Simple RSRP Fingerprint Collection Setup and Indoor Positioning in 5G

SOFI FLINK & ANNIE TALLUND

MASTER´S THESIS

DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY

FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

EXAMENSARBETE
Elektro- och informationsteknik

Simple RSRP Fingerprint Collection Setup and Indoor Positioning in 5G

Enkelt Automatiserat System För Insamling av RSRP Fingeravtryck samt Inomhuspositionering i 5G

Sofi Flink, Annie Tallund

Simple RSRP Fingerprint Collection Setup and Indoor Positioning in 5G

Sofi Flink
bmp13sfl@student.lu.se

Annie Tallund
an0284ta-s@student.lu.se

May 23, 2022

Master's thesis work carried out at Ericsson AB.

Supervisors: Jonas S Andersson, jonas.s.andersson@ericsson.com
Niclas Palm, niclas.palm@ericsson.com
Fredrik Tufvesson, fredrik.tufvesson@eit.lth.se

Examiner: Fredrik Rusek, fredrik.rusek@eit.lth.se

Abstract

Solving for a way to accurately predict the position of a user equipment is crucial in an array of applications within 5G new radio. One approach is to form unique identifiers, also known as fingerprints, and map them to positional data in an area. Previous research suggests that radio signal received strength is a promising indicator to use for the fingerprint, in order to achieve accurate indoor positioning.

The thesis proposes an automated data collection setup. It is capable of collecting radio signal information through modem logs. Additionally, it uses a navigation system which offers a way to map modem log data to a ground-truth coordinate. A data preprocessing pipeline is presented as to turn the raw modem logs into fingerprints.

The outcome is a data set for supervised learning. As a proof-of-concept the data is classified using two machine learning algorithms: naive bayes and k nearest neighbor. In the latter, a F1 (macro) score of 98% is obtained in predicting the user equipment position.

Keywords: 5G, indoor positioning, radio fingerprint, k nearest neighbor, naive bayes, automated data collection

Acknowledgements

We would like to extend our sincere thanks to Ericsson AB, for providing all of the necessary resources for this thesis. We would also want to express our heartfelt gratitude to our Ericsson supervisors Jonas S. Andersson and Niclas Palm for always taking time to share their knowledge and experience with us. We are grateful to them for being truly understanding and empathetic with any adversities we faced. They have demonstrated great confidence and commitment to our work.

We are truly indebted to our supervisor Fredrik Tufvesson at LTH, without him we would never have been able to complete this dissertation together.

Our dissertation would not have been completed without the help and dedication of Fredrik Brumme at Ericsson. He was always there to help with his extensive knowledge of various areas, whether it was on weekends, nights, or during daytime.

We would like to thank following colleagues at Ericsson for their availability and willingness to address our questions:

- Robert Marklund
- Ali Nader
- Ulf Eklund
- Martin Klitte
- Dino Pjanic
- Johan Berglund and the IT support team

Furthermore, we would like to thank the following people for their assistance and involvement:

- Our partners Andreas Hansson and Elias Rudberg, for their emotional support and valuable feedback
- Hassan Mahdi and Rosa Lindberg, for letting us set up the environment for collecting the data
- Bricks Eatery for always providing delicious and nutritious food at the site
- Workers installing the base station and workers building enclosure for robot
- Parents of Sofi, Peter Flink and Tarja Voipää, for the sacrifices they made in order to provide safety and comfort in trying times
- Grandmother of Sofi, Siri Charlez, for her encouragements
- Mother of Annie, Gunilla Tallund, for unconditional love and support

Contents

List of Abbreviations

IoT	Internet of Things
NR	New Radio
UE	user equipment
SS-RSRP	Synchronization Signal Reference Signal Receive Power
LTE	Long-Term Evolution
RSSI	Received Signal Strength Indication
MI	MobileInsight
SSB	Synchronization Signal Block
HA	HomeAssistant
YAML	YAML Ain't Markup Language
MQTT	Message Queuing Telemetry Transport
LiDAR	Light Detection and Ranging
CAD	Computer-Aided Design
API	Application Programming Interface
k-NN	k-Nearest Neighbor
SA	standalone
5GC	5G Core
NSA	non-standalone
EPC	Evolved Packet Core

MIMO	Multiple Input/Multiple Output
SU-MIMO	Single User-MIMO
RF	Radio Frequency
RSRQ	Reference Signal Received Quality
QC	Qualcomm
UI	user interface
RRC	Radio Resource Control
OTA	Over the Air
ML	machine learning
AI	artificial intelligence
OVA	One-vs-All
AVA	All-vs-All
GIGO	Garbage in, Garbage out
HTML	Hypertext Markup Language
MSE	mean square error
NaN	Not a Number
RMSE	root mean square error
PMF	Probability Mass Function
DNF	Probability Density Function
ROS	Robot Operating System
SS-RSRP	Synchronization Signal-Reference Signal Received Power
QXDM	QUALCOMM eXtensible Diagnostic Monitor
QCAT	Qualcomm Commercial Analysis Toolkit
CDF	cumulative distribution function

Chapter 1
Introduction

Positional awareness of mobile network devices is important in a number of different fields like Internet of Things (IoT), virtual and augmented reality, and much more. In recent years, the array of applications within fields requiring accurate indoor positioning has grown, especially with the commercial introduction of New Radio (NR) 5G.

The challenge of performing a task using accurate indoor position in user equipment (UE) - without GPS technology - is a relatively new area of research. By using the meta-data in the radio signals in a machine learning (ML) context, it is possible to predict the position of a UE.[12]

One of tasks at hand is therefore to design a radio fingerprint containing relevant radio signal meta-data. At the transmitter and receivers of the new radio, numerous antenna elements are operating at the same time. This means there is more information attained at each position compared to earlier generations of radio, potentially improving the accuracy of the models utilized.

As an initial hypothesis, it is anticipated that utilizing simply the strength of the Synchronization Signal-Reference Signal Received Power (SS-RSRP) values from the different beams created by the transmitter, it should be feasible to train a more accurate model than existing models for Long-Term Evolution (LTE) technology. Simulations, produced by the Ericsson supervisors, show that SS-RSRP values deviate with a linear relationship when observing from one designated coordinate, as seen in Figure 1.1.

Previous work in articles about radio fingerprinting have demonstrated that Received Signal Strength Indication (RSSI) - superset of SS-RSRP - signals are a useful values for identifying UE's.[12] [4] [18] It is also suggested that implementing a machine learning approach may be less expensive than implementing camera-based positioning. This provides motivation for a data-driven setup.[4] According to the work in [4] and [18], models with no learning process are sufficient to solve the positioning task accurately.

The open-source project MobileInsight (MI) has developed a software tool for extracting various types of network information on a UE. The project is based on research from Purdue University, University of California and Los Angeles in 2014.[21] It seems promising for

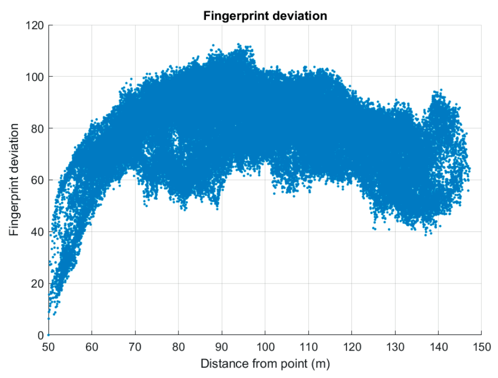


Figure 1.1: Radio fingerprint deviation measured from one selected coordinate in simulations.

further development and application for this thesis, and the project continues to expand its support regularly.

The advancement in creating a robust and inexpensive setup for data collection, data being transmission information made easily obtained for building a fingerprint, would assist in a lot of future research. This would enable more sophisticated research into 5G NR technology, resulting in greater technological advancement in the telecommunications industry.

Purpose and Aims

This thesis seeks to answer, among other questions, whether it is possible to improve the prediction accuracy of UE indoor position - using radio fingerprints with SS-RSRP in 5G NR - compared to previous attempts in this area of research.[19][12][4][18] The presence of multiple antennas on both the transmitter and receiver sides makes this a good starting hypothesis.

This necessitates the collection of data for the development of a machine learning model. Because this thesis implements supervised machine learning as a model, the data must come from a known location where positions can be checked for ground truth. The radio fingerprints are made up of 5G SS-RSRP values that are obtained during transmission.

The goal of the data extraction is to create an automation, capable of collecting data without human intervention. Rather than the currently available software for desktop computer systems, data should be extracted using tools and software on the UE itself. In order to evaluate the setup, it is necessary to look into how the distribution of the various Synchronization Signal Block (SSB) beams collected affects the signal received by the UE.

It is necessary to properly evaluate the machine learning model's accuracy, and how different UE's may affect the model's predictions. To accomplish this, the F1 Macro score and other well-established evaluation methods are used.

The primary goal for this project is to help pave the road for more sophisticated indoor positioning models, while also simplifying inexpensive NR 5G data collection. In this thesis, a setup is proposed as a proof of concept, along with some suggestions for potential future improvements.

Disposition

The following section, 1.1 "Technical Approach", describes the technical approach for the setup, as well as how each step is solved. It provides the overall framework used in terms of 5G signal emission, data collection and modelling.

The relevant theory to this project is described in chapter 2, "Background". This includes 5G NR and machine learning. The automation in chapter 3 presents the data collection setup in its entirety, with premise for decisions made along the way.

In chapter 4 the results are presented with respect to automation operability, data properties and model evaluation. The chapter is followed by a discussion of the decisions made throughout the running course of the project. The final chapter wraps up the conclusion for the thesis and potential future improvements.

1.1 Technical Approach

This chapter describes the general approach for solving each part of the project in order to ensure reproducibility. In chapter 2, the theory and evaluation of the described approach are elaborated upon.

1.1.1 5G Radio

For creating an ML model, fitted with SS-RSRP values, it is essential to have suitable data. For this purpose one 5G New Radio transmitter is needed, and one piece of user equipment, UE, capable of connecting to the 5G network. A location is necessary for installing the components and carrying out experiments for data collection. For this thesis, it was set up in the main entrance hall of the Ericsson building. The size of the operational area is $4.5x9$ meters and enclosed using a wooden frame.

The 5G radio transmitter AIR 5322, developed by Ericsson AB, is used. There may be several impediments that prevent the signals from reaching the antennas of the UE. As a result, the radio transmitter is positioned high above, with few disruptions facing the experimental area. It should be installed oriented towards the middle of the room, allowing some signal bouncing along the walls. The SSB beams are distinguished using their ID as explain in section 2.1.4.

Extracting Fingerprint Data

In order to match a UE position to the UE's information of the signals, it is necessary to choose suitable values in order to compose a radio fingerprint (explained in section 2.1.5). It is also imperative to find how to extract this information from the UE. For this project, the SS-RSRP values from the different SSB beams transmitted to the UE antennas are used.

For generating needed data through the modem, the UE must be connected to the mobile network alone, not using the WiFi connection. It needs some kind of application for the SS-RSRP extraction. The application needs root privileges in order to generate files and folders for data logs, and to access the connection to the modem. Hence, a rooted WNC MWH-5 prototype is used. In this thesis, installed on the UE is the Kivy Android application "Logger", described in section 3.7, MI with necessary dependencies, a compiled version of *diag_revealer* and a self-generated *.cfg* file.

1.1.2 Data Collection

In the matter of data collection, an automated procedure is ideal. Finding an easy and effective technique to acquire vast volumes of data - for machine learning research - is an important aspect for this thesis.

Robot

The automatic data gathering method is met by using a Xiaomi Roborock S5 robot vacuum cleaner with a HomeAssistant (HA) server running on a laptop. YAML Ain't Markup Language (YAML) scripts are updated within the HA server's virtual environment to control the robot's movement. A Python script is used for updating the YAML scripts, an example of this can be seen in section 3.2 on page 33.

To coordinate the setup, some form of communication between the robot and the UE is required. The UE must be alerted when the robot arrives at a coordinate, and the robot must be alerted when the UE completes data extraction at a coordinate. For communication, a light weight messaging protocol Message Queuing Telemetry Transport (MQTT) is used.

The robot need some sort of positional awareness of the real world, and the Xiaomi S5 robot vacuum cleaner possesses its own internal coordinate system. However, users cannot extract the robot's knowledge of its own position in the world. Fortunately, since the script's building blocks are more comparable to commands, a Python script is used to generate each "go to" command with coordinates. These coordinates are sent as MQTT messages to the broker server.

For mounting the UE on top of the robot, it has to avoid disrupting the Light Detection and Ranging (LiDAR), while also taking into account the pressure sensitivity of the sensor. In this thesis, a frame - 3D printed in Computer-Aided Design (CAD) - is used. The mount should disrupt the LiDAR as little as possible.

Data Storage

Since the purpose of the setup is to generate large volumes of data, storage is something to take into consideration. A cloud solution is proposed with OpenStack Object Storage, using swift Application Programming Interface (API). Object Storage is a data storage solution within OpenStack platform that employs clusters of standardized servers to store bytes of accessible data in a redundant, scaleable manner.

1.1.3 Model

For predicting the indoor positioning using collected data, a classification method is needed for comparing features between different coordinates. Also, since the scope of this thesis is quite extensive, the model needs to be simple to implement. For this purpose, the k-Nearest Neighbor (k-NN) is used as a model.

Equipment

The project demands a significant amount of processing power due to the volume of data required for the ML models and to evaluate performance. Three distinct machines are used to run the various scripts for this thesis. In addition, the plots in chapter 4 are generated using Matlab.

Preprocess Data

Some data may be contaminated with noise, while some data fields may be absent. Normalization and imputation are two approaches of dealing with invalid data. Since the model used in this project is k-NN, all data is normalized. For addressing missing values, this project employs k-NN imputation of the collected data.

Evaluation

The model must be evaluated before any conclusions can be drawn about its ability to forecast the indoor position for a radio fingerprint. A portion of the data set, 30 percent, is set aside for testing the model. This result is used during prediction to extract appropriate figures, graphs, and scores, making it possible assess the model.

Chapter 2

Background

This chapter describes relevant concepts as well as considerations made when approaching different problems. The sections describe 5G NR and some of its capabilities, as well as radio fingerprinting and its application in 5G NR. In addition, some basic theory on machine learning and its application within 5G indoor positioning are introduced.

2.1 5G New Radio

5G new radio is a set of standards that is to supersede the 4G and LTE network wireless communications standard.[3] A variety of novel techniques are being deployed to meet the specifications defined for 5G NR, some of which is detailed in this section.

2.1.1 Non-Standalone Architecture

For the purpose of implementing 5G, two architectures are used. One network deployment mode is known as standalone (SA). SA 5G network employs both the new 5G air interface, NR, and 5G Core (5GC). It provides users with an end-to-end 5G experience while also retaining the existing 4G/LTE network to assure service continuity between the two generations.

Non-standalone (NSA) 5G network, on the other hand, refers to having only 5G NR radios in place with Evolved Packet Core (EPC) as the core. 5G radios on these networks depend entirely on existing LTE network for all control functions and add-on services. The 5G NSA architecture operates under a master-secondary structure, where the 4G access node is the master and 5G access node is the secondary. This results in some limitations on the network, but allow for faster transitioning from 4G to 5G infrastructure.[37] A basic schematic describing the various architectures is shown below in figure 2.1 on page 18.

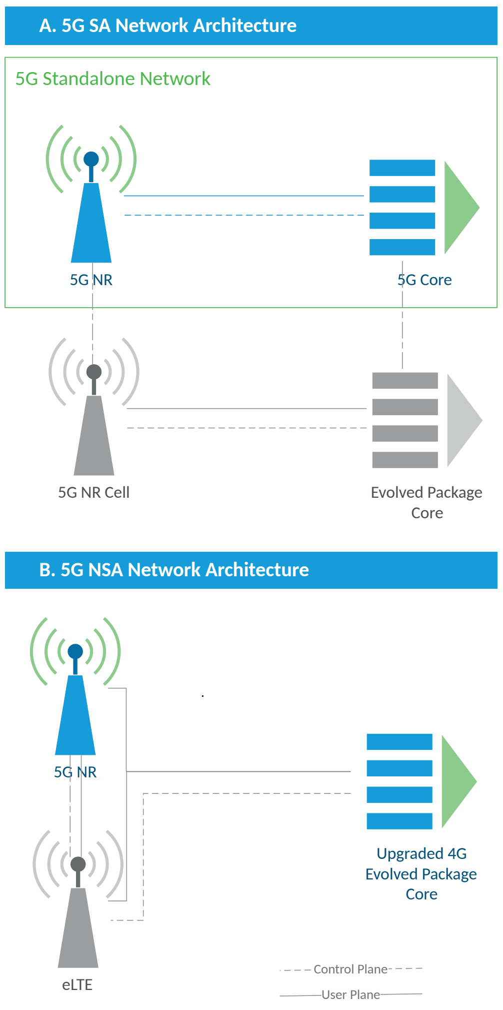


Figure 2.1: A simple diagram of 5G SA and NSA architecture.

2.1.2 Beamforming

In an analog context, beamforming is the ability to concentrate radio energy through a radio channel and direct it toward a specific receiver. It improves the strength of the received signal by altering the phase and amplitude of the transmitted signals. As a result, the signals are constructively added at the receiving UE, enhancing the coverage area and end-user throughput.

On the receiving end, beamforming is also used to collect the signal energy from a transmitter. These networks' antennas constantly adjust to their surroundings in order to manage high-speed up-link and down-link.

