budov s využitím komunikace ve viditelném světle. V úvodu práce je teoreticky popsána problematika navigace uvnitř budov z dostupných zdrojů, neboli rozdělení navigačních systémů, popsání jejich matematických interpretací a představení algoritmů použitých v literatuře. Dále je popsána technologie LoRa™, LoRaWAN™, MQTT protocol, Ionic a Angular, které jsou využity v navrženém komunikačním systému. Navržený systém obsahuje koncové zařízení (LED diody ovládané mikrokontrolérem), gateway a mobilní aplikaci. Gateway a mobilní aplikace je detailně popsána v druhé části této práce. Koncové zařízení lze nalézt v práci mého kolegy Štěpána Bosáka, uvedené ve zdroji [1].

Klíčová slova: VLC, komunikace viditelným světlem, LoRa™, LoRaWAN™, gateway, Ionic, Angular, mobilní aplikace

Překlad názvu: SW řešení systému pro navigaci uvnitř budov s využitím komunikace ve viditelném světle

Contents

1 Introduction	1	5.3 Network functionality test	42
Part I		6 Conclusions	45
Theoretical Part		Appendices	
2 Algorithms of indoor navigation using visible light communication	5	A List of Abbreviations	49
2.1 Algorithms of network-based systems	5	B Bibliography	53
2.1.1 Mathematical techniques used in IPs algorithms	6		
2.1.2 Received signal strength	7		
2.1.3 Angle of arrival	9		
2.1.4 Time of arrival and Time difference of arrival	10		
2.1.5 Fingerprinting	12		
2.1.6 Proximity	12		
2.2 Inertial based systems	13		
2.3 Hybrid systems	14		
2.4 Indoor positioning systems using VLC	15		
3 Overview of used technologies	19		
3.1 LoRa™	19		
3.2 LoRaWAN™ protocol	21		
3.3 MQTT protocol	22		
3.3.1 MQTT architecture	23		
3.3.2 Quality of Service (QoS)	23		
3.3.3 Message format	24		
3.4 Angular	25		
3.5 Ionic	25		
Part II			
Practical Part			
4 Communication design	29		
4.1 The gateway architecture	29		
4.2 The gateway implementation	31		
4.2.1 Components installation	32		
4.2.2 Components configuration	34		
4.2.3 The network setup	35		
5 Configuration App	39		
5.1 The prototype of web application	39		
5.1.1 The application architecture	39		
5.1.2 The application design	40		
5.2 The prototype of mobile application	41		
5.2.1 The application architecture	41		
5.2.2 The application design	42		

Figures

2.1 Classification of indoor navigation systems.	6
2.2 Trilateration and triangulation technique.	6
2.3 Multilateration technique.	7
2.4 RSS based system.	7
2.5 AoA structure scheme.	9
2.6 Model of the multi-element.	10
2.7 The TDoA visualization.	11
3.1 The up-chirp (left) and down-chirp (right).[2]	20
3.2 LoRa™ decoding.	20
3.3 The LoRaWAN™ protocol stack.	21
3.4 The typical LoRaWAN™ topology.	21
3.5 QoS communication timeline.	24
3.6 MQTT message format.	24
4.1 Architecture of the testing platform.	29
4.2 The gateway architecture.	30
4.3 Example of mosquitto.conf.	33
4.4 Example of service file.	33
4.5 Example of the service profile.	36
4.6 Example of the device profile.	36
4.7 Final end-device communication.	38
4.8 System structure.	38
5.1 The web application prototype design.	41
5.2 The mobile application prototype design.	42
6.1 Example of received messages in end-device.	46

Tables

2.1 An overview of network based algorithms.	13
5.1 Combinations of test data.	43

List of Listings

4.1	The PostgreSQL role and database creation	34
4.2	global-conf.json file changes.	35
4.3	chirpstack-network-server.toml file changes. .	35
4.4	chirpstack-application-server.toml file changes. .	35
4.5	Encode function example.	37
4.6	Key code parts in end-device implementation.	37



Chapter 1

Introduction

Increasing development in robotic systems, location-based services (LBS) and overall future technologies has necessitated a cost effective, easy and accurate implementation of indoor positioning systems (IPs).

First attempts to achieve this goal led to traditional indoor positioning methods. However, these techniques have not delivered desired results. Systems based on Radio frequency (RF-based) struggled with aspects of multipath fading, interference with other RF devices and had a poor accuracy. Moreover, they are limited in certain environments like hospitals and other areas with RF sensitive equipment.[3] Another technique used Infrared (IR) or ultrasonic signals. However, both aforementioned techniques need additional time and cost to deploy a network infrastructure of location sensors. WiFi, Bluetooth and Zigbee, just to name a few, were also attempted, but with the same outcome.[4]

In light of these circumstances, significant hopes are put in visible light communication (VLC) due to its physical properties and recent rapid development. VLC, often called visible light positioning (VLP), is a technology with many strengths that makes it a promising candidate for IPs.

This thesis provides an introduction into the field of indoor positioning. Furthermore, it will propose a system, that enables system configuration using a native mobile application. The system utilizes LoRaWAN™ and MQTT protocol in order to connect a mobile application with an end-device.

The thesis structure is divided into two main parts: Theoretical and Practical. The Theoretical part focuses on positioning techniques and algorithms, which are widely used in VLC. The VLC systems are organized based on their nature at the beginning of the part. Then, all the groups are described in detail. Moreover, this part introduces various technologies used in the system, which will be proposed in the Practical part of the work.

The Practical part contains two chapters: Communication design (Chapter 4) and Configuration app (Chapter 5). The Communication design chapter discusses the central unit (gateway) architecture, including hardware and software implementation. The Configuration app chapter presents two application prototypes, which use different code architectures and languages. The first prototype is a web application written in HTML, CSS and JavaScript. The second prototype is a cross-platform mobile app utilizing Ionic and

Angular framework. Finally, a test is performed in order to verify the correct functionality.



Part I

Theoretical Part

Chapter 2

Algorithms of indoor navigation using visible light communication

This chapter will introduce several mathematical techniques such as trilateration, triangulation and multilateration. It will also discuss the conventional VLC positioning algorithms and its practice in real-life applications.[5]

Traditionally, indoor positioning systems are classified into three groups: **Network based**, **Inertial based** and **Hybrid** systems. Hybrid systems usually combine more different algorithms together with the intention of increasing positioning accuracy. Inertial-based systems utilize integrated sensors in order to measure the motion of the target. The estimation of the position is evaluated from the sensor measurements. Therefore, the physical infrastructure deployed in the building is not required.[6] Hybrid and inertial-based systems will be introduced briefly in section 2.2. This chapter will focus on network-based systems because they are fundamental for VLC systems. Besides this classification, VLC systems can be also divided into two groups based on its receiver: Photodiode based (PD-based systems) and image sensor based systems. PD-based systems are usually unstable due to their vulnerability on ambient light or reflections from walls and furniture, which induce a steep accuracy fall. On the other hand, image sensor based structure can easily deal with the background light interference by utilizing an image processing method. Therefore, this technique has been deeply explored and the positioning accuracy can reach centimeters.[7]

2.1 Algorithms of network-based systems

The network-based systems use infrastructure of lights deployed in the building. Based on the information transmitted by the optical signals, they estimate the position of the target. There are two kinds of network-based systems: Range-based and Range-free systems. The full classification structure of IPs is shown in Figure 2.1. Firstly, this section will introduce range-based algorithms such as **received signal strength (RSS)**, **time of arrival (ToA)**, **time difference of arrival (TDoA)** and **angle of arrival (AoA)**. Secondly, the section will introduce the range-free algorithms: **Fingerprinting** and **Proximity**. However, before this, there are three main mathematical techniques:

function, that expresses the infinite possible Rx locations. These possible locations, if plotted, forms a hyperbolic curve as shown in Figure 2.3. Thus, multilateration is also called hyperbolic positioning.[10, 11] This technique is used in GPS, ToA or TDoA. The last two will be further discussed in section 2.1.4.

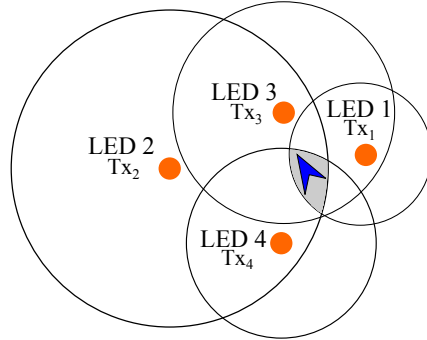


Figure 2.3: Multilateration technique.

2.1.2 Received signal strength

RSS-based systems are widely utilized in the field of indoor navigation because of its simple measurements and no need of additional hardware. However, the main disadvantage of RSS algorithm is the high dependency on received signal strength. Thus, in noisy environments, the signal strength will fluctuate, and the localization accuracy can severely fall.[12] Model of RSS-based system is shown in Figure 2.4a and 2.4b.

For better understanding, assume the following architecture: Tx (LED)

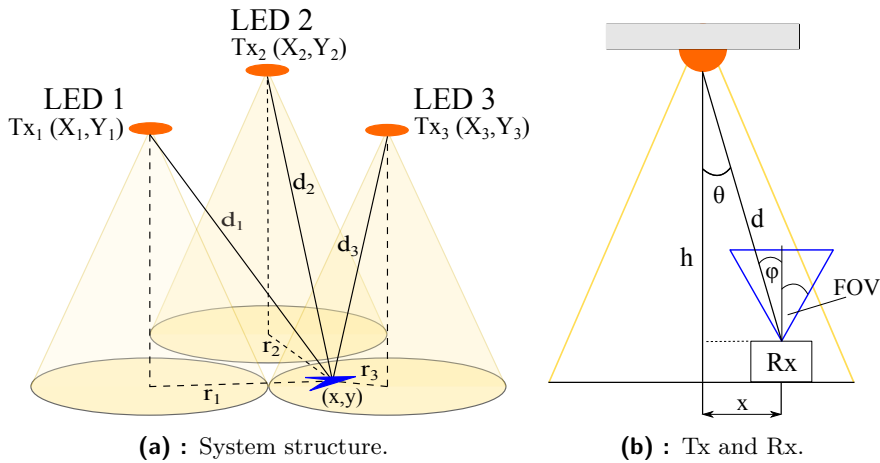


Figure 2.4: RSS based system.

transmits an optical signal, specific for the Tx. The signal is evaluated in Rx, which is formed by a PD-sensor or image sensor. The Rx's position is estimated based on the strength (power) of the signal at the Rx's location.

