Master Thesis

Design and implementation of a communication protocol and system for communication between the central control system and the urban service robot MURMEL

A thesis submitted in partial fulfillment of the requirement for the Degree of Master of Science in Mechanical Engineering

at the

Technische Universität Berlin
Institute of Machine Design and Systems Engineering
Department Methods for Product Development and Mechatronics

Submitted by: Yilmaz Naci Aslan 0377809

Submitted on April 24, 2021

Supervisor: Prof. Dr.-Ing Dietmar Göhlich
Abhishek Gupta M.Sc.

Disclaimer

☑ ”I hereby agree that my study, bachelor or master thesis may be displayed in the institute and university library and stored for inspection.”

□ ”I hereby declare that I do not agree to my study, bachelor’s or master’s thesis being displayed and viewed in the institute and / or university library.“

I have taken note of the choice to display my study, bachelor’s or master’s thesis in the institute and / or university library:

Berlin, April 24, 2021

. . . . . .

Yilmaz Naci Aslan

Affirmation in lieu of oath

I hereby declare in lieu of an oath that I have produced the aforementioned thesis independently and without using any other than the aids listed. Any thoughts directly or indirectly taken from published and unpublished texts are made discernible as such. To date, the thesis has not been submitted to any other board of examiners in the same or a similar format and has not been published yet.

Berlin, April 24, 2021

. . . . . .
Yilmaz Naci Aslan

Acknowledgments

I would like to express my gratitude to all the people that have been involved in helping and supporting me during the work of this thesis. In this regard, I would like to thank Prof. Dr.-Ing Dietmar Göhlich for his valuable feed-backs that he provided in the development phase of the project. I also would like to thank M.Sc. Abhishek Gupta for all his friendly and helpful disposition, which helped me a lot to keep my motivation at a high level throughout the semester.

Fachgebiet Methoden der Produktentwicklung und Mechatronik
Prof. Dr.-Ing. D. Göhlich

Institut für Maschinenkonstruktion und Systemtechnik
Fakultät V - Verkehrs- und Maschinensysteme
Technische Universität Berlin

Masterarbeit für Yilmaz Naci Aslan

„Entwurf und Umsetzung eines Kommunikationsprotokolls und Systems zur Kommunikation zwischen dem zentralen Kontrollsystem und dem Urbanen Service Roboter MURMEL“

Die Entwicklung der Großstädte über die letzten Jahre zeigt ein Wachstum auf, das sich auch in Zukunft weiter fortsetzen wird, wie aus einer Studie des Bundesinstituts für Bau-, Stadt-, und Raumforschung von 2017 hervorgeht. Darunter befindet sich auch Berlin als eines der Ballungszentren Deutschlands. Mit wachsender Größe und Einwohnerzahl steigen auch die Bedarfe und Anforderungen an die städtischen Institutionen wie den öffentlichen Nahverkehr, die Stadtreinigung und den Lieferverkehr. Um die steigende Nachfrage an Personen- und Gütertransport sowie Services im urbanen Raum weiterhin bedienen zu können, wird es notwendig sein, die Effizienz durch beispielsweise die Erhöhung des automatisierungsgrades zu steigern. Eine nachhaltige Entwicklung der städtischen Gebiete ist daher einer der Kernbausteine für ein zukunftsfähiges Energiekonzept und die allgemein anerkannten Klimaschutzziele.
In diesem Zusammenhang spielt das Voranschreiten der Elektromobilität und die Entwicklung autonomer Fahrzeuge eine wichtige Rolle. Auch die Mobilität abseits der Straße wird ein wichtiges Thema der Zukunft sein. Gerade in den Bereichen Güterverkehr und Service gibt es bereits innovative Ansätze, automatisierte Anwendungen zu nutzen, um beispielsweise der Last-Mile-Problematik zu begegnen oder Aufgaben im Dienstleistungssektor oder der Pflege zu übernehmen.
Auch bei der Stadtreinigung gibt es erste Ideen, einzelne Leistungen durch autonome oder teilautonome Fahrzeuge zu ersetzen oder zu unterstützen. Die Entsorgungsaufgaben setzen sich dabei aus unterschiedlichen Tätigkeiten wie dem Mülleimerleeren, Müllsammeln oder Straßenkehren zusammen.
Die Automatisierung bietet Potentiale zur Effizienzsteigerung und dem Einsparen von lokalen CO2-Emissionen durch das Übernehmen von Aufgaben, die momentan mit Hilfe von Kraftstoff betriebenen Nutzfahrzeugen erledigt werden.
Diese Arbeit erfolgt im Rahmen des Projektes zur Methodischen Entwicklung von mobilen Servicerobotern für Entsorgungsaufgaben im urbanen Umfeld. Um unterschiedliche Funktionen im Bereich Stadtreinigung zu automatisieren, soll das Konzept einer modularen Plattform genutzt werden und als Träger für entwickelte Servicemodule dienen. In Zusammenarbeit mit der Berliner Stadtreinigung (BSR) wird untersucht, welche Anforderungen sich aus den unterschiedlichen Tätigkeiten ableiten und welche Schritte die methodische Entwicklung solcher Module beinhalten muss.

Die Kommunikation und Koordination der MURMEL-Roboter ist eine der die wichtigen Herausforderungen im MURMEL-Projekt. Während des Betriebs eine Aufgabe ist es entscheidend, eine ständige Verbindung zu einem Roboter zu haben, um einen nahtlosen Betrieb zu erreichen. Um die Roboter flexibler programmieren zu können, ist die benötigte Zustandsinformationen anderer MURMELs können nur über einen Kommunikationskanal genutzt werden. Durch die Herstellung einer drahtlosen Verbindung zwischen einem Roboter und einem zentralen System kann es möglich sein, den Zustand der Roboter, indem sie die missionskritischen Informationen aus MURMELs abrufen.

Ziel dieser Arbeit ist die Entwicklung eines ersten Konzepts für die Kommunikation und die Umsetzung und Validierung erster Komponenten. Der Umfang der auszuführenden Komponenten soll den Roboter befähigen, gesammelte Information des aktuellen Stands von MURMEL mit einem zentralen Kontrollsystem zu ermitteln.

Schwerpunkte der Arbeit:

1. Recherche und Einarbeitung:
    - Recherche: Aktuelle Kommunikationssysteme und Protokolle, die in verschiedenen Outdoor-Robotern verwendet werden.
    - Recherche: Die Anforderungen an die Implementierung, die Funktionalität und die Zuverlässigkeit der oben untersuchten Systeme.
    - Recherche: Softwareentwicklungsmethoden für die Implementierung dieser Kommunikationssysteme
    - Einarbeitung in den bisherigen Stand des Projekts MURMEL

2. Methodik:
    - Methodischer Konzeptentwurf des mechatronischen Subsystems Kommunikation für MURMEL (V-Modell, Pahl/Beitz, VDI-2221, etc.)
    - Erstellen einer Übersicht aller geplanten mechatronischen Komponenten und deren Schnittstellen
    - Methodischer Konzeptentwurf der Software unter Einbeziehung der erstellten Übersicht
    - Evaluieren der erstellten Konzepte mit anschließender Auswahl einer Vorzugsvariante

3. Umsetzung:
    - Umsetzung des ausgewählten Konzepts zur Validierung der Funktionalität
    - Umsetzung eines modularen Schnittstellenprogramms zur Kommunikation von Aktoren und Sensoren sowie Einbindung dieses Software-Moduls ins Hauptsystem

Betreuer: Abhishek Gupta, M. Sc.

Berlin, den 26.11.2020

Göhlich

Prof. Dr.-Ing. Dietmar Göhlich
Leiter des Fachgebiets

TU Berlin • Sekr. H 10 • Methoden der Produktentwicklung und Mechatronik • Straße des 17. Juni 135 • D-10623 Berlin
Tel. (030) 314-23341 • Fax: -26481 • E-Mail: sekretariat@mpm.tu-berlin.de

Abstract

The increasing population in cities presents new challenges and increases the burden on municipal services such as public transport, city cleaning, and waste disposal systems. In order to adapt these services to changing conditions, it is necessary to improve the efficiency of the operation by increasing the rate of autonomous processes in the services provided. In this context, the MURMEL project aims to develop electric, autonomous service robots to reduce CO2 emissions of diesel-powered trucks used to empty garbage cans in Berlin. Since the communication and coordination of MURMEL robots are seen as one of the most critical challenges in the MURMEL project, this study aims to develop a solution architecture that meets the project's communication requirements. During the development phase, the latest developments in IoT technologies and smart city applications will be examined, and a highly efficient communication network will be developed in line with the project needs. Via the developed communication network, route information, sensor values, and actuator states will be instantly shared between robots, waste bins, and the central management center. Also, by integrating the OTA (over the air) update feature into the system, it will be possible to send software updates to robots without physical human interaction. In addition to online monitoring and software update features, it is aimed to minimize operational errors by accessing the robot's camera in emergencies and controlling the autopilot mode. As an output of the communication network developed, robot operators will be able to access all the developed services through the MURMEL application server with a user-friendly interface.

Abstract

Die wachsende Bevölkerung in Städten stellt neue Herausforderungen und erhöht die Belastung für kommunale Dienstleistungen wie öffentliche Verkehrsmittel, Stadtreinigung und Abfallentsorgungssysteme. Um diese Dienste an sich ändernde Bedingungen anzupassen, ist es notwendig, die Effizienz des Betriebs zu verbessern, indem die Rate autonomer Prozesse in den bereitgestellten Diensten erhöht wird. In diesem Zusammenhang zielt das MURMEL-Projekt darauf ab, elektrische, autonome Serviceroboter zu entwickeln, um den CO2-Ausstoß von dieselbetriebenen Lastkraftwagen zu reduzieren, mit denen in Berlin Mülleimer geleert werden. Da die Kommunikation und Koordination von MURMEL-Robotern als eine der kritischsten Herausforderungen im MURMEL-Projekt angesehen wird, zielt diese Studie darauf ab, eine Lösungsarchitektur zu entwickeln, die die Kommunikationsanforderungen des Projekts erfüllt. Während der Entwicklungsphase werden die neuesten Entwicklungen bei IoT-Technologien und Smart-City-Anwendungen untersucht und ein hocheffizientes Kommunikationsnetzwerk entsprechend den Projektanforderungen entwickelt. Über das entwickelte Kommunikationsnetz werden Routeninformationen, Sensorwerte und Aktorzustände sofort zwischen Robotern, Abfallbehältern und dem zentralen Verwaltungszentrum ausgetauscht. Durch die Integration der OTA-Update-Funktion (Over-the-Air) in das System ist es außerdem möglich, Software-Updates ohne physische menschliche Interaktion an Roboter zu senden. Zusätzlich zu den Funktionen zur Online-Überwachung und Software-Aktualisierung sollen Betriebsfehler minimiert werden, indem in Notfällen auf die Kamera des Roboters zugegriffen und der Autopilot-Modus gesteuert wird. Als Ergebnis des entwickelten Kommunikationsnetzwerks können Roboterbetreiber über den MURMEL-Anwendungsserver mit einer benutzerfreundlichen Oberfläche auf alle entwickelten Dienste zugreifen.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

With increasing size and population, the needs and requirements for urban institutions such as public transport, urban cleaning, and delivery are continuously growing. To overcome these new challenges in urban areas, improving operations' efficiency by integrating more autonomous services becomes necessary. Therefore improving the operational efficiency of waste management systems is a step forward to achieve a sustainable environment.

In cooperation with the Berliner Stadtreinigung (BSR), Project MURMEL has been set in motion by focusing on mobile service robots' methodical development for disposal tasks in urban environments. Under the scope of the project, in order to automate different functions in the area of city cleaning, a concept of service robot has been developed to carry service modules. In this context, communication and coordination of MURMEL robots stand out as some of the most important challenges in the MURMEL project. While operating a task, it is crucial to have a constant connection to robots to achieve seamless operation. During the process of a task, having continuous communication between robots and the central control station can significantly help minimize fatal errors and improve robot operators' control of robots. By establishing a wireless communication network between robots, dustbins, and a central management system, states of sensors, actuators, and route relevant information can be shared simultaneously.

The following work aims to develop a concept that can meet the communication needs of the project MURMEL. The primary approach to solve the problem will be focusing on solutions and technologies under the smart city concept's scope. According to the report *Smart City Strategy Berlin* published by **Senate Department for Urban Development and Environment**[52], an increasing number of cities and metropolitan areas around the world are embracing the "Smart City" concepts. This trend has also been set in motion in Berlin by the rapid development of powerful digital **information and communication technologies (ICT)**[52]. In this respect, during the design of a communication concept for the project MURMEL, emerging wireless communication technologies and communication protocols will be researched in detail.

When building up the communication stack, **economical sustainability** will be an essential factor in the design phase in addition to technical aspects. Considering the project's future scope where numerous robots are actively using the developed communication resources, chosen technologies in the physical layer and service providers in the network layer will have a considerable impact on the project's finance side. Therefore, the research's focus will be building a highly efficient solution without causing a highly cost infrastructure investment. At the end of the project, it is aimed to develop an innovative ICT application by using and linking together the data that has been sent from robots over mainly publicly accessible communications channels without high costs.

1.2 Foundation of Requirements

The requirements that formulate the need to have a dedicated communication network for service robots and dustbins in the project MURMEL are explained in 4 main sections.

- **Online Monitoring**

  The wireless connection established with robots will provide great convenience to the MURMEL control center for remote monitoring of the robot fleet. In addition to the instant sharing of ***GPS*** coordinates, incoming important ***sensor*** or ***actuator*** data from robots such as battery level and vehicle velocity will also increase robot operators' control over the operation by giving insight into the robots' current state.

- **Over the Air (OTA) Route Update**

  Once the project scope involves numerous robots, providing mission and route information to robots will become a complex task. Each time new tasks are required to upload to robots, robot operators must be present near the robots to create a local connection to upload a new task. In the case of hundreds of robots, providing task updates to robots will slow down operations and increase robot operators' workloads. This process can be automatized by enabling a constant wireless communication channel with robots and an application control center. Hence, through the developed route update service on the MURMEL application server, task-relevant information can be quickly sent to robots via a mobile network. This service can also be further utilized for updating route information on robots to dynamically changing conditions such as blocked paths or operational errors on dustbins. In these situations, new routes can be planned in the cloud and sent to robots. This way complexity of the autonomous driving algorithm can be reduced using cloud connections in such exceptional situations.

- **Implementation of Smart Dustbins**

  The smart dustbin idea is based on robots' route optimization to prevent unnecessary driving schedules to empty non-full dustbins. The dustbins ' state can be measured in regular intervals by installing sensors in dustbins to measure the fullness levels. However, for transmitting the sensor data to the central control center, dustbins must be equipped with wireless communication modules.

- **Emergency Access**

  Another benefit of having a direct connection to a robot can arise in emergencies. In case of an operational error such as blocked path or accident, controlling the ***autopilot mode*** and having a ***visual connection*** to the field by accessing to vehicle camera can help a lot to solve the problem remotely by saving time and energy. Instead of Mothership operators driving to the location of the robot for addressing the failure, by accessing the onboard camera Mothership operator can easily identify the issue in the surrounding area of the robot.

1.3 Defining Protocol Requirements

1.3.1 Communication Requirements for Robots

- **Uplink and Downlink Communication**

  The robots used in the MURMEL project should support both uplink and downlink communication with a network server. Uplink communication is defined as sending data from nodes to a network server, such as sensor values and GPS coordinates, whereas downlink communication is defined as sending data from server to nodes to control some of the actuators in exceptional cases. In a basic application scenario, both uplink and downlink messages are expected to be used to assure a safe and seamless application.

- **Acknowledgment of Received Messages**

  Acknowledgment of received messages that can also be defined as Quality of Service(QoS) is an important aspect of IoT-based applications. Especially in emergencies, it is very important to successfully identify if the receiver's send message has been received successfully. If a data collision or any data package is lost during the transmission period , the message should be resend until the receiver successfully receives the message.

- **Short Interval Communication**

  The communication between the MURMEL robots and mothership must support the data transfer in time intervals such as 1-5 seconds. Any delay or disconnectivity within the communication network might prevent access to robots in case of an accident or collision. Therefore the used communication technology should guarantee the data transfer in short intervals.

1.3.2 Communication Requirements for Dustbins

- **Uplink Communication**

  During the route planning phase, it is important to know the fullness levels of dustbins since the route planning algorithm takes into account only the full dustbins. This can only be possible by enabling uplink communication in the dustbins, where each dustbin sends its fullness level to the application server in intervals of 2-3 times per hour.

- **Low Power Consumption**

  While the communication unit's power consumption is not an issue for robots, it is a serious challenge for dustbins. Where robots have a chance to recharge or replace their batteries, this is not possible to realize on thousands of dustbins. For this reason, the power consumption of the communication units in dustbins should be minimized as much as possible.

- **Scalability**

  Once the scope of the project MURMEL goes beyond the prototyping and increases its capacities, there will be a need to connect thousands of dustbins to the internet [7]. When developing a communication protocol for dustbins, the created network should be able to handle thousands of connected nodes. In addition to the technical aspect, the financial feasibility and sustainability of the developed system should also be taken into account.

2 State of the Art

This chapter will give a short overview by researching the studies done in smart waste management systems and remote ROS communication applications.