The state of the network system has a significant impact on which direction is optimal for transmitting and receiving energy. Corners can sometimes impede energy, and surfaces can be used to redirect the signal. Figure 2.2 on page 20 shows a simple example of beamforming. In figure 2.2.B, you can see how the building surface is being exploited to reflect the beam in a different direction to the UE.[7]

2.1.3 Single-user Multiple-input and Multiple-output

Multiple Input/Multiple Output (MIMO) is a technology that uses spatial multiplexing. It transmits multiple data streams using the same time and frequency resource and each data stream can then be beamformed. This increases throughput, and as such it is not necessary to have as much power in each stream. When the received signal quality is strong, it is better to use multiple less powerful streams. Therefore, MIMO is a key element in the NR technology due to the higher requirements to fulfill the 5G promise.

Single User-MIMO (SU-MIMO) is the ability to transmit one or multiple data streams, referred to as layers, to a single UE. The UE has several antennas for receiving the different layers, and the beams are directed toward the UE or bounce on different surfaces before reaching the antennas.[7] As a result, this technique has the potential to be a suitable foundation for developing a radio fingerprint, since more information is available through more layers.

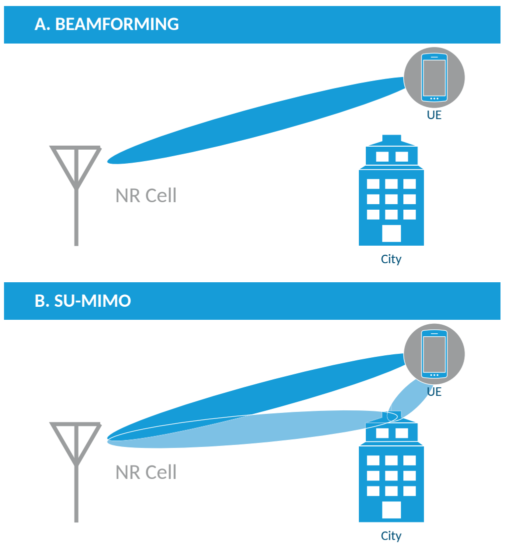


Figure 2.2: Beamforming and SU-MIMO examples.

2.1.4 Ericsson 5G Radio

It is possible to enable steerable high-gain beams using a rectangular antenna array, as shown in figure 2.3 on page 21. The more antenna elements there are, the higher the gain.

Steerability is achieved by individually controlling the amplitude and phase of the smaller sub-arrays of the antenna array. By applying two dedicated radio chains per sub-array (one per polarization) it enables control of the direction and other properties of the created antenna array beam.

Array gain is the gain achieved when all sub-array signals are constructively added in phase. The number of sub-arrays decides the possible size of the array gain. This gain can be directed in any trajectory path by switching phases in a certain pattern.[7] The result is a grid of radiating elements that are transmitted in different directions, also known as SSB beams. The main beams can be distinguished using an ID that is unique in each 5G radio, which are the numbers in the grid shown in 2.4. As a side effect, one observes side lobes for each of the elements.

The grid layout of radio AIR 5322 and corresponding beam IDs are illustrated in figure 2.4 on page 21. It has 24 SSB beams, radiating over 120° horizontally.

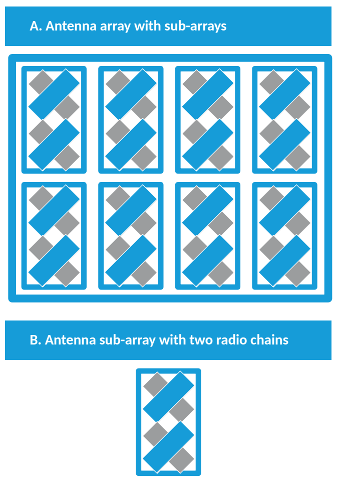
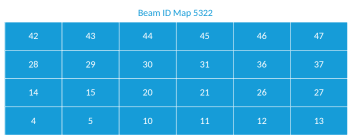

A. Antenna array with sub-arrays
B. Antenna sub-array with two radio chains

Figure 2.3: Antenna array where each sub-array is connected to two radio chains, normally one per polarization.



Beam ID Map 5322

42	43	44	45	46	47
28	29	30	31	36	37
14	15	20	21	26	27
4	5	10	11	12	13

Figure 2.4: Layout of the radio grid SSB IDs.

2.1.5 Radio Fingerprinting

This subsection describes the theory of radio fingerprinting, and how it is applied for training a model in 5G indoor positioning.

Definition

Radio fingerprinting is a method of identifying a UE or any other radio transmitter by a unique "fingerprint" that characterizes its signal transmission. A radio fingerprint makes it possible to identify a wireless device by its radio fingerprint transmission characteristics.

Radio fingerprinting refers to various physical layer classification approaches of radio signals. There are channel-specific ones, e.g. channel impulse response, and transmitter-specific ones that are independent of the channel, e.g. signal encoding. This dissertation focuses on transmitter specific features for the radio fingerprint.[5]

There are different ways to construct a radio fingerprint. Using transmission data as training input for a machine learning model, it is possible to predict a number of different qualities of an entity.[32]

Radio Signal Strength Indicator

In telecommunications, RSSI is a measurement of the power present in a received radio signal.[29] It is used internally in a wireless networking card. When radio energy falls below a specific level, the network card is "clear to transmit", and a packet with information is dispatched. When assessing the signal strength of a wireless network with a monitoring tool such as Wireshark, the end-user is able able to see a RSSI index. Additionally, each SSB beam results in an individual measured RSSI value.

SS-RSRP for Positioning

RSSI has previously been used in machine learning to achieve coarse-grained location estimates. [26] Because MIMO employs multiple antenna access points for both transmission and reception, it is possible to acquire additional fields of RSSI data. The hypothesis assumes that because each fingerprint has a larger likelihood of being unique, the result of using it in a ML context is a more accurate prediction of the UE location.

Based on the simulations described in chapter 1, the project's first target for RSSI data is to be able to collect and decode a subset of RSSI, namely SS-RSRP values. SS-RSRP reflects the average power received from a single reference signal. The range is typically around -44dBm to -140dBm. Using a variety of distinct RSSI information from the UE helps matching a specific radio fingerprint to its UE, and by extension its position. Due to the time constraints imposed by the scope of this thesis, the SS-RSRP values from the different beams form the fingerprints before introducing new variables.

Models presented in article [18], [4] and [12], illustrate how SS-RSRP values can be used to build fingerprints. A SS-RSRP fingerprint appears to be a reliable strategy when combined with previous simulations in the mentioned articles.

2.1.6 5G Message Format

Obtaining meta-data from 5G signals is a central concept in this thesis. The task is to choose a set of transmitted information, in form of a *message*, containing RSRP and Reference Signal Received Quality (RSRQ) values for the four strongest 5G SSB beams. The RSRP and RSRQ values are later used to construct the radio fingerprint. A *message* is a package of pre-determined information about a received signal. After a signal has been received in the antennas, a message can be thought of as a filter that sorts out relevant bytes, and packages them. The process is played out in the modem.

Logs

There is a risk of some confusion about the difference between two terms used frequently in this dissertation: logs and messages. A message is a single instance of information received by the modem from a single 5G radio. In other contexts, a message might carry a wide range of information, from a variety of layers. A message in this thesis represents SS-RSRP values about the four strongest SSB beams received by the UE, encoded in a byte stream. The format of the log type is the reason only four beams are recorded. The ground truth for the message is defined by Qualcomm (QC) - a company that produces wireless communication technologies.

A log, on the other hand, is a file that contains a collection of messages. The amount of messages varies depending on the message format and file size. In addition to the messages, the log also contains some additional meta data such as the creator of the log file format. Summarized, a log holds several messages.

The type of log used for the SS-RSRP values is known as *NR ML1 Serving Cell Beam Management*, and contains information about the management layer transmitted in the cell (meaning, radio). It is referred to as the *log type* throughout this dissertation.

2.1.7 Message Extraction

Two approaches for message extraction are worth considering. One constraint is the chipset in the 5G prototype UE's used is of type QC.

The first way to implement extraction is using QC who has developed software to analyze 5G data both in real-time and offline.[27] The detailed analysis and well-developed user interface (UI) opens up further opportunities for expanding the project beyond the scope of this thesis. QC enables a variety of 5G network analysis, including the needed log format for this dissertation.

The functionality can be integrated into a more mobile setup by using the associated API. An API is typically a set of functions used to access data or programs [33], which in this case results launching the data analysis trough a terminal window. This technique, however, has limitations due to the fact that data can only be collected in a file format readable by QC software. The log file can be extracted to Hypertext Markup Language (HTML) format for preprocessing. However, given the number of positional coordinates exceeding 1400, it is not a scalable solution.

The second approach to consider is the open-source project MI which provides some flexibility. It currently supports the QC chipsets used in the prototypes for this thesis, in addition to MediaTek chipsets. MI offers an Android application with prebuilt analyzers for a selection of network data. Support for 5G devices was added with Radio Resource Control (RRC) and Over the Air (OTA) packages in 2020, excluding the needed SS-RSRP messages.

Furthermore, MI has a desktop UI for offline analysis which creates a complete pipeline, from log collection to iteration and processing of the resulting data. Unfortunately, MI features are limited, requiring the implementation of some additional capabilities in order to analyze the acquired data. Also, functionality for decoding the specific log type required for this thesis are absent and must be included before it can be considered a working solution.

Another method for extracting data from a modem is to write a script from ground up. This makes it easier for adaptations to the code, especially if more features are to be implemented in the future. Though this is potentially more time consuming, and with other objectives in mind, it is not the best option for this thesis.

This project configuration uses the previously mentioned open source project MI, described above. The approach is more versatile and customizable than the QC API. Because some fundamental functionality is already implemented and available, more time may be spent on other aspects of the scope for this thesis. The data collection implementation is addressed in greater detail in chapter 3, whereas the following sections focuses on the format and procedure of obtaining SS-RSRP information from a UE.

Decoding Messages

It is necessary to know the contents of the messages in order to decode them. That is, defining what each segment of bytes in the stream represents. Typically, each manufacturer has its own method for encoding messages. The modems in this project are all manufactured by QC.

It is possible to learn the so-called ground-truth of the messages using QC's analysis software. The software discloses the message properties, order of bytes, and their representation. The ground-truth makes it possible to reverse-engineer how to decode the messages, by implementing separate analyzers in the MI desktop API. Further details of this process are described in [20].

By designing an array representing the order of the bytes mentioned above, the logs can be parsed to a humanly readable XML format. From there, it is possible to extract separate features for the radio fingerprint.

Q Number Format

The Q number format is used to encode some of the bytes in the SS-RSRP messages. Q is a fixed point format for binary streams that represents numbers with integer and fractional parts. A common notation is $Q(m.n)$ format, which refers to a number whose integer part is encoded with m bits and fractional part with n bits.

Some of the bytes in the messages are encoded using the Q number format. It is a fixed point format for binary streams that represent numbers with integer and fractional parts. A common notation is using $Q(m.n)$ format - meaning a number whose integer part is encoded using m bits and the fractional part uses n bits.[2] In the messages, the Q number format is represented by a integer that can be translated from bytes. It is referred to as the "raw value" and it is represented as n_{raw} below.

The following operation converts a floating point value n_{float}, encoded in $Q(m.n)$, to n_{raw}.

$$n_{float} = n_{raw} 2^{-n}$$

QC use the $Q(m.7)$ format, and both n_{raw} and n_{float} are accessible through QC software. Through trial and error it is possible to verify the exponent n. We omit calculating m for two reasons. Firstly, m is not used in the above mathematical operation, and therefore not relevant. Secondly, because the information is attained through the QC license provided, it is unwarranted to disclose more information than necessary.

2.2 Indoor Position Prediction Using Machine Learning

To predict the indoor position of a UE, a statistical model using acquired fingerprints as input data must be identified. This is a problem solved using ML. The next sections describe some basic principles of ML, such as data preprocessing, different models for training, and model evaluation strategies to consider.

2.2.1 Introduction to Machine Learning

Machine learning is a subset of artificial intelligence (AI) that refers to algorithms that can learn from experience. A function is implemented that can map input to output based on - commonly - vast volumes of data.[31] Typically, the field is split into three different subcategories [28]:

- Supervised learning, allows both input and result to be perceived and hence validated. Labeled data is a more typical term for the input.
- Reinforcement learning, occurs when the output receives feedback in order to learn the desired behavior.
- Unsupervised learning, an algorithm where no feedback is provided, resulting in unlabeled input data. Instead, the model detects hidden patterns in the data fed to the model.

When collecting data using a robot's internal coordinates, the ground truth for the position - that is information about the position that is known to be actual or true - is provided. This means supervised ML is a suitable approach since the test predictions can be compared to the target coordinates.

This to solves the classification problem of the data, using a grid region, rather than a probabilistic method where the positional data would be continuous. Article [12] has characteristics similar to collected data for this project, which further motivates for the classification approach.

Classification in Machine Learning

Classification is a field in statistics that is often applied in ML for prediction. It uses input - known as predictors or features - to output labels. The labels are categories, or classes, that describe the input[15]. In contrast there is regression, which aims to output estimates based on the features, while classification uses discrete values.

In the context of this thesis, the features are made out of SS-RSRP fingerprints. The label on the other hand is the UE position. Each coordinate is a class, and therefore the labeled data results in a large number of classes. As a result, each data point is denoted by a row in the data set. We illustrate the structure in the data using the following table.

Coordinate (x,y)	Fingerprint (RSRP from SSB beam #)			
	0	1	2	3
24300, 28300	−82.54	−86.62	−90.38	−84.74
24300, 28150	−83.77	−82.28	−84.15	−90.44
22950, 27600	−89.35	−84.71	−91.89	−84.74
22950, 27600	−89.35	−84.71	−91.89	−84.74
22950, 27600	−89.33	−84.74	−91.91	−84.74
24300, 28300	−82.55	−86.62	−90.38	−84.74

We illustrate the probability aspect in the following scenario. There are two classes: A and B. An instance of class A has a 1.0 chance of belonging to that class and a 0.0 chance of belonging to class B. A data point might belong to either A or B. The classifier predicts the chance of an unseen data point belonging to a class in a classification context, and the classification is based on the highest probability score. When two classes are present we have binary classification. In addition to A and B, more classes can be introduced. That is known as multiclass classifications, and the scenario for the data in this thesis.

Different classification techniques use separate methods for the decision-making. Each section in 2.2.3 describes the individual approach for that classifier.

2.2.2 Preprocessing

Garbage in, Garbage out (GIGO) is a data science term describing an essential topic in ML. It represents the importance of good quality data for any purpose of modelling.[36]

"Garbage" data may have an impact on the model's accuracy, contributing what is commonly referred to as noise and bias. Unexpected values, such as outliers, poorly formatted or incorrect data, can all be considered noise. In contrast, bias is an example of unbalanced data in which one or more classes are overrepresented in the data set.

Data may also be represented in an unfavorable manner for use in a programming context. All of these scenarios must be addressed before training a proper model, which is why data preprocessing is such an important element of any ML project. This section introduces some fundamental data preprocessing principles, and some comments on the data set used for this thesis.

Remark on Current Data Issues

The above mentioned concern with bias in the data set is prevalent in the data collection setup. This is related to the thread handling of the application explained in section 3.7 on page 40, responsible for data extraction and communication with the robot. There are some discrepancies in the amount of data retrieved at each coordinate, which replicated across the entire dataset. Considering the fact that the differences seem minor in comparison the data set, the assumption is that this aspect may be overlooked. It is, however, important to keep this in mind for future development of this set up.

Another finding is the existence of values that do not appear to represent any real-world value. The radio used in current configuration have beam IDs with ranges between 4 and 47 (see figure 2.4). Parsing results in some beams reading an ID of 25000. To deal with this, a known scope with valid values is defined, and all instances outside the scope are ignored.

To understand hidden structures in the data, a heat map over the classes is presented in 4. It aids in discovering any correlations between beams and radio fingerprints.

Normalization

When the size and shape of data in a data set has prominent fluctuations, numerous transforming procedures are available in order to scale them to a more comprehensible format.[25] Normalization is a well-known procedure in which data values are decreased to meet a specific range, resulting in reduction of data redundancy and improvement of data integrity. It is very crucial in distance measuring algorithms, such as k-NN used in this thesis.[1]

When data used in algorithms like k-NN is not normalized, there may be outliers which weight the result significantly during training. Additionally, normalization may help determining whether the model is capable of scaling the radio fingerprints for UEs with faded signals. Fading is an effect common with aging UE hardware. Two UEs may produce distinct SS-RSRP fingerprints, but the range between the signal strengths is preserved.

Normalization for the data set in this thesis uses MinMaxScaler from sklearn[1]. It scales each feature to be within a range passed as an argument. The range of the data is normalized between 0 to 1, from previous scale of -44dBm to -140dBm.

Imputation

When there is a large number of missing values in the data, it is critical to consider how this may effect the outcome, since it influences the prediction performance.[16] The technique of replacing missing values with estimations is known as imputation.

Various approaches can be used to calculate the estimates for the missing values. Above referenced article evaluates and compares several different imputation methods. In order to assess how effective the imputation is on the data set, you evaluate the model performance by calculating mean square error (MSE) for the predictions and the actual target values.

