BROWSER-BASED LIGHTWEIGHT CROSS-PLATFORM WEB CONTROL SYSTEM FOR AUTONOMOUS ROBOTS OPERATING WITH ROS

MEHMET EMRE SÖYÜNMEZ

Thesis for the Master's Program in Electrical and Electronics Engineering
Graduate School
Izmir University of Economics

İzmir
2023

BROWSER-BASED LIGHTWEIGHT CROSS-PLATFORM WEB CONTROL SYSTEM FOR AUTONOMOUS ROBOTS OPERATING WITH ROS

MEHMET EMRE SÖYÜNMEZ

THESIS ADVISOR: ASSOC. PROF. DR. PINAR OĞUZ EKİM

A Master’s Thesis Submitted to
The Graduate School of Izmir University of Economics
The Department of Electrical and Electronics Engineering

İzmir
2023

ETHICAL DECLARATION

I hereby declare that I am the sole author of this thesis and that I have conducted my work in accordance with academic rules and ethical behavior at every stage from the planning of the thesis to its defense. I confirm that I have cited all ideas, information and findings that are not specific to my study, as required by the code of ethical behavior, and that all statements not cited are my own.

Name, Surname: Mehmet Emre Söyünmez

Date: 22.06.2023

Signature:

ABSTRACT

BROWSER-BASED LIGHTWEIGHT CROSS-PLATFORM WEB CONTROL SYSTEM FOR AUTONOMOUS ROBOTS OPERATING WITH ROS

Söyünmez, Mehmet Emre

Master's Program in Electrical and Electronics Engineering

Advisor: Assoc. Prof. Dr. Pınar OĞUZ EKİM

June, 2023

With each day passing, autonomous robots are spreading more and more in our lives. They mostly started with logistics and heavy production lines but now they are being used in every sector and facility from hospitals to airports, cafeterias to libraries and even in people's houses. Initially there weren't used to be a standard to operate a robot, each robot developer was creating their own system and using it on their robots or buying it from other developers but lately the ROS (Robot Operating System) became a widely used tool. Even the biggest companies are using ROS on their robots. Remote controlling the robots on the other hand, required connecting the ROS system on the robot from another computer with a communication protocol such as SSH, then operating the robot by its command line, which also requires some Linux knowledge. To solve this, companies made applications that run on the remote PC that would work as a bridge. This however forces the remote device to have some specific requirements like having a Windows operating system, having 4GB of RAM, having administration access etc. To overcome this, I'm building a browser-based lightweight cross-platform system to control the robots that are using ROS from any kind of device or operating system that can run a browser with pure JavaScript, this could be an old laptop, mobile phone or even a smart watch.

Keywords: Remote Control, Browser, Lightweight, Cross-Platform, Autonomous Robotics, Robot Operating System.

ÖZET

ROS İLE ÇALIŞAN OTONOM ROBOTLAR İÇİN TARAYICI TABANLI HAFİF, ÇAPRAZ PLATFORMLU WEB KONTROL SİSTEMİ

Söyünmez, Mehmet Emre

Elektrik Elektronik Mühendisliği Yüksek Lisans Programı

Tez Danışmanı: Doç. Dr. Pınar OĞUZ EKİM

Haziran, 2023

Her geçen gün otonom robotlar hayatımızda daha da yaygınlaşmakta. Çoğunlukla lojistik ve ağır üretim hatlarıyla başlayan, ancak şimdi hastanelerden havalimanlarına, kafeteryalardan kütüphanelere ve hatta insanların evlerine kadar her sektörde ve tesiste kullanılmaktadırlar. İlk başlarda bir robotu kullanmak için herhangi bir standart bulunmaksızın, her robot geliştirici kendi sistemini oluşturup kendi robotlarında kullanmakta veya diğer geliştiricilerden satın almaktaydı. Ancak son zamanlarda Robot İşletim Sistemi (ROS) yaygın bir araç haline geldi. En büyük şirketler bile robotlarında ROS'u kullanmaya başladı. Öte yandan robotları uzaktan kontrol etmek, robot üzerindeki ROS sistemine başka bir bilgisayardan SSH gibi bir iletişim protokolü ile bağlanmayı gerektiriyor ve ardından robotu komut satırından çalıştırmak ya da emir vermek gerekiyordu, ki bu da belirli bir oranda Linux bilgisi gerektirmekte. Bu sorunu çözmek için şirketler, köprü görevi görecek uzaktan bir bilgisayarda çalışacak uygulamalar geliştirmeye başladılar. Ancak bununla birlikte, bu uzaktan

cihazın belirli gereksinimlere sahip olmasını gerektiriyor, örneğin Windows işletim sistemine sahip olması, 4 GB RAM'e sahip olması, yönetici erişimi olması vb. Bunun üstesinden gelmek için, ROS ile çalışan robotları her türlü cihaz ya da işletim sistemiyle, tarayıcı çalıştırabilen herhangi eski bir dizüstü bilgisayar, cep telefonu veya akıllı saat gibi, saf JavaScript kullanarak kontrol etmek için tarayıcı tabanlı hafif ve çapraz platformlu bir sistem oluşturmayı hedeflemekteyim.
Anahtar Kelimeler: Uzaktan Kontrol, Tarayıcı, Hafif, Çapraz Platform, Otonom Robotik, Robot İşletim Sistemi.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to Assoc. Prof. Dr. Pınar Oğuz Ekim for her guidance and insight throughout the research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ROS: Robot Operating System

Wi-Fi: Wireless Fidelity

SSH: Secure Shell

GB: Gigabyte

RAM: Random Access Memory

CLI: Command Line Interface

GUI: Graphical Unit Interface

CPU: Central Processing Unit

GPU: Graphics Processing Unit

HTTP: Hypertext Transfer Protocol

REST: Representational State Transfer

MQTT: Message Queuing Telemetry Transport

WebRTC: Web Real-Time Communication

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

GSM: Global System for Mobile Communications

GPRS: General Packet Radio Service

HTML: Hypertext Markup Language

CSS: Cascading Style Sheets

UI: User Interface

URL: Uniform Resource Locator

DNS: Domain Name System

IP: Internet Protocol

SDK: Software Development Kit

LiDAR: Light Detection And Ranging

CHAPTER 1: INTRODUCTION

Robotics system development and integration have significantly increased over the past several years across a variety of sectors such as automobile and auto parts manufacturing, mechanical processing, electrical electronics, rubber, plastic, food industries (Xiao and Xu, 2019). Due to its adaptability, modularity, and broad community support, the Robot Operating System (ROS) has become a widely used framework for creating and managing robots. The ROS system made the control of the robots much simpler however, since it runs on the device itself, controlling the device remotely started to become more important. To solve this, web-based control systems emerged and enabled the users to connect with and manage robots remotely using web browsers. With a simple internet connection, users can operate robots from anywhere around the world, creating new opportunities for remote operation, monitoring, and cooperation.

Using the web-based control systems allows users to build an interface to be accessed from a web browser and this eliminates the need for installing complex software on company machines. Also, this system makes it possible to control the robots from a safe distance where the robots are working in challenging or hazardous environments. With this the risky fields like space exploring, searching, and rescuing and hazardous material handling processes can be handled with more safety.

Additionally, web-based systems are able to handle real-time monitoring and data visualization such as robot statuses, sensor readings and video feeds remotely. With this the users can make decisions and adjustments any time during an operation. Also, multiple users can access the control system simultaneously which could encourage teamwork and knowledge sharing.

In this situation, using ROS with web-based control systems can significantly improve the capabilities and usability of robots. Web-based control systems can take advantage of the current ROS infrastructure and smoothly interact with the robot's control stack by utilizing the ROS ecosystem, which provides a comprehensive collection of tools and libraries for robotics development.

Overall, web-based control systems for robots with ROS can result in a path toward enhancing the functionality and accessibility of robotic systems. These technologies have potential to transform how we engage with robots and increase their power across a variety of sectors and areas by enabling remote operation, monitoring, and collaboration.

1.1. Problem Statement

Controlling the robots remotely provides a variety of advantages, yet these methods are usually not user-friendly, require high-spec devices and special software installations, this limits the accessibility and creates challenges for companies with low-spec computers or the students that want to improve themselves with limited budget (Fernando, Jayawardena and Rajapaksha, 2022). As a result, the demand for a specifically designed to be a lightweight system that will control ROS-based robots is increasing. This problem statement discusses benefits of developing such a system, with cost savings, compatibility improvements with existing systems and operational efficiency increase and the flexibility.

Firstly, in the scenario of acquiring a new ROS-based robot, the one might need a new device to support this robot’s software, as most of them have high-spec requirements. Upgrading or changing the existing computers can be a financial issue, especially for the companies with a large fleet of robots. By developing a lightweight system that can operate on a simple browser, companies or individuals who want to improve themselves can make use of their old low-spec devices. This can make the firms focus their resources on other aspects and it could also make them use ROS-based robots within their other operations.