2.1 Waste Management Systems

Technologies concerning the internet connectivity of devices and sensors have improved significantly over the past decade. Being these solutions are more efficient compared to existing connectivity options for constrained devices, new applications done in this field are emerging continuously.

One of the very simple applications done in this field is equipping trash cans with full-level sensors and transmitting the sensor data to the central waste management system over a wireless communication network. As a result of creating smart trash cans, trash-collecting schedules can be optimized[65] by only collecting the trash of the trash containers, reach out to a full level. This way, overall system efficiencies in trash collecting systems are expected to be increased by optimizing waste collection routes which would reduce the unnecessary energy consumption of tracks/robots deployed in waste collection operations. Edge computing and NB-IoT are some of the implemented technologies in such waste management systems [85].
The utilization of smart trash cans is already started to be implemented across Europe and also in Germany[85]. In cooperation with Deutsches Telekom, Bochum and Bonn[9][8] are the two pioneers cities in Germany in the implementations of smart trash cans into reality by using the Nb-IoT infrastructure as a communication technology. On the other hand, Darmstadt is also following the path to becoming a role model as a smart city in Germany[17]. Different than other cities mentioned, Digitalstadt Darmstadt GmbH is building a dedicated LoRa network to provide connectivity for specific applications such as smart bins, smart lights, and damage recognition[16].

In Amsterdam, the airport Schiphol has already up and running a smart waste management system based on a private LoRa network build in cooperation with French IoT Solution provider firm Kerlink[51]. In addition to the waste management system, the Schiphol airport's created network is also used for transmitting data from various devices such as accelerometers, temperature sensors, and barometers. The figure below shows the smart dustbins equipped with LoRa communication modules and ultrasound sensors.

Figure 1: Smart trash cans in Schiphol airport[51]

2.2 Outdoor Remote ROS Networks

ROS being a framework for writing flexible robot software, is widely used in lots of different applications[5]. Thanks to its underlying TCP/IP structure, users can develop applications across multiple machines and networks. Although ROS applications running in local networks are set up easily, for outdoor ROS applications, it is often complex and requires adjustments on different components[5]. As the communication protocol dedicated for the project MURMEL is closely interconnected with robots through ROS, it is important to highlight the research's done focusing on remote ROS networks.

One way to establish a communication channel to remote ROS networks is the port forwarding technique, enabling ROS master and clients to talk to each other over a public network. Port forwarding requires access to routers/gateways being used by ROS components for configuration purposes. However, this might be challenging for developers depending on the term and usage conditions of the service provided by the ISP (Internet Service Provider)[27].

An alternative way for communicating with remote ROS networks is the so-called *rosbridge* tool which enables non-ROS components to connect ROS network through cloud connections[27]. With this tool's help, web applications such as web browsers with java scripts or Android applications can be created to interact with ROS networks. An example of such a scenario is realized in RobotWeb Hackathon 2013 by controlling a robot through a web application over a long distance[27]. The figure below shows the architecture of a cloud-based web application for interacting with remote ROS networks.

Figure 2: Sample web application to a remote ROS network[3]

2.3 General Approach

In order to establish a communication where two or more users send data from one place to another, it is necessary to define a systematic set of rules that all the communications participants agree on. Computer networking's fundamentals rely on solving this simple problem by providing various methods and models to ease developed products' interoperability. Right at this point, computer network models offer solution bases for establishing communication between sender and receivers most efficiently [13].

TCP/IP and OSI are the most commonly used computer networking reference models on which the whole communication protocol is built[53]. OSI, which stands for *Open System Interconnection* was initially created to assure the compatibly of industrial and commercial computers from different manufacturers and product groups[13]. In the OSI reference model, the communications between a computing system are split into seven different abstraction layers: Physical, Data Link, Network, Transport, Session, Presentation, and Application. On the other hand, TCP/IP(Transmission Control Protocol Internet Protocol) is based on protocols on which the internet has developed[13]. In the designing phase of the communication protocol for the project MURMEL ,TCP/IP model below will be used as a guideline to approach the problems. Figure 3 shows the five layers of TCP IP.

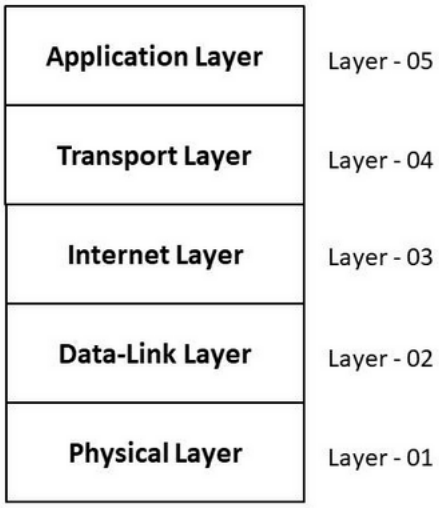

Figure 3: TCP/IP reference table with 5 layers[67]

It is also noted that TCP/IP model may not be the best fit for applications in the IoT Ecosystem due to the restricted environments of the devices being used in[53]. However, through the design process, the closest communication technologies and protocols will be chosen to fit into the TCP/IP model. Since some of the layers are already standardized in numerous internet applications, only the application and physical layers of the communication stack of the murmel project are studied in detail. The sections below will start with the definition of the problems and continue by examining the possible solution options to address the issues. At the end of each section, resulting decisions on the related topic will be declared following a comparison analysis.

3 Design of the Physical Layer of Dustbins

3.1 Overview of the Wireless Communication Systems

In order to choose the best suitable physical layer technology for both dustbins and robots, it is necessary to have a quick overview of the current wireless communication technologies. With its simplest definition, wireless communication is transferring information between two or more endpoints over a distance without using electrical wires or another solid medium for sending a signal[77]. Depending on the parameters such as transferring distance, bandwidth, latency, and power consumption, numerous wireless communication technologies are available. However, each of these communication technologies targets a specific application area with its unique advantages and disadvantages. In the figure below, wireless communication technologies are split into three categories based on the communication range and data transfer rates.

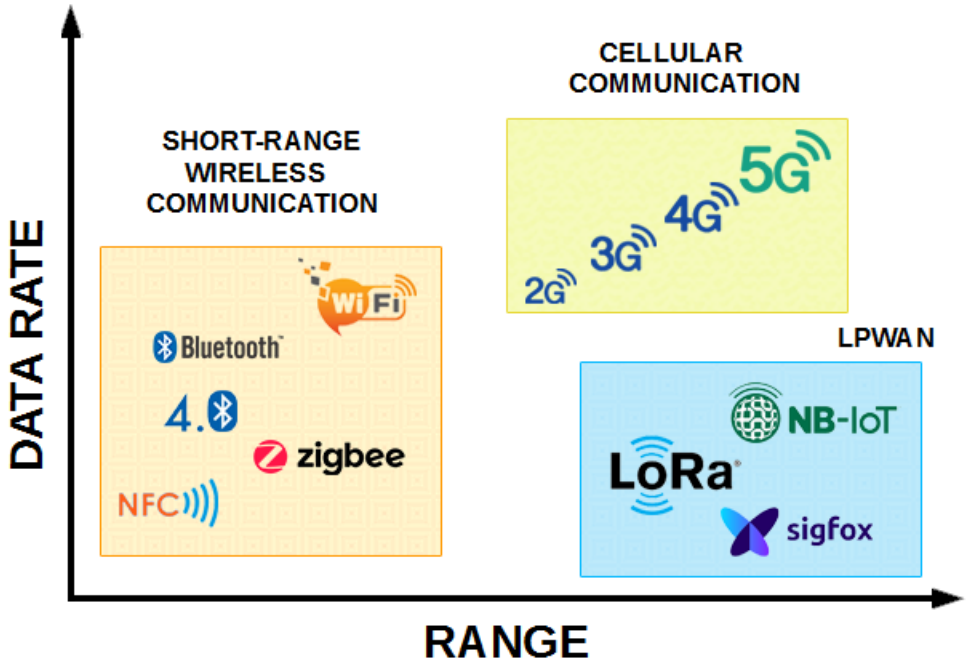

Figure 4: Wireless communication technologies overview [83]

Short-range wireless communications are mainly designed for transferring data in short distances ranging from a couple of centimeters to 250 [15] meters depending on the technology being used. While Wi-Fi becomes pro dominant when it comes data transfer in a short latency and high bandwidth, it is not very suitable for application that requires Machine to Machine(M2M) communications due to high power consumption[15]. NFC, Zigbee, and Bluetooth are also other communication technologies used in short-range communication applications. Bluetooth supports lower data transfer rates than Wi-Fi, and it is generally used to establish cable-less connections with electronic devices in 2-10m ranges. On the other hand, Zigbee is a communication protocol deployed in personal area networks for communication devices by sending and receiving a small amount of data. The common feature of all these named technologies is that there is no need to have a network operator or establish a costly infrastructure.

Cellular communication technologies and Low Power Wide Area Network(LPWAN) technologies, on the other hand, require an infrastructure that may result in fees in network usage. Since both robots and dustbins need long-range communication by the nature of the MURMEL application, the focus of the physical layer design of robots and dustbins will be on cellular and LPWAN technologies. In the following section, each of these wireless communication technologies will be studied in detail.

3.2 Low Power Wide Area Network Technologies

LPWAN technologies are a type of wireless communication where communication ranges are significantly greater than other technologies shown in the figure 4. Depending on the communication protocol, the range for LPWAN may reach up to 10 km in rural areas and 2-5 km in urban areas[42]. Despite offered long-range capabilities of LPWAN technologies, data transfer rate and bandwidth are highly limited compared to cellular communication technologies [42]. This makes LPWAN particularly suitable for applications that require sending a low amount of information like sensor readings where latency is not prioritized. Some of the known use cases for such technologies are assess tracking, smart water/electricity metering, pipe monitoring, and smart agricultural activities. Since the smart dustbins in project MURMEL are only expected to send fullness levels a couple of times per hour, dustbins' communication needs are good examples of use cases deployed in LPWAN technologies.

Currently, NB-IoT, Sigfox, and LoRa are the most used LPWAN technologies in IoT applications[42]. Especially the cheap radio chip costs for LPWAN modules on a scale of 2-10 dollars [42] makes these technologies more attractive for mobile IoT application in cases where deploying cellular network technologies appears to be much more expensive.

LPWAN technologies can be divided into two groups depending on the operating spectrum as licensed or unlicensed. Technologies running on licensed bands are more likely to have fewer restrictions in data transfer rate and data size, and permitted network usage time. Whereas technologies operating on unlicensed bands might be exposed to specific regulations depending on the country. In the table below, a categorization of the LPWAN technologies can be found.

Licensed Spectrum	Unlicensed Spectrum
NB-IoT	LoRa
LTE-m	Sigfox

Table 1: LPWAN in the licensed and unlicensed spectrum

In the following section, where the physical layer's design for dustbins is concerned, the LPWAN technologies will be studied in detail. In order to choose the best suitable communication technology for dustbins, different LPWAN technologies will be studied detailed in two main categories as the ones operating on the licensed band spectrum and as the ones operating on the unlicensed band spectrum.

3.3 LPWAN Technologies in the Unlicensed Band Spectrum

3.3.1 LoRa

What is LoRa ?

LoRa (short for long range) is low power and long-range communication technology covering the physical layer of communication protocols. It was first created by a french company in 2009, and later It was owned by a USA company Semtech[42]. LoRais operating in 868 MHz in Europe, 915 MHz in North America, and 433 MHz in Asia in the unlicensed spectrum of ISM bands. It is based on a spread spectrum modulation technology chip [82] which makes LoRa to be suited for application that requires low power consumption and long communication range. LoRa is currently one of the most widely used LPWA technology in IoT applications where already 150 Million devices are deployed in more than 99 countries by 140 network operators [82] . According to the report published by HS Market Insider(May 2019), 45% of LPWAN technologies are expected to run on LoRa infrastructure.

LoRa steps forward, especially by the support of long-range communication support. Even though it highly depends on the area's geographical conditions, it supports communication over a distance up to 48 km in rural areas[31] and 5 km in urban areas[42] which paves the road for many different IoT applications. The most known use cases are smart city applications, smart homes and buildings, agriculture, smart metering, assets tracking in supply chains.[31]

What is LoRaWAN ?

LoraWAN is a communication protocol based on LoRa radio modules on the physical layer of the communication structure[82]. It aims to connect battery-driven *sensors* to public internet using the LoRaWAN gateways[82]. The specifications of the LoRaWAN define different parameters to create a flawless connection between field devices and LoRa networks. Security measurements, communication range, and data transfer rates, network capacity, quality of the service are some of the important factors regulated by the LoRaWAN[4]. The standardization of the LoRaWAN is regulated under the nonprofit organization called *LoRa Alliance* to ensure the compatibility of LoRa based applications in all LoRa Wide Area Networks. Currently, more than 150 member LoRaWAN Network operators in LoRa Alliance are operating in more than 160 countries to improve the LoRaWAN network availability's around the world[35].

Since the LoRaWAN is an open-source network, users can build dedicated and customized private LoRaWAN networks for specific applications. Although the network usage is free of charge, it is required to establish an infrastructure by installing gateways to guarantee the network's communication range. Companies like *Zenner*[38] in Germany providing assistance to establish LoRaWAN network for companies and cities. Some of the known examples of such private networks are already launched in Darmstadt[16] and Bonn[9] under the scope of smart city projects.

Public LoRaWAN networks, on the other hand, are free to use by any users and don't require any special equipment. The only difference between private and public LoRaWAN networks is the trust issue of the network operator. For applications that include data transfer of sensitive information, the public LoRa network may not be the best solution as it is opened to access to anyone. However, dedicated servers can still be purchased in some of the public LoRa networks, as in the case of *The Things Network*.

Security

LoRaWAN implements security measurements in two layers. The network layer communication between an end node and a network server is encrypted using a 128-bit network session key. In another layer, communication between the network server and application server is encrypted using an application session key to prevent access from the network server to the end-user application.[4]. The usage of a two-layer key mechanism enables network operators to create multi-tenant architecture in a shared network without actually setting up an individual server for each tenant or user group [34]. This way, acceleration of the spreading LoRaWANs is aimed to accelerate without creating extra burdens on network operators.

Activation of devices before joining a network can be established in multiple ways. Either in the production line of the devices, activation can be done, or through the OTAA(Over the air activation), users can easily activate the end nodes [4]. However, just like any other wireless communication technology, LoRa Alliance strongly recommends using the certified devices and trusted service providers to prevent any possible security leak in users' applications. As more attacks on LoRaWAN becomes visible, improvements are also continuously being made on the network architecture. For building a secure LoRaWAN application, best practices are easily accessed from the LoRa Alliance[35].

Network Topology

In traditional networks, mostly mesh type of topologies is established in order to increase communication range as in the case of Zigbee[4]. While in this approach, nodes transmitting the received information further to another node results in an increase in the cell size and range, but it also adds lots of complexity, reduces the network capacity, and consumes a lot of power[4]. LoRaWAN, on the other hand, is utilizing star type of topology, which increases the range by keeping the power consumption in low-level [4]. A typical LoRaWAN network consists of end nodes, gateway, network server, application server, and the network architecture can be seen below

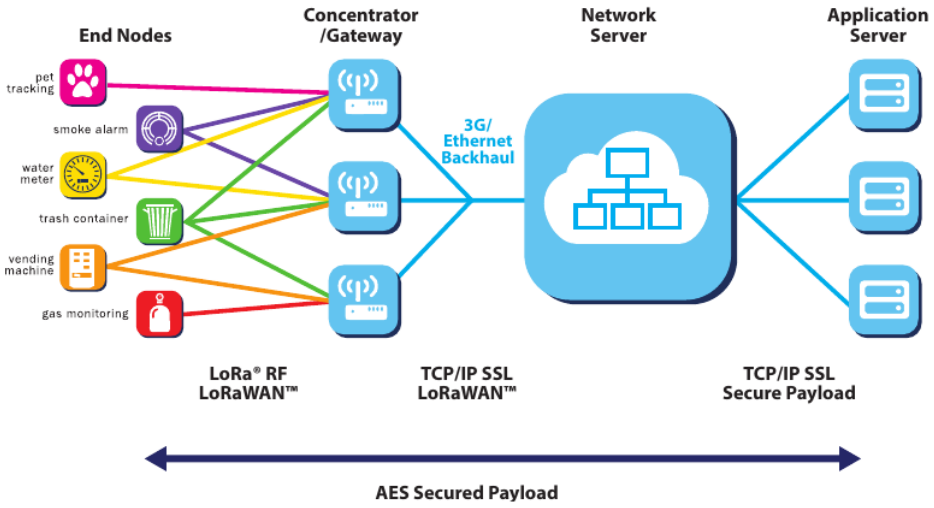

Figure 5: LoRaWAN topology[30]

Network Element: End Node

Node is any device or sensor that has a LoRaWAN supported communication module. In LoraWAN **end Nodes** are sending and receiving RF packages between LoRaWAN gateways. Under the MURMEL project's scope, after integrating the dustbins with communication modules, each dustbin would represent a **end node** in LoRaWAN [4].

In order to meet different application requirements, end-nodes in LoRaWAN are categorized into three classes as Class A, B, and C. The classifications' main logic lies in the trade-off between power consumption and latency on downlink messages [4].