Figure B.5 on 75 depicts a graph illustrating the percentages of missing data for each beam received. Because the QC chipset only saves the four strongest beams out of a total of 24, certain beams are missing as much as 90% of the data. Imputation attempts to generate suitable default values in place of these missing values.

For imputing the missing values in this thesis, the data set is processed with the *KNNImputer* by sklearn[2], which considers the k nearest neighbours and take their mean to estimate the missing value. It should be noted that this is unrelated to the k-NN model used for prediction detailed in section 2.2.3. Instead, the imputation uses the same algorithm, k-NN, for estimating each missing value.

The following procedure is used for evaluating the imputation, as explained in [16]. The first step is to select a subset of the data that is known to be complete. Next, values in one imputed column are replaced with empty Not a Number (NaN) placeholders. The imputation is applied on the empty columns after that. Finally, the predicted and target labels are compared for evaluation which is carried out with root mean square error (RMSE).

The authors of the above mentioned paper emphasize the importance of looking at more than simply the RMSE. To get the whole picture, it is also necessary to compare the model performance with the imputed data set and without. This completes the evaluation process described above and adds more information on how well-estimated the imputed values are.

2.2.3 Models

There are a vast variety of ML models to choose from that are suitable for the objectives of this thesis. Considering for other area objectives for the scope of this thesis, simpler models are used for predicting the UE position. This section describes and analyzes different models for the ML phase of this thesis.

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html`

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html`

k-Nearest Neighbour

In statistics, k-NN is a simple distance-based classification algorithm. Within the field of ML the algorithm demands a lot of computational power due to its time complexity of O(nxm), where n is the number of training examples and m is the number of dimensions in the training set.

In its application in the context of this thesis, a new fingerprint is compared to the k closest, already known, fingerprints. The new fingerprint is then assigned to the most prevalent class of fingerprints in that group of neighbors. An alternative to this is calculating an average value among the k nearest neighbours. In either case, the calculations are performed on a small number of fingerprints, a technique known as lazy learning.

Furthermore, the computations do not occur until the classification, making the approach quick to implement in comparison to other similar ML algorithms.[4]

The model needs a way to determine the distance between two data points, which in this thesis are rows of normalized values representing SS-RSRP values for each SSB ID. Different distance formulas can be used, and this thesis uses the Euclidean Distance metric. The Euclidean distance between two points p and q in a one dimensional space is defined as follows:

$$d(p,q) = |p - q| = \sqrt{(p - q)^2}$$

Similarly, the distance in an n-dimensional space is defined as [34]:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2}$$

It is worthwhile to consider which distance measure to use for k-NN model. The Euclidean distance would be an inaccurate representation of the similarity between the image of two faces, while it is an excellent statistic for assessing the similarity of two rows with floats or integers.

Given the nature of data used in this thesis, k-NN with Euclidean distance seems to be a good option. One argument for motivating this choice is that the distance concept is easy to implement and calculate for the fingerprints, since they are sets of rows with numerical values. Also the data set contains at least two identical instances for each coordinate, given the rate at which the messages are obtained.

Another option considered for k-NN was Manhattan distance, which is a grid-based distance measure. The Manhattan distance denotes shortest path in terms of steps. However, since the fingerprint data is continuous and one-dimensional, Euclidean persisted the better alternative.

In addition, the article [4] experiments to determine the most accurate method for indoor positioning, which describes a similar approach with k-NN used in this thesis. For example, they use regional positioning in an area the same size as the one suggested for the experiments. Additionally, the fingerprints consist of SS-RSRP information. The pairwise comparison between instances is a strength in terms of evaluation of a point's membership according to [9]. The algorithm provides a heuristic for similarity that is consistent, and as a result the k-NN algorithm offers several desirable characteristics for applying collected data in order to predict the indoor position.

The classifier is easy to implement, which is another requirement stated for the selected approach. Lastly, it is widely used for its simplicity and classification efficiency.[17]

Using k-NN, the performance between a range of different number of neighbours are compared. This is needed in order to explore what may be the optimum k-value to take in to account when running future simulations.

Naive Bayes Classifier

The Naive Bayes algorithm's decision making is based on Bayes Theorem, which is a significant theorem in the field of conditional probability. All features are assumed to be independent in Naive Bayes, which means they have no effect on each other. Furthermore, it is assumed that have an equal impact on the outcome.[40]

The Bayes Theorem expresses the conditional probability of event $\boldsymbol{A}$ occurring given that event $\boldsymbol{B}$ has occured.[35] It's defined as follows:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

The outcome of this formula in a machine learning context is a model that makes decisions based on the presence of features, one at a time. In our case, the calculation will be done for each feature $\boldsymbol{RSRP_i}$ based on the coordinate:

$$P(coordinate|RSRP_i) = \frac{P(RSRP_i|coordinate)P(coordinate)}{P(RSRP_i)}$$

As an example, if an object is red and round, the model might say it is an apple regardless of the relationship between or intensity of the two features. The classification denotes the probability of the object being an apple, *given* that it is red. In the context for the SS-RSRP values in the fingerprints, a numeric feature may be an indicator for the decision-making based on the value's span.

The experiment in [18] is carried out on RSSI fingerprint data and modelled using Naive Bayes and successfully manages to predict the UE position. This motivates using the model for the purpose of this thesis. Implementing more than one model preferable, since it is easier to compare and draw conclusions regarding the result. This thesis implements a simple Naive Bayes models, along with the k-NN model. Naive Bayes Classifer is also used in previously referenced article [4], which makes a good foundation for ML experiments as a proof of concept.

Probability Map

Some models use probabilities for prediction, mapping a state value to its probability of occurring. One such distribution for discrete data is the Probability Mass Function (PMF), and for continuous data the distribution Probability Density Function (DNF).[23]

Probability maps can be used for pattern recognition using various probability techniques. As new training data is supplied, the model analyses and modifies the probability using Bayesian inference methods. When precise answers are not feasible, the Bayesian approach presents approximation inference algorithms and, as a result, handles uncertainty in the data. In contrast, frequentist probability strategies make decisions on counts and confidence intervals using a statistical approach.[14]

In [12] probability maps are used as a probabilistic measure for prediction using neural networks. The premises are similar to those stated in this thesis, with a physical grid-like region in which position predictions are derived using RSSI fingerprints. The mentioned paper, on the other hand, uses fingerprints from two radios simultaneously as a resource. The data collected for this thesis is limited to one radio, which with certainty impairs the the accuracy of the neural network. Thus, this approach might be a subject for future work.

2.2.4 Evaluation

To understand how well the models can predict the indoor position of the UE, it is necessary to evaluate the performance. Prediction in classification tasks uses probability scores. Model evaluation techniques therefore require summarizing a model's performance based on predicted probabilities. This section explains different measurement scores for evaluation, along with presenting the strategy used for this thesis.

Preparation for Evaluation

It is important to test a model's performance on data that it has never observed before for verifying its behavior. Typically, this is accomplished by dividing the data into two parts, a training set and a test set. The majority of the data is used for the training phase, during which the model is tweaked to predict the classes. The remaining data points are utilized to test the model, and the results of this test are used for evaluation. As a remark, the k-NN model does not have a traning phase, but rather known and unknown entries. Therefore it is still necessary to split the data accordingly.

Due to the risk of bias mentioned previously in section 2.2.2, it is crucial for the data to be well-balanced in the two sets. This means there is representation of all classes, in this case the coordinates, in both sets. For this thesis, the input data is split between 70% for training and 30% for testing.

Evaluation Metrics

As stated above, there is a wide range of different metrics for evaluating the model:

- Accuracy: Describes the ratio of test data that were correctly classified across all classes.
- Precision: Indicates whether or not a system is over-fitted for a particular class. To calculate precision, the percentage of the classes labeled as positive by the model is used to determine what share of the positively labeled data that is correctly classified.
- Recall: A metric that indicates how well the model correctly identifies members of the specified class, that is, the ratio of true positives.
- F-score: The harmonic mean of the precision and recall. The F-score, like precision and recall, considers only one class at a time denoted as positive.
- Macro F1: The metric is calculated by considering the averages of the F1 scores of all classes. It is especially useful in systems with a large number of classes, where all are of equal importance.

- Micro F1: Like the Macro F1 metric, it measures the average of the harmonic mean with one exception: classes that appear more frequently in the data set are considered to be more significant for the result.

Since the data set for this thesis have a lot of classes with equal importance and relevance, the *Macro F1* is the main score used for evaluating and comparing the models. As a first iteration, accuracy is used to give an approximate of model performance during development.

Moreover, the deviation between falsely and correctly classified entries are explored. This is motivated for understanding how far away from the true position the predicted result deviates across the data set. The cumulative distribution function (CDF) denotes the probability that a variable X takes on a value lower than or equal to x:[6]

$$F_X(x) = P(X \leq x)$$

In terms of deviation it is a suitable measure.

Chapter 3

Automation

This chapter provides an overview of the various components of the data collection setup. The entire system is depicted in figure A.2 on page 72.

3.1 Physical Setup

The project needs an allocated area that remains consistent throughout the whole data collection procedure. For this thesis, Ericsson suggests using the main entrance hall of the corporate building to mount the 5G radio for the experiments. The designated area is illustrated in figure 3.1 on page 34.

The radio model itself is equipped with 24 wide SSB beams. Ericsson suggests tilting the radio at an angle of 35° so that the majority of the beams are directed towards the middle of the room. This causes the beams to not bounce off the surfaces as much, reducing noise in the data. Instead, the beams reaches the UE in line-of-sight, eliminating the need to account for propagation anomalies.

The area in which data is collected needs constraints due to how the LiDAR operates, which is explained in 3.2.

3.2 Navigation and Position Tracking

A moving, programmable device, capable of navigating the allocated area while keeping track of its position and surroundings, is required to automate the data collection method. Supervised learning simplifies the process of implementing a model, but it requires that the device's internal coordinates be consistent. These coordinates represents the data set's ground truth.

One alternative is to use a robot vacuum, several models employ internal map representations and are capable of adapting to their surroundings using LiDAR sensors. These two

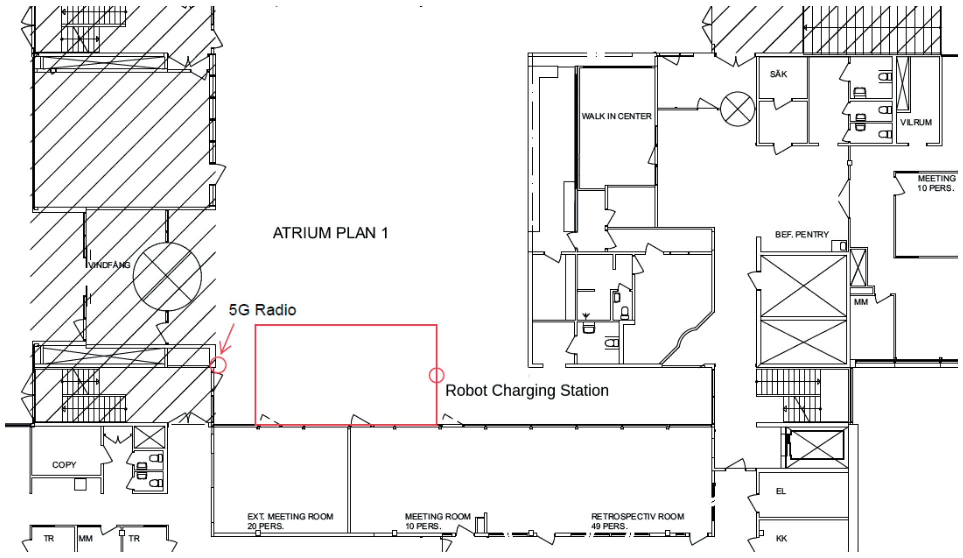


Figure 3.1: The main entrance hall marked with the robot charging station and the 5G radio location. The X-axis spans from right to left, and Y-axis from top to bottom.

characteristics make them suitable candidates for the task at hand. On the other hand, these devices frequently have limited system access, making them less programmable.

A robot grass mower is also an alternative. These make use of the Robot Operating System (ROS) framework and hence allow for customized functionality. The disadvantage is that the implementation process may prove time consuming, shifting focus from other parts of this thesis. The robot vacuum cleaners already have a lot of logic built in, making them a better fit for this project.

The Xiaomi Roborock S5, an older model from 2018 [39], is used for navigation during data collection. The robot uses an internal coordinate system, with the charging station as its control point with coordinates $(25500, 25500)$. Figure 3.2 on 35 shows a picture of the robot with relevant components highlighted.

It is possible to gain access to the robot vacuum's root directory by installing a custom firmware developed by Dennis Giese on the device.[11] This gives access to the sensors and internal data, but requires the development of an API to control all functions from within the robot's operating system. Also, there are limitations to how accessible these functions are, as well as knowledge about their structure.

There are a few open source projects available for customizing smart devices at home. One of them is HA, with features for both sending commands to the robot through scripts, and for communication with other devices. It has an active community, with regular updates. This integration is most suited for this proof of concept thesis because it already has a lot of functionality for accessing the robot's API in place. The following sections describes the configuration that is used for controlling the robot.

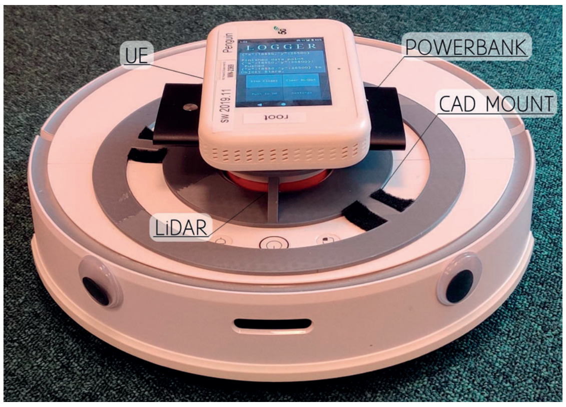


Figure 3.2: Overview of robot used for navigation.

3.2.1 HomeAssistant

HomeAssistant is an open-source project founded by Paulus Schoutsen.[30] It is a smart-home integration tool that supports a wide range of devices, each with its own integration. It has a well defined support for the Xiaomi Roborock S5.[13]

Every integration has its own functionality, for instance toggling the fan of the robot, or returning it to its charging station. The installed local HA server operates with the same commands sent from the Xiaomi Miio cloud, but allows the users to easily adapt the device to their own needs through scripts. In HA the functions are called *services*. In order to send commands to robot through HA, the robot token with its IP-address is needed, which can be extracted either through rooting the robot or using a script available in the open source community.

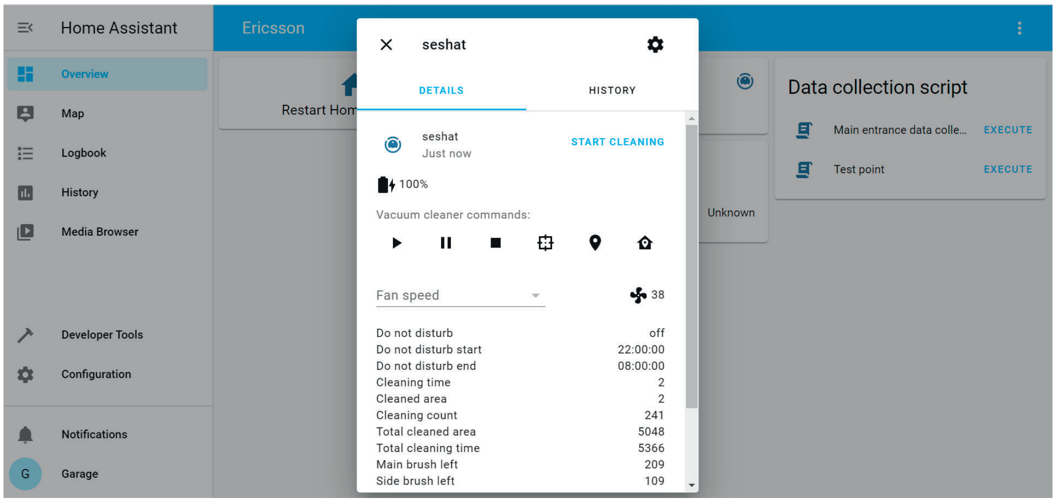
This is a useful feature for this project since that enables the possibility to control the device meticulously. Moreover, it is possible to use all the integrations in the same place. It also has support for different communication protocols to establish connection between different devices.

In HA, the command scripts are written in YAML, a data serialization syntax that orders objects and lists, and every command need to explicitly be described as their own service. [38] For moving the robot, a "go-to" command is used, and unfortunately there is no way to extract the robots coordinates internally from the robot. Instead it is possible to make use of the fact that when implementing the script for controlling the robot, to save the coordinate variable and send it to the UE when data is to be collected.

Since it is difficult to know when, in time, the robot has reached an extraction point, states are used. When moving, the robot reaches the state "cleaning", and when stopping, it switches to "idle". While charging, the robot's state is "charging".

The user base for HA is typically regular home owners who wish to personalize and synchronize their smart devices at home. As such, it is less common to use HA for purposes like this thesis.

When running a created script, the commands are added in a list of services, and all are executed at the same time. To workaround this, delays and waits are added, forcing the robot to wait for the next command until some chosen event triggers and resumes the execution.



seshat
Just now
100%
Vacuum cleaner commands:
Fan speed 38

Do not disturb	off
Do not disturb start	22:00:00
Do not disturb end	08:00:00
Cleaning time	2
Cleaned area	2
Cleaning count	241
Total cleaned area	5048
Total cleaning time	5366
Main brush left	209
Side brush left	109

Figure 3.3: The HomeAssistant user interface, displaying the Xiaomi S5 integration. To the right, two scripts are displayed

An example of a simple script is illustrated in listing 3.1.