2.1.3 Angle of arrival

AoA is a positioning technique that calculates the angles under which the incident signals fall on Rx from different Tx's. A great advantage of AoA-based systems is no necessity for time synchronization as it is in ToA or TDoA. Also, AoA does not require any signal strength measurements.[5] However, besides these advantages, there is a drawback. The cost of these systems is generally high. It is because the system needs an antenna array or receivers with orientation capability in case of VLC implementation. Aside from this drawback, AoA has interesting options.

Since light beams can be made directional, some AoA applications utilize

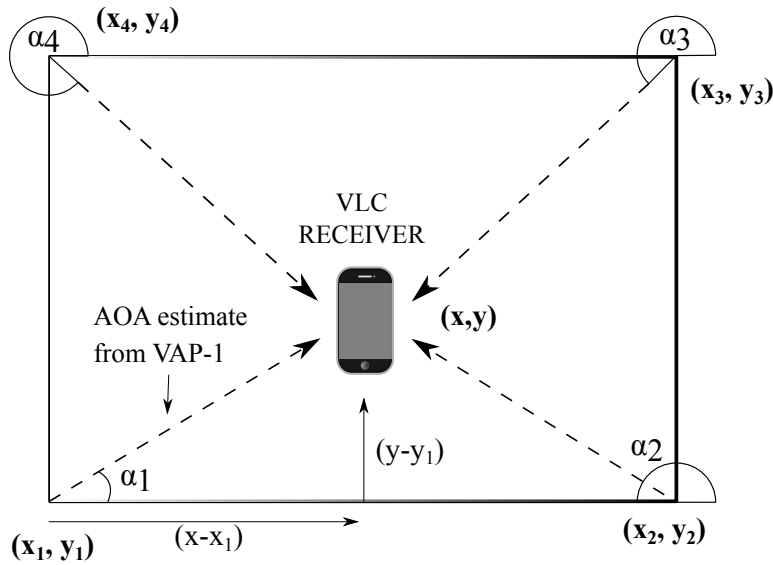


Figure 2.5: AoA structure scheme.

multi-element visible light access points (VAPs). Model of such multi-element light source is shown in Figure 2.3.[15]

For a deeper understanding of AoA, assume a setup from Reference [15] given in Figure 2.5. There is a multi-element VAP placed in each corner. To simplify this problem, consider a two-dimensional topology, where the VAP is on the same horizontal plane as the Rx. LEDs in VAP have the same field of view (FOV) and are directed to a different angle as shown in Figure 2.6. The different angle of each LED causes, that at a given time, there is just one LED beam per VAP received by Rx. This LED transmits message which has a specific header identifier and encrypted information about its position. The location is calculated as shown in Figure 2.5. Each LED transmits its message. The ones that are within line of sight (LOS) are connected and Rx will locate itself based on their message.

If there are just two LEDs connected to Rx (meaning Rx receives and evaluates their messages), the location will be estimated as the intersection of lines extending from the center of the LEDs' FOV. If the number of connected LEDs is greater than two, least square estimator (LSE) can be used to

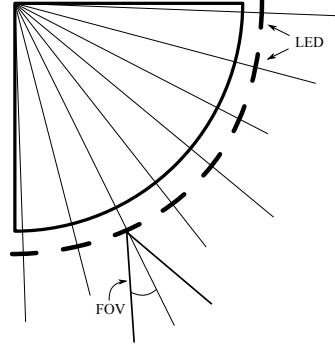


Figure 2.6: Model of the multi-element.

determine the location. From Figure 2.5, for $i = 1, \dots, 4$, can be stated:[15]

$$\frac{\cos \alpha_i}{\sin \alpha_i} = \frac{x - x_i}{y - y_i} \implies x \sin \alpha_i - x_i \sin \alpha_i = y \cos \alpha_i - y_i \cos \alpha_i. \quad (2.8)$$

Then a generalized matrix (2.9) for AoA can be created by evaluating the expression (2.8) for each VAP:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad (2.9)$$

where

$$\mathbf{A} = \begin{bmatrix} \sin \alpha_1 & -\cos \alpha_1 \\ \sin \alpha_2 & -\cos \alpha_2 \\ \vdots & \vdots \\ \sin \alpha_N & -\cos \alpha_N \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} x_1 \sin \alpha_1 - y_1 \cos \alpha_1 \\ x_2 \sin \alpha_2 - y_2 \cos \alpha_2 \\ \vdots \\ x_N \sin \alpha_N - y_N \cos \alpha_N \end{bmatrix}. \quad (2.10)$$

Assume $\mathbf{x} = [x, y]^T$ to be the Rx's location. Then LSE solution is given as:

$$\hat{\mathbf{x}} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}, \quad (2.11)$$

where $(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$ is Moore-Penrose pseudoinverse matrix, that can be used for LSE because \mathbf{A} is a invertible matrix and has right inverse.

■ 2.1.4 Time of arrival and Time difference of arrival

ToA is well-known range based technique that is used in GPS. ToA principle is based on measuring time delay of a transmitted signal from Tx to Rx. Thus, the transmitted message includes the time of transmission and Rx calculates the time of reception. The time delay can be computed from this information, and the true distance between Tx and Rx can be obtained by multiplying time delay and the propagation velocity of the utilized technology (e.g. speed of light in VLC) as in (2.12).

$$d = \Delta t v. \quad (2.12)$$

The target's location is evaluated by employing the lateration methods mentioned earlier. Accordingly, the location is the intersection of circles

whose center lies in reference points, and the radius is equal to the computed distance from Rx. For two dimensional space, at least three different reference points are required.[6]

The main drawback of ToA-based systems is the requirement of precise and strong time synchronization between all TxS and RxS, in order to achieve high positioning accuracy. Such synchronization can significantly increase the cost of the system though. Thus, most indoor navigation systems tend to use ToA's alternative approach instead. The alternative approach is called TDoA.[5] Both of these techniques are robust for the environment noise.[9]

TDoA is a range-based technique that relaxes the synchronization constraints. In TDoA, the synchronization is required just between TxS or RxS. In other words, there is no necessity for synchronization between Tx and Rx . TDoA can be categorized into two groups: multi-node TDoA and multi-signal TDoA. Multi-signal TDoA uses two signals with different propagation velocity to calculate the distance from Rx. The second group computes the difference in the ToA of signals transmitted from two TxS to Rx. The difference defines a hyperbolic (hyberboloid in 3D) curve on which the Rx should lie as in Figure 2.7. For two dimensions, the intersection of two hyperbolas can effectively point to the location of Rx. In the case of the three-dimensional system at

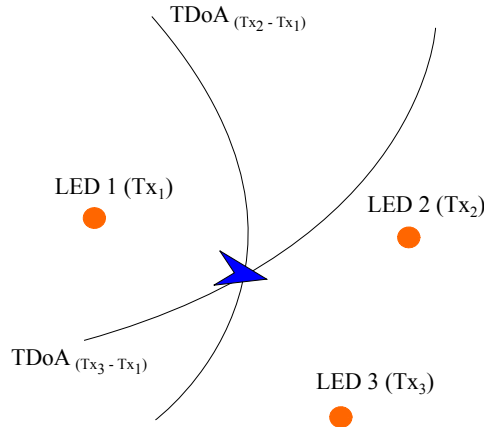


Figure 2.7: The TDoA visualization.

least three TDoAs are required to find the position of the target.[9, 6]

To better understand the ToA technique, assume four reference LEDs (TxS) which will emit a message at a time t_0 . The message contains, besides the LED identifier, also the time t_0 . The Rx receives four messages at the time t_i , for $i = 1, \dots, 4$. From this data, it is easy to calculate the offset between reception times: $\tau_2 = t_2 - t_1$, $\tau_3 = t_3 - t_1$, $\tau_4 = t_4 - t_1$. The Rx position (x, y, z) and t_1 can be computed from the system (2.13), where (X_i, Y_i, Z_i) is

the position of i -th LED.[10]

$$\begin{cases} (x - X_1)^2 + (y - Y_1)^2 + (z - z_1)^2 = (c(t_1 - t_0))^2 \\ (x - X_2)^2 + (y - Y_2)^2 + (z - z_2)^2 = (c(t_1 - t_0 + \tau_2))^2 \\ (x - X_3)^2 + (y - Y_3)^2 + (z - z_3)^2 = (c(t_1 - t_0 + \tau_3))^2 \\ (x - X_4)^2 + (y - Y_4)^2 + (z - z_4)^2 = (c(t_1 - t_0 + \tau_4))^2 \end{cases} \quad (2.13)$$

2.1.5 Fingerprinting

Fingerprinting is an algorithm that estimates the Rx's position based on a comparison of received Tx's values and stored values in a database. These stored values are called fingerprints. Fingerprints are basically, characteristics of the signals at different reference positions. This algorithm uses previously mentioned techniques (RSS, AoA, TDoA, ToA) for evaluation. The fingerprinting has two phases: calibration and operating phase. During the calibration phase, fingerprints are created and saved in the database. Whenever system settings are changed, the calibration has to be repeated because fingerprints may also differ. Besides this drawback, the fingerprinting algorithms are power and time-saving.[16, 6]

When in the operating phase, real-time measurements are matched with fingerprints in order to find the target's location.[5, 16] There are two kinds of fingerprinting algorithms: deterministic and probabilistic.

In the **deterministic** fingerprinting systems, the position is estimated as the fingerprint of a Tx which minimizes the Euclidean distance between the received measurements and the fingerprints. Some of the deterministic fingerprinting techniques employ the k-Nearest Neighbours (kNN) approach, which reduces the computational complexity of the technique and increases the accuracy of the system. kNN is an instance-based learning method that basically compares new measured instances with instances seen in training. Also, other approaches have been proposed, such as linear discriminant analysis or Support Vector Machines (SVM).[6]

The **probabilistic** technique finds the location with maximum likelihood and sets it as the target's location. Such probabilistic algorithm is used in Reference [16], where the location is calculated based on RSS measurements and Bayes's theorem.