- **Class A**
  Devices configured as type A has the lowest power consumption mode compared to the other classes. It only allows a device to listen to downlink messages following uplink messages. Any data sent to the node out of this schedule will not be received by the end node since it does not listen.

- **Class B**
  Devices configured as type B has the flexibility to be scheduled to listen to incoming messages at certain intervals rather than only listening following an uplink message. This reduces the latency in downlink messages by increasing the power consumption of the radio module.

- **Class C**
  Devices configured as type C consistently listens to the incoming messages to keep the latency on downlink messages as low as possible. This configuration can be useful for application which involves actuators in the end node device.

In the case of implementing LoRaWAN in dustbins, the most suitable class mode for LoRa radio modules would be class A since the application would not require pushing downlink messages to the dustbins. This way battery life of the radio modules on dustbins can be maximized.

Network Element: Gateway

In LoRaWAN, end nodes are **not** directly connected to a public internet network. At this point, Gateway act as a bridge between end nodes and a cloud-based LoRaWAN network server by converting IP packages and RF packages[35]. This way, uplink and downlink messages are routed in the LoRaWAN Gateway. A gateway can use different internet communication technologies such as LTE, IEEE 802.11, or Ethernet.

Network Element: LoRaWAN Network Server

In LoRaWAN, end-nodes are not linked with certain Gateways; instead, the transmitted messages might be received by multiple gateways simultaneously[4]. In order to reduce the complexity of gateways, all the data and device management issues are passed to the LoRaWAN servers [4].In a deployment of sensors communicating through a LoRaWAN network, the LoRaWAN Network Server plays a critical role in managing received and transmitted LoRa packages. The Network Server performs mainly security checks, handles the redundancy on received packets, adapts the data rate[4] and carry-outs the authorization of the LoRa devices [36]

Network Element: Application Server

In order to create IoT applications, the transferred data from field devices to network servers are needed to be transferred into application servers. Application servers can easily be realized using the cloud services such as AWS, Microsoft Azure, or any other services or servers. Specifications about the connection between a network server and an application server can be found in different APIs provided by the network server.

Network Operator and coverage

Since LoRaWAN is an open protocol, any organization can build up its own LoRaWAN network as long as regional ISM regulations are met during the network's operation. Thanks to the extensive coverage areas of LoRa signal, much fewer gateways are needed to create a local wide area network. These infrastructure services can be provided by system integration companies such as Zenner[38] in Germany, but also community-based initiatives can support the establishments of LoRa networks. Darmstadt, Karlsruhe, Lübeck, and Munich in Germany are already some of the cities that have already started to improve LoRaWAN infrastructure in their local regions[33]. Berlin, on the other hand, also has a growing LoRaWAN Network infrastructure provided by an open-source community-based initiative called *The Things Network(TTN) [37]*. In the following section, LoRaWAN network coverage of TTN in Berlin will be studied to be considered a network server provider for dustbins in the MURMEL project.

TTN availability in Berlin

Following the idea of creating the world's largest open LoRaWAN network, TTN has already created lots of LoRa networks in more than 150 countries around the world[37]. With 172 installed LoRa gateways, Berlin is the fourth largest TTN network in the world[74]. Although there isn't any official coverage test available yet, the current map of publicly available gateways in Berlin and their cover range can be seen below.

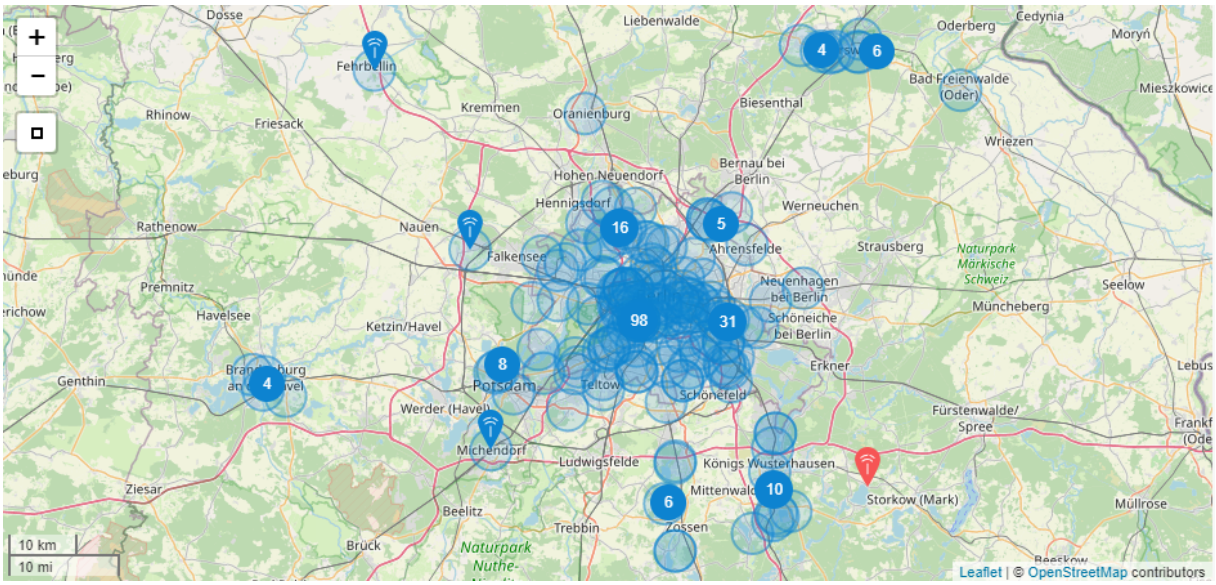

Figure 6: TTN gateways in Berlin [75]

3.3.2 Sigfox

What is Sigfox ?

Sigfox is a french **low power wide area network operator** company using its own developed wireless communication technology to send a small amount of data over long distances that can pass through objects such as buildings and walls.[60]. Very similar to LoRa, Sigfox is also using the Industrial, Scientific, and Medical ISM radio band, which uses 868MHz in Europe and 902MHz in the US. Its fundamentals are based on ”differential binary shifting key (DBPSK) and Gaussian Frequency Shifting Key(GFSK)” methods which result in consuming low power and passing freely through solid objects[62].

Network Overview

Sigfox network based on proprietary based stations that the local Sigfox operator operates. The general network operation principle is listed below.

1. A device with a Sigfox Ready module emits a radio signal
2. Signal is received by a Sigfox base station installed nearby
3. Base stations send the message to the Sigfox Cloud Platform
4. Sigfox Cloud provides API’s that can forward the message to different back end applications such as BSR MURMEL Project management system.

The Figure below illustrates the general network structure of Sigfox.

Figure 7: Illustration of the network structure of Sigfox defined above.[59]

Sigfox initially supported only uplink communication in the early phase of the deployment. This way, only end nodes were able to send data to the cloud. However, later on, a downlink communication feature is added to technology by only allowing users to send messages to the end nodes following an uplink message[42]. Just like LoRa, Sigfox also limits network usage. In this context, only 140 uplink messages per day are allowed to send, where each message payload can be a maximum of 12 Bytes[42]. Similarly, downlink communication is also limited to 4 messages per day, and the maximum message payload is 8 Bytes[42].

Network Coverage

Sigfox aims to establish a global IoT network to be deployed as a solution in Low Power Wide Area Network technologies. In order to reach this aim, it is cooperating with other network operators in countries to install base stations to increase the coverage areas. Currently Sigfox i available in more than 70 countries [58]. Currently, Sigfox is operating in Germany as Sigfox Germany GmhH and already reached 85% network coverage[57]. More detailed information regarding signal strength and coverage information can be found on the Sigfox Germany website, where it is possible to check individual addresses' network status. As an example, the figure below8 shows the signal strength at the main building of Technical of Berlin.

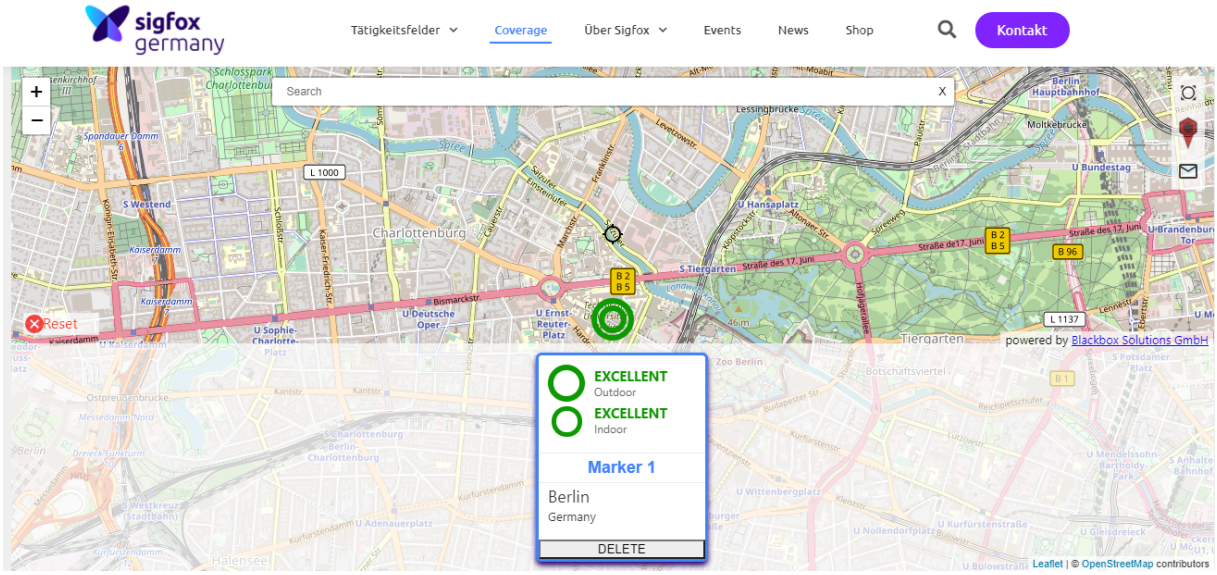

Figure 8: Sigfox signal check in Berlin [55]

Finance Model and Service Fee

Unlike The Things Network, which is based on an open-source LoRa network, Sigfox provides paid network services and subscription models that can be used in all countries where Sigfox base stations are installed. Depending on the number of devices and number of messages to be sent per day(2,50 or max 140), the tariffs are varying from € 9 to € 20 for a single device in a yearly period.[61]. For large-scale projects including more than 1000 devices, it is advised to get into contact with a local Sigfox operator.

On the official website of Sigfox, the complete list of communication modules and microcontrollers with a Sigfox certification can be found.[56] Even though there are a lot of hardware options, only some of the Sigfox Ready modules, such as MKRFOX1200 from Arduino, include a prepaid Sigfox subscription service that supports 140 uplink messages per day for a year. For other available devices, in order to activate the Sigfox connectivity, it is required to register the devices to the Sigfox portal.

3.4 LPWAN Technologies in Licensed Band Spectrum

Companies and network operators' interest in wireless communication technologies aimed at IoT applications is already researched in the previous section by focusing on LoRa and Sigfox in detail. Both of the technologies come with characteristic features that can be perceived as an advantage or disadvantage depending on the application type. However, the technologies running on unlicensed radio spectrum have some drawbacks in-network coverage and infrastructure adequateness. In this respect, The 3rd Generation Partnership Project (3GPP) defined in release 13[1] new communication standards to be deployed in LPWAN applications that can be considered an alternative to LoRa and Sigfox.

- LTE-M (Long Term Evolution Machine Type Communication)
- NB-IoT (Narrowband Internet of Things)

According to a report published by SNS Telekom in 2016[49], by the end of 2020, more than 35% of IoT devices used in LPWA networks are expected to be served by communication standards set by 3GPP(Nb-IoT, LTE-M, EC-GSM-IoT). Thanks to the cooperation between the Mobile Network Operators (MNOs),3GPP solutions are integrated into existing LTE infrastructures are increasing the usage scale of IoT communication standards in networks.

The following section will focus on NB-IoT and LTE-M's role in the IoT ecosystem by investigating their technical aspects, network availability, limitations, and financial scalability for the MURMEL project.

3.4.1 NB-IoT

What is NB-IoT

NB-IoT (Narrowband Internet of Things) is a Low Power Wide Area Network radio communication technology developed by 3GPP(3rd Generation Partnership Project)[66]. Low power consumption, wide-area coverage, and high connection density are the main highlights of the technology.[66], and already more than 140 network operators worldwide have implemented Nb-IoT Network into their infrastructures[66].

Different than other LPWAN solutions like LoRa and Sigfox, Nb-IoT is based on the LTE network. It uses the physical layer and some additional layers from LTE to standardize the technology as soon as possible[66]. NB-IoT stands out with its enhanced coverage range compared to existing GSM technology. NB-IoT has a particular advantage in bad connection areas due to simplifications made in the signal synchronization, and physical layers of LTE [66]. These made changes results in low system complicity, which also reduces the cost per device and power consumption during network usage.

Nb-IoT can coexist in 2G, 3G, and 4G networks without needing a new sim. The initial costs of Nb-IoT devices are expected to be compatible with GSM/GPRS device [25] in the early phase Nb-IoT deployments. However, as the underlying technology is much simpler than GSM/GPRS, it is expected to be decreased when demand is increased[25].

Network Coverage in Europe

As mentioned in a report published by Vodafone Germany[20], the installation of the Nb-IoT network can be done by a software update on the current basis stations without creating new network infrastructure. In this regard, NB-IoT is differentiating from Sigfox and LoRa-since they both require new basis stations to be installed to create a coverage area, which results in extra infrastructure costs.

Using this advantage, NB-IoT devices are already used in China, mainly in water and gas meters. In Europe, on the other hand, Vodafone and Deutsches Telekom are the leading operators in the region by expanding the coverage of NB-IoT with roaming agreements[28]. In April 2020, Deutsches Telekom signed NB-IoT roaming agreements with *"Swisscom, Telia Company and Vodafone"* in order to meet the customer need by increasing the coverage of the technology in Europe[69]. Currently, Deutsches Telekom NB-IoT Network services are available in 18 countries, and before the end of 2020, it is expected to reach a complete roaming coverage in Europe[69].

In the current state of the project, rooming is not required in MURMEL Robots since they are only designed to run in Berlin. However, in a larger-scale waste management system where waste bins are manufactured with built-in NB-IoT modules, roaming could be an important factor. In a scenario where dust bins are manufactured in a single country and then exported to other markets in Europe, no changes needed to be done on hardware or on the application provider side as long as roaming is supported[69].

In the map below, one can easily check the availability of NB-IoT or LTE-M network coverage in Germany.

Figure 9: Full coverage in Berlin, 90% NB-IoT coverage in Germany[14]

Security

Data security in the IoT ecosystem is an important topic to be considered when developing an application. Especially in mission-critical applications such as the communication stack of robots in the MURMEL Project, hijacking the device data could cause considerable significant financial and operational costs such as Interception of the communication and hijacking the sensitive data transmitted. In order to improve the security layer of LPWA network communication, Nb-IoT is designed with a feature called Non-IP Data Delivery. In a report published by GSMA [26] security features of NB-IoT are explained in detail, mainly focusing on NB-IoT infrastructures operated by Deutsches Telekom and Vodafone. An important aspect of this report is the term called *Managed Communications* services. According to IoT Technologist at Vodafone Martin Bell[26], in an IoT application, most of the time, devices are only communicating with certain application servers that are required. On the other hand, in daily life internet usage, people use computers or browsers to communicate with lots of different servers, which requires advanced routing protocols such as IP to control the data traffic. However, as also Mr.Bell explained, this is often not the case in an IoT application. As a result of this, a new feature called Non-IP Data Delivery(NIDD) is introduced with NB-IoT[26].

In IoT applications that use the IP protocol stack, the most significant proportion of the metadata size is the IP Headers. In contrast, the payload itself(sensor readings) is only a couple of Bytes. By enabling the Non-IP Data Delivery feature[26]

- amount of data in the network traffic,
- and power consumption of the device is reduced.

As a result of not utilizing IP stack, the possibilities to attack to a device using TCP/IP protocols are eliminated[26].

3.4.2 LTE-M

LTE-M is also another LPWA network technology for mobile IoT use developed and standardized by 3GPP. Similar to NB-IoT, LTE-m is also based on an existing LTE network, and it can be activated through a software update without hardware modifications in the network[39].

Although both LTE-m and NB-IoT are LPWA network technologies, they differ in the following aspects.

LTE-M	NB-IoT
High Data Transfer Rate(up to 1 Mbps)	Low Data Transfer Rate(up to 62.5 kbps)
Real time communication	High latency
More efficient battery use than 4G	Up to 10 year- battery life
Full mobility support	No connected mobility

Table 2: Key differences between LTE-M and NB-IoT[40]

As highlighted in the table above, the main differences between LTE-M and NB-IoT lie in latency, data rate, and connectivity in moving objects, with a trade-off between power consumption. Since communication modules on dustbins are planned to be run on batteries, power consumption criteria stand out as a significant deciding factor for the murmel project. In this respect, LTE-m technology doesn't seem to be the best fit for dustbins.