Another thing to discuss is that a browser-based system offers compatibility advantages. Currently almost all of the computers already support web browsers as default, making it easier to integrate the lightweight control system to their existing system. This eliminates the need for complex software installations, special drives, and additional training for employees since most of them are already familiar with web browsing and they can use the interface with less effort.

Furthermore, the proposed system increases the flexibility and accessibility of operations. With a browser based lightweight control interface the users can control their robots from any device that supports a web browser, such as smartphones, tablets, and even smart watches. Also makes the user not tied to the facility of the robots and allows them to control and observe the robots even from the other side of the world.

The following sections of this thesis will include 5 chapters as follows; preliminaries, related work, methodology, results, and conclusion. The second chapter will provide general information about ROS, communication interfaces and web browser working principles. Third chapter will overview the similar structures and common remote control of ROS-based robots. In the fourth chapter more detail will be given about the ROS environment, communicating, and storing the data to a database and design choices to make the browser system more lightweight. Followed by the fifth chapter, the results will be presented, and the proposed system will be discussed in the last chapter with its advantages, disadvantages, and improvable points.

CHAPTER 2: PRELIMINARIES

This thesis proposes a lightweight system to control and monitor ROS-based robotic systems and general information about this proposal will be explained in this chapter and more detail about them will be given in the next chapters.

2.1. ROS (Robot Operating System)

The Robot Operating System (ROS) is an open-source framework used on robotic projects that helps building applications with a set of software libraries and tools such as state-of-the-art algorithms and drivers (ROS, 2021). It has visualization and project packaging management tools, device drivers, libraries and more importantly a publisher/subscriber communication model. Currently it is widely used in the robotic community from small educational robots to heavy industry and logistic robots.

2.2. Robot Control Interfaces

Traditional robot control interfaces can be examined in 2 titles: Command Line Interfaces (CLI) and Graphical User Interfaces (GUI).

CLIs use text-based commands entered to the terminal or command prompt of the robots to control them. They offer simple and effective ways to give orders however these require familiarity with the commands and syntaxes.

GUIs on the other hand, are more visual and interactive. They can have buttons, sliders and visual feedback thus making them more user-friendly. Nevertheless, they demand more resources as they rely on graphical rendering.

2.3. Web-Based Interfaces

Web-Based Interfaces provide flexibility and accessibility when it comes to controlling the robots with the use of web technologies. These operate on web browsers and when it comes to accessibility, compatibility, and ease of use they give multiple advantages. There are some frameworks that specialize in robot controls such as Robot Web Tools, Blockly, Webots, Gobots etc. yet they are mostly not compatible with old devices and smartphones.

2.4. Web Communication Protocols

In order to communicate and exchange data between web browsers and servers, communication protocols are used. They simplify the process to transfer information over the internet and make the web applications interact with remote servers more reliably and standardized. Most used such protocols are HTTP (Hypertext Transfer Protocol), WebSocket, WebRTC (Web Real-Time Communication), MQTT (Message Queuing Telemetry Transport) and REST (Representational State Transfer). Each of these protocols has its own use case but almost all of them can be used in robotic applications, which will be discussed in the following chapters.

2.5. Low-Spec Device Challenges

Developing effective and usable control interfaces for web-based systems on devices with limited resources, such as low-end devices, presents a number of difficulties.

Limited Processing Power: Low-Spec devices generally don’t have enough processing power, fast CPUs, and much available memory. This causes decrease in responsibility and interface performance, thus resulting in delays in given commands and bad user experience.

Restricted Network Bandwidth: Devices with low network connectivity or bandwidth affects the communication between the interface and the robot, resulting in latencies or limited data transfers.

Small Display Size: Low-Spec devices such as old smartphones of smart watches generally don’t have a big screen thus making it crucial to consider when designing the GUIs.

Limited Battery Life: For portable devices, battery life can be a critical issue. Especially old devices have worn out batteries and the GUIs should be designed so they will minimize energy consumption and optimize power usage.

Browser Compatibility: Some Low-Spec devices don't support current frameworks or browser capabilities. Testing and being sure about the compatibility results in better user satisfaction rates.

Scalability and Responsiveness: Devices with limited resources can have problems when handling computationally high or complex algorithms so optimizing the system’s architecture and algorithms are necessary for scalability and responsibility.

Optimizing the system's performance, minimizing resource usage, and using lightweight frameworks can overcome these challenges and make control interactions even on low-end devices more efficient.

CHAPTER 3: RELATED WORK

Browser-based controlling and monitoring the robots has been an important topic even before the ROS era. Previous studies state that in order to control a robot browser can be used as a front-end system by allowing the user to connect to a server (Hiraishi, Ohwada and Mizoguchi, 1998). For the web-based robot controlling systems, the robot and the user need to be communicating in real time and it is very crucial to have valid data (Liu et al. 2003).

The most common computer networking protocols are TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). TCP often transfers with slightly more latency because of its error detection mechanisms but it is more reliable and used where the data is more crucial. UDP on the other hand, sacrifices reliability for speed and efficiency, however in the robot communication scenarios, it is not preferred since the data loss is mostly not acceptable. It is generally used at multimedia transfers between the robot and the user (Wang, Ma and Dai, 2004).

While WebSocket is the most suitable communication protocol in this case, some of the other protocols are worth mentioning as follows.

3.1. WebRTC (Web Real-Time Communication)

WebRTC is an open-source library that is used for real-time communication, mostly for video and audio streaming. It uses peer-to-peer communication, offers strong security with encryption, and supports web browsers and mobile applications. The downside is it may not support the old web browsers on some devices. (Anand, Kambhampaty and Rajalakshmi, 2022) mentions they used WebRTC to transmit LiDAR point cloud data with low latency and they encountered frame drops and slight jitters, and with the greater payload more latency occurred. Also not having sufficient bandwidth also increases the latency so it is not suitable for low-end devices.

3.2. MQTT (Message Queuing Telemetry Transport)

MQTT is a lightweight instant message protocol designed for the networks with low bandwidth. It is more useful when it comes to source-poor robots that have bad hardware and network (Yan, Shi, Wei and Pan, 2017). MQTT works on a publisher-subscriber model where the devices publish messages, and the other devices subscribe to these messages to transfer data. The architecture can be summarized as shown below in Figure 1.

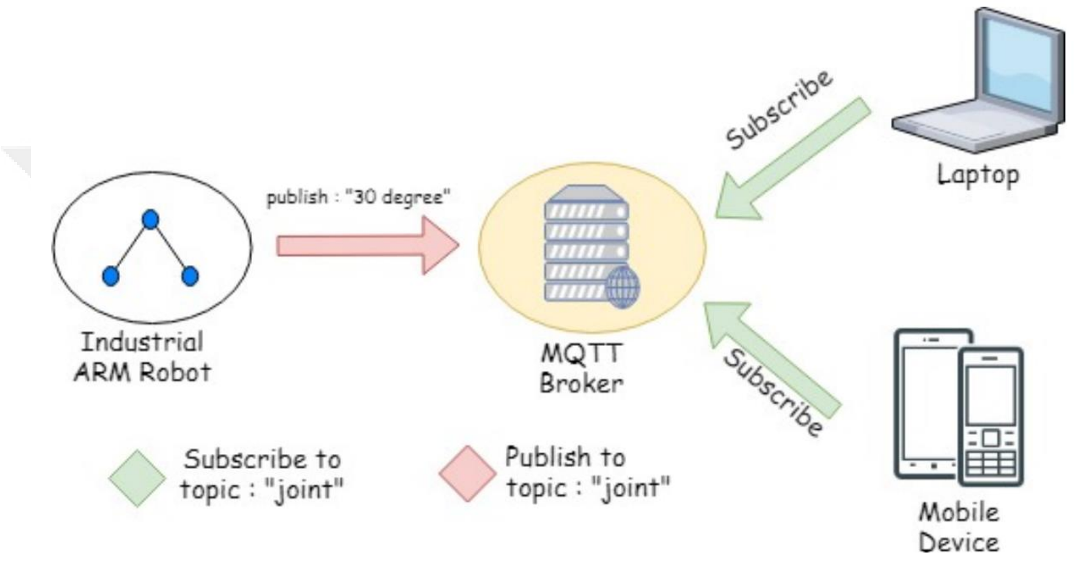


Figure 1. MQTT Architecture (Source: Atmoko and Yang, 2018)

It is mostly used in Internet of Things applications and suitable for low-spec devices however since it communicates through Wi-Fi, it doesn't fit the thesis's case where the user is expected to control the robot remotely by not being tied to geographical location. Also, as (Kato et al. 2021) states, the more devices connected to the broker, the larger delay becomes so again it is not suitable in this case.

3.3. Bluetooth

Bluetooth can also be a candidate for communication protocols for robotics. It can be used to control single robots as (Kuo, Cheng and Wu, 2014) did by remotely using a cleaning robot from an android phone paired with the robot through Bluetooth protocol. It is widely supported by various devices, and it is very friendly for energy-constrained devices with its low power consumption capabilities. Also, it provides a pairing mechanism thus making the connections more secure. However, it requires very specific cases as follows.