2.1.6 Proximity

The proximity algorithms are based on a simple idea: If the Rx receives a signal from a reference LED, it must be close by. In order to use the proximity algorithm, a large number of reference LEDs is required. Even then, however, the accuracy is far from perfect. Mostly in higher tens of centimeters. On the other hand, this method is easy to implement in comparison to other positioning algorithms. Thus, it is an ideal candidate for room accuracy applications.

After the Rx receives transmitted messages, the location is determined by

its proximity from the transmitted location. If Rx receives more than one message, the Rx position is to establish in terms of RSS.

The error committed by the proximity method is proportional to the size of the LED's coverage. If the LED's coverage is low, accuracy will rise. Although, if the coverage is reduced, the number of reference LEDs has to be increased in order to keep the whole area covered. That results in extra cost of the system.[6] The accuracy can be increased by overlapping LED beams, where more identifiable regions are created.

A significant role in proximity-based systems is played by the optical boundaries of LEDs. If the boundaries are well defined, it will prevent unnecessary delays in positioning time. The Reference [3] states, that a single LED transmitter with properly defined boundaries can reduce significantly the positioning delay. To be specific, with well-determined boundaries, the positioning delay is reduced by factor of 13 and 230 for 4-bit and 12-bit packet with no overlaps. In case of systems with LED overlapping beams, the reduce factor is 12 and 287 for 4-bit and 12-bit packet.

Algorithm	Accuracy	Additional hardware	cost
RSS	Low	None	Low
AoA	High	Receiver with orient. capability	High
ToA	High	precise clock	Medium/High
TDoA	High	precise clock	Medium
Fingerprinting	Medium	None	Low
Proximity	Poor	None	Low

Table 2.1: An overview of network based algorithms.

2.2 Inertial based systems

Unlike the network-based systems, the inertial based systems do not use the physical infrastructure in the building for computing the target's location. These systems measure the target movement using the forces applied to the inertial measurement unit (IMU) in the portable device. The IMU is usually formed by 3 axis gyroscope, 3 axis magnetometer and 3 axis accelerometer. The crucial estimation part is to compute the relative orientation of the IMU and the body of the target. Whenever IMU is placed on the body of the target, the axis of the IMU may not coincide with the axis of the target. Any misalignment results in errors of the measurement. The transformation is obtained by sequentially rotating around three axes. Angles of the rotation are expressed as Euler angles (roll, pitch, yaw).[6]

Inertial navigation can be classified into two groups: Strapdown systems and Step and Heading systems (SHS).[6]

The **Strapdown** systems use the fact that the position is double integration of acceleration. First, the angular velocity is measured by the gyroscope and orientation is evaluated. The actual integration is performed after the accelerometer's signal is adapted to the correct rotation and also the

smartphone-based system, but utilizing VLC is proposed in Reference [7] and will be further discussed in section 2.4.

2.4 Indoor positioning systems using VLC

This section is dedicated to the introduction of various indoor navigation projects using visible light communication. All of the mentioned projects utilize the Multiple LEDs Estimation Model (MLEM). MLEM is a positioning technique, that uses transmitted optical signals from multiple sources for estimating the target's location. The orientation of LEDs can be either with or without overlap. Thus, there are two types of receivers. The first type of receivers are made for single access, and they are used for the no-overlap approach. The second type is multiple access receivers, which are designed for MLEM with overlap. However, the overlap causes interference, since data transmitted are at the same wavelength and are asynchronous. This interference can be minimized by packet-based pulse duration multiplexing (PDM) and a low duty cycle transmission protocol. It is experimentally proven that MLEM with overlap increases positional accuracy, and the multiple access systems are shown to be more reliable for positioning. Therefore, this strategy is often utilized in VLC.[18]

An excellent example of the proximity algorithm is proposed in Reference [7]. The algorithm uses the properties of the CMOS rolling shutter to determine signals origin. For those who are not familiar with rolling shutter technology. There is a little introduction. The main idea of the rolling shutter is that it exposes pixels just in one row at a time and reads them out immediately after the exposure is finished. The same process repeats for each row individually until all rows are read. On the other hand, there is also another technology the Charge-Coupled Device sensor. The CCD sensor can often be found in ordinary cameras and it uses a global shutter. Unlike the rolling shutter, the global shutter exposes all the pixels at the same time, and after the exposure, it reads the data. As it will be explained later, the global shutter can not be used in the algorithm. The proposed scheme uses PWM with variable frequency and duty-ratio. This PWM modulates the LED transmitters with a unique signal. Each LED is then provided with a set of frequency and duty-ratio in order to differentiate them from each other. The CMOS image sensor with the rolling shutter can capture a specific number of bright stripes based on the unique PWM. This is possible because of the row by row exposure and readout of the rolling shutter. The LED's identification is calculated by Fisher discriminant analysis (FDA), which basically finds a projection vector that separates two or more classes of objects. The algorithm has proven to be sufficient as it can offer 1035 unique LED-ID with 100% recognition rate and maximum distance between Tx and Rx up to 6m. It was also shown, that the scheme is fitting for large-scale systems and is easily integrated with smartphone's camera.

Next algorithm uses similar system topology as will be proposed in this thesis. The structure consists of a controller part which controls the con-

SIFT descriptor recovery is in the Reference [23]. The scheme was experimentally demonstrated. The results showed successful localization of the Rx with positioning error 5 cm for a height of 170 cm.

The last algorithm, which is going to be discussed in this section, is proposed in Reference [24]. It is a hybrid architecture utilizing VLC, proximity and Zigbee network. The idea of the scheme is that visible light identification is sent from the light source and received by the target (smartphone), as it is in other algorithms. However, the received data are evaluated by the proximity method and reconstructed in the target. Then data are sent through the Zigbee network to the main node. They are displayed on the system controller screen, which is connected to the main node. This approach is significantly different from others. Unlike previous algorithms, this one sends the position information to the main node, where it is seen and can be evaluated from the user. This algorithm would be great for robot monitoring in manufactures.

Chapter 3

Overview of used technologies

This chapter will acquaint readers with technologies that will be later used in the system implementation. Particularly, the design utilizes LoRa™, LoRaWAN™ and MQTT protocol in the central unit. These three technologies will be discussed first. Then, this chapter will present a basic introduction into Angular and Ionic framework, that will be used in order to develop a cross-platform mobile app. This application will eventually be used for system configuration purposes.

3.1 LoRa™

LoRa™, which stands for "Long Range", is a proprietary PHY (physical) layer, developed by Semtech and now maintained by LoRa™ alliance. LoRa™ is using a wireless chirp spread spectrum (CSS) modulation technology to encode the messages. Generally, a spread spectrum modulations extent data rate in a wider bandwidth in order to increase the reach of the system. Besides wider bandwidth, CSS adds another feature. It uses frequency chips for encoding data. A LoRa™ chirp is a signal with a linear variation of frequency over time. The variation of the frequency can increase, then the chirp is an up-chirp. In the case of decreasing frequency, the chirp is called a down-chirp. Both chirp types are visualized in Figure 3.1.[2] The transmitted signal is composed of these chirps. Each chirp, in the signal, represents a symbol that holds the data. The symbol is unity for data.

The decoding process works as follows: The Rx receives the signal and generates a so-called inverse chirp signal. The inverse chirp signal is the opposite signal to the one that was received. In other words, for every up-chirp received, a down-chirp will be generated, and so forth. After the inverse chirp signal is generated, the Rx adds both signals together. The result is a flat lines signal, where each line represents the transmitted data. The process is shown in Figure 3.2.[25]

The great advantage of LoRa™ is the long-range capability as one gateway (central unit) can cover an entire city or hundreds of square kilometers. Another advantage is its high link budget and its low battery usage. The link budget is dependent on modulation and transmission power. In Europe, the transmission power limit is 14 dBm; therefore the link budget can get

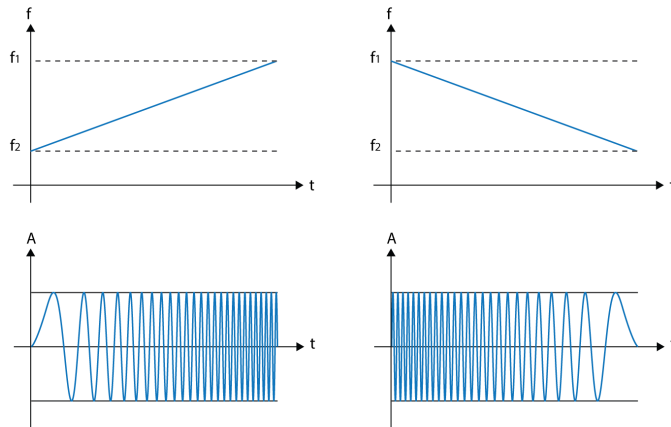


Figure 3.1: The up-chirp (left) and down-chirp (right).[2]

up to 156 dB. The LoRa™ communication is divided into several frequency channels and data rates. LoRa™ data rates are in a range of 0.3 kbps to 20 kbps. If channel aggregation is used, the data rate can reach up to 50 kbps.

In order to utilize LoRa efficiently, there are a few parameters that can be used to customize the communication. First is a Spreading Factor (SF). SF indicates the duration of a chirp. In other words, it measures how many chirps can be sent per second. LoRa™ operates with $SF \in \{7, \dots, 12\}$, where SF7 is the shortest time on-air, and SF12 is the longest.[26] This mechanism provides resistance to interference and multipath fading.[27] The Second parameter is Bandwidth (BW), which is the interval of minimum chirp frequency, to maximum chirp frequency as shown in Figure 3.2. BW for Europe is 125 kHz or 250 kHz. The last parameter is Coding Rate (CR), which is the number of forward error corrections.[28]

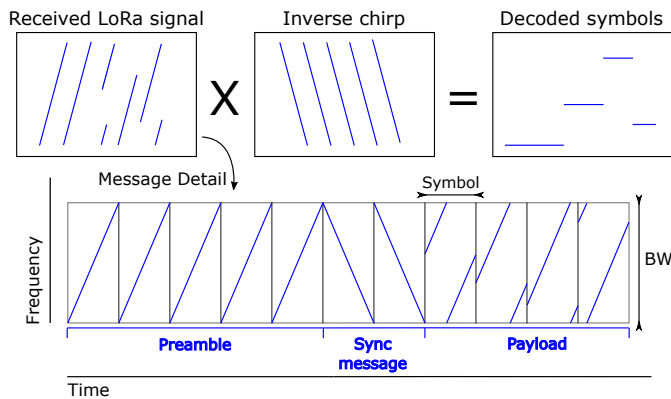


Figure 3.2: LoRa™ decoding.