This script first commands the robot to go to coordinates (25300, 27700). Coordinates (25500, 25500) is the position of the robot charging station. The added row with delay forces the script to wait before reading the next command. Without these delays, previous commands get overwritten.

```
main_entrance_script: # Name of the script
  alias: Main entrance data collection # Description
  sequence: # The list of services to be performed
  - service: xiaomi_miio.vacuum_goto # Go to position
    data:
      entity_id: vacuum.seshat # Robot vacuum name
      x_coord: 25300
      y_coord: 27700
  - delay: 00:00:10
  # Wait until robot vacuum has stopped
  - wait_template: "{{ is_state('vacuum.seshat', 'idle') }}"
  # Publish MQTT message with positional data
  - service: mqtt.publish
    data:
      topic: EricssonONE/egarage/5g_fingerprint_localization/data/
   collect
      payload: '{"x":25300,"y":27700}'
      retain: 'true'
  - wait_for_trigger: # Trigger with condition
      platform: mqtt # Wait for the payload 'RUN'
      topic: EricssonONE/egarage/5g_fingerprint_localization/data/
   run
```

```
        payload: RUN
```

Listing 3.1: YAML script with commands sent to the robot in a sequence with a list of services. Wait and delays were used to stop the commands from being executed at the same time.

The communication between the UE and robot is handles with MQTT, supported in HA. This is used to send the coordinates where the extraction is to be executed and is described further upon in 3.3.

3.2.2 LiDAR

For navigation, the robot uses a LiDAR. For the LiDAR to operate properly, it need surfaces for the laser to reflect back on the sensor of the robot. Due to the large room size of the main entrance hall, a fence is needed since the robot gets confused when there is no feedback for the LiDAR.

Since the robot navigates with the UE to collect data, a mount - seen in figure 3.2 - is used for placing the UE as to not obstruct the line-of-sight of the sensor. Placing a UE on top of the robot risks malfunction during operation, and also of the UE moving during data extraction.

The size of the allocated area is $4.5x9$ meters, allowing the LiDAR to receive proper feedback. In [12], rooms of maximum size $4.5x9$ meters are used. Since the area used for the experiments is larger in this thesis, it is safe to assume that a good amount of data is collected for the ML phase.

Any obstructions are also removed from the area before commencing data collection. Objects left in the area are at risk to be moved around during extraction, confusing the robot. They may also hinder the robot from reaching its location coordinate, which makes it recalculate its destination to an approximate one and as such creating noise in the data.

3.2.3 Positional Replicability

Experiments are carried out in order to have a better understanding of the precision of the robots' navigation. These experiments examine how different use cases influence the precision of the robot during data collection.

By letting the robot move between two pre-determined coordinates, it is possible to observe the error margins of the data collection. Doing this 30 times, noting the exact position for each position to observe an estimate of the deviation. During data extraction, the robot never moves between the same two coordinates, but it is a simple experiment to test the precision.

It is possible to observe the data collecting error margins by allowing the robot to move between two pre-determined coordinates. This is repeated 30 times, while noting the precise location for each position to get an idea of the deviation. The robot never moves between the same two coordinates throughout data extraction, however it is a simple experiment to assess the precision.

The robot traveling between the charging station and a coordinate is a more common example case for this thesis. By having the robot move 10 times between the charging station and one coordinate, it is possible to note some indication of deviation in its precision. Since many different units are working together in this thesis, errors may result in having to restart the data collection and let the robot return to its charging station. This experiment determines whether the accuracy is improved due to the robot returning to its reference point, the charging station.

In order to finish extraction of data for the whole allocated area, the experiments where done in separate runs. As such, the precision is also tested for when the robot is unplugged between coordinates. This experiment is repeated 10 times to estimate if a smaller error of margin is achieved.

3.3 Device Communication

With different devices dependent on each other for this setup, it is necessary to have a bridge of communication. The responsibility of such bridge is to submit information between the devices, information such as extraction point coordinates or prompts to start or resume different scripts.

An MQTT broker is used for meddling between the devices. It is a third-party intermediary that facilitates the communication between devices.[10]. The light-weight communication protocol is developed by Oasis Open, and is used as HA facilitates support for the protocol.[24] It uses topics to publish and subscribe messages between different entities.[22] The messages are published in topics to which devices can subscribe, which opens up for easy sorting and separation of information.

Because the 5G connection is a central part of the experiments, the UE would have to jump between 5G and Wi-Fi if the local network was used for the communication. This would drain the UE's battery faster. Additionally it is a possible source of malfuction in the automation, since it is difficult to overview and control for in runtime. The MQTT broker proves to be the most suitable solution to the internal communication for the setup using devices on different networks. An already existing MQTT broker is used, available through a public IP.

3.4 Extracting Data From Modem

One important mechanism for the setup is to obtain information about the connection that later is used for fingerprinting. The automated setup should run this process for each coordinate within an allocated area in the script. This section describes the technical details of data extraction with MobileInsight. An overview of the data flow is presented in 3.4.

For this configuration, a variety of various UEs are tested, since all 5G UE prototypes might not be supported. During development, a rooted LTE model (Google Pixel 3 with Snapdragon 845 modem) is used for testing. The decision to use LTE as a stepping stone is inspired by MobileInsight's extensive network analysis features for that generation of technology. However, UE selection support for 5G is restricted. Finding a UE that operates with 5G, while also supported in MobileInsight, is important for the extraction to be possible.

The script that takes care of the low-level communication that results in modem log files is called "diag_revealer". It is developed for Qualcomm and MediaTek chipsets and implements C-code in order to communicate with the modem. A configuration file is passed as an

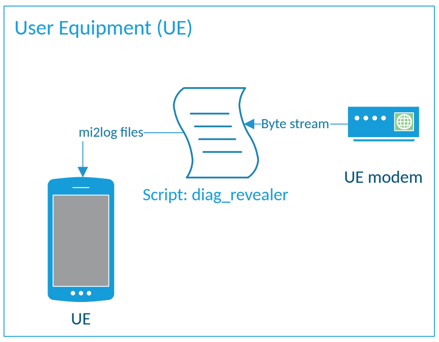


Figure 3.4: Internal data flow within the UE during extraction.

argument to parse byte streams into log messages, as the bits reach the modem. The script is developed in the MobileInsight project.

Using Qualcomm software QUALCOMM eXtensible Diagnostic Monitor (QXDM), it is possible to generate the needed configuration file for filtering selected log types. The support is limited to a selection of log types in MobileInsight's API today. The configuration file sends instructions to the modem to write the selected packages back to the diag_revealer, which then writes that to an output file. MobileInsight uses a file format for the output, "mi2log", that is implemented specifically for the project. When comparing a "mi2log" file to a log file generated directly from QXDM, it is clear that they contain the same byte stream for the packages. This enables a way to obtain and analyze packages that are not yet implemented in the MobileInsight Android application, by using the diag_revealer directly.

3.5 MobileInsight and Offline Analysis

In order to construct a radio fingerprint, it is required to extract specific metadata from the log files. The MobileInsight application keeps track of the messages coming from the modem by writing them to log files. These files are decoded using MobileInsight implementation for offline analysis. In the end, it is possible to look directly at the real SS-RSRP values collected at each coordinate.

The package decoding consists of two phases: define the contents of the package byte stream in C-code (referred to as "message parsers", with one message parser per package type), and translate that to XML files in Python coded analyzers. The packages in "mi2log" files are byte streams, where each byte represents one piece of information. The order and meaning of each byte in such a message can be viewed in a software known as Qualcomm Commercial Analysis Toolkit (QCAT). For instance, the number of detected beams are stored in one byte, and is represented with two hexadecimal digits. The defining phase starts with distinguishing the head - the first sequence of bytes that is unique - of a message. This step is necessary for the decoder to know when it encounters the message in the log files. Then proceed by defining the contents byte by byte as described above. A number of types are defined in the C source code, that automatically decodes the byte stream to the given types. By using QCAT message

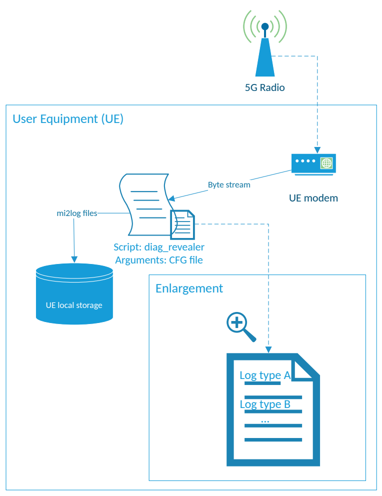


Figure 3.5: How the messages are obtained through logs in the UE

types as reference, it is possible to reverse engineer the process of decoding the message. The next step is to parse the bytes into a format that is comprehensible for humans, which is a process referred to as an "analyzer" in MobileInsight. By using the predefined order in the C-code, the analyzer assigns each data point of a package to an XML tag. These tags, with the associated data, are then written to an XML file.

3.6 OpenStack Object Storage

With a lot of data stored on different devices, it is necessary to have an accessible storage system. The goal is to utilize 5G to push the raw logs to the cloud in real-time. Looking at options that are easy to implement in Python - which is the language used for the Android application "Logger" described in section 3.7 - a Swift API seem to be the best alternative. Ericsson is able to provide such a server with storage through OpenStack Object Storage.

Object storage does not allow nesting of directories, which is required to keep all the information about the date and time of the instances for each coordinate. Instead Object Storage simulates a hierarchical structure within the container of data using "slashes", and as such it is possible to maintain all the information pushed to the container. With API calls it is possible to both save and retrieve data using the cloud storage.

3.7 Android Application

Currently, QC software only has support for computer systems, and there is no official application for Android systems. Thus, one limitation of a QC based approach is not being able to use the UE independently for data collection. This is due to some form of analysis needing to be done in order to filter packages for collection, and there is currently no integration compatibility on Android using the QC software.

There are some different possibilities to approach this problem, such as using a Raspberry Pi along with the UE and necessary scripts. This results in more dependencies for the automated setup. With some of the requirements for the proof of concept are simple setup and operation for future research, this is not desirable. Also the robot, with its pressure sensitivity, needs to move around with all units attached.

Implementing an Android application is a more flexible solution. One option to consider is the open source project MobileInsight.[21] For years the project has iteratively developed an application in Python Kivy to retrieve different data logs, using compiled C-files, for a variation of supported UEs. Among the supported chipsets are Qualcomm Snapdragon, along with new support for OTA 5G signals in a beta release. MobileInsight latest beta release does not collect the particular data logs needed for the radio fingerprint intended for the ML experiments. However, it has possibilities to further develop extra features on top of the existing application.

Implementing a new data extraction from the UE model is a large project in it self. Due to the broad scope of this thesis, instead of reinventing the wheel, it is more effective to go with existing available resources. Consequently, time can be spent on other parts of this project.

The requirements for the application is to be able to collect given logs from the UE, save said data on the UE itself, push data to a database for later data preprocessing in the ML stages. Additionally, it should be possible to view suitable feedback in GUI for users - such as status of script and error messages. Lastly, some essential changeable parameters within the application and communication with the MQTT broker are needed. Communication is necessary for accurate data collection as it would be undesirable for the robot to accidentally move during data extraction within the UE.

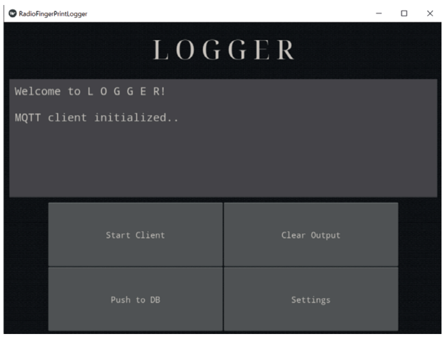
3.7.1 Kivy App

Kivy is an open source Python library, with cross-platform support. It is a simple syntax to grasp and testing is easily executed on different devices. With cross-platform support, it makes development of the application more effective. MobileInsight also uses the Kivy library. Therefore, integrating some working elements of their application source code in the "Logger" application, which is the name of the app developed for the automation setup, is possible.

The Kivy language uses simple syntax, similar to many other markup languages. The GUI in Logger is simple but informative, with a box for output, sending the user system information during execution. There are four buttons for interaction with the app. Figure **??** shows the GUI on Windows OS and Android OS.

Starting and stopping the client is in this context the MQTT client, subscribed to the topic that the robot publishes its coordinates to. The next button clears the output box from content. Additionally there is a button for pushing all the data to Object Storage container. The *Settings* button allows the user to adjust parameters for collecting data, such as amount of files per coordinate and corresponding file size. It is also possible to toggle if the user wants to clear the data logs from the UE while also pushing data to Object Storage in the cloud.

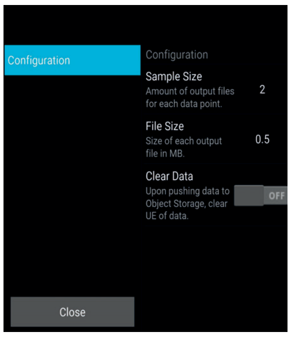
The dependencies, relations and general application structure can be seen in figure 3.8. The Controller in the app stores the MQTT client. When a new message arrives from the



Welcome to L O G G E R!

MQTT client initialized..

Figure 3.6: Windows OS GUI before starting MQTT client.



Configuration

Sample Size — Amount of output files for each data point. — 2

File Size — Size of each output file in MB. — 0.5

Clear Data — Upon pushing data to Object Storage, clear UE of data. — OFF

Figure 3.7: Android OS GUI, changeable user parameters.

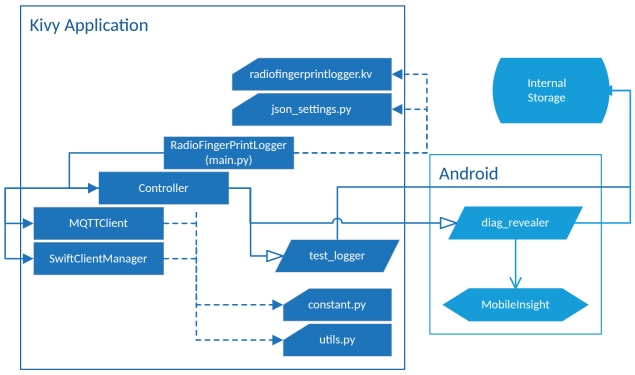


Figure 3.8: Diagram of application structure.

robot, through the broker, it sends commands to Android shell to start the external program *diag_revealer*. Along with the command, the parameters for diag _revealer are sent, either default values or updated from user in settings menu.

While the program executes, it uses busy-polling to keep track of number of files in a newly created map for this instance of starting the application. When the number of files requirement is met, the application prompts Android shell to kill the process diag _revealer. It then publishes to the topic the robot is subscribed to with a "run"-command, meaning that the setup should resume its script and move to the next coordinate.

The robot locates the new point. Meanwhile, the application pushes the data collected to Openstack Object Storage through a Swift Client API using keystoneauth1 for authorization. This is done over the 5G network.

Running this application on a computer system with either Windows OS or Ubuntu OS results in the program running in a test setting. It is necessary to first compile an execution binary program to run the test properly, since the file is not compiled in a freshly initiated setup. The test run results in the program creating dummy files in a folder, simulating data logs being generated. It keeps executing in an infinite while loop, until it is forced to shut down by the system. The program pushes its data during execution to a container for test purposes, as not to contaminate already collected data with dummy files.

With limited changes to the C-file diag _revealer, it is not possible to know when the program has completed the data extraction. Observer and Observable pattern is a good solution, but then it is necessary to adjust the source code of the MobileInsight program. Is is possible that with the changes, unexpected errors may occur and disrupt the extraction process. With time being a limitation, it is not a priority spending time on debugging for optimization.

Keeping track of the number of files is a simple solution to the problem. However, it may cause the last file of data, for each coordinate, varying in size. Since busy polling is used in the application, the sleeping threads may lead to some inconsistencies in timing when checking for the number of files in the folder.

3.8 Execution

Before starting the data extraction, the fence is put in place around the allocated area. The robot charging station is carefully placed in the exact position used for previous runs, and the robot is added. On the robot, as seen in figure 3.2, a powerbank is placed for supplying the UE with electricity during the run. Since the 5G prototypes are not optimized, the battery runs out before the session is completed without the extra energy source.

It is important that the UE is placed in the same direction and orientation before the data extraction starts. This is to eliminate any factors of inconsistencies within the collected data set. Velco is used to put the mount and UE in place, avoiding any accidental displacement during, or between, sessions.

Additionally, the coordinates are selected in such a way that the robot's orientation is consistent. For example, when a new row is recorded, it is necessary to use padding coordinates that are not included in the data collection. This way the robot can align itself to the new row, and therefore will not turn between two coordinates (changing the orientation of the UE as described above).

Due to the total time it takes for the robot to run all the coordinates, the experiments are done in sessions. The robot starts at its charging station, and the script is resumed at the next coordinate for extraction. The UE has the Android application up and running, and the MQTT client is listening for messages from one topic subscribed to.

When the robot arrives at a coordinate it changes state to "idle", and publishes a message on the UE subscribed to the corresponding topic with the coordinates as payload. The broker directs the message on the topic and passes it on to any subscribers, usually the UE and a shell terminal for observing runtime. This prompts the UE to execute the external script, diag _revealer, with received coordinates for extracting data from the modem.