Limited Range: Bluetooth has a very limited range up to 10 meters and also can be easily blocked by the obstacles.

Bandwidth Limitations: Bluetooth has a very limited bandwidth and causes the data with large volume to be transmitted very slowly.

Interference: As it uses 2.4 GHz frequency range, it can be easily interfered in environments with multiple Bluetooth devices.

Latency: Bluetooth is not very good when it comes to time sensitive applications due to protocol overhead.

3.4. GSM (Global System for Mobile Communications)

Often used together with GPRS (General Packet Radio Service), GSM was used in robotics, however it is not preferred in modern projects. It has extensive coverage in many regions and a simple protocol however limited bandwidth, subscription costs and licensing requirements makes it an undesired protocol. (Makaya, Chatelain and Snyman, 2004) proposed a GSM based robot control system and it could be decent for its time, but the approximate 4-5 seconds of action times will not be preferred for modern solutions.

3.5. HTTP (Hypertext Transfer Protocol)

HTTP is the most used communication protocol on the web that works on a client-server communication model. It is mainly used to retrieve web pages, send data between the browsers and servers, and take care of lots of web-based services and applications.

In robotics, the robot can act as a HTTP server, storing the user interface elements such as the HTML, CSS, and JavaScript files. The browser uses these files to show the GUI to the user, then the user can interact with the UI elements such as buttons, sliders etc. to make the browser send a HTTP request to the HTTP server, the robot in this case. The server side processes this request and after extracting the necessary information, it can perform actions accordingly. This could be moving motors, setting the sensors, capturing images or any other function that the robot has. Once the task is finished, the server can create a response and send it back to the browser as feedback and the user can see the outcome of their actions in real-time.

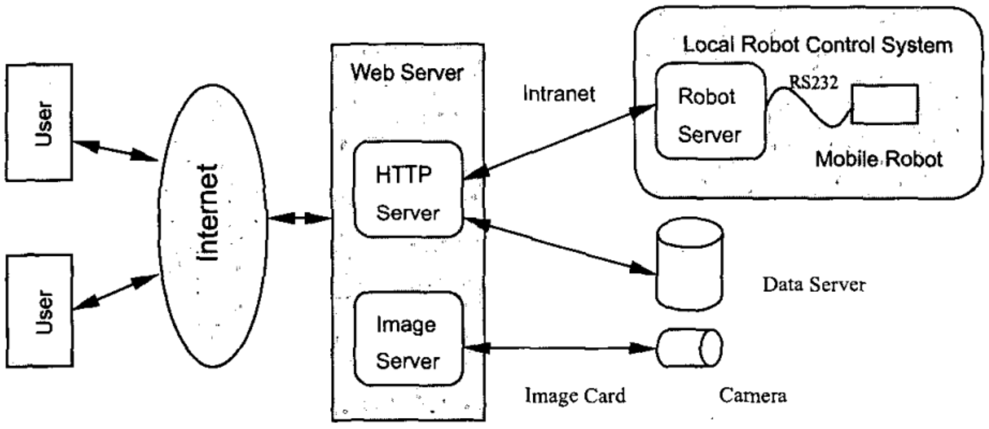


Figure 2. The structure of a web-based mobile robot system (Source: Chen, Geng and Woo, 2003)

The HTTP structure (shown in Figure 2.) is widely used in the robotics sector. (Gradil and Ferreira, 2016) developed a system to monitor both the simulation and robot environment on a browser, (Richtr and Farana, 2011) built a browser control system for a robotic arm with status responses. Another study in China (Ma et al. 2006) provided an '*Internet-based multi telerobot exploration system*' where the user can control multiple robots at one through the user interface.

3.6. WebSocket

WebSocket is another most used protocol along with HTTP, however they are mostly designed for separate scenarios. While the HTTP is used to retrieve data from a server in a traditional request-response protocol for various web-based interactions, WebSocket establishes bidirectional, full-duplex communication between the client and server in real time for scenarios with low-latency, efficiency and continuous communication is needed. The general architecture is shown as in Figure 3.

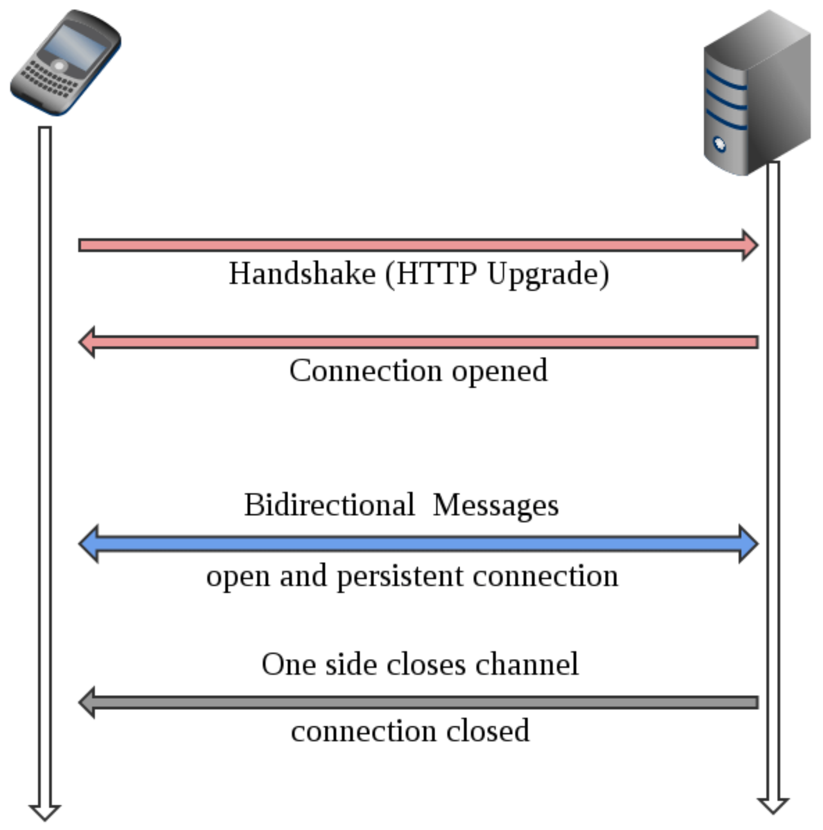


Figure 3. WebSocket architecture (Source: Chika and Esther, 2019)

The working principle is as follows (Hong et al. 2019); Client sends a specific request to the WebSocket server. The server parses the request then generates a response for the client. If the above two processes are successful, a persistent bi-directional connection is established. Client and server now can both send and receive data from each other, and if there are no new messages, they send probes to each other to keep the connection on hold. When one side closes the channel, both the client and the server close their own connection.

WebSocket is often used for applications needing real-time updates, low latency, and efficient data transfer. Previous studies show that there are multiple instances where WebSocket is used for robotic solutions. (Kohana and Okamoto, 2014) proposes a remote-control system for Sphero robots. The authors used WebSocket to connect a server and browser and also stated that the system works both on computers and smartphones. (Casãn et al. 2015) stated that they developed a remote education system where the user can connect to the robot then control and monitor the robot and even implement codes from their browsers. According to a study in Prague (Sikand et al. 2021) the authors developed a robotic two-way communication which is secure, reliable, and has low latency with the help of WebSocket to control a fleet of robots. Another study also states that they could give commands to their robot from a web browser or their smartphones (Ma et al. 2018). (Codd-Downey et al. 2014) proposed a WebSocket-based bridge between Unity and ROS systems to enter the simulation environment with Virtual Reality glasses and control the simulated robot. Also, a big Japanese robot company Rapyuta stated that they are using WebSocket on their fleet control system since it provides full-duplex, bidirectional communication with their robots and it allows server and robot to send data to each other (Mohanarajah et al. 2015).

3.6.1. ROSbridge

ROSbridge is a recent WebSocket-based tool that allows communication between the ROS-based robot and user through a web interface. (Toris et al. 2015) mentions they can use ROSbridge to control and monitor the simulation and real robot through a web browser. Another study proposes they are planning to build a smart wheelchair and control it with a web browser (Islam et al. 2023). They state they can autonomously navigate and monitor the simulated wheelchair in real-time by using web servers.

Table 1. below shows the comparison of these communication methods in a well-ordered way.

Table 1. Comparison of Connection Types

Connection Type	**Protocol**	**Communication Type**	**Data Transfer**	**Reliance**	**Latency**
WebRTC	UDP	Peer-to-peer	Real-time	High	Low
MQTT	TCP	Publisher - Subscriber	Small packets	Medium	Low
Bluetooth	Bluetooth	Peer-to-peer	Small data	Medium	Low
GSM	TCP/IP	Client/Server	Small to large	High	High
HTTP	TCP	Client/Server	Small to large	High	Medium
WebSocket	TCP	Full duplex	Real-time	High	Low
ROSbridge	TCP	Client/Server	Real-time	High	Low

CHAPTER 4: METHODOLOGY

This chapter will discuss more detail about the technology and techniques that the proposed method uses.