3.2 LoRaWAN™ protocol

LoRaWAN™ is a Media Access Control (MAC) layer on top of LoRa PHY layer. LoRaWAN™ manages the network architecture, security, capacity and battery lifetime of nodes. The LoRaWAN™ stack is shown in Figure 3.3.

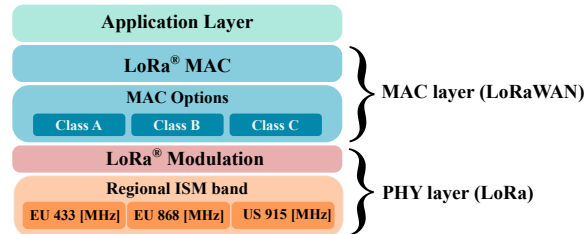


Figure 3.3: The LoRaWAN™ protocol stack.

To establish a proper connection between gateway and nodes, data from the application layer and MAC commands are encapsulated in MAC payload. The MAC layer then creates a MAC frame using the MAC payload. Besides MAC payload this frame consists MAC header and Message Integrity Code (MIC). The header defines the protocol version and type of the message (e.g. management or data frame and downlink or uplink). MIC is responsible for authentication of end-nodes and also prevents the message forgery. MAC frame is then used in LoRa™ PHY layer to create a PHY frame by inserting preamble, PHY header, Cyclic Redundancy check (CRC) of the PHY header and entire payload CRC.[2] Finally, the PHY frame is ready and transmitted.

The typical LoRaWAN™ architecture is star-of-stars topology as it is visualized in Figure 3.4.

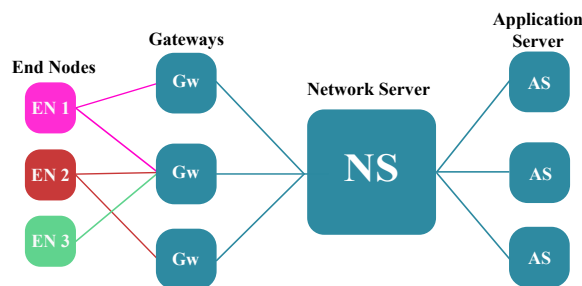


Figure 3.4: The typical LoRaWAN™ topology.

There are two types of authentications between the end-node and the network: Over-the-Air Activation (OTAA) and Activation by Personalization (ABP). ABP is a connection procedure, where the authentication data (DevAddr, NwksKey and AppSKey) are statically hardcoded in the device. Even though it is simple, there are serious security downsides and should be used cautiously. On the other hand, OTAA is authentication, where devices perform a join-procedure with the LoRaWAN network, during which the keys are negotiated and DevAddr dynamically assigned. Thus, OTAA is usually a

better option, but it requires more time and computational resources.

All communication in LoRaWAN™ is bi-directional in general. The uplink communication is associated with the node to the gateway direction. The opposite communication is called downlink as it goes from the gateway to nodes. The ordinary structure uses a gateway as a bridge between LoRa™ nodes and the network server. A complete solution of a gateway is provided in Chapter 4. This gateway utilizes LoRaWAN™ and MQTT protocol. Back to the architecture though. The network server is basically a brain that holds most of the system complexity and intelligence. It manages the security checks, schedules the acknowledgments, and performs ADR based on the node's location.[2] ADR allows the network server to adapt data rate by changing SF in order to find the best trade-off between power efficiency and link robustness.[29]

LoRaWAN™ networks distinguish three bi-directional classes of end-nodes based on the application:

- **Class A (for All)** nodes schedule uplink transmission based on their own needs. This communication type is strictly asynchronous. Each uplink transmission is followed by two short downlink receive windows, where the gateway can transmit packets to the node. Class A is the most power-saving bi-directional class provided in LoRaWAN™, but as a trade-off there is the least flexibility in downlink communication. These nodes are suited for applications that only require downlink communication shortly after the uplink transmission (e.g. temperature control sensors, traffic sensors).[2]
- **Class B (for Beacons)** nodes provide additional downlink windows at scheduled times. Besides the receiving slots after an uplink communication, the gateway sends a synchronized periodic beacon type message which activates the receiving slots. This class can be useful for battery-powered devices, which needs to be occasionally configured or controlled from the server. [26, 2]
- **Class C (for Continuously listening)** has the maximum downlink flexibility. The gateway can downlink data at any time except when the node is transmitting. However, this low downlink communication latency comes with the price of higher power consumption. That might be a crucial disadvantage for some applications.[2]

3.3 MQTT protocol

Message Queue Telemetry Transport also known as MQTT is an application layer protocol, which utilizes a broker-based publish/subscribe messaging transport. The protocol was developed by Andy Stanford-Clark of IBM and Arlen Nipper in 1999. In 2013, MQTT became a standardized protocol of OASIS. MQTT is mostly used on top of TCP/IP. Compared to HTTP, that also relies on TCP/IP, MQTT has lower protocol overhead.[30] MQTT is

extremely lightweight and easy to implement. Thus, it is an ideal protocol for constrained devices and high-latency, low-bandwidth, or unreliable networks, which can benefit from its design. The main principle of the protocol is to minimize network bandwidth and device resource requirements. These principles make it an outstanding candidate for Machine to Machine (M2M), Internet of Things (IoT), and mobile applications, where battery power efficiency and bandwidth are at a premium.[31]

■ 3.3.1 MQTT architecture

The MQTT based systems consist of clients and MQTT broker (server). A client can be an IoT device that sends or receive telemetry data. The client's role varies based on the application. For instance, the structure, proposed in the second part of this thesis, the client will mainly publish messages. Either way, in order to start the communication, MQTT client needs to connect to the MQTT broker first. It will do so by using a particular type of message (CONNECT). Message types will be further discussed in section 3.3.3. After the connection is established, the client has to declare its role in the network.[32]

In the case of being a publisher, a topic has to be set. The topic is a string identifier, by which different messaging channels can be distinguished. In other words, topics organize and filter messages to different groups. On the other hand, if the client wants to receive data, it will need to subscribe to a specific topic.[32]

In order to create a client from a device, a MQTT library has to be properly installed. There are multiple open source MQTT client's libraries, that can be used for this purpose. Another fundamental piece of MQTT structure is a MQTT broker. MQTT broker is mainly responsible for handling the communication between clients. It can manage thousands of clients at the same time. Whenever a message is received, MQTT broker has to find all clients subscribing to this particular topic and distribute the message to them. Besides handling communication, brokers have to manage the authentication of clients. The authentication is accomplished by including username and password in the CONNECT message and topic permissions are implemented on the broker side in order to confine clients to publish or subscribe. To secure the communication between clients and broker TLS and SSL encryption is used. There are a few implementations of MQTT brokers. In this thesis, we will use an open-source MQTT broker called Mosquitto.[32]

■ 3.3.2 Quality of Service (QoS)

There are three QoS in MQTT, and they are distinguished by a number from 0 to 2. If QoS is zero, the publisher will send a message at most once and will not check the delivery of the message. The receiver will not send an acknowledgment and the sender will not resend the message. This QoS structure is also called At most once option.

If $QoS = 1$, the publisher will store and send the message. MQTT Broker

publishes the message to subscribers and sends an acknowledgment to the publisher. The publisher deletes the message as soon as he receives the acknowledgment. Some literature refers to $QoS = 1$ as At least once QoS option.

The last QoS, sometimes also called Exactly once option, is the most reliable QoS scheme. As in $QoS = 1$, the publisher stores and sends a message. MQTT broker stores and publishes the message to subscribers. The broker also sends PUBREC (publish received) message to the publisher and the publisher sends PUBREL (publish release) to the broker, when he receives the PUBREC. The broker receives the PUBREL, deletes the stored message and sends the PUBCOMP to the publisher. The publisher deletes the message and the communication is completed.[32] The timeline of each QoS is shown in Figure 3.5.

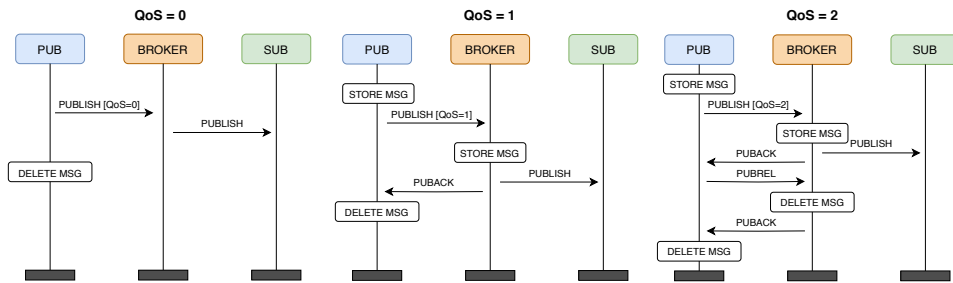


Figure 3.5: QoS communication timeline.

3.3.3 Message format

Every MQTT message is composed of a fixed header (at minimum 2 bytes), variable header and message payload, which are not always present. The message structure is depicted in Figure 3.6.

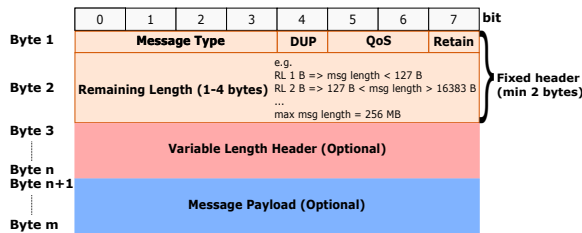


Figure 3.6: MQTT message format.

The first byte of the fixed header consists of the message type in four most significant bits (MSB) and flags in the last four bits. The message type is for instance: CONNECT (client request to connect to the broker), CONNACK (connection acknowledgment), PUBLISH (publish message), PUBACK (publish acknowledgment) and SUBSCRIBE. A full list of message types can be found in the official documentation. Flags examples are DUP

(duplicate message), QoS (Quality of Service), and RETAIN (Retain message). The second part of the fixed header informs about the remaining length of the message. This part can be up to 4 bytes long, where each byte uses 7 bits for the length information, and MSB is a continuation bit. If the continuation bit is zero, there will be no other length information behind this byte. These length bytes inform about the length of the variable header and the payload together.[32, 33] Another piece of the message is the variable header. It is used if the message needs additional control information. The structure of the variable header differs based on the message type.