When conditions are met for the of number of files generated at the coordinate, the external script is terminated and the UE publishes a message on the topic subscribed to by the robot. The robot resumes it script and continue to the next coordinate for data extraction, meanwhile the UE pushes the collected data to Object Storage. When it is time to end the session, the last coordinate used for extractions is manually saved as starting point for the next session.

The time needed to collect data for one coordinate is about 50 seconds. Therefore, an estimate for the total elapsed time for 1432 coordinates is $24 hours$.

Chapter 4
Result and Analysis

4.1 Automation

Some restrictions is apparent with this setup, but fulfills its objectives for this thesis. In the following sections the end result the data collection setup is presented along with data analysis and the machine learning evaluation scores.

4.1.1 Robot

The particular robot model chosen for the experiments is a restriction it self, further described in these following sections. There are also inconsistencies between the HA server connection and the robot, forcing to restart the server and sometimes reconnecting the robot. These problems are manageable and does not disrupt the completion of the data extraction process.

Positional Replicability

Here, we present and analyze the outcome of the experiments described in 3.2.3. Negative values denote the robot positioning itself closer to the second point. In the first experiment, the second point is the other reference point tested. In contrast to this, the base station is considered the second point for experiment two and three.

The first test shows that the error accumulates over time. After the 30 runs of moving between two coordinates the deviation from the original position is measured. Along the x-axis it differs 5 cm, and y-axis -8 cm.

The results of the second experiments are shown in table 4.1, where the difference along each axis is presented in cm.

Thirdly, when restarting the robot between each test, a lower difference is observed. No axis deviation exceeds ± 0.5 cm, which is the reason it is not included as a table.

	x	y
1	1.5	−2.5
2	2.0	0.0
3	1.0	−2.5
4	1.0	−2.5
5	1.0	−3.0
6	1.5	−2.5
7	1.5	−2.0
8	0.5	−2.5
9	1.0	−2.5
10	1.0	−2.5

Table 4.1: The axis deviation from original position per test in centimeters.

Operational Area

The LiDAR of the robot needs constant feedback for the robot to be aware of its position, as such an enclosure is needed to define the operational area. For this setup, a rectangle of boards measuring $9.75x4.50$ meters is used in the main entrance hall of Ericsson. Without this feedback, the robot assumes there is a LiDAR sensor malfunction.

Because of the size and accuracy of the robot, the operational the extraction points are constructed as a grid with 15 centimeters between each coordinate. The overall area is described in 3.1 on page 34. With less spacing between coordinates, there is a risk that the robot will assume the new position is reached before arrival.

4.1.2 User Equipment

As mentioned in 3.4, it is crucial to find an appropriate UE model from which it is possible to extract the 5G meta-data. This process provides two limitations: MobileInsight support and wavelength compatibility.

The first point is caused by MobileInsight not being fully developed for 5G, and hence, the support is bounded. Therefore, we are limited to models with a Qualcomm Snapdragon X50 modem[1]. For newer modems, for example with the X55 model, the diag_revealer currently lacks driver support, but the MobileInsight project is updated from time to time. In addition, MobileInsight requires a rooted phone for proper privileges in order to access the Android system.

Secondly, the selected 5G radio uses high-band - millimeter wave spectrum - signals which at the time of writing is less common in Europe as opposed to mid-band frequencies.[8]

To conclude, the setup is limited to mostly 5G prototypes with broader frequency ranges and X50 modems. Therefore, a rooted WNC MWH-5 the main model used for experiments.

In the end, The WNC MWH-5 proves a bit unpredictable. Sometimes the mobile network drops the NR connectivity and switches to 4G, disrupting the collection process. Other times, some features and accessibility are blocked and forces a restart of UE before resuming the data collection.

[1]https://www.qualcomm.com/products/snapdragon-x50-5g-modem

4.1.3 Android Application

The application successfully fulfills all requirements for the automation setup. There are however some drawbacks and restrictions with this application, which is explained in this section.

MQTT

The compatibility of the MQTT client used in Python on Android has some complications. Sometimes the UE, especially in regards to the prototypes, is unable to receive messages from the broker. Restarting the UE usually resolves this problem by itself. As for now, the bug has not successfully been replicated on a computer system.

Data Extraction Inconsistencies

For the experiments, 5 files each with 0.1 MB (totalling 0.5 MB, in mi2log format) of data is retrieved. Observing the collected data shows that there are some inconsistencies between extracted file sizes, that is the amount of collected data for each coordinate. A script counting the total XML file size for each coordinate provides insight of that dispersion. The XML files are less compressed than the mi2log format, meaning the size of each file is expected to increase.

On average, each coordinate has files that on average totals to 2.7 MB, with a maximum of 3.44 and minimum 2.56 MB. The results are presented in figure 4.1.

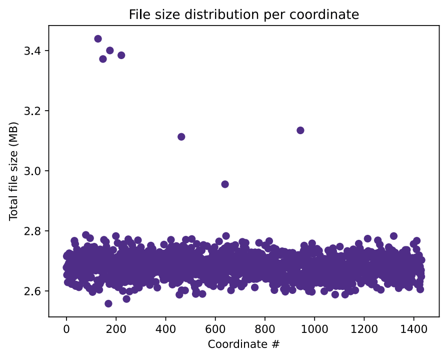


Figure 4.1: Each point represents one coordinate and its total file size.

4.2 Data Properties

This section presents some interesting insights of collected data.

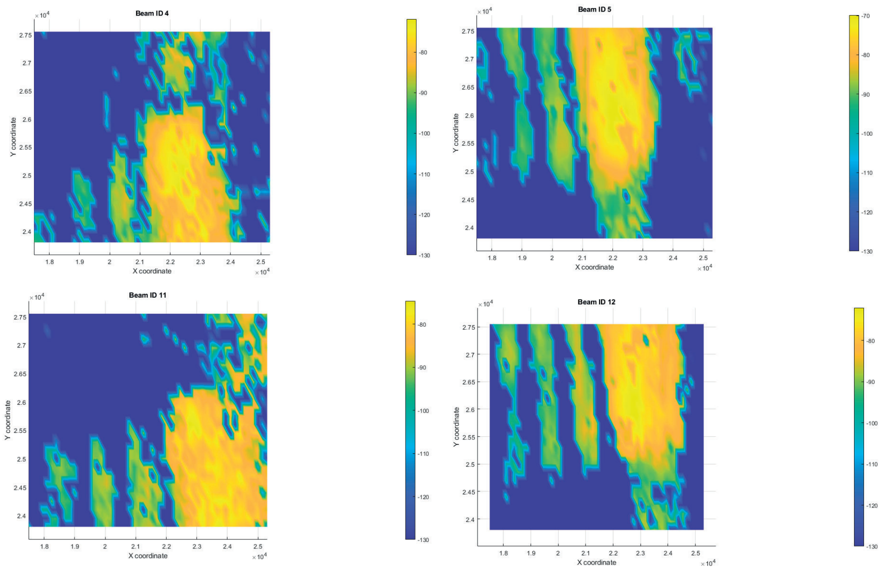


Figure 4.2: The beam IDs with the least missing data in the data set. The plot is in relation to their coordinates and the highest SS-RSRP value. The 5G radio is located to the left and robot dock to the right.

4.2.1 SSB Distribution

Figure 4.2 illustrates the distribution of the four most commonly occurring beams in the physical space. The radio is located at approximately at the "Y-coordinate" label with a tilt of 35°. The rest of the beam heat maps are presented in B.6, B.7 and B.8.

Additionally, figures B.1, B.3 and B.4 all show the same plot but from different angles. In the three figures, the radio is positioned at $x = 15000, y = 26500$. The graph represents the envelope, meaning the highest SS-RSRP value measured for each coordinate. The purpose of presenting the plot is to describe the distribution of the overall coverage in the room.

We are able to confirm the presence of some side lobes. Refer to figure 3.1 to compare the radio's position in the entrance hall. It is also possible to observe that close to the radio we have no coverage, which is an expected behaviour. The uppermost part of the figure shows what is likely the line of sight of the beams. The radio is angled towards the middle of the room and the experimental area is located at the bottom corner. Therefore, the line of sight is not centered in the coordinate grid.

4.2.2 Imputation Results

Running the imputation evaluation on 4% of the data in one column, we get a RMSE of 0.58 dBm. Out of the evaluated data, 1.62% of the classification was correct, and the errors spanned between 0.01 and 17.75 dBm deviation from the actual values.

As shown in figure B.5, some of the columns in the data lack up to 90% of their fields. This means that, in some cases, the predictions for 90% of the data is based on the 10% of

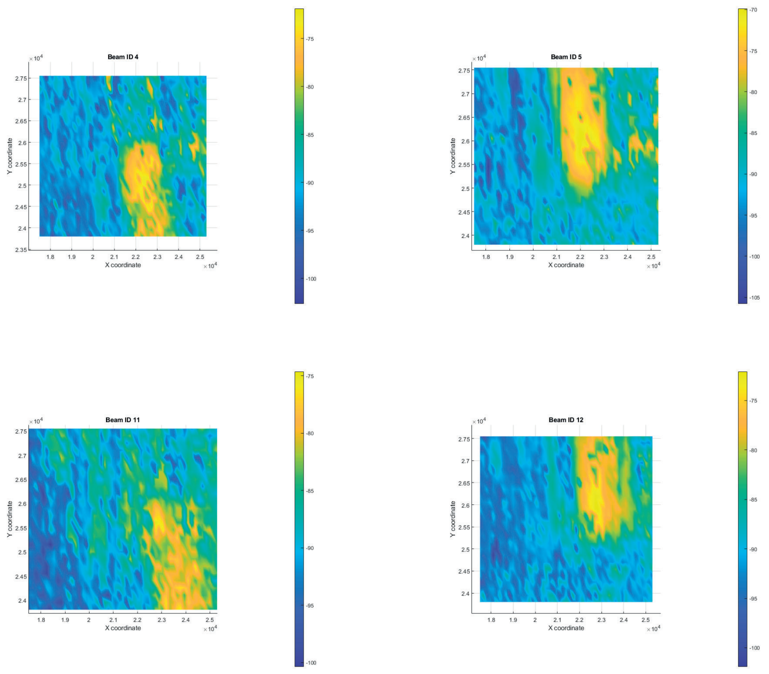


Figure 4.3: A heatmap of the imputed data set for four beams, which may be compared to figure 4.2

data actually in the data set.

To illustrate the imputation results, graphs similar to those in figure 4.2, B.3 and B.4 were generated for the imputed dataset:

4.3 Model

In this section the results of the ML phase are presented.

4.3.1 F1 Score

The score is obtained using *f1_score* by sklearn[2]. The sets consists of two different versions of the data, one normalized and imputed, and one only normalized. Therefore, we are able to compare the model performance and thus, the result of the imputation. In general, each model was run three times with a random seed.

[2]`https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html`

Data set version	Bayes	1-NN	2-NN	3-NN	4-NN
Normalized	0.86	0.98	0.98	0.98	0.98
Normalized, imputed	0.88	0.98	0.97	0.97	0.96

Table 4.2: Model results per dataset

Score	Accuracy	F1
k-NN	0.0007	< 0.000001
NB	0.0006	< 0.000001

Table 4.3: Results from fading experiment

An important remark is the difference in scores between the complete imputed version and incomplete version of the data set. The sklearn k-NN classifier used does not accept missing values. Thus, it is necessary to fill the empty cells with selected values.

4.3.2 Signal Fading

The ability to perceive transmitted signals may vary with the UE model. Thus, a model must be adaptable to different fading aspects that may occur. To test this, a trained model with the imputed dataset in section 4.3.1 is used to predict the position for the dataset with adjustments for fading. The test set of the imputed data is subtracted with -5dBmW from each RSRP value. The evaluation score for predicting the faded data is presented in tabular 4.3.

4.3.3 k Nearest Neighbour

In the graph below, figure 4.4 and 4.5, we observe the obtained evaluation scores for k-NN. An important remark is that the name "training accuracy" is misleading given the fact that k-NN does not utilize a training phase, as described in section 2.2.3. The figure refers to a process of feeding already known data into the fitted model.

4.3.4 Naive Bayes Classifier

No difference is notable when using the imputed version of the data set, as compared to the initial version with only four SS-RSRP values per fingerprint.

4.3.5 Deviation in Misclassified Data

In 4.6 and 4.7 we can observe the CDF from the misclassified instances in the data set for the two models respectively. In other words, we observe the distribution of the distance in the misclassified data.

Additionally, we observe that most misclassified instances occur along one side of the experimental area, for both models.

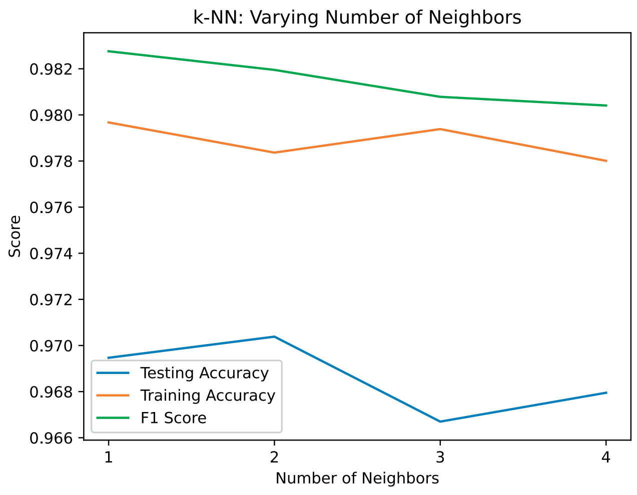


Figure 4.4: Evaluation scores on k-NN per k, incomplete dataset

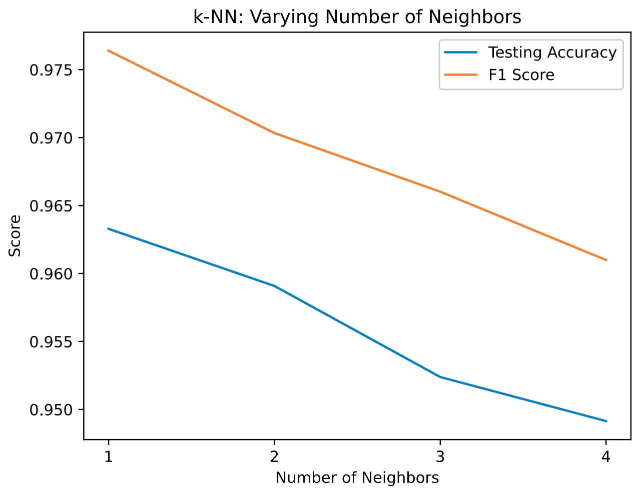


Figure 4.5: Evaluation scores on k-NN per k, imputed dataset

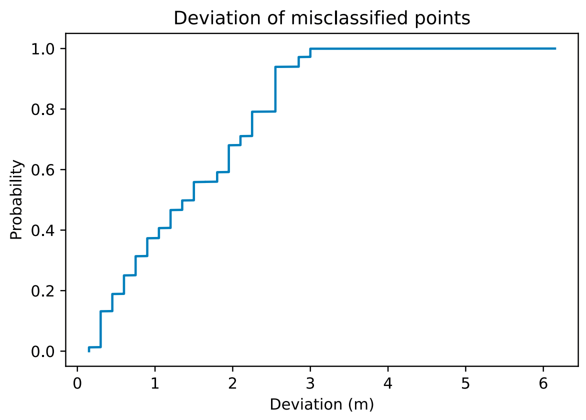


Figure 4.6: CDF of the misclassified instances over the deviation in meters, k-NN

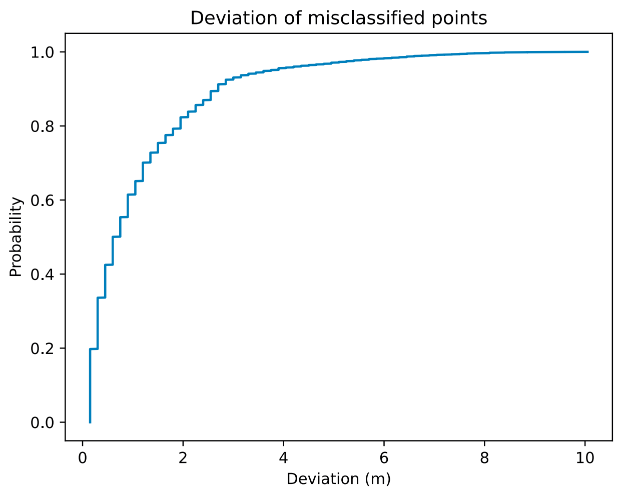


Figure 4.7: CDF of the misclassified instances over the deviation in meters, Naive Bayes

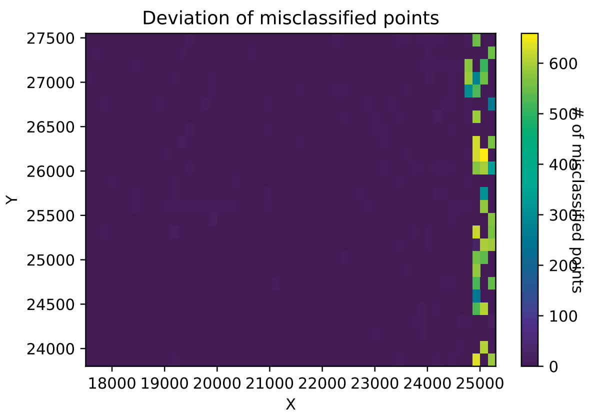


Figure 4.8: The placement of incorrectly classified instances in the k-NN normalized dataset