3.5 Discussion

The studied technologies so far all come up with different advantages and disadvantages. When choosing the right solution for any IoT application, including the Project MURMEL, the decision should be taken over the essential needs of the application. The following table summarizes some of the essential parameters to be considered as deciding factors for dustbins.

Parameters	LoRa	Sigfox	NB-IoT	LTE-M
Spectrum	Unlicensed	Unlicensed	Licensed LTE Bandwidth	Licensed LTE Bandwidth
Power Efficiency	Very High	High	Medium	Medium-Low
Bandwidth	145 kHz& 250 kHz	100 Hz	200 Hz	1.4 MHz
Max.Data Rate	50kbs	100 kbps	200 kbps	300-380 kbps
Max.Message per day	Unlimited	140(UL),4(DL)	Unlimited	Unlimited
Mobility	Supported	Supported	Not Supported	Supported
Standardization	LoRa-Alliance	Sigfox with ETSI	3GPP	3GPP
private network	yes	no	no	no
End device cost	<2$	<3-5$	>20$	20-30$
Service cost	free	paid	paid	paid

Table 3: Comparison between LPWAN technologies [42]

One of the most important factors in the communication needs of dustbins is the radio modules' power consumption. As also indicated in many technical reports and overviews [63] LoRa hast best performance in terms of power efficiency among other LPWA technologies where the life cycle of the battery may be up to 10 years[4]. Under the scope of financial feasibility, two factors vital to be considered; end device cost and service cost. In this category, LoRa is also significantly advantageous by having the lowest end device cost. Compared to other network services where service usage is subjected to fees, LoRa is also more appealing since it doesn't require a fee. Hence the maximum data sending rate in MURMEL Project is not expected to be more than 3-5 messages per hour, it is not a vital parameter to be considered.

3.5.1 Field Test

As described in the previous network coverage section, Berlin has infrastructure support for all the mentioned LPWAN technologies. In this section, a prototype communication module to be placed in a dustbin has been designed to test the network availability of the LoRaWAN using TTN gateways. In order to connect to TTN gateways, it is required to have a certified LoRaWAN device[76]. Since creating the communication network of dustbins is in the prototyping phase, it is a good start to use development boards with the support of TTN in order to keep the complexity and time effort at a low level. For this reason, LoRaWAN supported *Lopy4*[48] from Pycom will be used as a communication unit in the smart dustbin prototype. The device is MicroPython programmable[48], and all the necessary documentation regarding LoRaWAN and TTN connection issues are easily accessible on the official site of the Pycom.

However, for large-scale applications such as implementing LoRaWAN connectivity on thousands of dustbins, more price efficient and smaller size chips should be preferred. For guaranteeing the proper functionality of the application, LoRa Alliance certified device should be chosen. The complete list of supported device and SDK's are also available on the official website of *The Things Network*. The developed communication module can be seen below.

Figure 10: LoRaWAN test unit on a dustbin

The main component of the prototype above is a Lopy4 microcontroller which supports the wireless communication standards WiFi, Bluetooth, Sigfox, and LoRa[48]. In addition to the Lopy4, a standard WiFi antenna, power unit, and a GPS receiver module (NEO-6M) [78] has been installed on a breadboard. Even though there isn't any GPS tracking need in the smart dustbin application, GPS module has been installed on the test unit in order to verify LoRa availability in different locations of Berlin.

In order to map the locations of accessing points to TTN, a simple application has been prepared. As the first step of the application, a connection on Lopy4 to TTN Console is created using the MicroPython tool. Due to the fair access policy of TTN [76], the GPS location of the hardware is only attempted to be sent in 4-minute intervals. In order to make use of the sent LoRa messages from Lopy4 to TTN Server, MQTT API is used for retrieving the published MQTT messages[73]. After successfully connecting to data Data API of TTN, received messages send by Lopy4 are further pushed to Thingsboard io for visualizing the send locations of LoRa messages on a map. The resulting field test map can be found below.

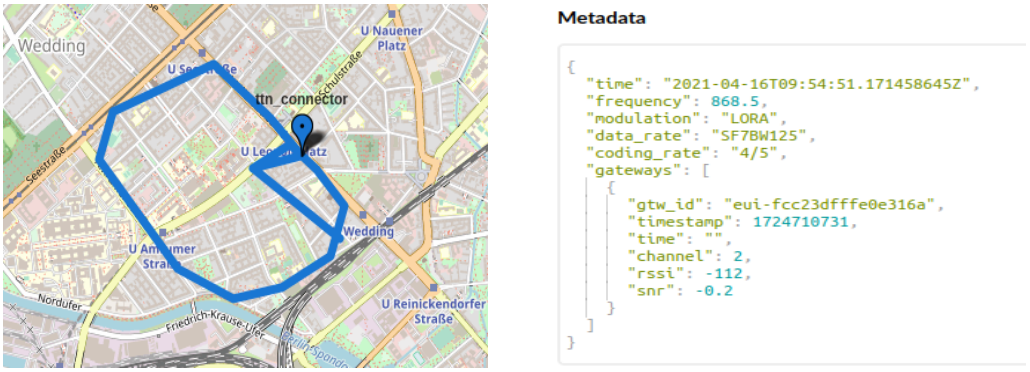

Figure 11: TTN coverage test in Berlin Wedding and TTN console view

In this section, connection details, IoT Platform comparisons, and assessments of MQTT are not given. More information regarding MQTT and thingsboard service are further discussed in the application layer design section.

3.5.2 Conclusion

As a result of a detailed analysis of the currently available low-power wide-area network technologies, LoRa seems to be the best fit to be deployed in the physical layer of dustbin communication architecture. Besides low power usage and low-end device cost, it also gives business or end-users to deploy their own private LoRa networks. Even though this can be an option to be considered in the future phases of the MURMEL project, it is not necessary to create a private LoRa network for MURMEL dustbins. Since the already mentioned crowd-sourced *The Things Network(TTN)* has a significant tested and verified coverage area in Berlin, using TTN infrastructure seems to be the best fit for smart dustbins.

4 Design of the Physical Layer of Robots

Wireless communication technologies that have been discussed so far are the solutions designed specifically for IoT applications that require low data rates and low power consumption. However, due to restrictions in different topics such as low uplink and downlink data rates, limited data sending intervals, high latency, and low bandwidths, LPWA technologies are not meeting the communication protocol requirements of MURMEL Robots. As a result of the previous comparisons, it is clear the need for cellular communication technology to be deployed in MURMEL Robots.

4.1 Cellular Communication Technologies Overview

Cellular communication with its simplest definition is a technology that enables mobile phones to communication with each other in a bi directional way. The fundamentals of the cellular technology based on the geographical division of the coverage areas into cells. This structure enables network operators to provide service to More subscriber while reducing the number of channels being used.

The wireless cellular communication technologies that have been developed are denoted by the generation indicator 'G' as 1G 2G 3G 4G and 5G[23]. In the figure below, one can see the historical evolution of the cellular technology generations. Just like any other technology, cellular communication is also developed throughout history by trying to improve people's experience in mobile services. In this respect, improving the communication quality, multimedia streaming, and achieving a lower latency have always remained the priorities of the developments.

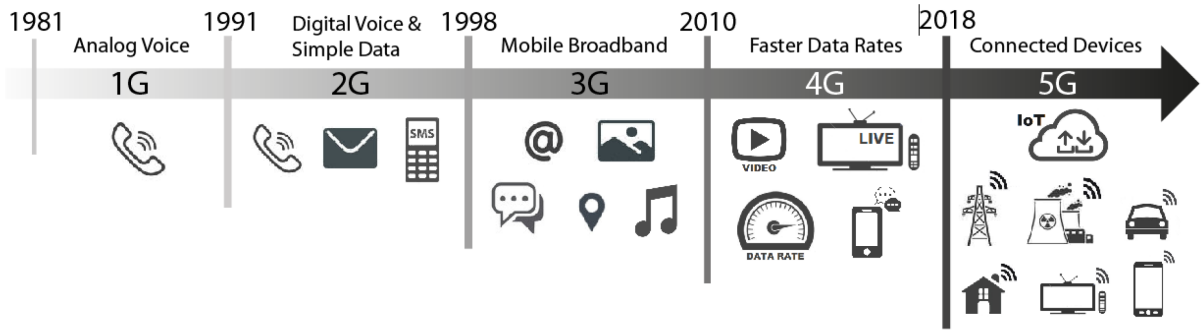

Figure 12: Cellular communication generations[41]

The most important aspects of the generations are shortly explained below.

- **1G**
  The first generation of mobile communication networks only allows to transmit voice through modulation of analog signal. As a commercial application it was initially released in Japan in 1977[23].

- **2G**
  2G mobile network allows for the first time use of digital signals in mobile telecommunications. In addition to transmitting voice in digital signals it is also the first generation supports to send shot message serve(SMS) and multimedia message services(MMS)[54]. 2G networks for intialy started on GSM Standards in Finland. TDMA(Time Division Multipleaccess) and CDMA(Code Division Multiple Access) are the used modulation technices used in 2G.

- **2G - GSM(Global Systems for mobile Communications**
  GSM is the most widely used communication standard introduced by 2G[23] and it is the first standard the supports international roamining [23]

- **2.5G - GPRS (General Packet Radio Services**
  The major difference of GPRS compared to GSM is the Packet switched technology. All the informations to be send to the destinations are divided into packets and all packets can be send in parallel.[23] It also supports IP to connect to Internet. In the GPRS network, devices must be equipped with GPRS modem in order to communicate with GPRS Gateways which are connected to world wide web(public internet).

- **2.75G - GPRS - EDGE**
  The last 2G standard was Edge(Enhanced Data rated for GSM Evolution) which is also called 2.75G.Edge Standards was developed by AT&T in 2003[22]. EDGE introduced digital modulation and packet switching which opened the doors to high speed data transfer[54].

- **3G - Third Generation of Mobile Networks**
  The developments done under the term of third generation aimed to increase network communication speed. The first introduced standards are WCDMA (Wideband CDMA) and UMTS(Universal Mobile Telecommunication System)[23]. 3G is the first generation which enables live streaming over mobile networks[32].

- **4G - Fourth Generation of Mobile Networks**
  The wireless communication technology as LTE(Long Term Evolution) standardisation has been started by ITU in 2004 and the first commercial LTE network is deployed in Oslo-Norway and Stockholm/Sweden. 4G standards also includes WiMax standards but LTE has been prompted more[23].

- **5G - Fifth Generation of Mobile Networks**
  5G is the latest cellular communication technology developed and standardized by 3GPP[1]. It enables high speed data transfer and supports more connected devices in mobile networks[81].

4.2 Discussion

The technical reviews and comparisons made in the previous sections on mobile communication technologies revealed the need for a communication module that can support communication in at least a 3G network to meet the communication requirements of MURMEL Robots. However, the recent developments in cellular communication technologies showed that frequencies that are blocked by the 3G network could be better utilized in 4G and 5G infrastructures to reach lower latency and to support more devices in the mobile network.[79]. In the light of these developments, network operators worldwide are already started to plan to shut down 3G networks. In the table below, some mobile network operators worldwide and their closure dates regarding the 3G network can be found.

Switch-Off Date	Network Operator	Country
December 2020	Verizon[79]	USA
February 2022	AT&T [6]	USA
June 2019	Airtel [2]	India
December 2012	Telenor[70]	Norway & Sweden
June 2021	Vodafone [80]	Germany /Netherlands
June 2021	Deutsches Telekom[68]	Germany

Table 4: Some of the network operators shutting down 3G

In this context, in Germany, mobile network operators in the market such as Vodafone, T-Mobile, and Telefonica Deutschland have already announced the termination of the 3G network starting from summer 2021[68]. One of the critical results of these developments is that devices that had been manufactured with built-in 3G communication modules(phones, laptops, modems, vehicles, etc.) will not be able to use cellular communication anymore, or they would require a hardware update on devices as in the case of Tesla Model SX series.[71] As a result of these recent announcements regarding the tendencies of network operators to shut down 3G networks and considering the timeline of project, it is required to have hardware that would support at least 4G communication.

5 Design of the Application Layer

5.1 Introduction

The application layer is the highest abstraction layer defined both in TCP/IP and OSI computer networking models that specifies a set of communication protocols and provides interfaces to users for transferring data in the application[84]. Also, in the project of the MURMEL, it is a crucial element for the management of the operations as it sits in the center of the network of connected devices. The application layer developed in this work is targets the prototyping phase of the MURMEL project. In this context, it is designed to keep the complexity level as minimum as possible by only implementing the core functions to create a qualified application layer for murmel robots and dustbins. The created servers and services are listed below.

1. A lightweight communication server for creating a communication between MURMEL application layer and connected devices such as robots and dustbins to send telemetry data.
2. For managing the communication between connected devices through dynamic dashboards, it is necessary to deploy an IoT Platform service in the MURMEL Server.
3. A video streaming server for accessing and controlling the live streaming from robots to application layer..
4. A route planing service which sends robots route and task informations.

5.2 Messaging Protocol

Common application layer protocols used in generic TCP/IP models such as HTTP, FTP, and SSH are not always to best fit for IoT-based applications [53]. HTTP is a standard client-server type of network based on requests and responses where a client is generally referred as an end-user, and a server is a place that hosts web pages. By using a web browser or supported HTTP APIs in script programming languages, clients initiate HTTP requests to the server. These made requests establish a TCP connection on a certain port of the server where the server consistently listens to the incoming requests. Upon receiving the request, the server sends back the client's response as an HTTP status line along with the requested message body. Like HTTP, FTP is another client-server type of communication protocol for transferring files between clients and servers.

Although both HTTP and FTP are common communication protocols deployed in the application layer, they are not the best fits for IoT applications there require M2M communications. The high requests header size in terms of bytes and more needed resources makes it harder for microcontrollers with limited hard sources to handle such complex communications. As a result of the different IoT applications requirements, new communication protocols have been emerged to meet the need for M2M communication. In the following section, one of the most used and known communication protocols used in IoT applications[72] will be studied and integrated into the application layer of the murmel project. Protocols of interest for sending telemetry data between robots and murmel server are listed as follows;

1. MQTT (Message Queue Telemetry Port)
2. CoAP (Constrained Application Protocol)

5.2.1 CoAP

Constrained Application Protocol(CoAP) is an application layer communication protocol developed by Internet Engineering Task Force in 2014[11]. It is designed to meet the needs of M2M communication in IoT applications where devices with limited resources such as 8-bit microcontrollers with a small amount of ROM and RAM are operating in constrained networks such as Ipv6 based Low Power Personal Area Networks (6LoWPANs) [21]. The protocol is essentially designed to be deployed in M2M communications for smart metering and building automation applications.

What is CoAP ?

CoAP has a similar structure as HTTP, and it is based on a server-client network where sensor nodes are designed to be servers. Unlike HTTP, CoAP is not intended to send documents through the network, but only sensor reading data in small amounts by using the existing HTTP methods GET, POST, PUT, and DELETE[12]. From this aspect, CoAP clients can simply request relevant information from CoAP servers through a web application that uses HTTP.

CoAP is intentionally designed to run in UDP protocol through which it has a quite smaller packet header size compared to other messaging protocols that run on TCP[11]. From the perspective of the underlying UDP protocol, messages transfer in CoAP is not as reliable as TCP since the messages might get lost during the transmission, and the system is lacking of precaution to prevent data loss or acknowledgment mechanism. However, all these features enable very lightweight communication, especially useful constrained devices such as microcontrollers with limited RAM and ROM's[84].

Architecture

CoAp is based on client-server type communication where mostly sensor nodes represent a server.[12]. CoAP is mostly suitable to deploy when all the nodes of the communication are in the same constrained network. In order to access a CoAP server from the outside of a server, NAT capable router is required for port forwarding. In the current mobile networks, end-users do not have access to the router since they are mostly running on IPv4 networks[47]. To overcome the NAT issue, nodes must be connected via the IPv6 network, which is not fully implemented yet in mobile networks. The figure below shows the networking architecture of CoAP.

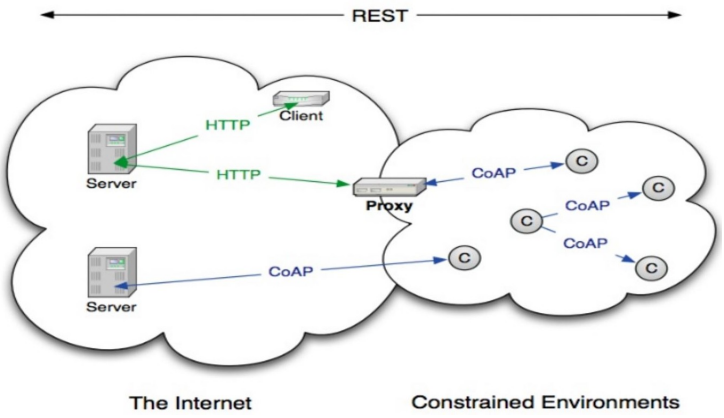

Figure 13: CoAP server client architecture [29]

5.2.2 MQTT

MQTT is an open-source lightweight messaging protocol based on publish-subscribe architecture, and it is ideal for connecting remote devices by using very low bandwidth and minimum code footprint. It is widely [44] used in IoT applications and deployed in different industries such as[44] automotive,oil&gas and manufacturing, where numerous connected devices are used for remote controlling and monitoring purposes.

How does it work?