■ 3.4 Angular

Angular is an open-source JavaScript Framework for creating reactive Single-Page-Applications (SPAs). SPA is a web page that seems to visit different pages based on the changing URL; however, in the end, the page has never changed. SPA is one HTML file wrapped with JavaScript code that is downloaded from the server. Every change on the page is then rendered in the browser. Thus, there is no need to reach out to the server for every page change or every piece of new data. However, if the application requires new data from the server, it can be downloaded in the background. Moreover, such a web page gives the user a very reactive user experience with a feeling of being in a native mobile application. To expand on Angular applications, they are written in TypeScript, which is a super-set of JavaScript. TypeScript features help to write more robust code. Angular, as a framework, can be classified into two groups: AngularJS (Angular1), which will not be discussed in this thesis and Angular. Angular refers to all versions of Angular since Angular2, which was realized in 2016 and it fixed all the issues AngularJS had. It is important to say that Angular2 was rewritten from the ground up; therefore AngularJS is a significantly different framework. In this thesis latest Angular v9.1.1 was used with Ionic framework to build a native mobile application.

■ 3.5 Ionic

Ionic is an open-source framework that allows building preferment mobile and desktop applications using web technologies (HTML, CSS, JavaScript). It can also be easily integrated with Angular or React framework as it will be shown in the practical part of this thesis. Ionic focuses mainly on the front-end of an application, such as UI animations, gestures, and controls. Thus, it introduces a set of web components, which is a technology supported by modern browsers. A web component is basically, a HTML element, that has behind the scenes more complex logic (e.g. menu bar, tabs,...). The great advantage is that one codebase runs on all platforms with just minor adjustments. The ionic platform also consists of a Capacitor and Ionic CLI. Capacitor is a tool, that is capable of taking an existing web application and

wrapping it into so-called web-view into a native application. In other words, Capacitor gives a native mobile application, that runs the web application in the side of it, but in a way, it is indistinguishable from a native application written in Swift or Java.



Part II

Practical Part

Chapter 4

Communication design

This chapter will provide a communication design for the indoor navigation testing platform. The whole system structure is shown in Figure 4.1. End-nodes are formed by 10W LED which is controlled by NUCLEO-F446RE with a LoRaWAN™ extension shield (I-NUCLEO-LRWAN1). The detailed architecture of the end-nodes is documented in the Reference [1]. The previously mentioned LoRaWAN™ shield on the end-node will be crucial for the communication design, which will be introduced here in detail. The core idea of the structure is well visualized in Figure 4.1.

The application responsible for system arrangement sends configuration data to the gateway via MQTT protocol (section 3.3). The gateway restructures the received MQTT payload to a readable format for LoRaWAN™ protocol (section 3.2). The configuration data are then transmitted to the particular end-node using LoRa™ modulation (section 3.1), where they are also processed. The gateway is crucial for the correct data transmission; therefore the architecture and realization will be proposed in the next section.

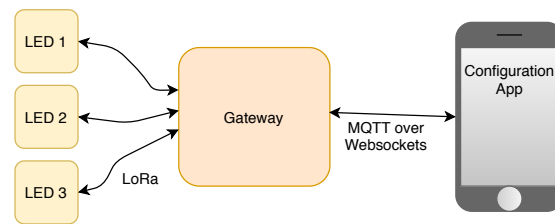


Figure 4.1: Architecture of the testing platform.

4.1 The gateway architecture

As it was mentioned in the introduction, the gateway is a central unit of the system. It allows transferring data between two discrete networks. Another key feature of gateways is its ability to connect various networks using more communication protocols. Our gateway connects end-nodes (LEDs), that use LoRaWAN™ protocol, with mobile or web application, which utilizes MQTT

Server, as one of the subscribers, takes the message and encodes. The further action is similar to the uplink communication, just opposite.

4.2 The gateway implementation

For the gateway creation, a Raspberry Pi 3 was used as a host, PRI 2 Bridge RHF4T002 as a connection between Raspberry Pi 3 and the Gateway module RHF0M301–868 supplied with 0 dBi Rubber Duck Antenna. The Gateway module RHF0M301–868 enables LoRa™ communication on the Raspberry Pi 3, which was required. In order to test the created network, a Seeduino LoRaWAN with GPS was used as a temporary end-node.

The first step in gateway development was choosing an appropriate OS, which would run on the Raspberry Pi 3. The main requirement was the light weightness of OS because Raspberry Pi 3 has limited memory storage. For that matter, Raspbian Lite GNU/Linux 10 (buster) was chosen. Raspbian Lite is an open-source operating system based on Debian, which is optimized for Raspberry Pi with no desktop environment. Thus, it is really lightweight and an excellent candidate for this design. In order to install Rasberian, a USB keyboard and a display was required.

First of all, an image of the OS was made. A nice tool to make an image of the Raspberry Pi OS is the Raspberry Pi Imager. A great advantage of this tool is that it automatically configures the SD card and downloads the chosen OS release.

Secondly, the actual installation was made on Raspberry Pi. There was minor Raspbian configuration required after the installation. For instance, SPI protocol was enabled (`sudo raspi-config` → Interfacing Options → SPI), which is fundamental for communication between the concentrator and packet forwarder. The Wi-Fi connection was set. There are two ways to do so. Either, it could be done through `raspi-config` or by editing the actual network configuration file manually¹. In order to use the edited network settings, the network has to be reconfigured with a command². Sometimes the reconfiguration or `raspi-config` will not succeed and the Raspberry Pi has to be rebooted. After the reboot, the network connection should be set and the IP address assigned³.

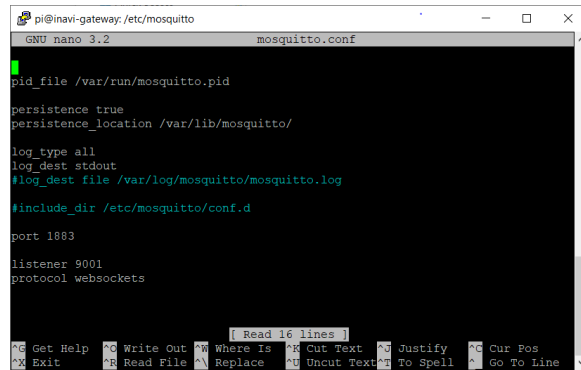
The last configuration was made in order to control the Raspberry Pi without external peripherals (the keyboard and the display). In that matter, the SSH protocol was enabled through `raspi-config` (`sudo raspi-config` → Interfacing Options → SSH). After that, the Raspberry Pi was ready for the installation of the gateway components.

¹`/etc/wpa_supplicant/wpa_supplicant.conf`

²`wpa_cli -i wlan0 reconfigure`

³`ifconfig wlan0 | grep inet`

After this minor edit, the mosquitto can be built by the "make" command and installed by "make install" command. In order to run mosquitto with websockets, there are two more things which have to be set. Firstly, the main mosquitto configuration file (/etc/mosquitto/mosquitto.conf) must contain default port 1883, which is used for the gateway components communication. Also, there has to be a listener on a websocket port. In this design 9001 is used. An example of such mosquitto.conf is shown in Figure 4.3. Secondly,



```

pi@inavi-gateway: /etc/mosquitto
GNU nano 3.2 mosquitto.conf
pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_type all
log_dest stdout
#log_dest file /var/log/mosquitto/mosquitto.log

#include_dir /etc/mosquitto/conf.d

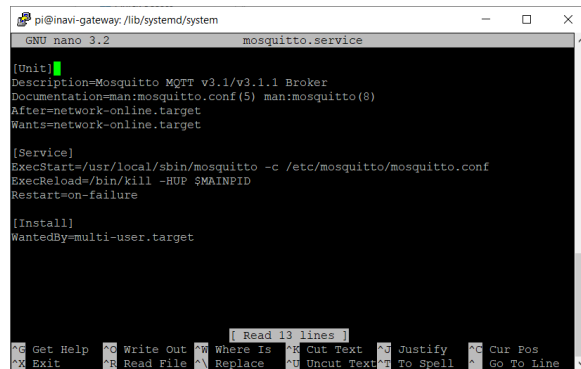
port 1883

listener 9001
protocol websockets

```

Figure 4.3: Example of mosquitto.conf.

mosquitto was set as a service. The easiest way to do so, is copying the service template included in the downloaded mosquitto directory⁸ to the directory, where all the services are stored (e.g. /lib/systemd/system). The final service file is shown in Figure 4.4. Then, the MQTT broker is ready.



```

pi@inavi-gateway: /lib/systemd/system
GNU nano 3.2 mosquitto.service
[Unit]
Description=Mosquitto MQTT v3.1/v3.1.1 Broker
Documentation=man:mosquitto.conf(5) man:mosquitto(8)
After=network-online.target
Wants=network-online.target

[Service]
ExecStart=/usr/local/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
ExecReload=/bin/kill -RUP $MAINPID
Restart=on-failure

[Install]
WantedBy=multi-user.target

```

Figure 4.4: Example of service file.

Last dependencies, that were required for the components, were Redis and PostgreSQL. Redis is an open-source, in-memory data structure store, used as a database. In the gateway, Redis stores transient data such as device-sessions. On the other hand, PostgreSQL is long-term storage and the gateway uses it for data, which should not expire. The PostgreSQL stores gateway's settings for instance. Redis and PostgreSQL was installed through apt package manager.

The last step was the LoRa Gateway Bridge, LoRa Server and LoRa App

⁸mosquitto/service/systemd/mosquitto.service.simple

Server installation. Dirmngr and apt-transport-https package was used for downloading the software repository as follows: Firstly, the repository key was set up⁹. Secondly, the created repository was added to the repository list by creating a new file¹⁰. Finally, the apt package cache was updated and components were installed via apt package manager¹¹.