As mentioned in the previous chapters, in order to control a robot remotely and efficiently, proper communication methods must be chosen (Söyünmez, 2023). I also want the system to be able to work even when the robot is not online, thus a database is added to the equation that is able to store every kind of data that both the user and the robot can offer.

For a database, Google's developed NoSQL platform Firebase is used, as it works in real-time, supports WebSocket, simple to use, has freemium package and high performance. A Server is also added to be able to help the browser reach the website files more easily from different devices. Figure 4. shows the architecture in a basic way.

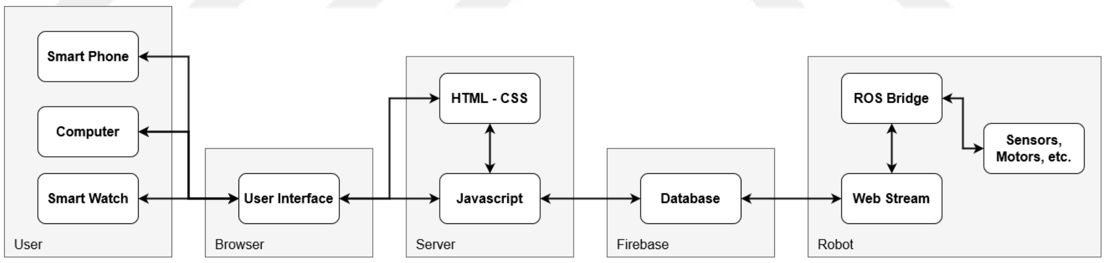


Figure 4. General architecture of the proposal.

The functions of the user interface will be explained in the next chapter.

4.1. ROS

Robot Operating System is an open-source tool to help the developers by offering low-level device control, hardware abstraction and communication. It has various libraries, and the community is creating more libraries to be publicly available with each passing day. ROS is widely used from commercial products like cleaning robots to robotic arms, drones, and mobile platforms. It works as a bridge with pub-sub communication principle and does not limit the user with specific programming languages.

The subscribed or published data can be a sensor data, custom message, state of robot, direction orders and many more. To be able to send or receive such messages remotely may require a flexible communication principle. Firebase’s NoSQL database structure shines here as it doesn’t restrict the user and supports various data models.

I cannot share much of the code on the robot that was used to test the proposed method as much of it is classified information, however the robot’s structure is irrelevant in this case. It is enough to mention that the robot has a running stream that listens to the Firebase database continuously with a REST API, which is a protocol that allows computer programs to communicate through the internet, and if there is a given command, the stream directs it to the relevant packages with the help of ROS. Also, the stream can push various of the robot's data such as sensor readings, position and path information, diagnostics data etc. Means the robot has the capability to send and receive data with the database without any problem.

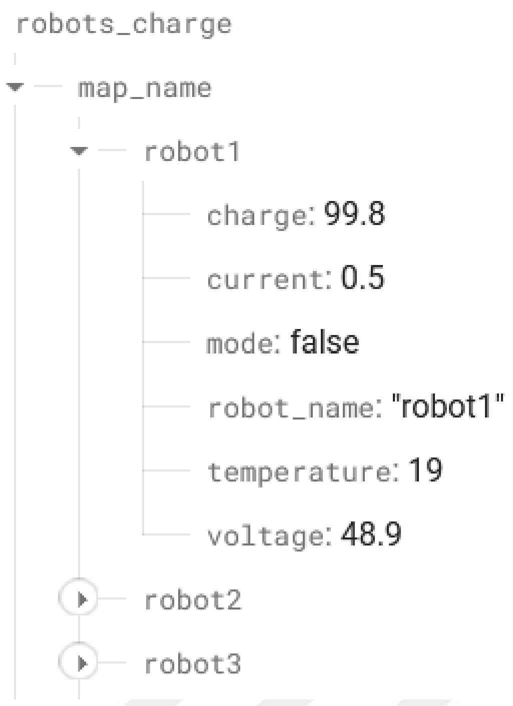
4.2. Firebase

Firebase is a tool developed by Google to offer several cloud-based services for helping the developers to build and manage web applications in an easier way. It has lots of features and functions such as real-time database, hosting, cloud storage, authentication of users and more.

The real-time database, called the Cloud Firestore, is a NoSQL database which allows to store different types of information without a strict plan in a very flexible way. The NoSQL structure also allows the database to be accessed by multiple users at once. This means that both the robot and the user can store or request data at the same time.

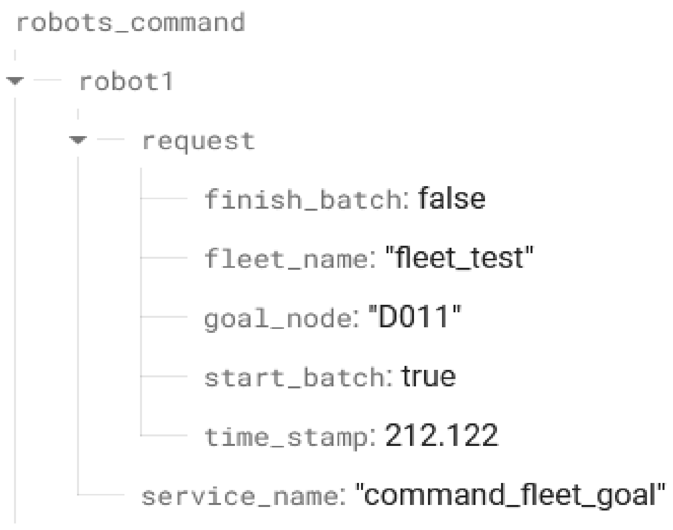
Firebase also allows authentication and user management. The authentication can be traditional methods like email and password structure but also it allows social media logins such as Google accounts, Facebook, and Twitter. The admins can also manage which user can access which part of the database. Means the robot can reach some parts of the database that are unavailable to the users or other robots and vice versa.

Firebase can integrate well with popular frameworks and SDKs for several programming languages. It can be accessed with WebSocket so makes it a valid candidate for the proposed method.



```
robots_charge
▾ map_name
    ▾ robot1
        charge: 99.8
        current: 0.5
        mode: false
        robot_name: "robot1"
        temperature: 19
        voltage: 48.9
    ▸ robot2
    ▸ robot3
```

Figure 5. Firebase database data example received from the robot.



```
robots_command
▾ robot1
    ▾ request
        finish_batch: false
        fleet_name: "fleet_test"
        goal_node: "D011"
        start_batch: true
        time_stamp: 212.122
    service_name: "command_fleet_goal"
```

Figure 6. Firebase database data example for order given to the robot.

A sample data sent by the robot is shown in Figure 5. and a command sent by the user shown in Figure 6. in Firebase Interface. As can be seen the data could be stored and accessed easily even when it is in a complex form.

4.3. Server

A server is a computer or a system that is used to store and transfer web content to the clients upon their requests. It can store various files such as HTML, CSS and JavaScript files, images, videos, and documents. The server can be a physical computer or virtual environment.

When a user accesses a website through their browser, the browser sends a request to the hosting's server. The server then transfers the necessary files back to the browser to render and display the web pages. The browser renders the main structure of the web page with the HTML and uses CSS to visualize the appearance of these structures. JavaScript on the other hand handles the interactivity of the websites.

The servers can handle multiple requests simultaneously efficiently through the internet and this makes it really useful when it comes to control multiple robots at once or multiple users want to access a single robot.

4.4. Web Browser

Web browser is a software that can run on a computer or a smart device that is used to access and view the websites. It's working principle is as follows:

The user inputs the URL (Uniform Resource Locator) address of the website they want to visit. Then the browser sends a request to the DNS (Domain Name System) to transform the URL address into an IP address. After knowing the IP address, the browser can establish a connection with the server at that IP.

The browser then sends a HTTP request to the server, which is generally a request to receive the web site structure files like HTML. The server creates a response to the request with the files and then the browser starts to render the website. After the rendering process is complete the user can interact with the website and if necessary, the browser sends additional HTTP requests to gather more data from the server.

The process goes on as the user navigates through different pages or interacts with the website elements such as buttons, sliders, input boxes etc. Here to make the system more low-spec friendly a few things should be considered.

Performance Optimization: The loading times can be improved by minimizing the use of large images, compressing the files, and making use of the cache of the websites. Also, some web tools can be used to find the bottlenecks and work on those.

Responsive Design: Rendering the website on high resolution while the screen is small and has low resolution can take a while. Detecting the device's specs and adapting the screen accordingly can improve the performance of the website.