In MQTT protocol, messages are published by clients to an MQTT Broker/Server under a topic name, and all other clients who are subscribed to that topic receive the message. Due to its publish/subscribe method, messages published to the MQTT Server could remain anonymous, and Broker handles the transferring of the messages. Even though IP addresses of clients remain anonymous, MQTT runs on top of the TCP/IP Protocol.[44]

In the image below, one can see how the temperature reading values from a sensor are published to a broker, and other clients subscribed to the topic *temperature* receives the information.

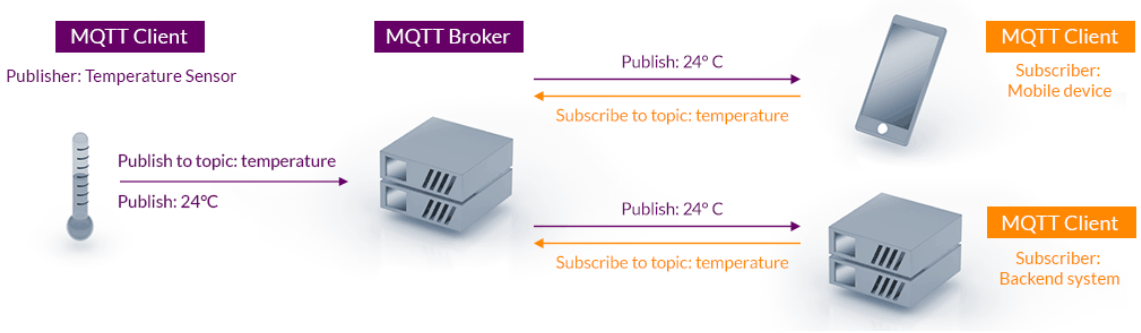

Figure 14: MQTT communication architecture [44]

Features and Benefits

Some characteristics of the MQTT messaging protocol according to the official homepage of MQTT can be listed below.

- Light-weight
  MQTT is designed to consume less power and less resources[44] so that MQTT clients can be easily implemented on microcontrollers that run on a battery. The light-weight term comes from the small size of headers(only 2 Bytes [44] used in MQTT packages. This aspect is especially useful in the MURMEL Project for the connectivity of Robots where resource and power are important.

- Bi-directional
  The communication between broker and client is bi-directional. So that data flow can be utilized in two ways, from client to broker and from broker to client.

- Scale-ability
  MQTT communication protocol can support millions of connected devices according to

official developers of MQTT[44]. Considering the future steps of the MURMEL project where thousands[7] of dustbins joined to the communication network, scaleability of the communication would become particularly important.

- Reliable Message Transport
  MQTT provides three different levels of quality of service(QoS) as 0, 1, and 2[44] for guaranteeing the delivery of published messages in different levels.

- Secure Transportation
  MQTT supports TLS encryption between clients and brokers so that messages in the air are encrypted.

Communication Parameters and Settings

In the MQTT network, there are available different settings and parameters to meet various applications' requirements. In this part, some of the critical parameters used in MQTT are explained.

Last-will and Testimony

In mobile IoT applications, the communication between clients and server mostly runs over unstable mobile networks where sudden communication disconnectivity could happen for a moment.[45]. MQTT is designed to minimize possible application failures that could arise as a result of disconnectivity by notifying the network that I have lost the connection in a graceful manner. In a normal MQTT session, clients send **DISCONNECT** packet to the server prior to a normal disconnecting procedure. However, in sudden network disconnection situations, MQTT clients disconnect from the server without sending the **DISCONNECT** package. In such situations a message called **last-will message** a.k.a (*Testimony*) under the predefined **last-Will topic** is published to server to notify that one client has lost the connection . The settings for the Last-will message and Last-will topic are optional and configured on the client-side prior to connecting with server. This feature can be especially useful in MURMEL Project for notifying the disconnectivity issues in dustbins that could arise as a result of low battery, device damage, or network issues.

Quality of Service

MQTT Protocol defines three different QoS(Quality of Service), which guarantees the delivery of messages from sender to receiver at different levels. In a publish mechanism message sender is the client and broker is the receiver, whereas in a subscribe mechanism broker is the sender, and the subscripted client is the receiver. Each client defines the level of QoS prior to establishing connection to the broker and are available as bellow;

- QoS 0 - Delivery at most one time
- QoS 1 - Delivery at least one time
- QoS 2 - Delivery exactly one time

Persistent vs Non-Persistent Sessions

When an MQTT client wants to connect to a broker, it has two connection options as a persistent and non-persistent session. If the client connects to the broker with the persistent session, the broker saves the client's subscription information. So that if the client disconnects

from the broker for a while, the broker saves the messages that are normally supposed to be delivered to the client. Once the client connects again to the broker, it receives the messages from the broker while it was offline. However, this is only possible for messages that are published with QoS 1 and 2. In a non-persistent session, on the other hand, the broker does not store any subscription information of the client, hence won't deliver the messages to the client that it missed during the time being offline. This type of connection is ideal for the client that is not required to subscribe to any topics. The configuration for persistent sessions is done by setting the CLEAN_SESSION_FLAG to false. In the MURMEL Project, Robots are critical elements in an application, and it is necessary to configure robots with persistence sessions to avoid any possible operational problem.

5.2.3 Conclusion

Even though both CoAP and MQTT are explicitly designed for IoT applications, the underlying architectures are different. While MQTT is a many-to-many communication protocol, it enables multiples clients to communicate with each other. Through the underlying decoupling, the mechanism broker decides which messages to publish to which client. CoAP, on the other hand, is a one-to-one communication protocol that is more suitable for state information transfer rather than event details [46]. Considering the network topology of MURMEL Communication architecture, implementing server nodes on mobile networks for representing the Robots is not feasible due to the IPv4 infrastructure. The effect of these critical differences on the choice of end-users can also be seen in Figure 15 which shows the apparent interest of users on MQTT more than CoAP.

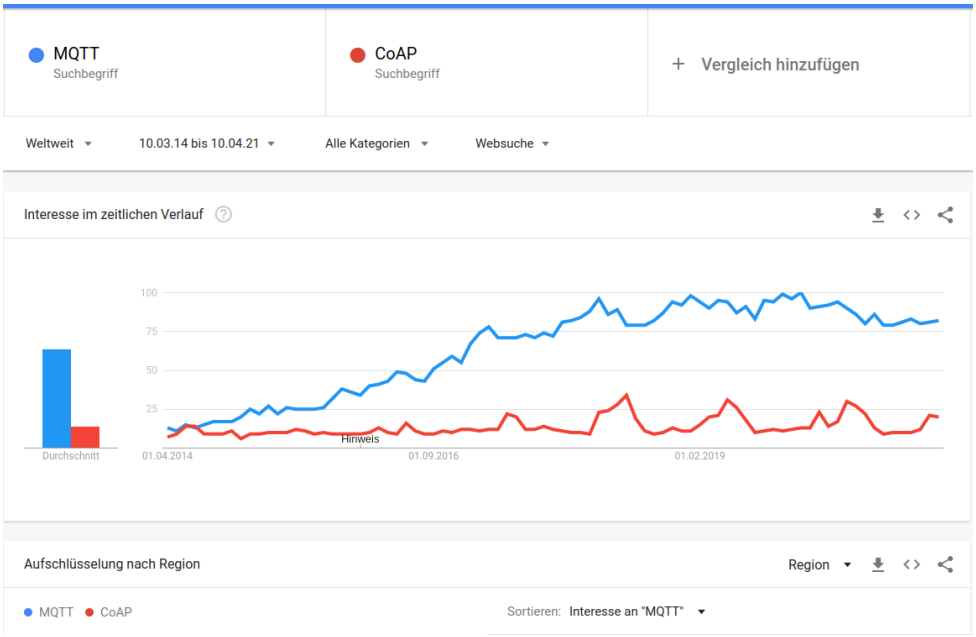

Figure 15: Google Trends: CoAP vs MQTT[24]

Also, the request-response mechanism of the CoAP creates complications in the implementations of the dashboards where more than one user might use the dashboards at the same time. Because of these reasons, MQTT will be used as a messaging protocol in the application layer of the MURMEL communication protocol.

5.3 Video Streaming Protocol

Accessing robot camera frames during an operation is an essential need in a MURMEL application for handling possible emergencies. However, the communication protocols studied so far are only capable of sending and receiving telemetry data, which requires fewer resources and less band within a network. In order to stream video frames from a vehicle camera to the MURMEL application server requires a more sophisticated solution. In the scope of this work, the following solutions are researched.

1. Running a local webserver on a robot.
2. Utilizing a web socket connection

5.3.1 Running a Local Webserver on a Robot.

One way to access the vehicle camera frame without inter-venting the camera's running tasks is by embedding a webserver node in the robot's ROS environment. After establishing the running webserver node on the robot, any client in the network can control the streaming video's flow from the robot to the client. Since the webserver is not running in a data center that directly connects to the world wide web, accessing the webserver through the internet can only be possible by exposing the local webserver to the public internet through port forwarding. The architecture of the designed system can be seen below.

However, accessing a local webserver through port forwarding in a mobile network is a challenge to be overcome. The problem lies in the architecture of the Internet Protocol(IPv4) being deployed in mobile networks. Due to the design of the current IPv4 protocol maximum of 4,294,967,296 public IP addresses are available to be used public internet, and as a result of increasing internet-connected devices, the last available IPv4 blocks were given in 2019[50]. Since this was already known that this problem was coming out, Internet Service Providers(ISP) are installed NAT-enabled gateways in base stations to provide internet access to as many users as possible[10]. However, the downside of this approach is that connected devices to the same gateway share the same public IP addresses, and due to security reasons, end users are not allowed to change gateway configuration settings for port forwarding. Hence, using IPv4-based mobile networks prevents running a local webserver on a robot for accessing it from the public internet. The working principle of running a local webserver on ROS of the robots is shown in the figure 16

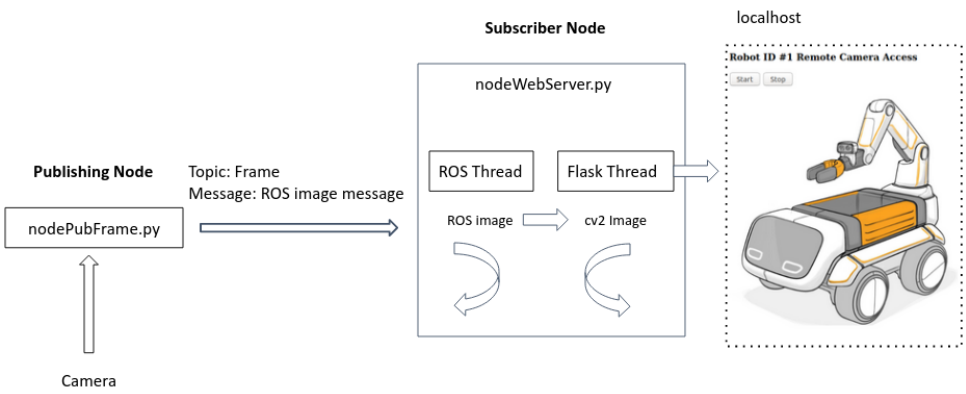

Figure 16: ROS based local webserver

5.3.2 Utilizing the RFC 6455 WebSocket Protocol

Using HTTP protocol for sending and receiving data in real time required frequent opened and closed TCP connections which increases the data usage and load on network[64].Another way for sending and receiving data in a network is to utilize socket connections. However for applications such as Streaming video frames from one end point to another end point in computer networking requires a stable and persistent connection between the participants. Even though it is possible send frames in binary format in regular TCP based computer networks, it becomes a challenge for browsers to initiate TCP connection with sockets for creating continuous data flows[64]. Right at this point, the RFC 6455 WebSocket Protocol standardised by IETF(Internet Engineering Task Force) in 2011[64] enables web browsers as a client to connect websocket servers for streaming data over a single TCP connection.

A WebSocket is a persistent connection between a client and server. Fundamentally it provides a full-duplex and bi-directional communication channel that enables applications to communicate with each other both in server to client and client to server directions. Like HTTP, it operates on a TCP/IP socket but different than HTTP, the socket connection stays open instead of closing right away as in the case of HTTP. Although by using the *Long pulling* technique, users can create long live sessions in HTTP by increasing the timeout in the created TCP/IP socket, this method causes servers resources to keep occupied even if there is no data transfer between the client and server. With the introduction, WebSocket need to use *Long Polling* method is eliminated.

In the project MURMEL, in order to initiate video streaming from robots to the application server, a persistent WebSocket connection between robots and the MURMEL application server must be created. This way streaming video contents to web browsers can be easily controlled via commands send to robots via the WebSocket connection.

5.3.3 Conclusion

Due to the restrictions in IPv4 Internet Protocol and NAT enabled gateways in mobile network operators, running a webserver on robots for streaming the video frames is not possible. In contrast to IPv4 protocol, IPv6 protocol has much more capability in terms of available IP addresses. IPv6 uses 128-bit (2128) addresses, allowing 3.4 x 1038 unique IP addresses, which corresponds to 340 trillion trillion trillion IP addresses.[50] Due to this abundance of available IP addresses in the IPv6 network, mobile operators can directly connect the end devices to the world wide web without using a NAT-enabled gateway. This way, any end device can run a server application in the mobile network without port forwarding.
However, due to high infrastructure costs, mobile network operators' transition from IPv4 to IPv6 has not been completed yet. Currently, in Germany, Deutsches Telekom mobile network infrastructure fully supports IPv6, which only became available at the beginning of 2020[19]. It is reported that Vodafone and o2 also are implementing IPv6 in mobile network, so far there isn't an official statement about the current status of the deployment[18].

Even though IPv6 gives the possibility to deploy servers on mobile networks by providing unique public IP addresses, the installed 4G modem on Robot is not supporting the IPv6 protocol. In the light of these considerations, implementing the RFC 6455 WebSocket Protocol for video streaming seems to be a better option in the MURMEL application.

5.4 HTTP Server

In order to create an interactive user interface for communicating to robots and dustbins through a web browser, an HTTP server is needed to be deployed in MURMEL Application Server. By implementing an HTTP server, mothership operators or any other users depending on the management of accessing rights, can create a communication channel with field devices. The primary services expected to be provided by the HTTP server in the MURMEL Application server are explained below.

5.4.1 Video Streaming Service

Video streaming service is an interface provided by the HTTP server, where users can access the selected vehicle's camera and control the flow of the video stream. Each time users want to access to vehicle camera through this service, the web browser will establish a connection to the WebSocket server. WebSocket Server, on the other hand, will send start or stop commands to robots to control the flow of video streaming. Once the camera frames are arrived at the WebSocket server, based on a publish-subscribe method, they will be shared with the camera service clients.

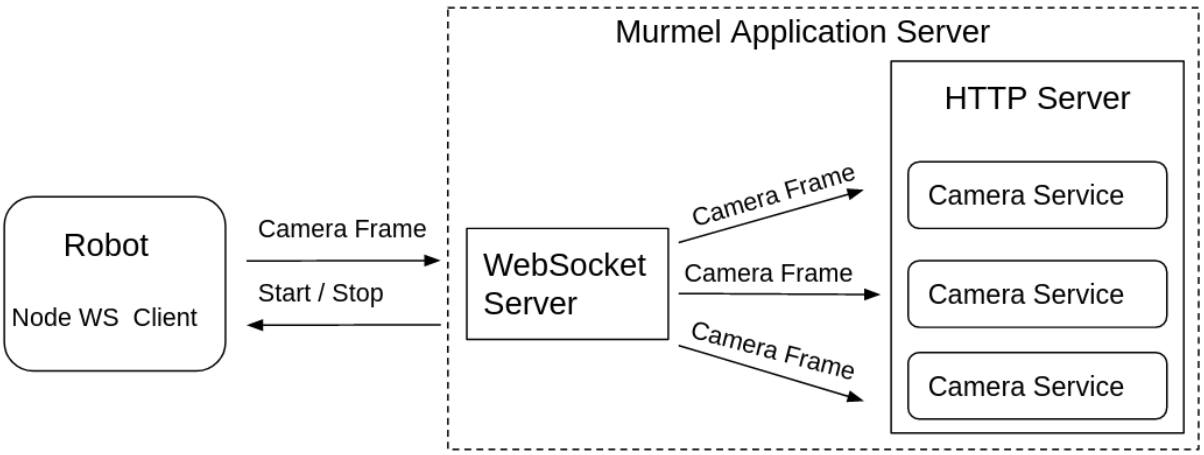

Figure 17: Video broadcasting architecture

As a result of the publish-subscribe method applied on the WebSocket server, it is aimed to keep the data usage from robot the cloud at a minimum level. Independent from the number of clients connected to the WebSocket server, the robot's frame transfer will always be kept alive as long as there is at least one camera service client subscribed to the camera frame. Once the last client sends a stop command, the WebSocket server will forward this message to the robot to terminate the camera frame transfer. This way, the outgoing IP traffic from the robot server is kept under control to prevent excessive mobile data usage.

As an alternative to the publish/subscribe mechanism applied on the WebSocket server, it is also possible to deploy a Real-Time Streaming Protocol (RSTP) server combined with the WebSocket server. However, since the project's current state only includes a single robot and a single camera, the WebSocket server can be meet the requirements easily.