4.2.2 Components configuration

With all the dependencies and components installed, the configuration process could begin. Firstly, as mentioned earlier, the LoRa Server and LoRa App Server store data in PostgreSQL database. Both components need their own database; therefore a role with a database was created for each of them via PostgreSQL prompt as it is shown in Listing 4.1. Before executing commands from Listing 4.1, the PostgreSQL prompt was entered using the proper command¹². The LoRa App Server database requires two additional extensions, which are not needed in LoRa Server database. Thus, a few extra steps were made in the case of LoRa App Server database creation.

```
-- create a role
create role role_name with login password 'dbpassword';

-- create database with the created role as the owner
create database database_name with owner role_name;

-- following is only for the LoRa App Server database
-- connect to the LoRa App Server database
\c as_database_name

-- create required extensions
create extension pg_trgm;
create extension hstore;

-- leave PostgreSQL
\q
```

Listing 4.1: The PostgreSQL role and database creation

Secondly, all components were configured manually through their configuration files. The packet forwarder's global-conf.json file changes are in the Listing 4.2. The MQTT broker address and port was edited in the LoRa Gateway Bridge configuration file¹³ (lora-gateway-bridge.toml). The changes in the LoRa Server configuration file (chirpstack-network-server.toml) are shown in Listing 4.3 and changes made in the LoRa App Server configuration file (chirpstack-application-server.toml) are listed in Listing 4.4. The jwt

⁹sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 1CE2AFD36DBCCA00

¹⁰sudo echo "deb https://artifacts.chirpstack.io/packages/3.x/deb stable main" | sudo tee /etc/apt/sources.list.d/chirpstack.list

¹¹sudo apt install chirpstack-gateway-bridge chirpstack-network-server chirpstack-application-server

¹²**first entry:** sudo -u postgres psql, **database:** psql -h localhost -U role_name -W database_name

¹³server="tcp://localhost:1883"

token was generated¹⁴ in the terminal before the LoRa App Server's configuration file was edited. Finally, after the configuration, all components were started¹⁵ as a service. In order to make certain of correct functionality, the logging file¹⁶ was investigated and occurred errors were solved.

```
"serv_port_down": 1700,
"serv_port_up": 1700,
"server_address": "localhost",
```

Listing 4.2: global-conf.json file changes.

```
dsn="postgres://ns_role_name:dbpassword@localhost/
ns_database_name?sslmode=disable" # database dsn
automigrate=true
name="EU_863_870" # EU LoRa freq
timezone="Local"
# MQTT broker address and port
server="tcp://localhost:1883"
```

Listing 4.3: chirpstack-network-server.toml file changes.

```
dsn="postgres://as_role_name:dbpassword@localhost/
as_database_name?sslmode=disable" # database dsn
jwt_secret= "generated_token"
# MQTT broker address and port
server="tcp://localhost:1883"
# internal API used by LoRa Server
public_host="localhost:8001"
```

Listing 4.4: chirpstack-application-server.toml file changes.

4.2.3 The network setup

All components were running at this point; thus I moved to the last step, which was the gateway and end-nodes registration. The LoRa App Server Web interface was used for that matter. In order to open the Web interface, the IP address with the appropriate port was entered into the browser. Login was accomplished by entering the default username and password¹⁷, which were changed immediately after the first login. The first step in the network setup was creating a network server. It was created by choosing the Network-server tab, followed by an add button. There, in the General configuration, the name, the server (localhost:8000), and enable gateway discovery were filled.

After that, an organization was created, where the organization name and display name were filled, and the "Organization can have gateways" option was checked. Next, a service profile, device profile and a gateway were created.

¹⁴openssl rand -base64 32

¹⁵sudo systemctl start <service_name>

¹⁶tail -f /var/log/syslog

¹⁷username: admin, password: admin

in Listing 4.5. The gateway configuration was on its end and ready to use.

```
function Encode(fPort, obj, variables) {
  // json to string conversion
  var strmsg = JSON.stringify(obj);
  // bytes array
  var bytes = []; // char codes

  // filling the bytes array
  for (var i = 0; i < strmsg.length; ++i) {
    var code = strmsg.charCodeAt(i);
    bytes = bytes.concat([code]);
  }
  return bytes;
}
```

Listing 4.5: Encode function example.

The test end-node implementation is inspired by the code from brady-aiello repository¹⁸. The crucial parts of the end-node implementation are presented in Listing 4.6.

```
// LORA SETUP
const float EU_channels[8] = {868.1, 868.3, 868.5, 867.1, 867.3,
  867.5, 867.7, 867.9}; // EU channels
char buffer[256]; //buffer of 256 bits
lora.setId(DEV_ADDR, DEV_EUI, APP_EUI); // sets Ids
lora.setKey(NWK_S_KEY, APP_S_KEY, NULL); // sets keys
lora.setDeciveMode(LWABP); // sets ABP mode
lora.setDataRate(DR0, EU868); // sets EU data rate
lora.setPower(MAX_EIRP_NDX_EU); // sends at+power=
  MAX_EIRP_NDX_EU
setChannelsFreq(EU_channels); // loop of lora.setChannel

// LORA COMMUNICATION
lora.transferPacket("UpAck!", 10); // uplink
lora.receivePacket(buffer, 256, &rssi); // downlink receive
SerialUSB.println(rssi); // communication's RSSI
// output received data
for(unsigned char i = 0; i < length; i ++){
  {
    SerialUSB.print(buffer[i], HEX); // hex
    SerialUSB.print(buffer[i]); // ascii
  }
}
```

Listing 4.6: Key code parts in end-device implementation.

The end-device implementation, presented in Reference [1], utilizes the I-NUCLEO-LRWAN1 library¹⁹, which has a similar code structure. The communication of the final end-device was tested. The uplink was performed without malfunction, but downlink communication reported minor issues. The device did not receive all configuration files that were sent. I believe the

¹⁸https://github.com/brady-aiello/Seeeduino_LoRaWAN_for_hybrid_gateways/tree/master/Seeeduino-LoRaWAN-ABP

¹⁹<https://github.com/stm32duino/I-NUCLEO-LRWAN1>

4. Communication design

solution to this malfunction lies in the LoRaWAN™ configuration, particularly in the receiving window duration, which needs to be changed. The example of communication is depicted in Figure 4.7. The entire system structure is shown in Figure 4.8, where the gateway is on the left side, the test end-device is in the middle and the final end-device is on the right side.

```
12:59:26.492 -> Online ack sent
12:59:27.642 -> 49 65 6C 6C 6F 2C 20 57 6F 72 6C 64 on port 1
12:59:47.430 -> Online ack sent
12:59:12.218 -> Online ack sent
12:59:36.940 -> Online ack sent
13:00:01.707 -> Online ack sent
13:00:26.448 -> Online ack sent
13:00:51.200 -> Online ack sent
13:01:15.982 -> Online ack sent
13:01:40.732 -> Online ack sent
13:02:05.463 -> Online ack sent
13:02:30.262 -> Online ack sent
13:02:31.449 -> 7B 22 64 75 74 79 43 79 63 6C 65 22 3A 30 2E 35 2C 22 66 72 65 71 75 65 6E 63 79 22 3A 31 30 30 2C 22 6D 6F 64 75 6C 61 74 69 6F 6E 54 79 70 65 22 3A 22 70 77 6D 22 7D on port 1
13:02:51.390 -> Online ack sent
```

Figure 4.7: Final end-device communication.



Figure 4.8: System structure.

Chapter 5

Configuration App

This chapter provides two possible app prototypes, which utilize MQTT protocol in order to communicate with end-nodes through the gateway (Chapter 4). In other words, they are able to connect to the gateway and send downlink messages with a configuration data required for system configuration. Both apps implement a MQTT client, but using different MQTT libraries. The first prototype is a web application. Second is a cross-platform mobile application based on Ionic and Angular framework. Prototypes will be further introduced in the following sections.

5.1 The prototype of web application

The web application prototype is written in HTML, CSS, and JavaScript. On top of these languages, the application utilizes the Eclipse Paho JavaScript Client¹, which is an open-source MQTT browser-based client library. This client library uses websockets for connecting to the MQTT broker. That is why the MQTT broker on the gateway had to support websockets in the first place. This prototype is inspired by the steves-internet-guide website², where Paho JavaScript library is nicely introduced with real-life examples.

5.1.1 The application architecture

The proposed web application consists of four source files: `index.html`, `styles.css`, `vender.js` and `app.js`. The `index.html` wraps the whole HTML code required in the application. The `styles.css` file preserves the additional application design, which was not satisfactory in the bootstrap library. The bootstrap is an open-source CSS framework, containing the design of components such as typography, forms, buttons just to name a few. It significantly speeds up the work; thus it was used as the primary design template also in this web application. The two remaining files hold the entire website logic. First is the `vender.js` file. This file is mainly responsible for rendering all required DOM objects to the JavaScript. On top of that, the `vender.js` includes the function, which controls the visibility of certain objects. It

¹<https://github.com/eclipse/paho.mqtt.javascript>

²<http://www.steves-internet-guide.com/mqtt-websockets/>

will be further discussed in section 5.1.2. Finally, the last file contains the main intelligence of the application. The `app.js` implements the MQTT over websockets communication; thus it connects with the MQTT broker on the gateway, evaluates the user input, and creates the MQTT message based on the data input. Moreover, it interactively outputs the current status of the MQTT communication.

■ 5.1.2 The application design

The website design is divided into five sections as it is shown in Figure 5.1. The first section is responsible for the connection. A user inputs the gateway's IP address, the appropriate port number, and in case of secured MQTT communication, also username and password. On top of that, the user can decide whether the communication is led as a clean session or not. When a clean session is set to one, that means the checked option in the application, the MQTT broker does not store any data about the MQTT client. However, the clean session set to zero (unchecked Clean Session option in the application), the MQTT broker stores information such as subscribed topics or messages with certain QoS and topic, which the client subscribed to.

The second section is used for testing the topic subscription. The user inserts the name of the topic, which is intended to be subscribed, and QoS. Then, upon clicking the subscribe button, the new topic is subscribed.

The third section is for sending a string message to a specific topic. In other words, publishing to a topic. To publish the message, the user needs to insert the message payload (a string in this case), QoS, and topic, the message is published to. Before sending the string payload, the user can decide whether the message will contain a retain flag or not. If the retain flag is set to zero, MQTT broker discards the message if no subscriber is in the topic. On the other hand, if the retain flag is set to one (checked Retain Message option), MQTT broker will store the last message in the topic. This may become handy if an end device sends data just a few times a day. With a retain flag equal to zero, any new subscriber would have to wait for a new publish in order to get the current status. However, with the retain flag set to one, a new subscriber does not have to wait for a new publish, because it will receive the last retained message from the broker.

