Bachelor of Engineering with Honours in

Mechanical Design and Manufacturing Engineering

MME3191 Capstone Project AY2021/22

Final Report
Waiter Robot Implementation

Date of Submission: 29 July 2022

Serrano Kurt Yves Espenilla

2001307

Acknowledgement

I would like to express my utmost gratitude to my supervisors, Dr. Michael Lau and Dr. Edwin Foo, for their guidance, support and wisdom throughout the capstone project. They have also monitored my progress throughout the project and encourages me in my best interests. Thank you for allowing me to take on this project and to grow as an engineer.

Abstract

There was a boom in robot in all part of the industries during and after the recent Covid-19 pandemic. Especially the service industries to solve manpower issue, to increase labour productivity and service quality. However, some of the robotic restaurant centers around its unique selling point, the robotic experience. To add on that there seem to be a lack of implementation techniques that further increase productivity and lessen workload for human waiters. To improve the waiter robot capability, the OSI model were explored and established a tech stack that works together providing minimal modifications to the restaurant to enables them to personalize their robot without the need of the supplier help and a new way of deploying orders to robots.

Thereafter, a literature review was conducted to evaluate the various software implemented on the market and how to develop it using open-source software. Subsequently, the conceptual design listed the techniques and ideas on how to implement features that were researched beforehand. Afterwards, methodology was discussed as this project was divided into development phase and testing phase. It was portrayed in the results and discussion how the application created react to the environment and robot. Serveur was able to be deployed in the testing environment and able to deliver food through automatic mode. In conclusion, the objective of the project was achieved.

Table of Contents

List of Figures

Abbreviations and Symbols

AMCL	Adaptive Monte Carlo Localization
API	Application Programming Interface
CBOR	Concise Binary Object Representation
CRUD	Create, Read, Update and Delete
HIL	Hardware in the Loop
HTTP	Hypertext Transfer Protocol
ICR	Instantaneous Center of Rotation
IOT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
MBD	Model-Based Design
NU	Newcastle University
NYP	Nanyang Polytechnic
OOP	Object Oriented Programming
OS	Operating System
OSI	Open Systems Interconnection
ROS	Robot Operating System
RVIZ	ROS Visualization
SIL	Software in the Loop
SIT	Singapore Institute of Technology
SQL	Standard Query Language
TCP	Transmission Control Protocol
UI	User Interface
XML	Extensible Markup Language
SSL	Secure Sockets Layer

1. Introduction

Machines are a part of us as an individual. With various machines from phones to toaster, it has helped us established a luxurious environment that enables us to be productive. With the recent Covid-19 pandemic, there were a boom in popularity of robots in all parts of the industry. They were used for applications ranging from medical assistance to food deliveries to mitigate the effect of the pandemic [1]. Service industry have taken a huge impact during the pandemic. Some have closed their doors permanently [2], others persevered [3] and a few took the opportunity to use technology such as robots [4] and delivery apps [5] to help them recuperate losses.

Service industry have already incorporated robotics [6] even before the pandemic had happened like the Savioke's Relay Robot on hotels and inventory delivery [7], Pudu's Waiter Robots in Haidilao hotpot restaurant in Singapore [8] and many more. The pandemic on the technological perspective have only boosted its applicability as more restaurants begins their journey with waiter robots like the partnership between SoftBank Robotics Group (SBRG) and Keenon Robotics which they deployed their Keenbot to the Grab Kitchen at Hillview, Singapore [9].

Waiter robots has become a hot topic in the service industry as of late. As more establishments deployed their robots to solve manpower issue [6,10], to increase labour productivity and service quality [6,11] and even allow people with disability to work remotely with Dawn robot café in Japan [12]. This hype train might come to a halt as the number of robots continues to increase. A survey study has been done regarding the effects it has caused in the robotic restaurants in India. The results were good however, "the robotic restaurant somehow revolves around their unique selling proposition that is the robotic experience" as some of the restaurants failed to provide the basic services in their survey [13].