5.4.2 Route Planning Service

Since the route planning algorithm is still under development in MURMEL Project, only a framework is designed under the scope of this thesis. Depending on the project's needs and scale, route planning services can be used in different ways. The usage scenarios can be listed below.

1. **Creating route plans based on smart dustbins**.
   In this service, routes are created by using the fullness information from the dustbin database. After collecting the fullness levels from dustbins, a route planning algorithm is expected to create a set of routes for the robot. Since the route planning algorithm is still under development, it will not be covered within this work scope.

2. **Creating route plans based on operator input**
   Another way to use route planning service is to update the existing route by considering the robot operator's requests. In a scenario where a robot can not continue to operate its original route for any reason, a new route might be needed to be created. This can be done by the robot operator using the Route Planning Service interface.

In the figure 18 architecture of the route planning service for updating the route of a robot is shown. The process of updating the route starts with the request of the Mothership operator by specifying the constraints. Route Planning Service takes the constraints as an input to the route planning algorithm and sends the output route to SUMO Server for verification. SUMO server simulates the created route and sends the confirmation back to Route Planning Service, which passes this information to the Mothership Operator. In the final step, the Mothership operator sends the resulting route to the WebSocket server, and webserver further forwards it to the Robot.

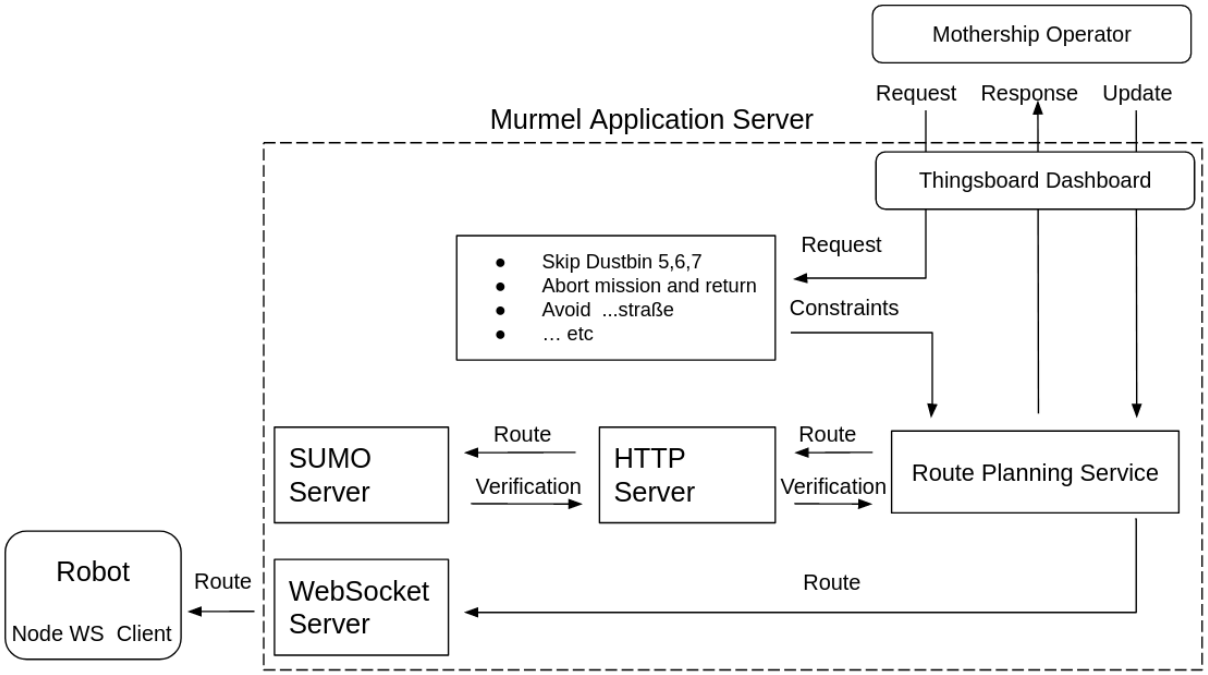

Figure 18: Route planning and route update schematic

5.4.3 Visualization Service

Visualization of the received telemetry information from vehicles and dustbins is one of the vital needs in the MURMEL Application Server. The purpose of using a visualization tool is to create dashboards for monitoring the states of the robots and dustbins. There are numerous solutions available for meeting the visualization requirements of a web application. For a better understanding, the solutions are packed into three groups as below.

1. **Cloud computing platforms**
   These are non-free services that bill the service usage, and the billings can be on a scale of 200€/Month depending on the amount of the service usage[43]. Some examples of such platforms are Microsoft Azure, AWS, Google Cloud IoT Core, and IBM Watson IoT Platform.

2. **Open source/limited IoT Services**
   These are open-source(free) services that enable to use of services such as device management, data collection, processing, and visualization for IoT projects. Some examples of such services are Thingsboard (free/limited), Kaa IoT platform (limited), Thinger.io (limited). Depending on the service provider, it is possible to use these servers in two different ways: either accessing and using the service directly through the cloud or embedding the services into local machines. Especially for applications that involve the transfer of sensitive data, it is a good practice to deploy dedicated servers and services. However, for low-scale and experimenting applications building these services on dedicated servers can be unnecessary.

3. **Creating a visualization tool from scratch**
   It is also possible to design and create a custom visualization tool without using any third-party service. Considering a web application, Javascript provides many dynamic widgets to be placed on a web page. In order to create a minimalist dashboard for the MURMEL server, only vital widgets such as map, chart, control buttons, status indicators, and tables are enough to satisfy the requirements of the visualization service. Even though it is doable from a technical point of view, it would cost extra time and effort to the project group.

In addition to the visualization services, cloud computing platforms mentioned above provide various services to their users such as data analytic, device authentication, and project management tools[43]. However, these services come with a price that could create unnecessary financial burdens to the project MURMEL. Another aspect of integrating such a service is the deployment efforts. Since this project covers only the prototyping phase of the project, integrating such a big and complicated system to the project MURMEL is not effective in terms of utilizing the project resources. Open source or limited services, on the other hand, are a better fit for the needs of the project MURMEL considering the finance and complexity factors. In this respect ***thingsboard.io*** is selected for visualization service.

The main reason for deciding on *Thingsboard.io* for visualization service is the simplicity of the system architecture and richness of the documentation. Unlike to more advanced cloud computing platforms, it doesn't have hidden costs and gives users the ability to use the free version as long as limitations are not exceeded. In this context, considering the fact that the current state of the project MURMEL doesn't require very complex and advanced solutions, thingsboard service is decided as a good start.

5.5 MURMEL Application Server Architecture

The resulting MURMEL Application Server consists of different servers and services to create the communication stack of the MURMEL Project. In the back end, MQTT and WebSocket servers were established to create communication channels with Dustbins and Robots. Since the dustbins are using LoRa communication technology, data transfer between dustbins and the MURMEL Application server is set via the MQTT API of the TTN Network Server. For storage purposes of telemetry data, a database server is used. Depending on the needs, different pieces of information can be stored in the database for later analysis. On the front end, HTTP Server and Thingsboard are used for interacting with connected devices. The general architecture of the system can be seen below.

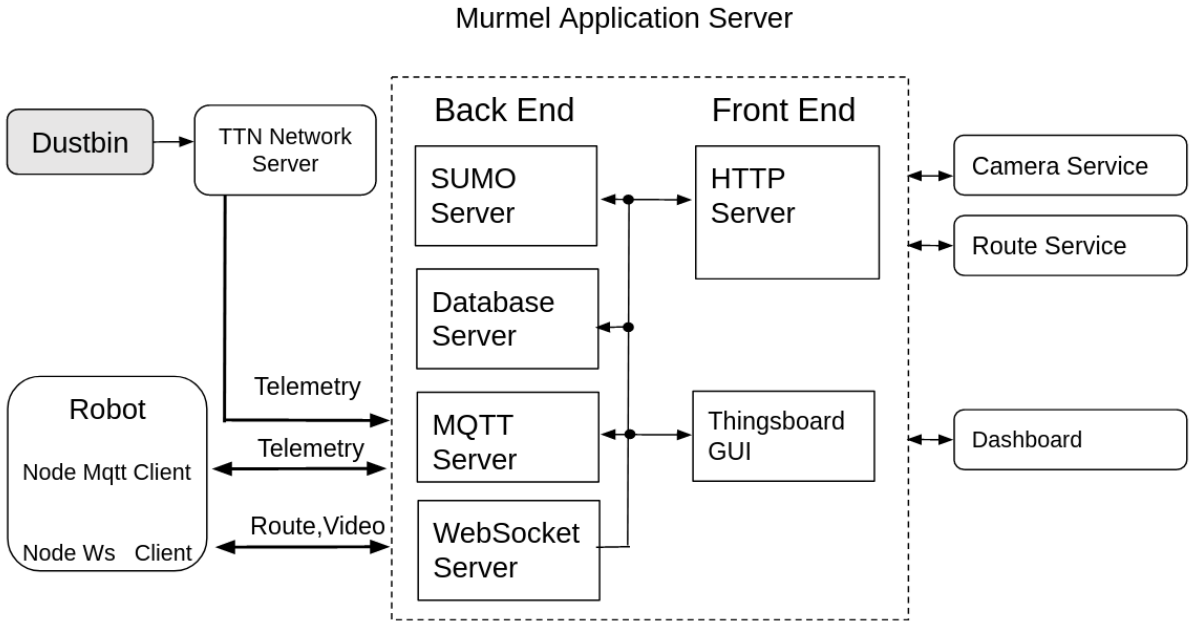

Figure 19: MURMEL Application Server Architecture

6 Simulation of a MURMEL application

In the course of the simulation, a simple MURMEL application has been simulated by integrating the services and servers of the MURMEL application server. At the end of the simulation, a dashboard was created to control the running task on Robot. For a better understanding of the simulation, a simple scenario has been prepared as below;

6.1 Simulation Scenario of a Sample MURMEL Task

1. Before starting to MURMEL operation, a sample simulation in SUMO has been created to simulate the predefined routes for MURMEL vehicles consisting of two Robots and one Mothership truck. After successfully ending the SUMO application, route files are stored in the ROS files systems of Robots.

2. The simulation starts with a Mothership truck's arrival that carries two MURMEL Robots to the mission area of concern.

3. Once the robots are brought down from the Mothership and become ready to drive, the Mothership operator activates the autopilot driving modes on the robots.

4. Each Robot starts to follow its path within its mission area by using the route files in ROS murmel communication package.

5. During their missions, robots drive to the assigned dustbins one by one by following a set of GPS coordinates, and once they arrive at the target, vehicles wait a certain amount of time for emptying dustbins. Once the dustbins are emptied, robots start to drive their next target and publish messages to the server to update the dustbin fullness level as empty.

6. Throughout the simulation, the IoT Dashboard service server receives telemetry data from vehicles and displays some of the important diagnostic parameters.

7. In order to utilize the dynamic route planning feature, a dummy error or failure on the robots will be initialized during the simulation. Upon initializing the error, the mothership operator will request a new route to route service, and a new route will be sent to the robot through the WebSocket connection. Since the route planning algorithm still in process, the route service will simply send the predefined route to the robot.

6.2 Components of the Simulation Environment

The simulation program for a sample MURMEL task consists of three main sections as listed below. For simulating the Robot, Dustbin, and mothership instances in the MQTT and WebSocket connection, the ROS framework is used. However, for illustrating easy and fast simulations, another python-based lightweight simulation is also created.

1. Route planning and verification in SUMO
2. Cloud integration of robots in ROS
3. Cloud integration of dustbins

6.2.1 Route Verification in SUMO

- The first step to creating a MURMEL simulation in SUMO is downloading the map of an area of concern from the *openstreetmap* API. Before importing the downloaded map to SUMO, the .osm map is converted to .net.xml, and corrections are done if necessary. For this task, an area close to TU Berlin has been chosen.
- Since route planning is beyond this work's scope, route configuration of SUMO is done manually by choosing the edges that pass by the GPS coordinates of dustbins. For each robot, separate routes are defined.
- After defining the vehicle types for robots and mothership, sumo simulations are run from a python script, and in each simulation step GPS, positions of robots are saved into a file called xxx. The resulting set of GPS coordinates of robots along their mission are later used in IoT Dashboard for defining the mission areas of robots.

6.2.2 Cloud Integration of the Robots in ROS

Under the package of *murmel_communication* in Robots' ROS framework, two nodes are created to establish the MQTT and WebSocket connections between the Robot and corresponding servers. The additional Robot and Mothership Instances are also created and connected to the MQTT server in the node called *simulation*. Following a successful connection to the server, robots publish the telemetry data such as GPS position, sensor states, and mission detail to the MQTT server.

6.2.3 Cloud Integration of the Dustbins

Cloud integration of dustbins is also done in the node called *simulation* by creating instances for dustbins and connecting dustbins to the MQTT server. Upon successful connection of Dustbins, GPS positions and their initial fullness levels are sent to the server.

6.3 Finalized Dashboard

The finalized dashboard for accessing the vehicle's telemetry messages and controlling specific actions consists of several GUI widgets. Since the whole dashboard is too big to fit on one page, it will be explained in multiple steps. The following widgets are needed for a good grasp of a MURMEL simulation.

6.3.1 Map Widget

The map widget below consists of two identical maps specific to a mission. The first map on the left side is used for monitoring the locations of dynamic objects, in this case specifically for Robots and Mothership. Each robot's predefined routes are colored differently to highlight working areas for each robot, and vehicle status is indicated with four different indicators.

- Steering (Green): Robot is steering to a dustbin
- Operating (Orange): Robot is operating on a dustbin
- Error (Red): Robot encountered an error
- Return (Blue): Robot is returning to the Mothership

The colors on the graph left side are indicating the states of the dustbins as full or empty. Once the robots clean the dustbins, colors are also updating their values.

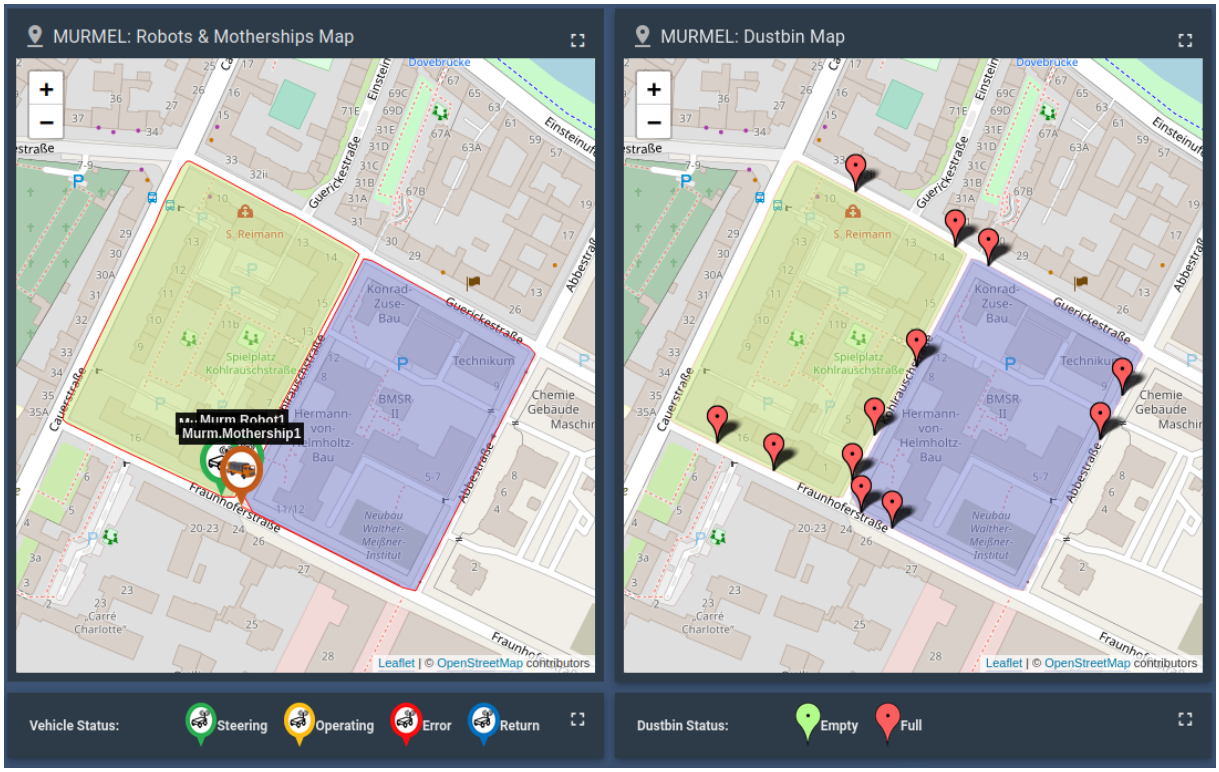

Figure 20: IoT Dashboard - Vehicle Monitoring

6.3.2 Route, Sensor and Actuator Details in Robots

Telemetry information from robots to the central server has been split into three categories: route details, sensor details, and actuator details. In the route details, parameters such as distance recovered target and distance to the target are highlighted. In the sensor details, some of the important sensor values/states such as battery level and container level are shared. In the actuator card, movements of the actuators are conveyed to closely follow the details in operation mode.