The fourth section is responsible for publishing the configuration data. This data is meant to be sent to the end-nodes. End-nodes process this data and reconfigure itself. The fourth section has a modulation dropdown selection. A form parameters are updated based on the selected modulation type. Whenever modulation type changes, the form parameters updates. When the publish button is pressed, the data are rendered and inserted into a JSON message. The JSON message is sent to the MQTT broker, where it is translated by the encode function, presented in the Listing 4.5 of Chapter 4.

The last section is placed on the bottom of the website, and it informs the user about the MQTT communication status. It outputs connection status, new subscriptions and message payloads.

Figure 5.1: The web application prototype design.

5.2 The prototype of mobile application

The mobile application prototype is built in the combination of Ionic and Angular framework. This approach was chosen, because of its robustness, great documentation, and its ability to build cross-platform native mobile applications with minor code adjustments for the most part. Moreover, they are fairly new, fast-growing frameworks, which are supported by big companies like Google. This framework combination might play a significant role in future app development. The code itself is written in TypeScript. Where TypeScript is a super-set of JavaScript, which enables to write more robust code.

5.2.1 The application architecture

The application utilizes the object-oriented architecture. The core of the application is designed to be divided into page tabs, which should help the scalability in the future. The configuration page, which will also be the main topic of this section, is shown in Figure 5.2. I believe, as the project continues, more functionality will be needed, and new tabs will be implemented. However, focus on the configuration page for now. The configuration page contains a modulations component, where all the prototype modulation forms are located. The modulations component is responsible for sending the user input to the configuration page, where it is processed. This design is easily scalable in case of new modulation is required. So far, the application implements three prototypes: PWM, SIN and OOK. Each prototype contains its own parameters, which will be configured. Another key part of the page is the `config-message.service.ts`, which is responsible for restructuring the configuration payload. The rest of the logic is implemented in the `config.page.ts`, where MQTT client is created and controlled.

5.2.2 The application design

The mobile application follows a similar design concept as the web application introduced earlier. The pages, carrying various functionality, are ordered into tabs. This practice is usual in mobile applications because it allows us to clearly organize the functionality around the app. The page of main interest is the configuration page, whose architecture was discussed in the previous section 5.2.1.

This page is divided into three sections. The first section is the header and it only plays the design role. The second section is responsible for the connection between the app and MQTT broker. The last section contains the configuration variable form, which updates based on the modulation type selection. The same practice was utilized in the first web prototype. There are three modulation types implemented in the application so far: PWM, SIN and OOK. Each modulation form contains "Number of nodes" input field. The vision behind this field is, that the app will create original configuration data for every single end device. Thus, no configuration duplicates are possible, and it is positive, that each node will transmit its own signal, distinguishable for the receiver.

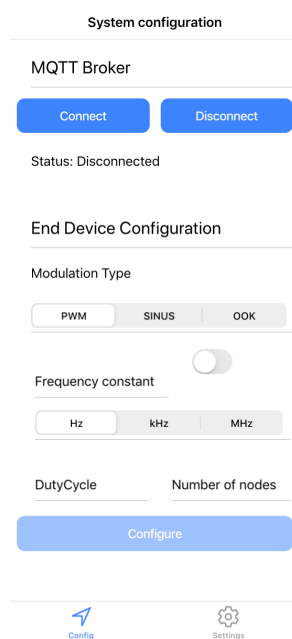


Figure 5.2: The mobile application prototype design.

5.3 Network functionality test

An end-to-end test was performed in order to confirm the correct functionality of the proposed system. The vision of this project is to configure the system through the mobile application. Therefore, the following tests were performed only on the second prototype. The communication was traced in the gateway

log files and on the end device.

The first test case verified the connection procedure. The connect button was clicked, the status of the connection changed from "Disconnected" to "Connecting". The gateway's IP address and port appeared in the browser console and connection status eventually updated to "Connected". A new client connection arose in the gateway log file. Then, the connection was interrupted by clicking the Disconnect button. The status changed to "Disconnected".

The next test case was focused on the data validators. If the inserted data are invalid, the "Configure" button will fade, and it can not be clicked. For instance, the duty cycle parameter acquires values between zero and one. Thus, if 1.2 is inserted the "Configure" button will fade and the input field will be marked as invalid. All validators passed the testing values.

The last test case was aimed at data transmission. Different combinations of configuration data were chosen, sent to the end device and tracked along the way. The full list of the tested combinations is shown in Table 5.1. Every sent message was verified in the gateway log file and then in the end device.

Modulation	Parameters			Expected
PWM	Freq.	Duty cycle	Num. of nodes	
	100	0.5	1	sent ✓
	1000	1.2	50	not sent ✓
	0	0.7	1	not sent ✓
	100 KHz	0.5	50	sent ✓
	10 MHz	0.5	50	sent ✓
	variable	0.2	1	sent ✓
SIN	Freq.	Amplitude	Phase shift	
	1000	2	180	sent ✓
	10 kHz	3	0	sent ✓
	100	1	370	not sent ✓
	variable	2	120	sent ✓
	0	2	270	not sent ✓
OOK	Freq.	Log1	Log0	
	100 MHz	3	0	sent ✓
	2 kHz	1.5	-1.5	sent ✓
	100	5	0	not sent ✓
	0	3	1	not sent ✓
	variable	1.5	-1.5	sent ✓

Table 5.1: Combinations of test data.

Chapter 6

Conclusions

The goals of this work were successfully accomplished. Firstly, the VLP techniques and algorithms were closely investigated and then described at the beginning of the thesis. The research significantly helped with designing the system, which was proposed later in this work. The system supposed to enable a configuration of end-devices in the VLC network. Various communication approaches were considered and studied in order to pick the most favorable for this project. Eventually, the decision has been made, and the proposed system utilizes the LoRaWAN™ and MQTT communication protocol. These protocols were chosen, because LoRaWAN™ uses the license-free sub-gigahertz radio frequency bands and it is a powerful LPWAN. The MQTT protocol was chosen because of its light-weight, which is very important in IoT. Both protocols and other technologies used in the system were studied and described within Chapter 3.

Secondly, the proposed system was implemented. The system structure was formed by the gateway (central unit), end-devices and mobile application. The gateway software is hosted on a Raspberry Pi 3, which is equipped with the PRI 2 Bridge RHF4T002 and the Gateway module RHF0M301–868. All gateway components such as packet forwarder, LoRa Gateway Bridge, and mosquitto MQTT Broker are implemented on the Raspberry Pi 3 alongside the LoRa Server and LoRa App Server. Each component was properly configured and the network was set up. Verification of the correct functionality was checked by uplink/downlink communication between the end-device and the LoRa App Server.

The last piece of the system is the mobile application. There were two prototypes proposed. The first prototype was implemented as a web application, which was used mainly for testing purposes. Moreover, this prototype helped to understand the more advanced MQTT Client library, which was used in the mobile application. The mobile prototype was implemented with the help of Ionic and Angular framework. These frameworks enable to build native cross-platform mobile apps with a robust codebase. Eventually, the mobile application was used in the final end-to-end test that verified the full functionality of the system. Configuration messages were sent to the end-device through the gateway, and it was successfully received and processed in the end-device. The specification of the entire test is available

in section 5.3 and the example of communication is shown in Figure 6.1.

Even though the system was fully functional, I believe that the potential of this project is far more promising. The vision is to design an indoor positioning system, that would be able to locate the target in real-time. There is still a huge amount of work to be done and obstacles to overcome in order to achieve such objective. However, I believe, employing the proposed system, with a few additional features, could be a great start to do so.

The next step would be definitely implementing class B end-devices, because class A devices have large downlink latency. This latency could result in significant unwanted configuration delay. Then, I would try to utilize the mobile phone camera in order to capture visible light signals, transmitted from end-devices to the mobile phone. There comes the tricky part though. There must be a powerful algorithm implemented on the phone. The algorithm has to localize the target fast and precisely. The speed is crucial because the algorithm must evaluate the target's position in real-time. Otherwise, it would be poor navigation. There are numerous techniques proposed in the literature to achieve the correct target localization, but many are too slow to use in the real-time positioning. However, I believe, employing an image processing method (e.g. Optical flow) with a probabilistic technique (e.g. Bayesian forecast) as it is proposed in the Reference [34], might be the way to solve this complicated problem.

```
+MSG: Start
+MSG: PORT: 8; RX: "7B22647574754379636C65223A302E352C226672657175656E6379223A3130303030302C226D6F64756C6174696FEE54797065223A2270774D227D"
+MSG: RXWIN1, RSSI -43, SNR 7.5
+MSG: Done
Length is: 59
RSSI is: -43
Data is: {"dutyCycle":0.5,"frequency":100000,"modulationType":"pwm"}
+MSG: Start
+MSG: Done
+MSG: Start
+MSG: Done
+MSG: Start
+MSG: PORT: 8; RX: "7B22616D70223A302C226672657175656E6379223A747275652C226D6F64756C6174696FEE54797065223A2273696E222C227069617365223A3132307D"
+MSG: RXWIN1, RSSI -38, SNR 6.3
+MSG: Done
Length is: 61
RSSI is: -38
Data is: {"amp":12,"frequency":true,"modulationType":"sin","phase":120}
+MSG: Start
+MSG: Done
+MSG: Start
+MSG: Done
+MSG: Start
+MSG: PORT: 8; RX: "7B226672657175656E6379223A31303030303030302C226C6F674FEE65223A332C226C6F675A65726F223A302C226D6F64756C6174696FEE54797065223A226F6E8227D"
+MSG: RXWIN1, RSSI -38, SNR 6.8
+MSG: Done
Length is: 69
RSSI is: -38
Data is: {"frequency":100000000,"logOne":3,"logZero":0,"modulationType":"ook"}
+MSG: Start
+MSG: Done
+MSG: Start
+MSG: Done
+MSG: Start
+MSG: PORT: 8; RX: "7B22616D70223A302C226672657175656E6379223A3130302C226D6F64756C6174696FEE54797065223A2273696E222C227069617365223A32307D"
+MSG: RXWIN1, RSSI -40, SNR 7.0
+MSG: Done
Length is: 60
RSSI is: -40
Data is: {"amp":2,"frequency":100,"modulationType":"sin","phase":270}
+MSG: Start
+MSG: Done
```

Figure 6.1: Example of received messages in end-device.