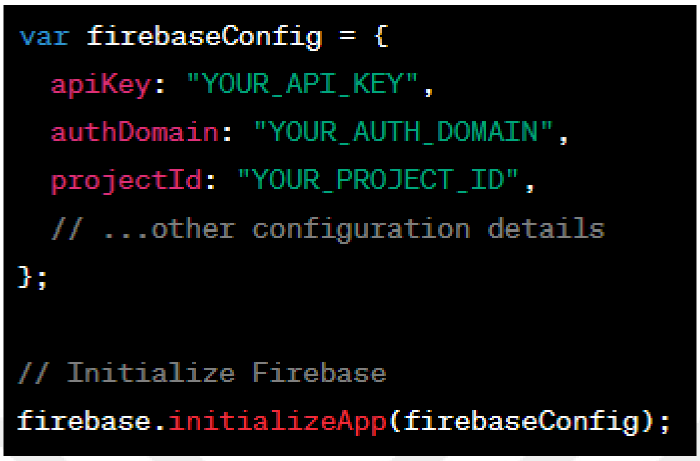
Keeping Simplicity: Getting rid of unnecessary plugins, libraries, and scripts can increase the lightweightness of the website significantly. The animations also can take too much resource on low-spec devices especially if they are JavaScript animations and not CSS animations. Decreasing the amount of those animations and transitions improve the experience greatly.

4.5. JavaScript Frameworks

JavaScript is a programming language used for increasing the interactivity and dynamicity of the web pages. It is a client-side scripting tool, meaning that it runs on the user's browser and not on the server. JavaScript can manipulate the website without needing to request the server again. It can create animations, send emails, visualize data and much more.

In order to simplify and accelerate the development of web-based technologies, some libraries or preset tools that are developed to be able to provide pre-written code, which are called the JavaScript Frameworks. Some of the most popular frameworks are React, AngularJS, Vue.js, Ember.js, and Node.js. However, since the proposed method is aimed to be used on both the low-spec and old devices, using these frameworks are not preferred, since they may require some installations or constrain the project. Instead, the JavaScript codes in the web page will be implemented with pure JavaScript which means that it will not rely on any external libraries or frameworks. Yet it needs to be noted that the Firebase JavaScript SDK will be used.

In order to use the Firebase SDK, firstly it needs to be included in the html file. Then Firebase needs to be initialized from the JavaScript. A simple code to handle this is shown in Figure 7.

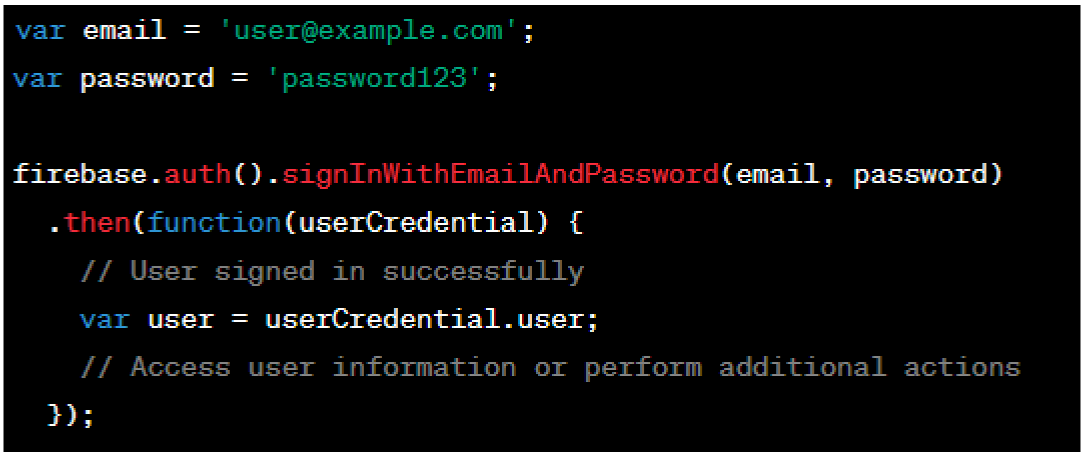


```
var firebaseConfig = {
  apiKey: "YOUR_API_KEY",
  authDomain: "YOUR_AUTH_DOMAIN",
  projectId: "YOUR_PROJECT_ID",
  // ...other configuration details
};

// Initialize Firebase
firebase.initializeApp(firebaseConfig);
```

Figure 7. Firebase initialization code for JavaScript.

Here the apiKey, authDomain, and projectId will be obtained from the Firebase interface and only necessary to be initialized once per web page. If your database requires authentication, you need to sign in after the initialization to be able to communicate with your database. Figure 8. shows a basic signing code.

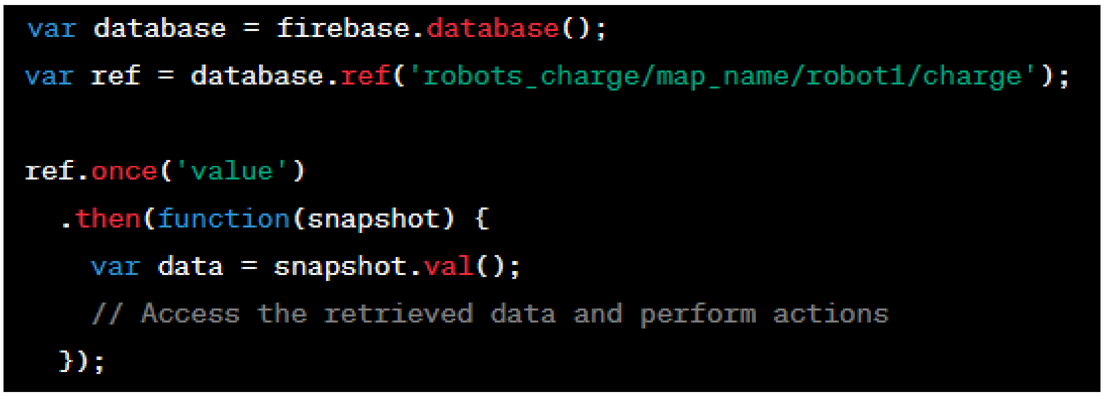


```
var email = 'user@example.com';
var password = 'password123';

firebase.auth().signInWithEmailAndPassword(email, password)
  .then(function(userCredential) {
    // User signed in successfully
    var user = userCredential.user;
    // Access user information or perform additional actions
  });
```

Figure 8. Firebase authentication code for JavaScript.

Now the database is ready to use, and we can read or store data. If the user wants to read the charge data from Figure 5, the following code can be used to retrieve its value in the float or double form (Figure 9.)

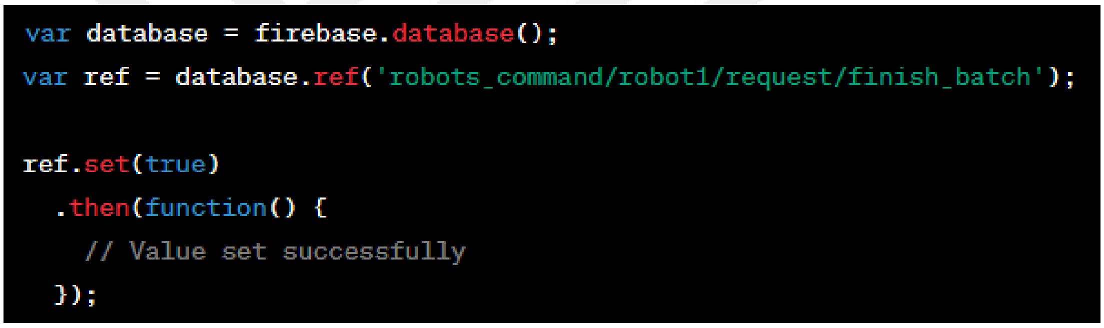


```
var database = firebase.database();
var ref = database.ref('robots_charge/map_name/robot1/charge');

ref.once('value')
  .then(function(snapshot) {
    var data = snapshot.val();
    // Access the retrieved data and perform actions
  });
```

Figure 9. Reading sample data from Firebase database.

Now if the user wants to store a value to the database, they can use database.ref.set() function. Below (Figure 10.) shows a way to set the finish_batch variable to ‘true’ at the command data in Figure 6.



```
var database = firebase.database();
var ref = database.ref('robots_command/robot1/request/finish_batch');

ref.set(true)
  .then(function() {
    // Value set successfully
  });
```

Figure 10. Setting data to the Firebase database.

The next chapter will present a web control system by basically using these functions and also the methodology and technology mentioned in this and previous chapter and tests of them on a real robot. Various functions of the system will be shown without going much in detail and showing the codes for each of them.

CHAPTER 5: TEST ENVIRONMENT AND RESULTS

This chapter will explain the testing and the results. For testing purposes, an Advoard brand mobile differential drive mobile robot platform will be used as shown in Figure 11. For testing the browsed based system, a decent Monster brand laptop, a Samsung J4 Core smartphone and a Samsung Active 2 model smartwatch will be used.