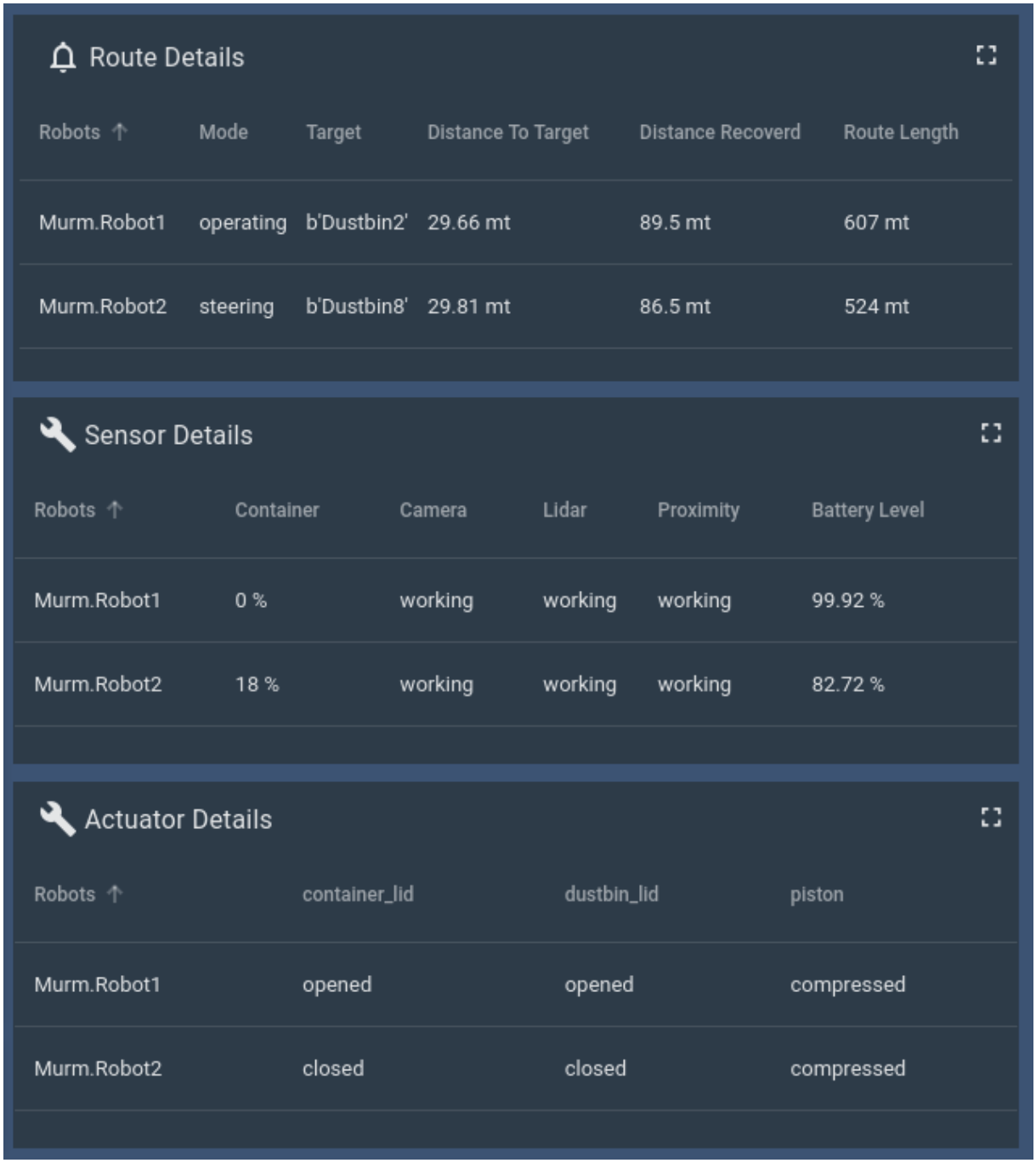

Figure 21: IoT Dashboard - Sensor, actuator and route details

6.3.3 Autopilot Control Panel

During the operation of a robot, it might be necessary to disable the autopilot feature under certain circumstances such as blocked path, sensor failure, or any other action that prevents the normal operation of the murmel. For such cases, the created control panel sends messages to the robot either to able or to disable the autopilot feature. Once the autopilot feature is deactivated, the robot stops, and the red LED light turns on. When the autopilot gets activated again, the green led turns on to show the status of autopilot. Since deactivating the autopilot mode stops the robot, it can also be utilized as an emergency brake as well.

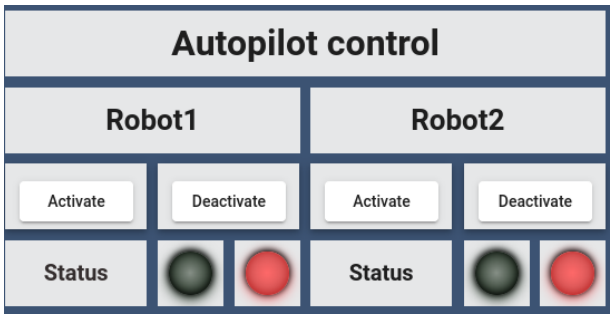

Figure 22: IoT Dashboard - Autopilot Control

6.3.4 Camera Streaming Service

For accessing to camera and route services of the MURMEL application server, two HTML iframe elements are embedded into the dashboard. Through the streaming service, one can easily start and stop the video stream from robots to the application server. Route configuration service, on the other hand, provides an interface for monitoring and updating the route information on the robots. Figure 23 shows the user interfaces of both services.

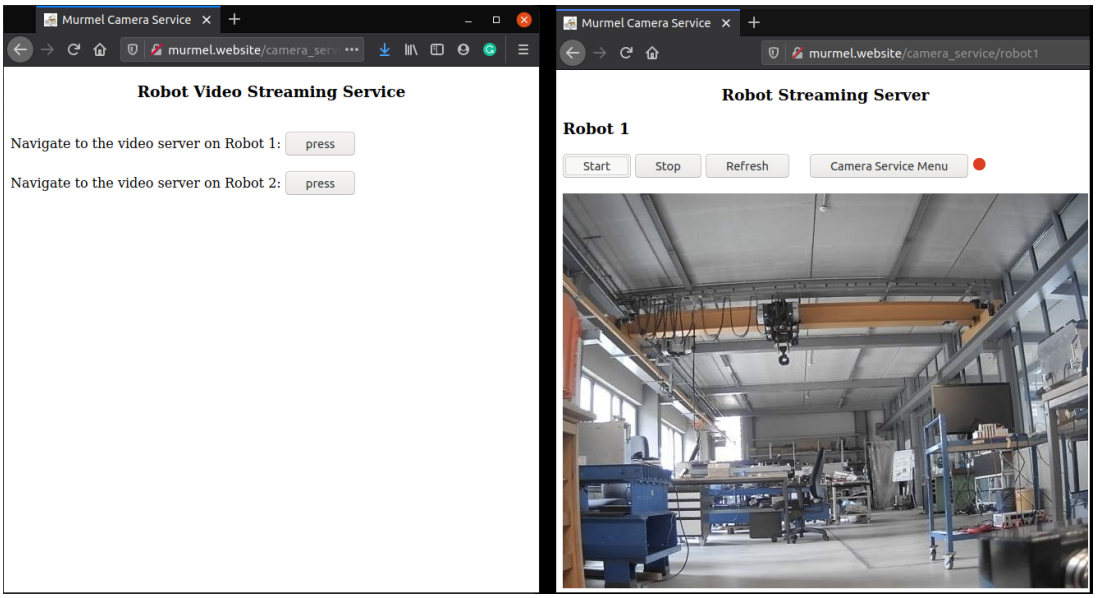

Figure 23: IoT Dashboard - Accessing to the Robot Streaming Server

6.3.5 Route Planning Service

For accessing the MURMEL application server's route planning service, HTML iframe elements are embedded into the dashboard. Through the route planning service, one can easily monitor and update the route information on the robots. For updating the route plan dynamically, a user interface is designed to send constraints to the route planning algorithm. Figure 23 shows the route information monitoring and route update user interfaces of the planning services.

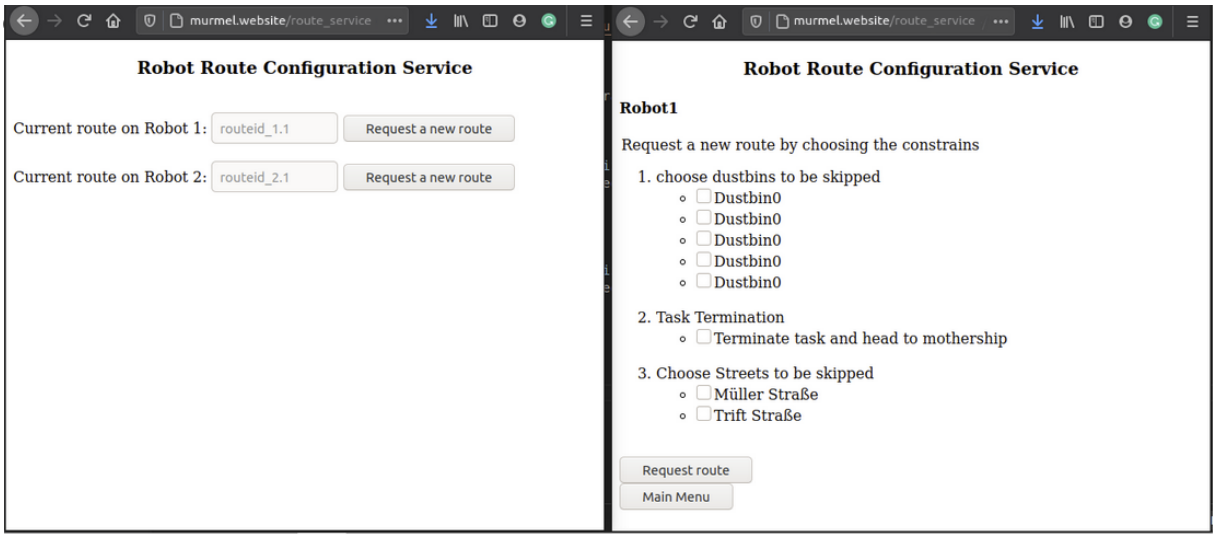

Figure 24: IoT Dashboard - Accessing to the Route Planning Service

Figure 25 shows the original and updated route plans after using the route planning service. Following a successful route planning simulation in SUMO based on the user inputs, new route information is sent to the robot through the WebSocket connection.

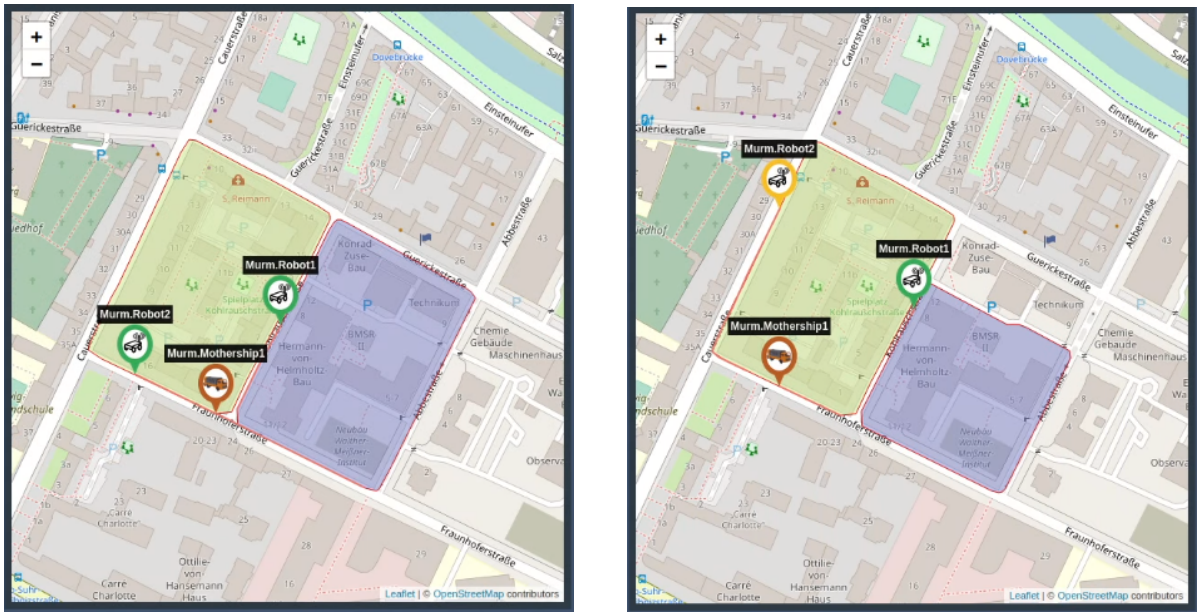

Figure 25: Original(left) and during run-time updated(right) route plans

7 Conclusion/Outlook

The communication protocol designed for the project MURMEL has been successfully tested on software and hardware prototypes and met the predefined communication requirements of a standard MURMEL application. At the current state of the project, communication modules both on the test robot and test dustbin are enabled, and all the data being transferred can be easily controlled and monitored through the IoT-based dashboard. As a conclusion of the developed solution architecture, a short review of the works done in the project's scope is given below.

Uplink Communication of the Robots

As the robot of concern is still under developing process and its components are not fully integrated, most of the vehicle parameters are not available yet. Because of that, throughout the tests, only available parameters such as GPS, battery level, and velocity information had been sent to the server as telemetry data. The rest of the parameters defined in the sensor, actuator, and route details sections of the dashboard had been sent as dummy data from the robot to the application server. It is also worthy to state that all the captured or simulated parameters used in telemetry data transfer had been chosen by focusing on the current state and the needs of the application. However, once all the robot components are integrated, the new emerging vital parameters can also easily be shared with the application server.

Downlink Communication of the Robots

The downlink communication, on the other side, had been only utilized on robots as there is no need to push data from the cloud to dustbins. Even though the ability to downlink communication of the designed protocol brings many opportunities to the robot, only a dynamic route update feature had been installed on the system due to security and safety concerns. However, it is important to note that the downlink communication on robots can be further expanded to the applications such as controlling the motion of certain actuators or even controlling the robot's movement itself. Although these features are not mandatory requirements for the robot's operation, these implementations would provide great convenience to the robot operator in exceptional cases where manual control over the robot is required. Exactly at this point, safety concerns arise due to the unpredicted results of the remote controlling actions. Any accidentally sent faulty command could cause the robot to hit a person or any foreign object where damages are likely to occur. Because of these reasons, downlink communication on robots is only limited to the update of route information. However, in future work, if the actions of such remote controlling applications are limited to certain safety boundaries through an enhanced algorithm, these features can be easily adapted by the system.

Physical Layer of the Dustbins

Designing the smart dustbins' physical layer was one of the most demanding challenges in the development phase of the communication protocol. Considering the possible implementation of wireless connectivity feature on thousands of dustbins[7] around the city, scalability and low power consumption requirements of the application have been stood out as the most critical factors to be considered. Lack of the high capacity power source on dustbins and financial aspects of the large-scale hardware implementations leads to the physical layer of dustbins on LPWAN(Low Power Wide Area Network) technologies. Among all the alternatives of LPWAN technologies, a comprehensive study was made on Nb-IoT, LoRa, Sigfox, and LTE-M

standards, comparing the network availability, data transfer limitations, and power consumption aspects. As a result of this research work, and LoRa has been chosen as the most suitable technology to be deployed in smart dustbins due to the extremely low power consumption option and open source support of the LoRaWAN Network.

Choosing the LoRaWAN Network Operator

The regulations of the LoRaWAN enables devices to be operated both on public and private LoRa networks as long as users obey the limitations[4]. This leads the design of the network infrastructure of the dustbins to be deployed on either a private or public LoRa network. At this point, *TTN(The Things Network)* public LoRa network stood out as a promising solution for dustbins to be considered. The field tests were done using the developed hardware prototype, and the coverage map supplied by the TTN Berlin community [74] proved the availability of the LoRa network in Berlin. In line with these developments, there couldn't be determined any downside of using the TTN network. However, in case of any special requirement such as increased data transfer interval or a further deployment of LoRa on other applications might require setting up a private LoRa network under the hoot of Berlin Stadt Reinigung. In such a situation, the existing hardware can be easily configured and continued to be used on a newly created private network without any additional hardware modification.

Application Layer Communication Protocols

The communication protocols deployed in the application layer had been decided based on the requirements of a standard MURMEL application. In this context, due to its lightweight [44] natures, two common messaging protocols for IoT applications MQTT and CoAP, are deeply studied to be deployed for sending telemetry data within the network. Although both options are widely used in IoT applications, it was decided to use MQTT due to the CoAP topology requirement to the IPv6 network.

In addition to using a messaging protocol for handling telemetry data, another protocol for streaming camera frames was also needed in the MURMEL network architecture. Since the purpose of the streaming protocol is to enable robot operators to access robot camera frames from mobile devices such as smartphones and tablets, the end solution must work on even simple web browsers. Under the scope of this limitation, solutions based on web applications are studied, and the WebSocket protocol stepped forth as a solution for this purpose. Through a simple publish-subscribe alike algorithm on the server-side, mobile data consumption of the robot has been kept minimum to prevent excessive data usage.