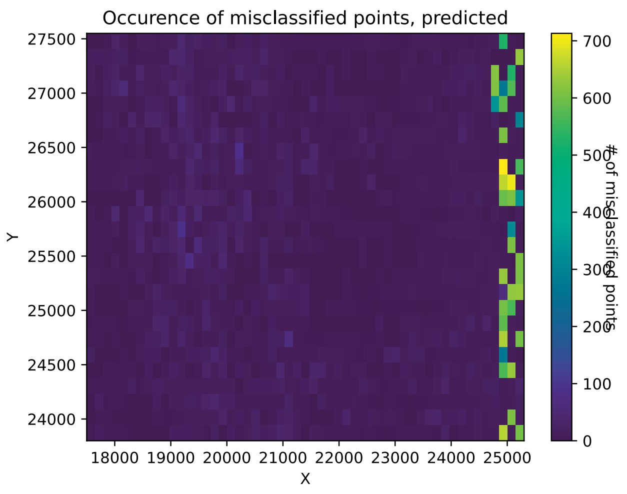


Figure 4.9: The placement of incorrectly classified instances in the k-Nearest Neighbour imputed dataset

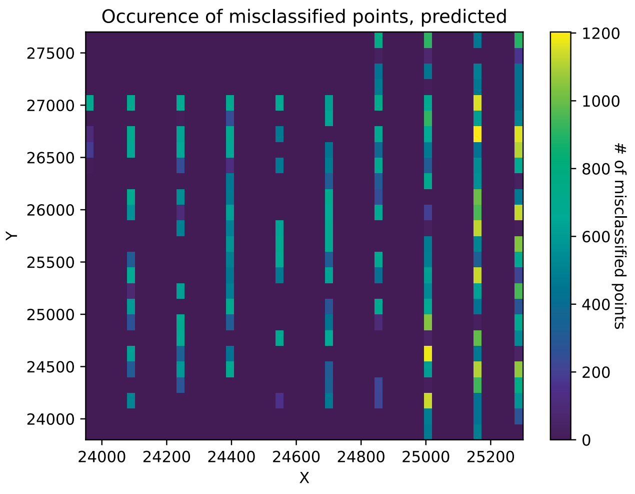


Figure 4.10: The placement of incorrectly classified instances in the Naive Bayes imputed dataset

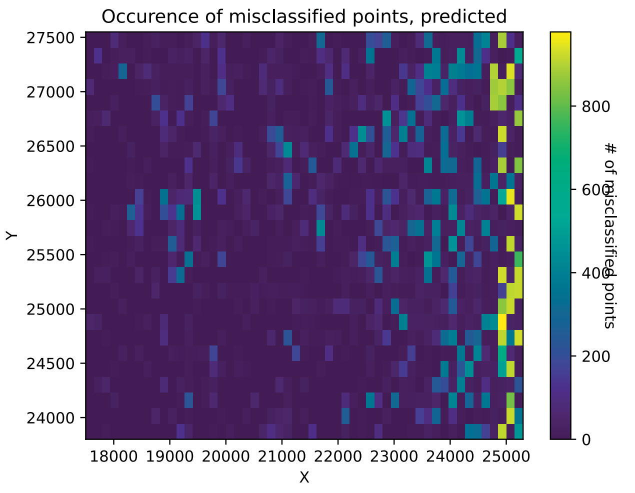


Figure 4.11: The placement of incorrectly classified instances in the Naive Bayes normalized dataset

Chapter 5

Discussion

In this chapter we interpret the results presented in previous chapter 4. We will also evaluate if we fulfill our proof of concept problem statements for this thesis.

5.1 Automation

One of the objectives of this thesis, addressed in 1, is to create an automated data collection setup using software deployed on a NR UE. The proposed system lives up to the stated requirements. However, it is necessary to discuss any weaknesses of the final solution. This section aims to focus on the aspects that are subject for future iterations.

5.1.1 Robot

For this first iteration of the data collection setup, the Xiaomi robot manages to collect the data with a crude coordinate grid. This setup should be possible to expand with more complex experiments, but there are however limitations that need to be addressed.

Positional Replicability

The results of the experiments described in 4.1.1 provides a few insights that are presented in this subsection.

Firstly, the positional accuracy decreases when the robot moves between the same coordinates several times. In the experiments conducted for the data collection, each coordinate is visited one time only. Consequently, the potential source of error is not relevant to consider, since the first experiment is never recreated in the data collection. Choosing to collect more data at one time - rather than smaller amounts in several instances - proves to be a good decision. This needs to be considered, should the setup be used for other experiments.

The map representation that contains the coordinate system is consistent stored within the robot. However, the ability to get as physically close to the coordinate as possible may vary. Tests show that if the robot is not reset, the orientation is slightly affected in terms of replicability. However, this deviation does not accumulate over time. In other words, the margin of error seems to be lower if the robot is restarted between each round.

Regardless, the robot has been continuously rebooted throughout the running course of the data collection. In conclusion, this aspect together with the chosen distance between coordinates should not result in significant noise in the collected data.

Another reason for why adding enough space between coordinates, or risk affecting the replicability of the data collection, is to avoid the imprecision of the Xiami robot. When the robot executes a "go to" command that is too close to its current position, it assumes it has already arrived without moving its position or stops before reaching the coordinate destination.

5.1.2 Kivy Application: Logger

The application operates as intended and the features include, but are not limited to, communication between robot and UE, starting a process for extracting data from UE modem, and exiting the process when requirements are fulfilled. There are some unpredictability regarding the MQTT communication. There are also some inconsistencies with the data extraction of the modem via the application.

For the MQTT communication, the problem seems to largely depend on how MQTT is integrated within Python-for-Android, since the error of inconsistencies retrieving messages from the broker could not be reproduced on a computer system. This also makes it difficult to debug what triggers the error.

Data extraction is controlled in the application using busy-polling as previously mentioned in section 3.7. The sub-process forces termination when the application finds the requirements met. This lead to some variances in the size of the last generated file between different coordinates. For training the model, equal amount of data for all coordinates are crucial to avoid biased training for one feature in the data.

5.1.3 Modem Log Extraction

The data extraction, done using the diag_revealer, works as intended. Given a configuration file that specifies the modem log type, the script is able to read the correct byte streams from the modem and save them on the phone. Several modem log types, spanning from LTE to 5G, have been tested for the configuration files. All of them have worked, which indicates that the diag_revealer can handle any QXDM affiliated modem log type as of today.

However, the current implementation is dependent on specific modems. Out of the tested UE models, the X50 modem on a WNC UE is the only working one. Other models show incompability in diag_revealer related to the drivers. As the MobileInsight project evolves, this aspect may be affected. On the MobileInsight side, more support is likely to be added, especially on the 5G spectra. On the other hand, QC might limit the privileges to read directly from the modem, forcing MobileInsight to change their implementation.

5.2 Input Data

The data used for feeding the machine learning models during training is the most important part for good results. As explained in previous sections, preprocessing of data is a large part for achieving good accurate predictions. In this section we discuss the result from analyzing the data collected with the setup.

5.2.1 Radio Fingerprint

As further described in section 5.3 and presented in section 4.3, the radio fingerprint using RSRP values seem to form uniqueness between different positions for a UE. This is why it is possible to train a model to make predictions based on these values.

This should of course still be expanded upon, since a finer grid of coordinates with less space padded in between may result in lower accuracy. This is a result of the fingerprints having less differences as the distance between the coordinates is reduced, and more noise is introduced due to limitations of the implemented setup.

Currently there is a restriction, as intended by QC's design, where the SS-RSRP values collected only retrieve information for the four strongest SSB beams. This may be changed in the future, but for now there is no way to retrieve information about the weaker beams for the selected modem log type.

5.2.2 SSB Distribution

As stated in section 1, one aspect to consider is the distribution of the beams. This is necessary in order to verify and validate the collected data. By looking at the raw data, it is easier to understand any constraints or unwanted behaviour in the data collection. Additionally we can use those insights to understand the modelling results.

Comparing the occurrence of beam ID's collected to the layout of the SSB ID's is another aspect to discuss. The beams have better coverage in some areas than others, which is shown in the figures in section 4.2. Comparing figures 2.4 and 4.2, some phenomenon are observed.

Firstly, the selected beams - being the most common ones in the data set according to figure B.5 - seem to be placed on the bottom row of the beam array matrix. By changing the tilt of the radio, we might see other beams more frequently in the data set.

Other than that, we observe that an individual beam's line of sight align with their placement on the beam array. For example, compare beam 4 and 5 in figure 4.2, to their position in 2.4. This is valuable information in potential additions to the fingerprints. The topic relates to the conclusions drawn in 5.2.1.

Secondly, we observe a line of sight in our data in B.1, with most energy focused in the top-right region of the experimental area. In contrast, the coverage drops in the area closest to the radio.

5.2.3 Imputation and Missing Data

As mentioned in section 4, empty values are initially filled with default values. There is a marginal difference between the models score for the different data sets for when the set value varied. This shows that the fingerprints in the data set are truly unique and therefore easy to compare.

Regarding imputation, we see that only 1.62% of the missing data is estimated correctly, presented in 4.2.2. In addition, a slight decrease in the scores is observed according to results in 4.3.1. This is likely due to the results of the imputation not being a good enough prediction for missing values. Instead, it seems to slightly throw the model off. However, the difference in the scores does not seem significant.

The reason for the imputation still generating good model predictions, despite the values being so off, is likely connected to the first point in this section. As mentioned, varying the placeholder values do not affect model results. Instead, the significant values, meaning the four strongest beam signals at each point, are able to make the prediction independently. The same principle as for the placeholders applies to the imputation, and the predicted imputed values do not play a notable role.

Ideally, all values would be recorded at each point. With the current modem log type about 84% of the data can be missing, considering the 24 SSB configuration of the 5G radio. However, the results show that four are sufficient to make a prediction.

An illustration of the differences can be observed in figures 4.2 and 4.3 (imputed data). The beams and side lobes are still visible, although the values contains more noise and the contrasts are not as strong as for the unimputed data.

5.3 Model

The models selected for the ML phase are k-NN and Naive Bayes, these two simple algorithms are easy to implement as a first iteration of this setup. Here we compare the two ML models to further evaluate the setup.

5.3.1 K-NN

k-NN does not have a training phase. However, it is interesting to consider how the algorithms performs on known data, in figure 4.4. The reason a score of 1.0 is not obtained when testing on training (in other words, already known data, "Training Accuracy") can be explained by the nature of the data, and by the way k-NN performs the classification. As mentioned in 2.2.3 the data set contains at least two identical instances per coordinate. For more neighbors than one, the algorithm performs a majority vote between the neighbors. If for instance three fingerprints are very similar, the majority may "vote" the wrong class.

However, for one neighbor, a score of 1.0 is expected in figure 4.4. This is an aspect that should be further examined. The model is anticipated to find at least one identical instance in the training data, given that we feed already known instances for prediction. However, for the absolute majority of the instances, the classification is correct, which is why this aspect does not invalidate the model.

5.3.2 Naive Bayes

The overall results of the Naive Bayes model are lower than earlier attempts and previous studies mentioned in section 2.2.3. On possible approach to improve the scores is to introduce hyper-parameters and optimize them. In addition to this, it is possible that the missing data points - that are filled out with default or imputed values - affects the score.

5.3.3 Comparison: k-NN & Naive Bayes

The two models have different trade-offs. For k-NN, the predictions take a lot of time and consumes more resources than Naive Bayes. While for Naive Bayes it achieves a lower accuracy but prediction time is smaller in comparison.

Accuracy

Naive Bayes achieves lower scores on evaluation than k-NN. One possible explanation is the assumption of feature independence described in 2.2.3. The essence in the fingerprint approach is to see relationships between the features, in other words for them to depend on one another. This is contrary to the assumption made for calculating the probabilities.

Another theory for the lower scores, of the Naive Bayes classifier, is the equal contribution of features mentioned in section 2.2.3. The format of the modem log type with only four out of 24 beams collected results in many empty features for the unimputed data set. The four strongest beams - generally the 4 in line of sight of the experiment area - are contributing more for the prediction due to the fact that the other 20 do not exist in our model. This contradicts the equal outcome assumption and therefore the modem log type used may not be optimized for the fingerprint approach.

Misclassified Data

As seen in figure 4.8 through 4.11, the misclassified values are focused to the area of the wooden fence which makes up the enclosure. Since the radio beams are reflected on any surface, the behaviour can be a result of such reflections. The effect is more apparent when using k-NN as opposed to Naive Bayes.

Another observation is seen in figure 4.11, where the Naive Bayes model seems to spread out the reflected effect, although it is still most prominent closer to the enclosure's side. Additionally, the difference between 4.10 and 4.11 visualizes how the dataset is affected by the imputation. For the imputed NB case, the misclassified values are more consistent and not as spread out. The imputation process creates average values of already existing ones. Thus, it can be argued that this is the effect that imputation has on the model interference, where it replicates the error through out the data set with missing values.

Beam Significance

As previously mentioned in 2.1.6, only the four strongest beams are collected in each message. Each row in the data set has 20 empty fields, since the radio used has 24 different SSB beams. One aspect to consider is therefore the significance of the presence of beams in the fingerprint, as opposed to the actual SS-RSRP values.

In other words, is the combination of occurring beam IDs in this case a more unique identifier for a location in the room, in contrast to the levels of recorded signals? One possible approach to answer the question is described in section 6.

5.3.4 Faded Signals

The purpose of the experiment is to assess whether the models can account for signal strength caused by damaged or aged antenna elements. The evaluation scores were well below 0.1%.

One theory to explain the low scores is that since the model looks at the values themselves rather than the relationship between them, there is no way it can account for a change in values.

The discussion presented in this section shows that the proof of concept did show successful: both in terms of a working data collection setup and model performance. However, some aspects need to be investigated further to improve the project for the future, and to open up to real life applications.

Chapter 6

Conclusions and Future Work

This chapter wraps up the different points addressed in previous chapter 5, and presents some suggestion on how to approach to expand upon this work in the future.

6.1 Automation

6.1.1 Robot

The physical size of the Xiaomi S5 is a current limitation for a finer grained coordinate system. With a more accurate robot, this aspect could be ignored. Nonetheless, this aspect is a constraint in understanding just how accurate the positioning is. As a future iteration is is important to examine how the model evaluation scores change, should the grid have a higher resolution (meaning, less space between the coordinates).

The robot manages as a proof of concept to some degree consistently successfully navigate between extraction points. It is easy to operate and install, though some knowledge about the configuration is needed.

For future work there probably is an interest in collecting data from a finer grid of coordinates. For that purpose, the Xiaomi S5 robot probably would have to be exchanged with another robot for navigation. A more precise solution would be to program a robot with necessary sensors, and carefully configure for the precision of its navigation.

However, the current robot model can be used for future research to explore different expansions of the RF fingerprint by collecting more data. For this reason, more experiments are needed to test the error of margin for the robots positioning. Current padding of 15 cm between coordinates adjusts for some lack of the robots precision. In order to interpret the results after processing the data, the knowledge of the limitations of the robot is crucial.

6.1.2 Improve Logger

One alternative for the future is to use a different language for the application development. The problem with inconsistent MQTT communication might also lie with Android systems themselves, and in this case there is not time to explore this option. In that case, it would be necessary to improve the application by exchanging the communication protocol.

Furthermore, busy polling should be addressed in future iterations. One possible solution is the Observer and Observable pattern. However, that would require changing the source code of diag _revealer for the script to notify the observer - that is, the application Logger - when the extraction is finished. Alternatively, checking for the amount of data collected could be a way of achieving this.

The Android application implemented for this thesis properly manages the pipeline of the data collection process. However, some parts as described in section 4.1.3, and elaborated upon in section 5.1.2 creates some noise in the data, and slows down the data collection process.

For the future, a better method for regulating the volume of the data extraction should be explored. Currently, the periodic check based on the system clock results in irregular volumes of data for each coordinate as mentioned in section 4.1.3.

One approach could be to adjust the diag_revealer script to make it notify the application when the extraction is complete. Another one could be to check the file sizes, but then the response of the Android application is still affected due to sleeping threads.

The application could also use more adjustable parameters for choosing container to save the data within the GUI, along with the possibility to change between configuration files when filtering for different messages. Usability is important to make the setup accessible for many to use and helps future research. As such, more features, and developing an intuitive GUI for the application, should be expanded upon in future work.

A different framework might also be good to explore, since there are some limitations with the Kivy framework. This might also solve the inconsistencies with the communication between the MQTT client and broker.

6.1.3 Outcome

The data collection setup executes as intended, we are able to retrieve data extracted from the modem through an automated process. This allows for many possible expansions upon this thesis in the future.

It is also important to try and improve the usability of the setup. A result of having easier access to operate it, more may continuing the expand upon the work from this thesis.

6.2 Modem Log Extraction

As mentioned in 5, the current modem log is a limitations since it only offers the SS-RSRP values of the four strongest beams. Finding a modem log type that is more detailed is one possible way forward in the process of building a more robust fingerprint. It is suitable to find a modem log type that records all beam ID values and form fingerprints with all 24 fields. Then, after modelling the new fingerprints, validate by comparing the new model's performance with the current figures presented in 4. Additionally, data from several modem log types may be combined to one fingerprint.