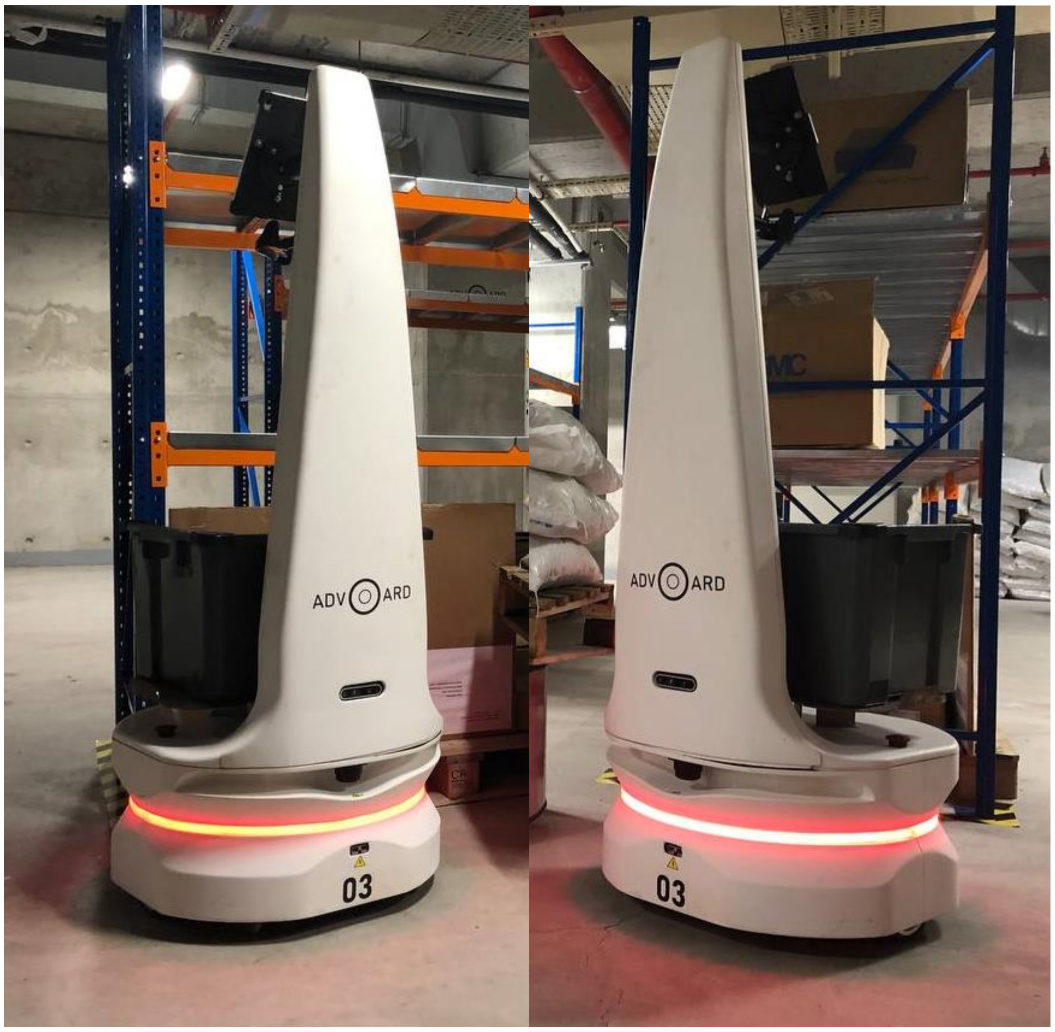


Figure 11. Advoard ORCA model autonomous mobile piece picking robot.

Since the robot uses LiDAR and coordination based 2D maps to navigate itself, the leaflet library is used to visualize the robot data better and give commands more easily.

5.1. Leaflet

Leaflet is an open-source JavaScript library that provides lightweight and flexible interactive maps that can be integrated with web applications. Leaflet allows users to create dynamic maps by having markers, polylines, polygons, and interactive tools such as pan and zoom and also can integrate with map providers like Google Maps to create the best map the user might need.

One of the things to consider here is that the robot uses the x y coordination system while the leaflet uses the longitude and latitude coordinate system. So, transformations should be taken into account when developing the JavaScript code. The origin [0,0] coordinate will be at the center of the map on the robot side however when it comes to the leaflet, the origin is the left top side of the screen, and the resolutions are most likely not the same so a proper scaling will be necessary as well. Since the robot codes are considered classified information, I'm not allowed to present a sample code in this part. A sample leaflet map can be seen in Figure 12.

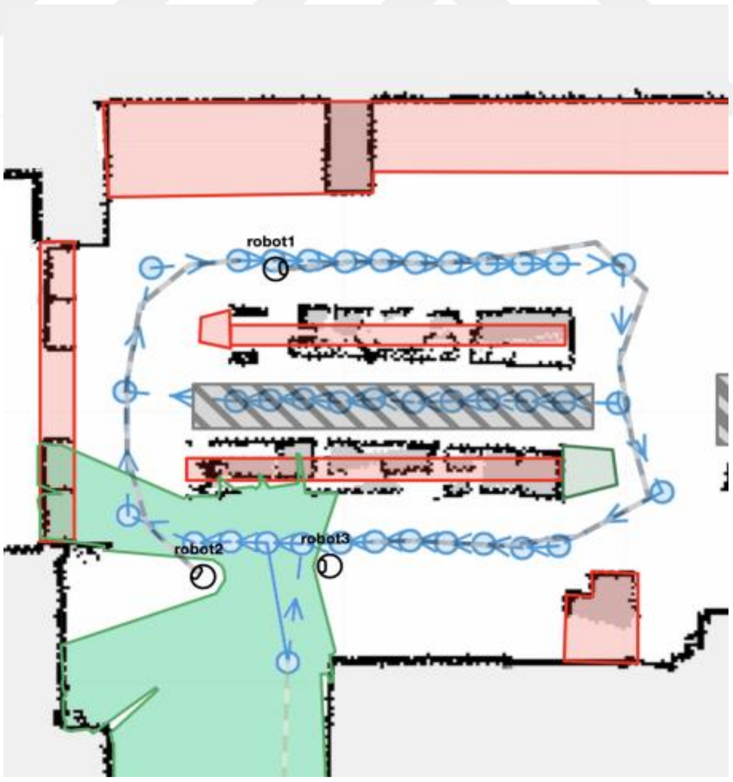


Figure 12. Leaflet map with preset regions, polygons, and markers.

5.2. Control Interface

The web control system of the robot offers a variety of functions for both monitoring the robot such as diagnostics, battery state, position, and scan data, and giving it orders like starting SLAM, declaring tasks, sending to a point with free navigation and remote emergency stop.

5.2.1. Interactive Map

The web system has a leaflet based interactive map as shown in Figure 13. The user can pan and zoom the map, initiate free navigation with a simple click and drag function, declare restricted areas as can be seen as the red areas, green dotted occupancy areas where only one robot at a time can be present and show the path of the robot. The user can also define speed zones where the robot speeds up or slows down according to the given input.