Visualization Service

In order to ease the monitoring and controlling actions on robots, a user-friendly visualization service became a fundamental need for the MURMEL server. At this point, various options are evaluated, from simple open-source platforms to complex cloud platforms such as AWS(Amazon Web Services) and Microsoft Azure. While IoT services of the cloud platforms of AWS and Azure provide a rich set of features, these solutions are relatively more expensive than other assessed solutions and have a way more complicated architecture. Therefore as a visualization tool, a free service provided by *Thingsboard* has been utilized in the front end of the MURMEL server.

8 Reference to Future Work

In this work's scope, a solution architecture had been designed for meeting the communication requirements of Project MURMEL. At the end of the work, a sample simulation has been created to illustrate the architecture of the murmel application server, and an application has been created for dustbins and robots. However, to achieve a fully integrated murmel application server, some improvements still need to be made. These are explained as below;

1. **Improvements of the Dutsbins' Communication Module**

   The first field tests done using a LoRaWAN module and TTN network have successfully demonstrated LPWAN technologies' ability to work on smart dustbins. However, as also seen in the test, the TTN network does not have %100 coverage in Berlin, and at some points near the Ernst Reuter Platz, for instance, the device fails to connect a TTN Gateway. Considering the financial advantage of using the TTN network and the early phase of the MURMEL project, such coverage issues can be tolerated. However, for achieving a truthfully working system, either deployment of more LoRaWAN gateways or transition to other LPWAN technologies such as Sigfox or NB-IoT should be considered. such as Sigfox or NB-IoT should be considered.

2. **SUMO Integration to the Back End**

   The route planning and route pushing methods are made manual in this work. For sending the route plans automatically to the robot, the being developed route planning algorithm should be integrated into the HTTP server's route planning service. This way, only validated route plans should be sent to the robot. One way to achieve this behavior would be integrating the SUMO simulation server into the MURMEL application server.

3. **Integrating Project Management Tools with MURMEL Application Server**

   Currently, project development files and all necessary documentations are stored in the TU Berlin cloud service. Even though it is not a must for creating a fully compact MURMEL Application Server, Project Management tools can also be integrated so that all the necessary tools and Robot controlling services(route planning, camera service) can be accessed via a single server.

4. **Improvements in the security layer**

   Throughout the work, for certain needs such a creating control panel dashboard and MQTT server, third party solution providers are used to enable the rapid prototyping in the project development. Cyber security aspects of the created solution are not analysed deeply hence it is beyond the scope of written thesis.However considering the MURMEL Project as a critical IoT application, more specialized custom designed server is needed to have more control on security layer.

References

[1] 3GPP. *Standards for the IoT*. June 2016. URL: https://www.3gpp.org/news-events/1805-iot_r14. (Accesed October 08, 2020).

[2] Airtel. *Airtel shuts down 3G network in Kolkata*. URL: https://www.airtel.in/press-release/06-2019/airtel-shuts-down-3g-network-in-kolkata. (Accesed October 31, 2020).

[3] Brandon Alexander et al. "Robot web tools". In: *IEEE Robotics & Automation Magazine* 19.4 (2012), pp. 20–23.

[4] LoRa Alliance. "A technical overview of LoRa and LoRaWAN". In: *White Paper, November* 20 (2015).

[5] asd. *ROS*. URL: http://wiki.ros.org/ROS/Introduction. (Accesed April 4 , 2021).

[6] ATT. *Get ready, 3G is going away in 2022*. URL: https://www.att.com/support/article/wireless/KM1324171/. (Accesed October 31, 2020).

[7] Dustbin Number Berlin. *Berlin Mülleimer*. URL: https://www.rbb24.de/panorama/beitrag/2019/03/muelleimer-verteilung-bsr-papierkoerbe-berlin.html.

[8] *BonnSmartCitStartup*. URL: https://www.businessinsider.de/gruenderszene/news/ticker-13042018/. (Accesed April 4 , 2021).

[9] *BonnSmartCity*. URL: https://analysedeutschland.de/article/6-smart-cities-04-20.html. (Accesed April 4 , 2021).

[10] Stuart Cheshire, Marc Krochmal, and Kiren Sekar. "Nat port mapping protocol (natpmp)". In: *Work in Progress* (2008).

[11] CoApIETF. *CoApIETF*. URL: https://tools.ietf.org/html/rfc7252. (Accesed January 15, 2021).

[12] CoApTech. *CoAPTech*. URL: https://coap.technology/. (Accesed January 15, 2021).

[13] *ComputerNetworking*. URL: https://minigranth.in/computer-networks-tutorial/computer-network-models. (Accesed April 11 , 2021).

[14] IoT Creators. *Nb-IoT Germany Map*. URL: https://iotcreators.com/en/coverage-eu/. (Accesed October 15, 2020).

[15] K. Dar et al. "Wireless communication technologies for ITS applications [Topics in Automotive Networking]". In: *IEEE Communications Magazine* 48.5 (2010), pp. 156–162. DOI: 10.1109/MCOM.2010.5458377.

[16] *DarmstadtSmartCity*. URL: https://www.digitalstadt-darmstadt.de/. (Accesed April 4 , 2021).

[17] *DarmstadtSmartCity2*. URL: https://logistik-aktuell.com/2019/01/15/what-makes-a-smart-city/. (Accesed April 4 , 2021).

[18] Deutschem. *IPv6 Deutschland*. URL: https://www.datamate.org/aus-ipv4-in-ipv6-serverzugriff-aus-dem-mobilfunknetz/. (Accesed March 7, 2021).

[19] DeutschesTelekom. *NeuerIPv6Zugang*. URL: https://telekomhilft.telekom.de/t5/Blog/Neuer-IPv6-Zugang-zum-mobilen-Internet-im-Netz-der-Telekom/ba-p/4254741. (Accesed March 7, 2021).

[20] Vodafone Deutschland. *Narrowband-IoT:Internet of Things*. URL: https://www.vodafone.de/media/downloads/pdf/vodafone-whitepaper-narrowband-iot.pdf. (Accesed October 08, 2020).

[21] Eclipse. *CoAPEclipse*. URL: https://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php. (Accesed January 15, 2021).

[22] Clayton Foster. "Key drivers of success for 3G". In: *Competition for the Mobile Internet*. Springer, 2003, pp. 145–167.

[23] Anju Uttam Gawas. "An overview on evolution of mobile wireless communication networks: 1G-6G". In: *International Journal on Recent and Innovation Trends in Computing and Communication* 3.5 (2015), pp. 3130–3133.

[24] Google. *MQTTvsCoAP*. URL: https://trends.google.com/trends/explore?date=2014-03-10%5C%202021-04-10&q=MQTT,CoAP. (Accesed April 10, 2021).

[25] GSMA. *Narrow Band Internet of Things*. URL: https://www.gsma.com/iot/narrow-band-internet-of-things-nb-iot/.

[26] GSMA. *Security Features of LTE-M and NB-IoT Networks*. URL: https://www.gsma.com/iot/wp-content/uploads/2019/09/Security-Features-of-LTE-M-and-NB-IoT-Networks.pdf. (Accesed October 15, 2020).

[27] Sami Salama Hussen Hajjaj and Khairul Saleh Mohamed Sahari. "Establishing remote networks for ROS applications via Port Forwarding: A detailed tutorial". In: *International Journal of Advanced Robotic Systems* 14.3 (2017), p. 1729881417703355.

[28] Berg insight. *The Global M2M/IoTCommunications Market*. URL: http://www.berginsight.com/ReportPDF/ProductSheet/bi-globaliot4-ps.pdf. (Accesed October 15, 2020).

[29] Texas Insturment. *Getting started with CoAP development*. URL: https://e2e.ti.com/support/wireless-connectivity/zigbee-and-thread/f/zigbee-thread-forum/495960/getting-started-with-coap-development. (Accesed January 2, 2021).

[30] Hadi Jamali-Rad et al. "IoT-based wireless seismic quality control". In: *The Leading Edge* 37.3 (2018), pp. 214–221.

[31] Donald Knuth. *Knuth: Computers and Typesetting*. URL: http://www-cs-faculty.stanford.edu/~uno/abcde.html. (accessed: 01.09.2016).

[32] Dhananjay Kumar. "Video streaming in 3G wireless network for telemedicine application". In: ().

[33] Marcel Linnemann, Alexander Sommer, and Ralf Leufkes. *Einsatzpotentiale von LoRaWAN in der Energiewirtschaft*. International series of monographs on physics. Springer Fachmedien Wiesbaden, 2019, pp. 71–99. ISBN: 978-3-658-26917-3. DOI: 10.1007/978-3-658-26917-3_5. URL: https://doi.org/10.1007/978-3-658-26917-3_5.

[34] LoRa-Alliance. *LoRa Products*. URL: https://lora-alliance.org/showcase/search/?_sfm_lorawan_certified_device=certified. (Accesed April 3, 2021).

[35] LoraAlliance. *Lora Alliance*. URL: https://lora-alliance.org/. (Accesed February 15, 2021).

[36] *LoRaWAN Architecture*. URL: https://iotfactory.eu.

[37] *LoRaWAN Network Availability*. URL: https://www.thethingsnetwork.org.

[38] *LoRaWAN System Integrator*. URL: https://www.zenner.de/iot/loesungen/eigenes-lorawan.html.

[39] Deutsches Telekom LTE-M. *Deutsches Telekom LTE-M*. URL: https://www.telekom.com/en/company/details/long-term-evolution-for-machines-563208. (Accesed October 16, 2020).

[40] Mavoco. *LTEvsNBIoT*. URL: https://www.mavoco.com/lte-m-vs-narrowband-iot-and-the-internet-of-things/. (Accesed October 18, 2020).

[41] Mdpi. *Challenges in Smart Cities and Intelligent Transportation Systems*. URL: https://www.mdpi.com/2071-1050/12/16/6469/htm. (Accesed February 18, 2021).

[42] Kais Mekki et al. "A comparative study of LPWAN technologies for large-scale IoT deployment". In: *ICT Express* 5.1 (2019), pp. 1–7. ISSN: 2405-9595. DOI: https://doi.org/10.1016/j.icte.2017.12.005. URL: https://www.sciencedirect.com/science/article/pii/S2405959517302953.

[43] Microsoft. *MicrosoftAHub*. URL: https://azure.microsoft.com/en-us/pricing/details/iot-hub/. (Accesed March 7, 2021).

[44] MQTT. *What is MQTT*. URL: https://mqtt.org/. (Accesed January 2, 2021).

[45] MQTTBRoker. *hiveMQ*. URL: https://www.hivemq.com/tags/mqtt-essentials/. (Accesed January 2, 2021).

[46] MQTTCoAP. *MQTTvsCoAp*. URL: https://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php. (Accesed February 19, 2021).

[47] Oscar Novo. "Making constrained things reachable: A secure ip-agnostic nat traversal approach for iot". In: *ACM Transactions on Internet Technology (TOIT)* 19.1 (2018), pp. 1–21.

[48] Pycom. *Lopy4*. URL: https://pycom.io/product/lopy4/. (Accesed April 3, 2021).

[49] SNS Reserach. *The LPWA (Low Power Wide Area) Networks Ecosystem: 2017 – 2030 – Opportunities, Challenges, Strategies, Industry Verticals & Forecasts*. Nov. 2016. URL: https://www.snstelecom.com/lpwa. (Accesed October 08, 2020).

[50] RipeNet. *Ip4runsout*. URL: https://www.ripe.net/about-us/press-centre/understanding-ip-addressing. (Accesed March 7, 2021).

[51] *SchipholLoraNetwork*. URL: https://www.kerlink.com/blog/2019/12/17/kerlink-and-dutch-partner-mcs-help-deploy-versatile-iot-network-coverage-of-amsterdam-airport-schiphol/. (Accesed April 4 , 2021).

[52] Berlin Senat. *Smart City Strategy Berlin*. Senate Department for Urban Development and the Environment, 2015.

[53] Wentao Shang et al. “Challenges in IoT networking via TCP/IP architecture”. In: *NDN Project* (2016).

[54] Sapna Shukla et al. “Comparative Study of 1G, 2G, 3G and 4G”. In: *J. Eng. Comput. Appl. Sci* 2.4 (2013), pp. 55–63.

[55] *Sigfox Berlin Coverage*. URL: https://sigfox.de/coverage/.

[56] *Sigfox Device List*. URL: https://partners.sigfox.com/search/products.

[57] *Sigfox Germyn Coverage*. URL: https://sigfox.de/sigfox-0g-erreicht-85-netzabdeckung-in-deutschland/.

[58] *Sigfox Global Coverage*. URL: https://www.sigfox.com/sites/default/files/og-guide/Sigfox%5C%20-%5C%20Introducing%5C%200G_Sept2020.pdf.

[59] *Sigfox Network Overview*. URL: https://build.sigfox.com/sigfox.

[60] *Sigfox Our Story*. URL: https://www.sigfox.com/en/sigfox-story.

[61] *Sigfox Tariffs*. URL: https://buy.sigfox.com/buy/offers/DE.

[62] *Sigfox Technology*. URL: https://paganresearch.io/details/sigfox.

[63] Rashmi Sharan Sinha, Yiqiao Wei, and Seung-Hoon Hwang. “A survey on LPWA technology: LoRa and NB-IoT”. In: *ICT Express* 3.1 (2017), pp. 14–21. ISSN: 2405-9595. DOI: https://doi.org/10.1016/j.icte.2017.03.004. URL: https://www.sciencedirect.com/science/article/pii/S2405959517300061.

[64] Dejan Skvorc, Matija Horvat, and Sinisa Srbljic. “Performance evaluation of Websocket protocol for implementation of full-duplex web streams”. In: *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE. 2014, pp. 1003–1008.

[65] *SmartWasteTelekom*. URL: https://www.telekom.com/en/blog/group/article/smarten-up-your-waste-management-587336. (Accesed April 4 , 2021).

[66] Grant Svetlana. *3GPP Low Power Wdie Are Technologies- GSMA White Paper*. URL: https://www.gsma.com/iot/wp-content/uploads/2016/10/3GPP-Low-Power-Wide-Area-Technologies-GSMA-White-Paper.pdf. (Accesed September 1, 2020).

[67] *tcpipModel*. URL: https://afteracademy.com/blog/what-is-the-tcp-ip-model-and-how-it-works. (Accesed March 9 , 2021).

[68] TELEKOM. *telekom3G*. URL: https://www.telekom.com/en/media/media-information/archive/bye-bye-3g-now-lte-is-coming-for-everyone-608220. (Accesed October 31, 2020).

[69] Deutsches Telekom. *Nb-IoT Roaming Announcement*. URL: https://www.telekom.com/en/media/media-information/archive/mobile-iot-roaming-goes-live-across-europe-598700. (Accesed October 15, 2020).

[70] TELENOR. *telenor3G*. URL: https://www.telenor.no/privat/dekning/hvorfor-stenger-vi-3g-nettet/. (Accesed October 31, 2020).

[71] tesla. *Tesla3G*. URL: https://staceyoniot.com/how-to-handle-the-end-of-3g-networks-for-your-iot-devices/. (Accesed October 31, 2020).

[72] Priyanka Thota and Yoohwan Kim. "Implementation and Comparison of M2M Protocols for Internet of Things". In: Dec. 2016, pp. 43–48. DOI: 10.1109/ACIT-CSII-BCD.2016.021.

[73] TTN. *TTNMQTT*. URL: https://www.thethingsnetwork.org/docs/applications/python/index.html. (Accesed April 3, 2021).

[74] *TTN Community*. URL: https://www.thethingsnetwork.org/community.

[75] *TTN CommunityBerlin*. URL: https://www.thethingsnetwork.org/community/berlin/.

[76] *TTN Limitations*. URL: https://www.thethingsnetwork.org/docs/lorawan/limitations.html.

[77] Wireless Communication Technologies TUTORIAL. "tutorialspoint. com". In: *Tutorial Point* (2017).

[78] ublox. *NEO6Series*. URL: https://www.u-blox.com/en/product/neo-6-series. (Accesed April 3, 2021).

[79] verizon. *verizon3G*. URL: https://www.verizon.com/support/knowledge-base-218813/. (Accesed October 31, 2020).

[80] Vodafone. *vodafone3G*. URL: https://www.vodafone.de/newsroom/netz/bye-bye-3g-vodafone-setzt-kuenftig-noch-mehr-auf-lte-und-5g/. (Accesed October 31, 2020).

[81] Lopa J Vora. "Evolution of mobile generation technology: 1G to 5G and review of upcoming wireless technology 5G". In: *International journal of modern trends in engineering and research* 2.10 (2015), pp. 281–290.

[82] *What is LoRa*. URL: https://www.semtech.com/lora/what-is-lora. (Accesed February 15, 2021).

[83] *Wireless Communication Technologies*. URL: https://www.embien.com/blog/introduction-to-lora-technology/. (Accesed February 7, 2021).

[84] Muneer Bani Yassein, Mohammed Q Shatnawi, et al. "Application layer protocols for the Internet of Things: A survey". In: *2016 International Conference on Engineering & MIS (ICEMIS)*. IEEE. 2016, pp. 1–4.

[85] Yujie Zhu et al. "An NB-IoT-based smart trash can system for improved health in smart cities". In: *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE. 2019, pp. 763–768.