Editing the diag_revealer script is another way to improve the modem log extraction step. The current configuration of the setup is dependent on the MI framework. This proved to be a limitation in terms of what UE models could be used. The setup is currently only compatible with a WNC 5G prototype using a X50 modem out of the models that were tested. By cherry-picking the functionality in the diag_revealer, and expanding upon existing features that is used to extract the modem logs and messages, it is possible that the modem log extraction process can be extended to work on more and newer 5G high-band UE models.

With MobileInsight being an open source-project, it's under constant development. The beta with 5G support was released late 2020. During the running course of this project, more functionality has been released in terms of supported models and bug fixes for the offline analysis. Potentially, this offers more stability to the pipeline, making it possible to extend the work to other UE modems and log types. Also, the relevance of the work may increase since this thesis offers a solution to collect and analyze data in a full setup.

However, the work requires the diag_revealer, developed in MobileInsight, being independently executable. One risk is that the MI team restricts the C-binary to run only from the MI app, in which case the setup is not usable. In addition, it depends on configuration files from QXDM, and MI may restrict or change the format of those to the log types implemented in the application. The general approach relies heavily on MobileInsight in its current state, which may be considered a weakness. All in all, the development of MI must be closely followed to determine how this project may be affected by any changes made to the original one.

6.3 Model

The models in this thesis are used as a proof of concept to test and evaluate prediction of the indoor positioning with 5G RF fingerprints consisting of SS-RSRP values. The models manages to predict with a higher probability than guessing the position.

For future expansion of the ML phase, we propose to use a similar model described in article [23]. The authors of the article manages to achieve high accuracy with low error margins.

In section 4.3.2 an experiment is presented where the trained models are faced with faded data. Since those model scores are very low, it can be concluded that the current model setups are insufficient for that task. The reasoning being that both k-NN and NB looks at the actual feature values, as opposed to the relationship between them. Instead, a suggestion for the future is to try ML techniques that can account for the feature dependencies. That way it is possible to further investigate using UE models with damaged or aged antenna elements.

In conclusion, a natural step is to find more models and see how they perform with regards to the aspects presented.

6.3.1 Accuracy

Due to the fact that k-NN is an uncomplicated algorithm, it may yield lower scores for the accuracy as a trade-off for the simplicity. The conclusion is therefore that testing other models and compare performance is an important step in future work.

The k-NN model manages with high accuracy to predict the indoor positioning of the UE. The accuracy may be negatively impacted if the distance between the coordinates is reduced. This could be countered with expanding the fingerprint as mentioned in previous sections. The k-NN model is a good baseline to compare other models to, since it is a simple algorithm easy to understand and evaluate.

For the Naive Bayes classifier, it achieved a lower F1-score and accuracy. It may be that the classifier is a poor choice for the nature of our data. The SS-RSRP values seem to have some dependency between different SSB beams. Some adjustments to the Naive Bayes classifier may be enough to also raise the accuracy of the models prediction.

Additionally, the imputation process needs to be reviewed as to improve the score rather than lower it. It is possible that small adjustments and considerations is sufficient. Another way is to find a better log type that records more than just the four strongest beams in each message, as mentioned above in section 6.2 Modem Log Extraction.

6.3.2 Misclassification

As mentioned in 4 the distribution, of the misclassified data points shown in figure 4.8, is skew. All falsely classified instances are located along the short side of the enclosure. An explanation for that behaviour is signal reflections in the rink. It can be hypothesized that the fingerprint values are thrown off due to interfering reflection signals, with the radio facing that particular part of the enclosure. Thus, one suggestion to minimize such patterns in the data is to use a material that absorbs signals.

6.3.3 Outcome

In the introduction, section 1, it is stated that a goal for the thesis is to see if we can improve the results of earlier attempts [19][12][4][18]. Although the obtained scores do not surpass the referenced articles, the approach in this thesis is not identical to the above. Therefore we may conclude that, as a proof of concept, it is possible to use SS-RSRP values in fingerprints for indoor positioning.

References

[1] Luai Al Shalabi and Zyad Shaaban. Normalization as a preprocessing engine for data mining and the approach of preference matrix. In *2006 International Conference on Dependability of Computer Systems*, 2006.

[2] Arar, Steve. Fixed-point representation: The q format and addition examples. `https://www.allaboutcircuits.com/technical-articles/fixed-point-representation-the-q-format-and-addition-examples/`, accessed: 2021-05-30, 2017.

[3] Badman, Lee and Fitzgibbons, Laura. Definition 5g new radio (nr). Online. *"TechTarget"*, `https://whatis.techtarget.com/definition/5G-New-Radio-NR`, accessed: 2021-05-30.

[4] Sinem Bozkurt, Gulin Elibol, Serkan Günal, and Ugur Yayan. A comparative study on machine learning algorithms for indoor positioning. *2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)*, 2015.

[5] Vladimir Brik, Suman Banerjee, Marco Gruteser, and Sangho Oh. Wireless device identification with radiometric signatures. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*. Association for Computing Machinery, 2008.

[6] Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong. *Mathematics for Machine Learning*. Cambridge University Press, 2020.

[7] *"Ericsson"*. Ericsson white papers: Advanced antenna systems for 5g networks. Online, 2018. `https://www.ericsson.com/en/reports-and-papers/white-papers/advanced-antenna-systems-for-5g-networks`, accessed: 2021-05-30.

[8] *"everything RF"*. 5g frequency bands in europe. Online. `https://www.everythingrf.com/community/5g-frequency-bands-in-europe`, accessed: 2021-06-22.

[9] Alex et. al Finkel. Optimizing indoor location recognition through wireless fingerprinting at the ian potter museum of art. In *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2014.

[10] *"Gartner"*. Gartner glossary: Ib (integration broker). Online. `https://www.gartner.com/en/information-technology/glossary/ib-integration-broker`, accessed: 2021-05-30.

[11] Giese, Dennis. Dustcloud. Online. `https://dontvacuum.me/`, accessed: 2021-02-25.

[12] Emre Gönültaş, Eric Lei, Jack Langerman, Howard Huang, and Christoph Studer. Csi-based multi-antenna and multi-point indoor positioning using probability fusion, 2020.

[13] *"Home Assistant"*. Xiaomi miio. Online. `https://www.home-assistant.io/integrations/xiaomi_miio/#xiaomi-mi-robot-vacuum`, accessed 2021-05-30.

[14] Jaokar, Ajit. Understanding the applications of probability in machine learning. `https://www.datasciencecentral.com/profiles/blogs/understanding-the-applications-of-probability-in-machine-learning`, accessed: 2021-05-30, 2019.

[15] G. Kesavaraj and S. Sukumaran. A study on classification techniques in data mining. In *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, 2013.

[16] Wooyoung et. al Kim. A comparison of the effects of data imputation methods on model performance. In *2019 21st International Conference on Advanced Communication Technology (ICACT)*, 2019.

[17] J. Laaksonen and E. Oja. Classification with learning k-nearest neighbors. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, 1996.

[18] Reza Firsandaya Malik, Eko Pratama, Huda Ubaya, Rido Zulfahmi, Deris Stiawan, and Kemahyanto Exaudi. Object position estimation using naive bayes classifier algorithm. In *2018 International Conference on Electrical Engineering and Computer Science (ICECOS)*, 2018.

[19] Piotr Mirowski, Dimitrios Milioris, Philip Whiting, and Tin Kam Ho. Probabilistic radio-frequency fingerprinting and localization on the run. *Bell Labs Technical Journal*, 2014.

[20] *"MobileInsight"*. How to develop a message parser? Online. `http://www.mobileinsight.net/message-parser.html`, accessed: 2021-06-03.

[21] *"MobileInsight"*. Who are we - mobileinsight official website. Online. `http://www.mobileinsight.net/about.html`, accessed: 2021-06-04.

[22] *"MQTT: The Standard for IoT Messaging"*. Online. `https://mqtt.org/`, accessed: 2021-05-30.

[23] Nimish, Mishra. Probability theory for machine/deep learning. `https://towardsdatascience.com/probability-theory-for-deep-learning-9551b9255cf0`, accessed: 2021-05-30, 2019.

[24] *"OASIS Open"*. Online. `https://www.oasis-open.org/org/`, accessed: 2021-05-30.

[25] Hadeel S. Obaid, Saad Ahmed Dheyab, and Sana Sabah Sabry. The impact of data pre-processing techniques and dimensionality reduction on the accuracy of machine learning. In *2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)*, 2019.

[26] Ambili et. al Parameswaran. Is rssi a reliable parameter in sensor localization algorithms: an experimental study. January 2009. `https://cse.buffalo.edu/srds2009/F2DA/f2da09_RSSI_Parameswaran.pdf`, accessed: 2021-05-30.

[27] *"Qualcomm"*. About qualcomm, overview. Online. `https://investor.qualcomm.com/about-qualcomm/overview`, accessed: 2021-05-30.

[28] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition, 2010.

[29] Sauter, Martin. *Universal Mobile Telecommunications Systems (UMTS) and High-Speed Packet Access (HSPA)*, chapter 3.7. John Wiley & Sons, Ltd, 2010.

[30] Schoutsen, Paulus. Online. `https://paulusschoutsen.nl/`, accessed: 2021-06-02.

[31] James G. Shanahan. *Machine Learning*. Springer US, 2000.

[32] Yan Shi and Michael A. Jensen. Improved radiometric identification of wireless devices using mimo transmission. *IEEE Transactions on Information Forensics and Security*, 6(4), 2011.

[33] Siegel, Camille and Dorairajan, Arun. What is an api? Online. `https://apifriends.com/api-management/what-is-an-api/`, accessed: 2021-06-03.

[34] K. Smith. *Precalculus: A Functional Approach to Graphing and Problem Solving*. The Jones & Bartlett learning series in mathematics. Jones & Bartlett Learning, 2013.

[35] *"Stanford University"*. Stanford encyclopedia of philosophy: Bayes' theorem. Online, 2003. `https://plato.stanford.edu/entries/bayes-theorem/`, accessed: 2021-05-30.

[36] *"TechTerms"*. Gigo. Online. `https://techterms.com/definition/gigo`, accessed: 2021-06-04.

[37] Teral, Stephan. 5g best choice architecture. Technical report, IHS Markit Technology, 2019. `https://res-www.zte.com.cn/mediares/zte/Files/PDF/white_book/5g-best-choice-architecture.pdf`, accessed: 2021-05-30.

[38] *"The Official Yaml Website"*. Online. `https://yaml.org/`, accessed: 2021-05-30.

[39] *"Xiaomi"*. Roborock robotic vacuum cleaner manual. Online. `https://www.mistore.se/sv/manualer`, accessed: 2021-05-30.

[40] Harry Zhang. The optimality of naive bayes. *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*, 2004.

Populärvetenskaplig Sammanfattning

Har du någon gång gått vilse i ett köpcentrum och känt dig sviken av din mobiltelefons GPS-funktion? Det är en universell upplevelse, som skulle kunna vara möjlig att lösa med hjälp av maskininlärning och 5G.

Den smarta tekniken är ett allt mer vanligt förekommande inslag i vårt samhälle. Allt från smarta dammsugare till inbrottslarm och mobiltelefoner som förenklar i vår vardag. I takt med att högre krav ställs på enheternas svarstid och pålitlighet behövs 5G-tekniken. En aspekt i detta område är att med hög noggrannhet kunna bestämma var någonstans en enhet befinner sig. Utomhus går det att använda satelliter med GPS-teknik för lokalisering. Inomhus är denna typ av lokalisering inte möjlig då satellitsignaler sällan ger någon bra täckning. Vi har undersökt en metod för att samla 5G data i syfte att sedan kunna förutse var någonstans en smartenhet befinner sig i en inomhusmiljö. Genom att öka noggrannheten i inomhuspositionering med 5G så öppnas möjligheten för fler tillämpningar i framtiden för smarta enheter.

Arbetet består av tre olika faser: datainsamling, databearbetning och tolkning av resultaten. I den första fasen har vi med hjälp av en smartdammsugare och en 5G-telefon konstruerat ett system som kan samla in data helt utan översyn. Datan innerhåller dels ett så kallat fingeravtryck, som är en unik signatur för 5G-signalen i en viss tidpunkt. Dessutom finns det för varje fingeravtryck en koordinatposition, det vill säga den punkt i rummet där 5G-telefonen befinner sig.

I den andra fasen undersöker vi olika maskininlärningstekniker. Målet är att hitta den metod som med högst noggrannhet kan gissa var någonstans vår enhet har befunnit sig, med hjälp av de olika fingeravtrycken. Den sista fasen går slutligen ut på att evaluera hur bra maskininlärningen presterat, och hur exakt den kan gissa en position baserat på ett fingeravtryck.

Syftet med examensarbetet är att testa om det är möjligt att genomföra en prognos, baserat på den insamlade mobildatan, var i rummet mobilenheten befinner sig. Vår slutsats är att det är möjligt att uppskatta en inomhusposition med hjälp av fingeravtrycket i 5G-signalen. Genom att fortsätta utveckla mer sofistikerade insamlingsmetoder och maskininlärningsalgoritmer bör det vara möjligt att minska felmarginalerna ytterligare. Ännu ett exempel på förbättring är att fundera på hur just fingeravtrycket skulle kunna göras ännu mer unikt. Avslutningsvis vore det intressant att undersöka flera fingeravtryck i samma punkt kan förbättra exaktheten.

Möjligheten att lokalisera en mobil enhet med hjälp av dess mobildata kommer öppna upp för fler tillämpningar inom smart teknik. Dessutom kan det också vara användbart i att optimera hur 5G-signalen skickas ut i ett rum, för att få starkast möjliga mottagning. Fler enkla metoder för datainsamling kommer dessutom göra denna typ av undersökningar lättare samt tillgängliga för fler. Förhoppningsvis kommer detta leda till mer innovation inom området.

Vår datainsamlingsassistent - Seshat

Appendices

Appendix A

Figures

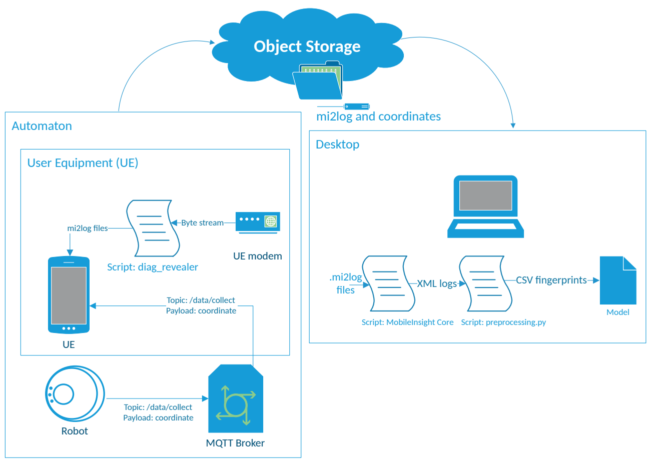


Figure A.1: Data flow

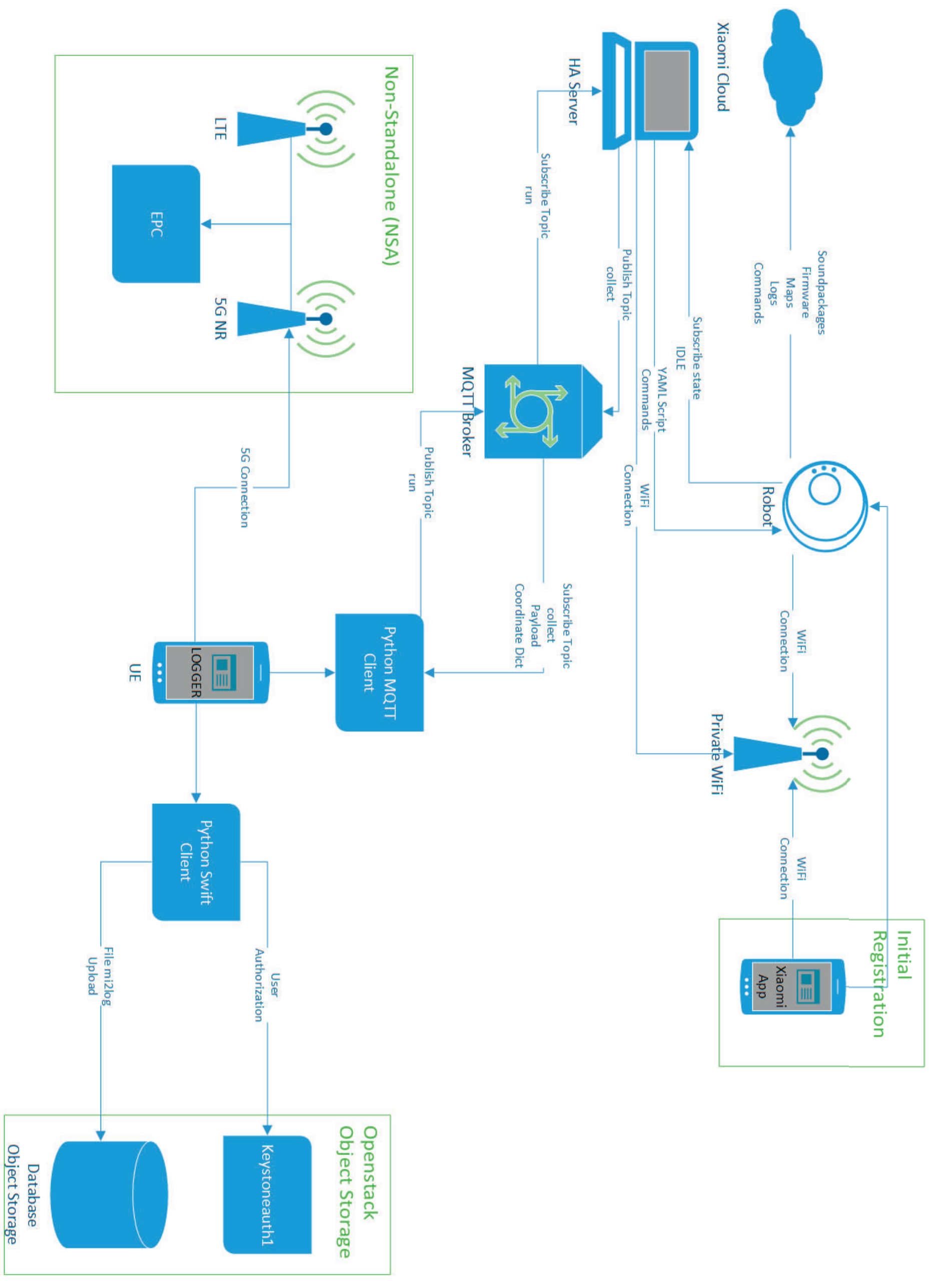


Figure A.2: Full system architecture.