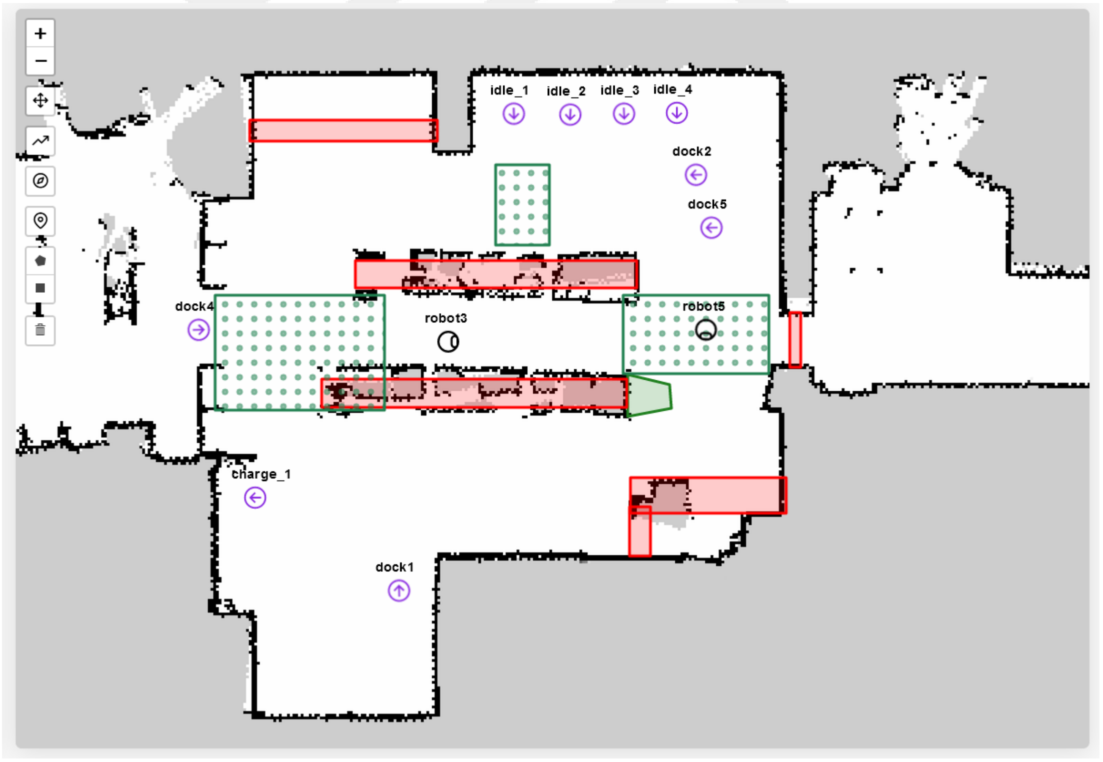
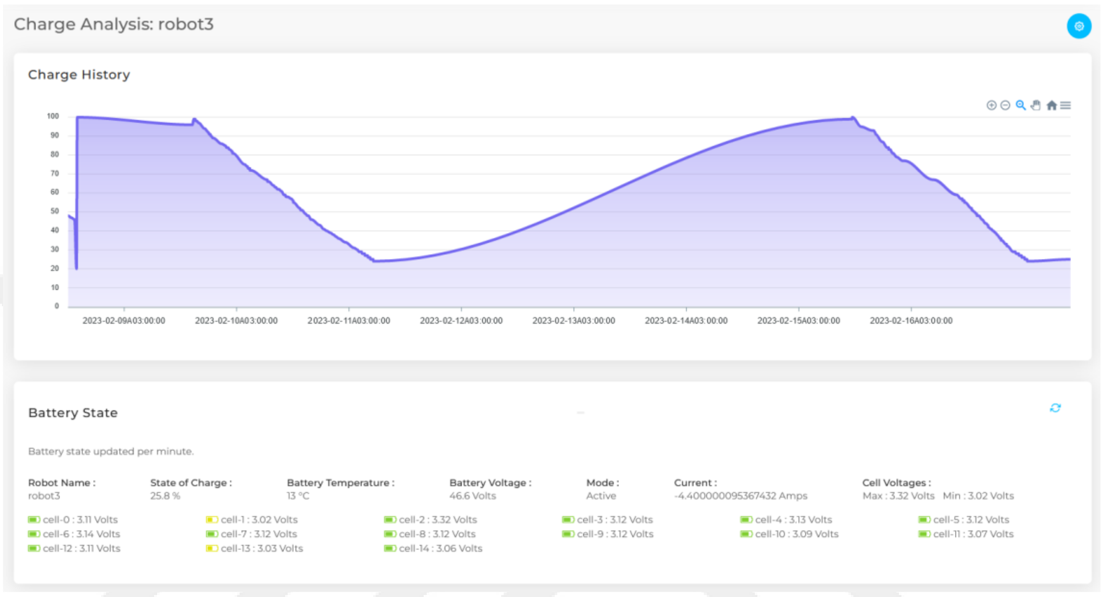


Figure 13. A sample monitoring screen showing the location of the robots, stations, and the restricted zones.

5.2.2. Monitored Data

The robot shares lots of information while it is active such as the live battery status and charge history (Figure 14.), active robot nodes and topics, current parameters, and an option to set them in real time, show the diagnostics and status notifications (Figure 15.).


Charge Analysis: robot3

Charge History

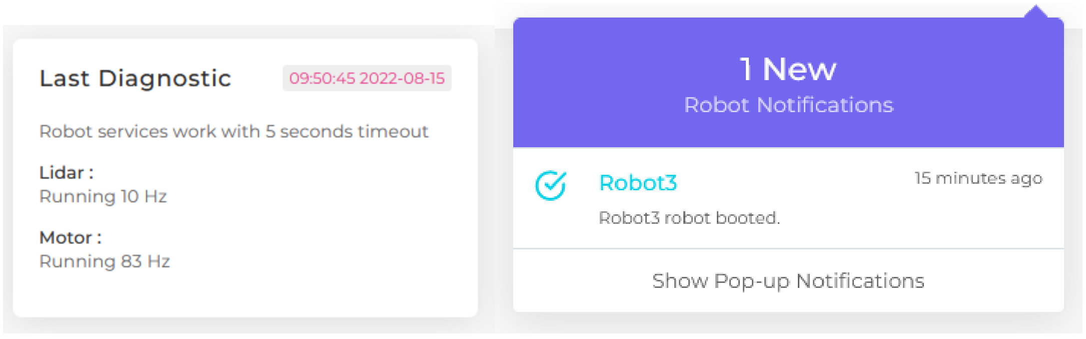
Battery State

Battery state updated per minute.

Robot Name :	State of Charge :	Battery Temperature :	Battery Voltage :	Mode :	Current :	Cell Voltages :
robot3	25.8 %	13 °C	46.6 Volts	Active	-4.400000095367432 Amps	Max : 3.32 Volts Min : 3.02 Volts

cell-0 : 3.11 Volts	cell-1 : 3.02 Volts	cell-2 : 3.32 Volts	cell-3 : 3.12 Volts	cell-4 : 3.13 Volts	cell-5 : 3.12 Volts
cell-6 : 3.14 Volts	cell-7 : 3.12 Volts	cell-8 : 3.12 Volts	cell-9 : 3.12 Volts	cell-10 : 3.09 Volts	cell-11 : 3.07 Volts
cell-12 : 3.11 Volts	cell-13 : 3.03 Volts	cell-14 : 3.06 Volts			

Figure 14. A monitored battery data of the mobile robot.


Last Diagnostic 09:50:45 2022-08-15

Robot services work with 5 seconds timeout

Lidar :
Running 10 Hz

Motor :
Running 83 Hz

1 New
Robot Notifications

Robot3 15 minutes ago
Robot3 robot booted.

Show Pop-up Notifications

Figure 15. Live diagnostics and status reports of the robot.

An issue encountered here is when the charge history data is too big, generally more than one month of accumulated data, some low-spec devices, such as the smartphone in this case, can struggle while processing the data. However, after the processing is complete the web page works as intended with no other problems.

5.2.3. Testing Devices

Three testing devices are used to check the compatibility of the system. The first one is a Monster brand Laptop with 9th Generation i5 Core and 8GB RAM, can handle everything smoothly but this was expected and will not be used as a benchmark.

The smartphone Samsung J4 Core is a fairly new phone released in 2018 however it can be considered a low-spec device with the limited 1GB ram with a big 6-inch screen makes it struggle in some cases. The phone can handle the web control system finely as shown in Figure 16. Nevertheless, when the device needs to do resource hungry processes in the background, (accumulated charge history over a long period for example) it can show some frame drops and freezing or leggings until the process is complete.

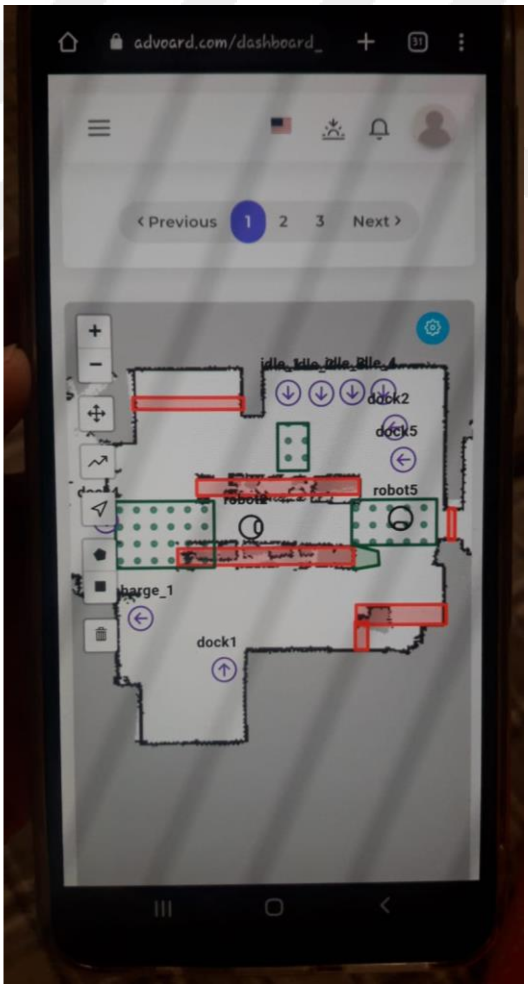


Figure 16. Low-Spec smartphone showing no problems with the proposed method.

Final testing device, Samsung Active 2 smartwatch on the other hand didn't show any sign of struggles even when the big charge history data is processed. The device had 768MB RAM with 4GB storage and Exynos 9110 dual-core processor. The downside was the round shaped interactive screen made it impossible to access some features of the website that are normally at the edges of the screen, since they were considered to be working on rectangular screens, but this can be overcome with adjusting the places of the said function's placement on the screen when the resolution is too low.