The Gartner Hype Cycle represents how technologies develops throughout its life cycle as seen in Figure 1 [14]. Though each technology experiences different types of slopes with some have smoother curve than others, the waiter robots may soon reach the peak of inflated expectations as robots in the restaurants becomes more of a norm than an exceptional experience [13]. Delivery drones a distant relative of waiter robots also was not a stranger to hype cycle. Amazon had introduced the concept of being able to deliver parcel right outside your door using a drone in 2013 [15]. It was the hype at its time but quickly died down as the several challenges like controlled airspace [16] and noise [17] remains to hinder the implementation for domestic purpose. However, it has redeemed itself to be applicable in delivering relief supplies such as medicines as demonstrated by Zipline's autonomous aircraft during the pandemic [1,18]. Though the waiter robot will not experience the same challenges, mitigating the steepness of trough of disillusionment slope will help to quickly translate superficial value to a tangible benefit [14] for both restaurants and the customers.

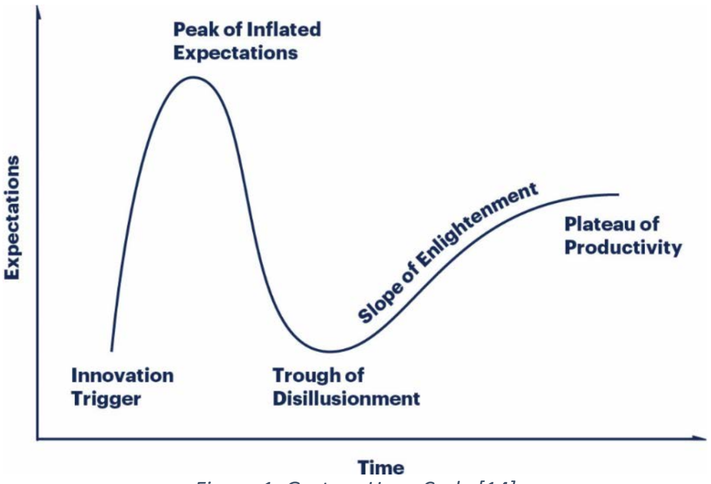


Figure 1: Gartner Hype Cycle [14]

1.1 Motivation

Recent researches and latest releases of Pudu robotics [19] were more centered on how the waiter robot locate itself and avoid obstacles whether its dynamic or static in the environment it has been placed on via RFID [11], stickers [19] or camera [19,20] and experience it provide to the customer [13,21]. There seem to be a lack of implementation techniques that further increase productivity and lessen workload for human waiters.

Implementation of waiter robots requires table to be fixed as it affects how the robot navigate. This in turn leads to reoccurring designs layouts of restaurant which can kill off creativity in restaurant designs. Furthermore, modifications in restaurants heavily depends on the robot company that implemented the waiter robot. Nowadays, the younger generation are generally much more tech-savvy compared to few generations ago [13,22]. With proper instructions, things such as simple modification can be delegated to the restaurant to save cost and resources.

Moreover, software used in the implementation of robots using ROS software currently centralized in using JavaScript or its frameworks. It is due to its flexibility to work with several software building tools and its ability to work with protocols that follows the OSI model. As tons of tools are available for free and maintained to help the developers build their robot and implement it [23]. This locks developer into learning one framework which can attract and deter new developers into learning robotics. Having an alternative will diversify and enriches the developer experience for people implementing the waiter robots.

1.2 Objectives

This project aims to make use of the OSI model to create a tech stack for building a robot application using websocket protocol to integrate ROS framework with Flutter UI framework. Using UI, introduce a new way of deploying orders to robots. UI will be done for Robot and Staffs. In addition, use a centralized network approach using Raspberry PI to enable multiple devices to communicate with each other with the same protocol using Python. Furthermore, scheduler script with Python and Mariadb SQL Database to provide simple status for the robot to prevent and allow it to be controlled. This will be developed and tested through ROS TurtleBot simulation before implementing to the Servour Waiter Robot. The goal is to delegate minimal modifications to the restaurant to enables them to personalize their robot without the need of the supplier help.