Appendix B
Data Properties

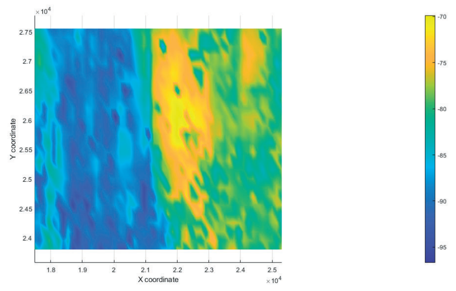


Figure B.1: Envelope showing energy distribution of experimental area, top view, incomplete dataset.

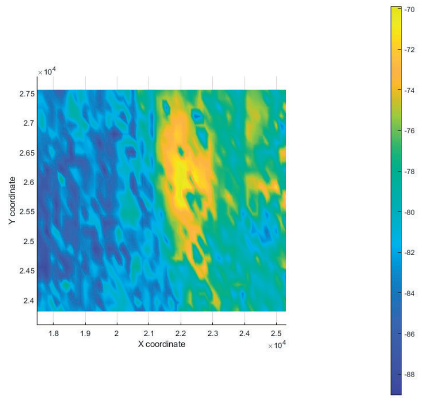


Figure B.2: Envelope showing energy distribution of experimental area, top view, complete (imputed) dataset.

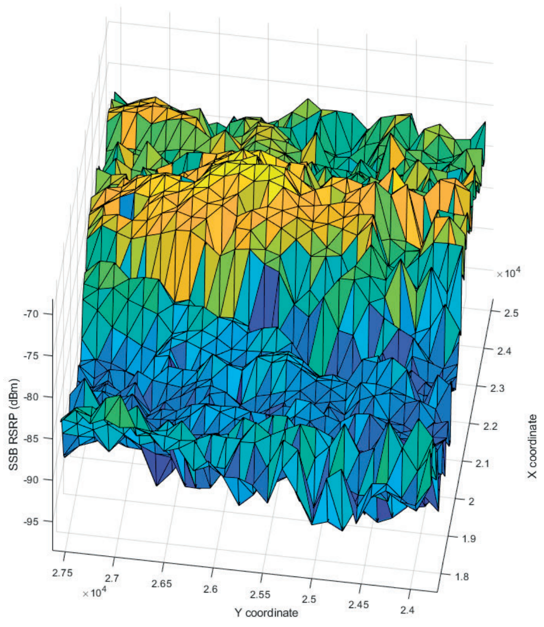


Figure B.3: Envelope showing energy distribution of experimental area slightly overhead from a 5G radio perspective, incomplete dataset.

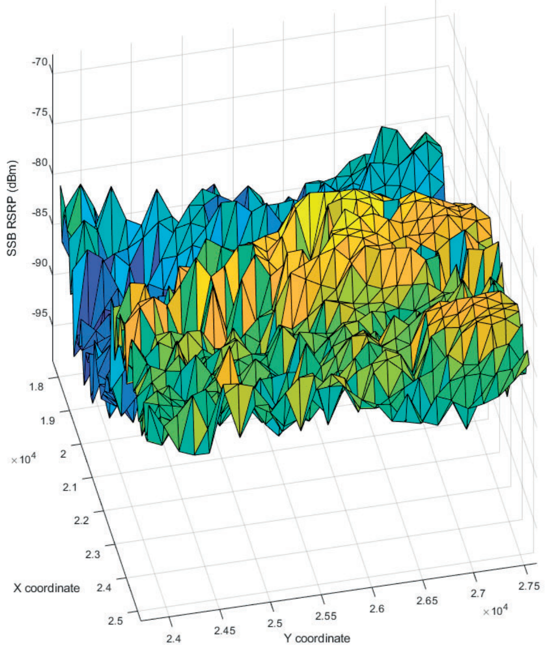


Figure B.4: Envelope showing energy distribution of experimental area in perspective of the robot charging station, incomplete dataset.

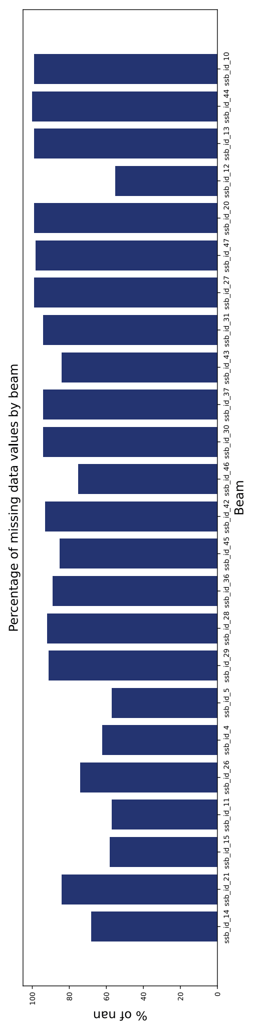


Figure B.5: Percentage of missing data per beam

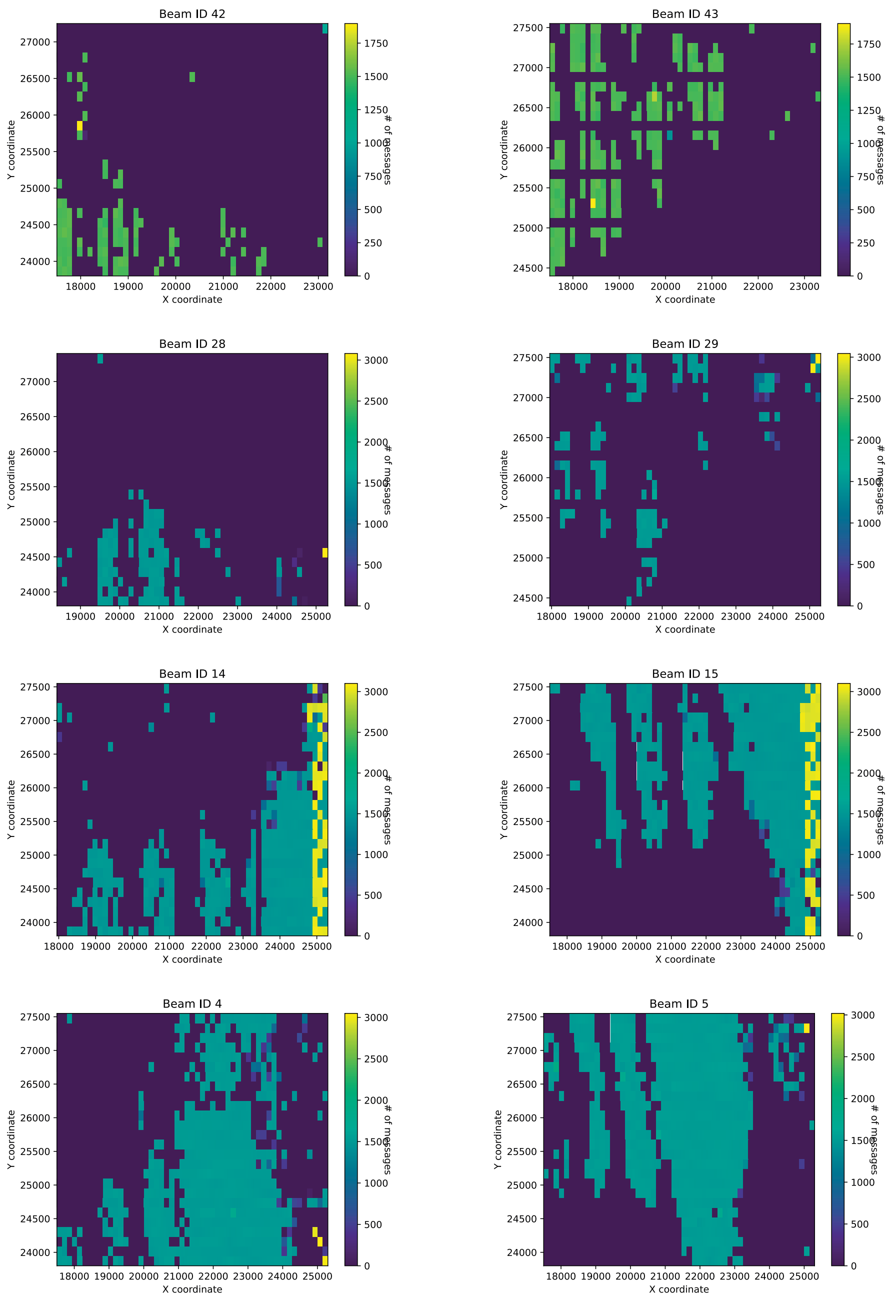


Figure B.6: First two columns in the antenna array

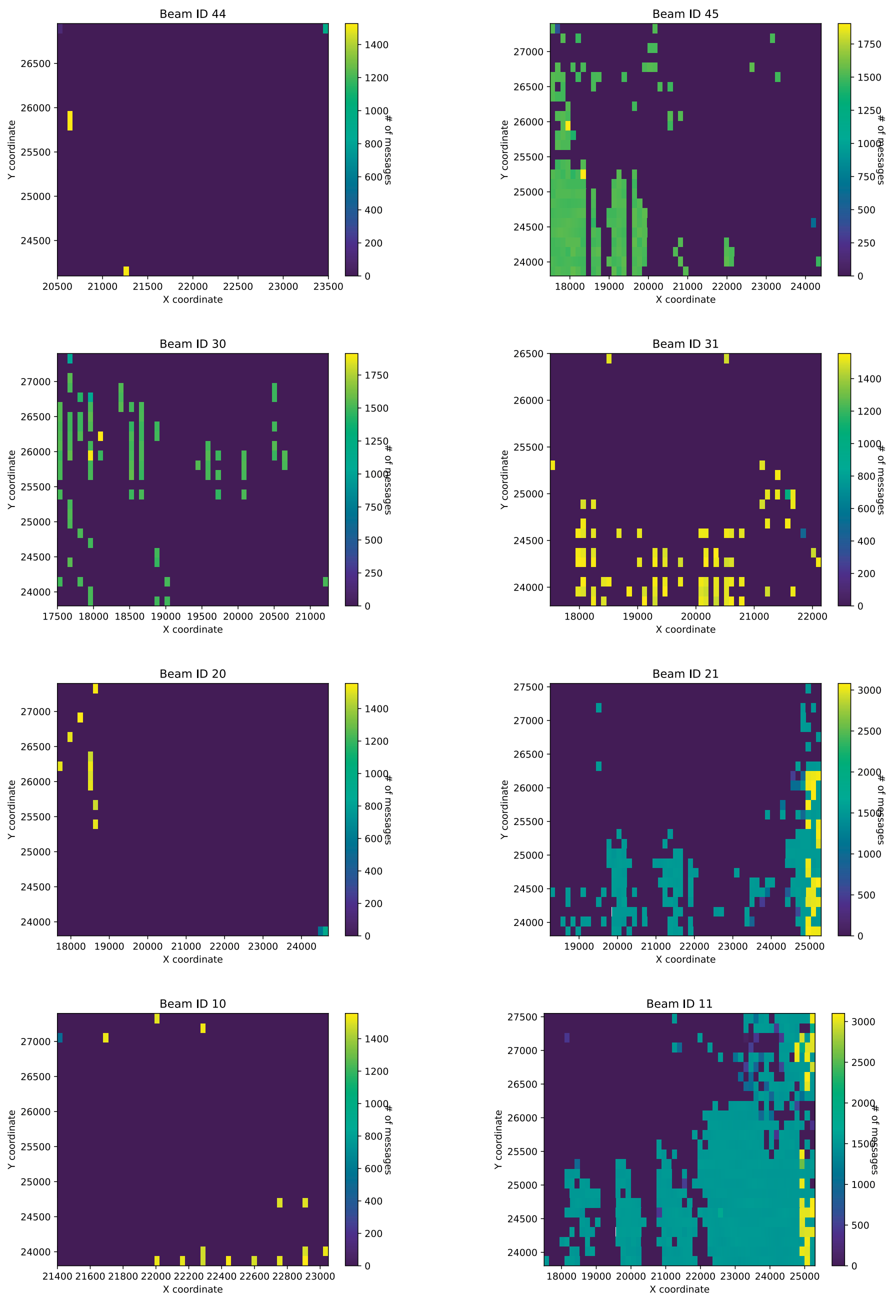


Figure B.7: Third and fourth columns in antenna array

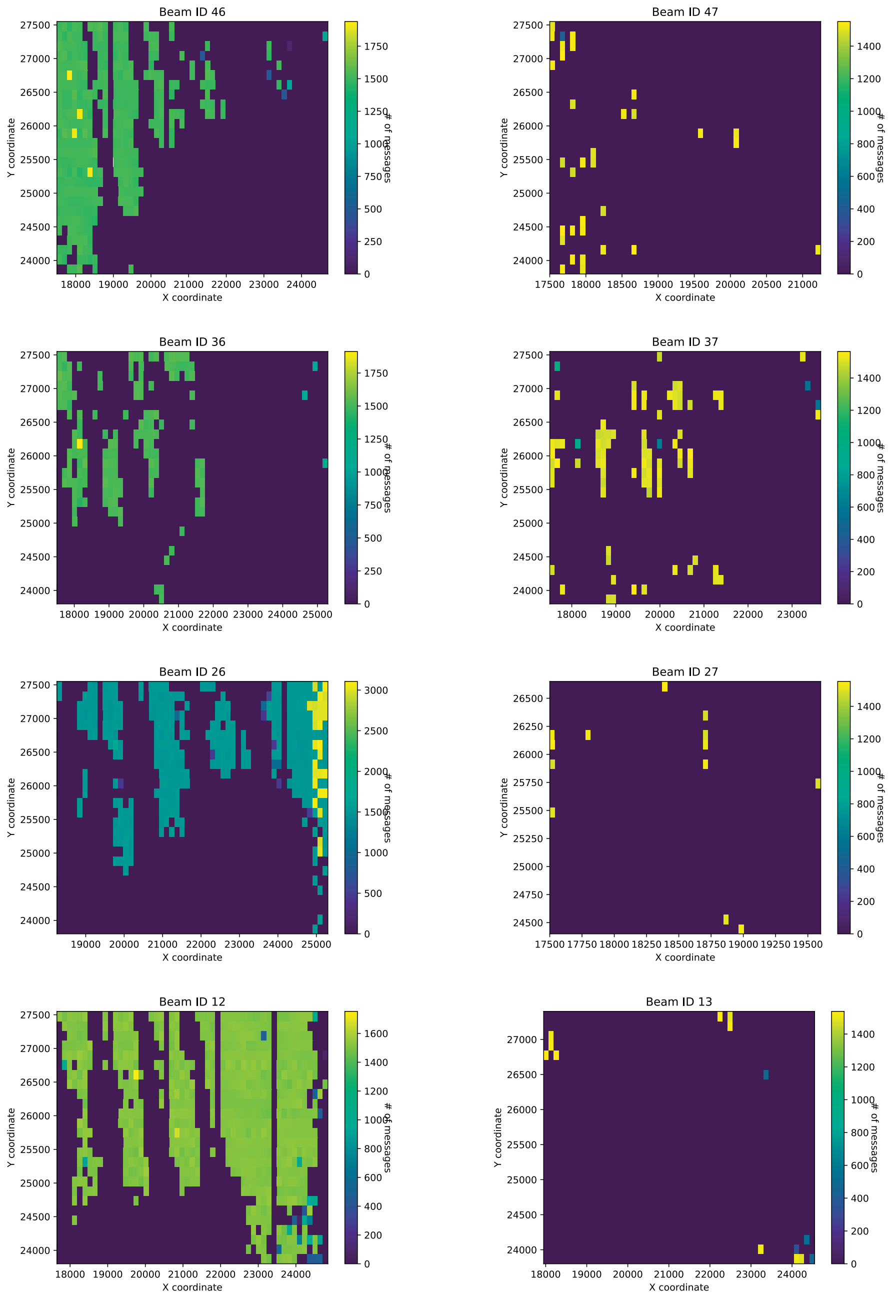


Figure B.8: The two last columns in the antenna array

Printed by Tryckeriet i E-huset, Lund 2022

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2022-880
http://www.eit.lth.se