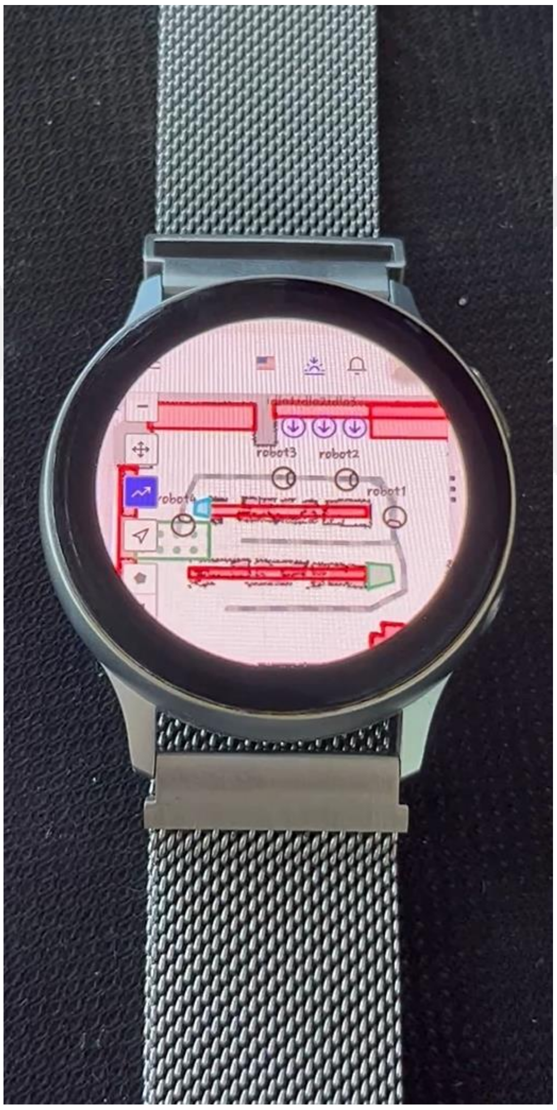


Figure 17. Smartwatch running the proposed solution flawlessly.

CHAPTER 6: CONCLUSION AND FUTURE WORK

This thesis shows that developing a lightweight browser-based control system for robots operating with ROS will be beneficial for both the robot providers and their customers by reducing the need for a high-level control unit for robotic solutions, especially when it comes to operating multiple robots at once. Instead, the solution can lead the saved money to be spent in other aspects and maybe encourage the customers to invest in the robotic solutions more.

The proposed solution is by using a lightweight pure JavaScript structure to make the browser application use less resources on both the device itself and the network. Furthermore, installing a database between the robot and the remote-control unit makes it possible to store various information about the robot and makes the system usable even when there are no active robots. Also in the test scenario, using Firebase as a database did not create any struggles for the low-end devices furthermore, it allowed the data transfer traffic to be more organized.

Although the proposed solution is working as it is, for future work the processing concept of big, accumulated data can be optimized, as right now some devices with low memory can experience freezes and lags while working with long data. Thus, in order to have a better and more robust system, the high calculation processes could be executed in a remote server rather than the client’s browser.

REFERENCES

Anand, B., Kambhampaty, H. R. and Rajalakshmi, P., (2022) *A Novel Real-Time LiDAR Data Streaming Framework*. IEEE Sensors Journal, Vol. 22, no. 23, pp. 23476-23485.

Atmoko, R. A. and Yang, D. (2018) *Online Monitoring & Controlling Industrial Arm Robot Using MQTT Protocol.* IEEE International Conference on Robotics, Biomimetics, and Intelligent Computational Systems (Robionetics), Bandung, Indonesia, pp. 12-16.

Casañ, G. A., Cervera, E., Moughlbay, A. A., Alemany, J. and Martinet, P., (2015) *ROS-based online robot programming for remote education and training*. IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, pp. 6101-6106.

Chen, Q., Geng, H. and Woo, P. -Y., (2003) *Research on and pure Java realization of a Web-based mobile robot system.* Proceedings of the American Control Conference, 2003., Denver, CO, USA, pp. 615-620 Vol.1.

Chika, Y. -B. and Esther, O. K., (2019) *Financial stock application using WebSocket in real time application.* International Journal of Informatics and Communication Technology (IJ-ICT), University of Lagos, Nigeria, pp. 139-151.

Codd-Downey, R., Forooshani, P. M., Speers, A., Wang H. and Jenkin, M., (2014) *From ROS to unity: Leveraging robot and virtual environment middleware for immersive teleoperation.* IEEE International Conference on Information and Automation (ICIA), Hailar, China, pp. 932-936.

Fernando, W. A. M., Jayawardena, C. and Rajapaksha, U. U. S. (2022) *Developing A User-Friendly Interface from Robotic Applications Development.* Smart Computing and Systems Engineering, University of Kelaniya, Sri Lanka, pp. 196-204.

Gradil, A. and Ferreira, J. F., (2016) *A visualization and simulation framework for local and remote HRI experimentation.* 23rd Portuguese Meeting on Computer Graphics and Interaction (EPCGI), Covilha, Portugal, pp. 1-8.

Hiraishi, H., Ohwada, H. and Mizoguchi, F. (1998, October) *Web-based Communication and Control for Multiagent Robots.* IEEE/RSJ International Conference on Intelligent Robots and Systems, Canada, pp. 120-125.

Hong, H., Wen, Z., Bi, S., Zhang, Y. and Yang, W., (2019) *RoverOS: Linking ROS with WebSocket for mobile robot.* IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CY*BER),* Suzhou, China, pp. 626-630.

Islam, M. T., Hameem, I. R., Saha, S., Chowdhury, M. J. R. and Deowan, M. E., (2023) *A Simulation of a Robot Operating System Based Autonomous Wheelchair with Web Based HMI Using ROSbridge.* 3rd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), Dhaka, Bangladesh, pp. 175-180.

Kato, K., Nakamura, Y., Matsuhira, N. and Narita, M. (2021) *Remote control experiment of multiple robots using RSNP unit.* 21st International Conference on Control, Automation and Systems (ICCAS), Jeju, Korea, pp. 1866-1871.

Kohana, M. and Okamoto, S., (2014) *Sphero Control System Using a Web Browser.* 17th International Conference on Network-Based Information Systems, Salerno, Italy, pp. 606-610.

Kuo, G. -H., Cheng, C. -Y. and Wu, C. -J., (2014) *Design and implementation of a remote monitoring cleaning robot.* CACS International Automatic Control Conference (CACS 2014), Kaohsiung, Taiwan, pp. 281-286.

Liu, P. X., Meng, M. Q., Gu, J., Yang S. X. and Hu, C. (2003) *Control and Data Transmission for Internet Robots.* IEEE International Conference on Robotics and Automation, Taipei, Taiwan, pp. 1659-1664 Vol.2.

Ma, X., Zhang, F., Meng, F. and Li, Y., (2006) *Collaborative Control for Internet-based Multi-robot Exploration*. IEEE International Conference on Information Acquisition, Veihai, China, pp. 516-521.

Ma, X., Fang, F., Qian, K. and Liang, C., (2018) *Networked robot systems for indoor service enhanced via ROS middleware*. 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), Wuhan, China, pp. 852-857.

Makaya, F. D., Chatelain, D. and Snyman, L. W., (2004) *Design and performance assessment of a prototype wireless controlled robot.* 12th International Symposium on Electron Devices for Microwave and Optoelectronic Applications, EDMO 2004., Kruger National Park, South Africa, pp. 115-118.

Mohanarajah, G., Hunziker, D., D'Andrea, R. and Waibel, M., (2015) *Rapyuta: A Cloud Robotics Platform.* IEEE Transactions on Automation Science and Engineering, Vol. 12, pp. 481-493.

Richtr, L. and Farana, R., (2011) *Remote control the robot using web service.* 12th International Carpathian Control Conference (ICCC), Velke Karlovice, Czech Republic, pp. 326-330.

Robot Operating System. (2021) *ROS - Robot Operating System* [Online]. Available at: https://www.ros.org/. (Accessed: 5 June 2023)

Sikand, K. S., Zartman, L., Rabiee, S. and Biswas, J., (2021) *Robofleet: Open Source Communication and Management for Fleets of Autonomous Robots.* IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, pp. 406-412.

Söyünmez, M. E., (2023) *Browser Based Lightweight Web Control System for Autonomous Robots With Ros*, ASYU 2023, Sivas, Turkey [Submitted]

Toris, R., Kammerl, J., Lu, D. V., Lee, J., Jenkins, O. C., Osentoski, S., Wills, M. and Chernova, S., (2015) *Robot Web Tools: Efficient messaging for cloud robotics*.

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, pp. 4530-4537.

Wang, D., Ma, X. and Dai, X. (2004) *Web-based robotic control system with flexible framework.* IEEE International Conference on Robotics and Automation, New Orleans, LA, USA, pp. 3351-3356 Vol.4.

Xiao, Z. and Xu Y. (2018) *Web-Based Robot Control Interface.* IOP Conference Series: Earth and Environmental Science Available at: https://iopscience.iop.org/article/10.1088/1755-1315/252/4/042112

Yan, B., Shi, D., Wei, J. and Pan, C. (2017) *HiBot: A generic ROS-based robot-remote-control framework*. 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), Wuhan, China, pp. 221-226.