Appendices



Appendix A

List of Abbreviations

ABP	Activation by Personalization.
ADR	Adaptive Data Rate.
AoA	Angle of Arrival.
APD	avalanche photodiode.
API	Application Programming Interface.
CCD	Charge-Coupled Device.
CSS	chirp spread spectrum.
CSS	Cascading Style Sheets.
DOM	Document Object Model.
FDA	Fisher discriminant analysis.
GPS	Global Positioning System.
gRPC	Remote Procedure Calls.
HTML	Hypertext Markup Language.
HTTP	Hypertext Transfer Protocol.
IMU	inertial measurement unit.
IoT	Internet of Things.
IPs	indoor positioning systems.
JSON	JavaScript Object Notation.
kNN	k-Nearest Neighbours.
LBS	location-based services.
LED	light emitting diode.
LoRa™	Long Range.
LoRaWAN™	Long Range Wide Area Network.

A. List of Abbreviations

LOS	line of sight.
LSE	least square estimator.
MAC	Media Access Control.
MLEM	Multiple LEDs Estimation Model.
MQTT	Message Queue Telemetry Transport.
MSB	most significant bits.
OASIS	Organization for the Advancement of Structured Information Standards.
OS	operation system.
OTAA	Over-the-Air Activation.
PDM	pulse duration multiplexing.
PIN	positive-intrinsic-negative.
PWM	pulse-width modulation.
RF	radio frequency.
RSF	receive signal feature.
RSS	received signal strength.
Rx	receiver.
SHS	Step and Heading systems.
SIFT	scale-invariant feature transform.
SPA	Single-Page-Application.
SPI	Serial Peripheral Interface.
SSH	Secure Shell.
SSL	Secure Socket Layer.
TCP/IP	Transmission Control Protocol/Internet Protocol.
TDoA	time difference of arrival.
TLS	Transport Layer Security.
ToA	time of arrival.
Tx	transmitter.
UDP	User Datagram Protocol.
USB	Universal Serial Bus.
VAP	visible light access point.
VLC	visible light communication.
VLP	visible light positioning.

Appendix B

Bibliography

- [1] Štěpán Bosák, “Hw for indoor visible light positioning testbed,” Bachelor’s Thesis, CTU FEE, Technická 2, 5 2020.
- [2] E. R. Lin, “Lora protocol. evaluations, limitations and practical test,” Master’s Thesis, Universitat Politècnica de Catalunya, 2016.
- [3] O. Popoola, S. Sinanović, W. Popoola, and R. Ramirez-Iniguez, “Optical boundaries for led-based indoor positioning system,” *Computation*, vol. 7, no. 1, p. 7, Jan 2019. [Online]. Available: <http://dx.doi.org/10.3390/computation7010007>
- [4] S. Jung, S. Hann, and C. Park, “Tdoa-based optical wireless indoor localization using led ceiling lamps,” *IEEE Transactions on Consumer Electronics*, vol. 57, no. 4, pp. 1592–1597, 2011.
- [5] C. Neha, A. Luis Nero, and Z. Ghassemlooy, “Current Trends on Visible Light Positioning Techniques,” in *The 2nd West Asian Colloquium on Optical Wireless Communications (WACOWC2019)*, Tehran, Iran, Apr. 2019. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-02135266>
- [6] A. Correa Vila, M. Barceló, A. Morell, and J. Lopez Vicario, “A review of pedestrian indoor positioning systems for mass market applications,” *Sensors (Switzerland)*, vol. 17, 08 2017.
- [7] C. Xie, G. Weipeng, X. Wu, L. Fang, and Y. Cai, “The led-id detection and recognition method based on visible light positioning using proximity method,” *IEEE Photonics Journal*, vol. PP, pp. 1–1, 02 2018.
- [8] N. A. Azmi, S. Samsul, Y. Yamada, M. F. Mohd Yakub, M. I. Mohd Ismail, and R. A. Dziauddin, “A survey of localization using rssi and tdoa techniques in wireless sensor network: System architecture,” in *2018 2nd International Conference on Telematics and Future Generation Networks (TAFGEN)*, 2018, pp. 131–136.
- [9] A. R. Kulaib, R. M. Shubair, M. A. Al-Qutayri, and J. W. P. Ng, “An overview of localization techniques for wireless sensor networks,” in *2011 International Conference on Innovations in Information Technology*, 2011, pp. 167–172.

- [10] L. Jaulin, “5 - instantaneous localization,” in *Mobile Robotics*, L. Jaulin, Ed. Elsevier, 2015, pp. 171 – 196. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/B978178548048550005X>
- [11] Wikipedia contributors, “Multilateration — Wikipedia, the free encyclopedia,” 2020, [Online; accessed 27-March-2020]. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=Multilateration&oldid=947549616>
- [12] M. Horiba, E. Okamoto, T. Shinohara, and K. Matsumura, “An improved nlos detection scheme using stochastic characteristics for indoor localization,” vol. 2015, 03 2015, pp. 478–482.
- [13] H. Lv, L. Feng, A. Yang, P. Guo, H. Huang, and S. Chen, “High accuracy vlc indoor positioning system with differential detection,” *IEEE Photonics Journal*, vol. 9, no. 3, pp. 1–13, 2017.
- [14] W. Xu, J. Wang, H. Shen, H. Zhang, and X. You, “Indoor positioning for multiphotodiode device using visible-light communications,” *IEEE Photonics Journal*, vol. 8, no. 1, pp. 1–11, 2016.
- [15] Y. S. Eroglu, I. Guvenc, N. Pala, and M. Yuksel, “Aoa-based localization and tracking in multi-element vlc systems,” in *2015 IEEE 16th Annual Wireless and Microwave Technology Conference (WAMICON)*, 2015, pp. 1–5.
- [16] J. Machaj, P. Brida, and R. Piché, “Rank based fingerprinting algorithm for indoor positioning,” in *2011 International Conference on Indoor Positioning and Indoor Navigation*, 2011, pp. 1–6.
- [17] T. Stockx, B. Hecht, and J. Schöning, “Subwayps: Towards enabling smartphone positioning in underground public transportation systems,” 11 2014.
- [18] O. R. Popoola, F. B. Ogunkoya, W. O. Popoola, R. Ramirez-Iniguez, and S. Sinanović, “Indoor localization based on multiple leds position estimation,” in *2016 IEEE 17th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2016, pp. 1–6.
- [19] M. Xu, W. Xia, Z. Jia, Y. Zhu, and L. Shen, “A vlc-based 3-d indoor positioning system using fingerprinting and k-nearest neighbor,” in *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, 2017, pp. 1–5.
- [20] S. Jung, S. Hann, and C. Park, “Tdoa-based optical wireless indoor localization using led ceiling lamps,” *IEEE Transactions on Consumer Electronics*, vol. 57, no. 4, pp. 1592–1597, 2011.
- [21] A. Hajihoseini, A. Dargahi, and A. Ghorashi, “3d indoor localization using visible light communications,” *IJIREEICE*, vol. 4, pp. 119–122, 07 2016.

- [22] P. K. Aswin, P. Shyama, and L. B. Das, "Indoor localization using visible light communication and image processing," in *2018 IEEE International Conference on Consumer Electronics (ICCE)*, 2018, pp. 1–6.
- [23] B. Zhang, M. Zhang, Z. Ghassemlooy, D. Han, and P. Yu, "A visible light positioning system with a novel positioning algorithm and two leds," in *2019 24th OptoElectronics and Communications Conference (OECC) and 2019 International Conference on Photonics in Switching and Computing (PSC)*, 2019, pp. 1–3.
- [24] Y. U. Lee and M. Kavehrad, "Long-range indoor hybrid localization system design with visible light communications and wireless network," in *2012 IEEE Photonics Society Summer Topical Meeting Series*, 2012, pp. 82–83.
- [25] "Lora crash course by thomas telkamp," USA, 2016. [Online]. Available: https://www.youtube.com/watch?time_continue=68&v=T3dGLqZrjIQ&feature=emb_logo
- [26] A. Augustin, J. Yi, T. H. Clausen, and W. Townsley, "A study of lora: Long range & low power networks for the internet of things," *Sensors*, vol. 16, p. 1466, 10 2016.
- [27] A. Lavric and V. Popa, "Internet of things and loraTM low-power wide-area networks: A survey," in *2017 International Symposium on Signals, Circuits and Systems (ISSCS)*, 2017, pp. 1–5.
- [28] L. Angrisani, P. Arpaia, F. Bonavolontà, M. Conti, and A. Liccardo, "Lora protocol performance assessment in critical noise conditions," in *2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI)*, 2017, pp. 1–5.
- [29] L. Vangelista, A. Zanella, and M. Zorzi, "Long-range iot technologies: The dawn of loraTM ," in *Future Access Enablers for Ubiquitous and Intelligent Infrastructures*, V. Atanasovski and A. Leon-Garcia, Eds. Cham: Springer International Publishing, 2015, pp. 51–58.
- [30] D. Thangavel, X. Ma, A. Valera, H. Tan, and C. K. Tan, "Performance evaluation of mqtt and coap via a common middleware," in *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2014, pp. 1–6.
- [31] "Foundational iot messaging protocol, mqtt, becomes international oasis standard," USA, 2020. [Online]. Available: <https://www.oasis-open.org/news/pr/foundational-iot-messaging-protocol-mqtt-becomes-international-oasis-standard>
- [32] K. Grgić, I. Špeh, and I. Hedi, "A web-based iot solution for monitoring data using mqtt protocol," in *2016 International Conference on Smart Systems and Technologies (SST)*, 2016, pp. 249–253.

- [33] S. Cope, “Understanding the mqtt protocol packet structure,” 2011-2020. [Online]. Available: <http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/>
- [34] W. Guan, X. Chen, M. Huang, Z. Liu, Y. Wu, and Y. Chen, “High-speed robust dynamic positioning and tracking method based on visual visible light communication using optical flow detection and bayesian forecast,” *IEEE Photonics Journal*, vol. 10, no. 3, pp. 1–22, 2018.