2. Literature Review

2.1 Current Implementation in Dining outlets

What can be seen on social medias about the implementation of Waiter Robots is very one sided. It is because little data about its architecture is shown by the supplier which is normal to protect its business. Thus, its necessary to build the implementation from scratch using frameworks that are abide the OSI model.

2.1.1 Scheduling Robot

Scheduling allows multiple robots and orders to be managed and delivered. Pudu Robotics did a decentralized approach where robots can communicate with each other within the same network. This is to segregate workload to robots to eliminate the need of a central server [19]. Keenbot on the other hand, specified that their robots are controlled via an intelligent schedule system. In addition, capable of delivering food to maximum 4 tables at once [24]. There is no definite way to know which implementation for scheduling is better. Since both worked for them, centralized network approach will be used in this project because of its innate scalability.

2.1.2 Deploying Robot

For deploying Pudu Robots, place the food on the tray and enter table location as seen in Figure 2. This information would require the staffs to know what table the customer is in and the food that they have ordered. Thereafter, press a button to acknowledge you have collected the food

[25]. Mozo from Marses Robotics and 1st Century Robot alternatively incorporates contactless acknowledgement of collection of food [26] as seen on Figure 3 and ask the customer to pat its head to continue working [27] respectively. Each company has its gimmick as to how to acknowledge the collection. However, for informing the location of table seen to be relying on the waiter or chef to input the parameters. Thus, with the use of database and server instruct the robot to deliver food without telling the robot the table number will be done in this project.

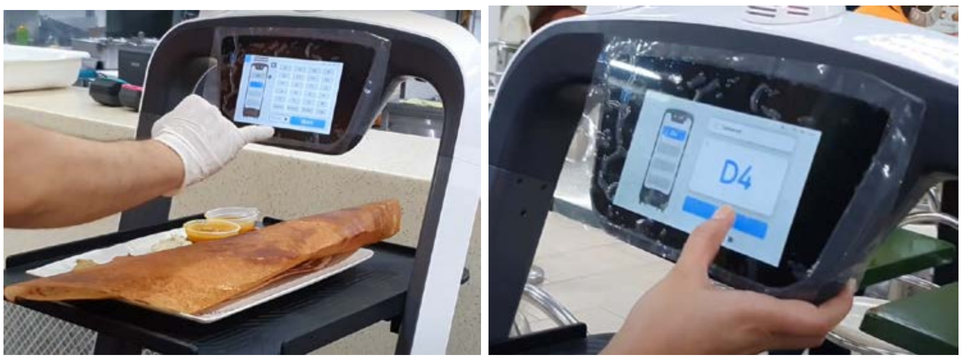


Figure 2: From left, deliver food to the specified table location and acknowledgement of collection [25]

Figure 3: Contactless acknowledgement of collection of food [26]

2.2 Current State of Serveur Waiter Robot

Serveur the second-generation waiter bot, as seen in Figure 4, created by students and staffs from SIT/NU and NYP. It has a 3-Swedish 90º wheeled circular base that allows it to travel through compact places with ease, leaving a small footprint and allowing access to the tray from all directions [28]. It uses Ubuntu OS to run ROS. It has a Velocity State Machine, a low-level robot control, to overcome the jerk motion that the robot does using the default control. However, it trades accuracy for stableness [29] which can be compensated by environmental landmarks such as RFID [11] and stickers [19]. Serveur also has an obstacle avoidance module that incorporates Nvidia Jetson Nano with ROS to interactively identify static and dynamic objects in its path though machine learning and avoid accordingly [20]. Furthermore, an algorithm to make Serveur to a Cocktail Robot [30]. This robot will be used to implement the system made through OSI model and centralized network.

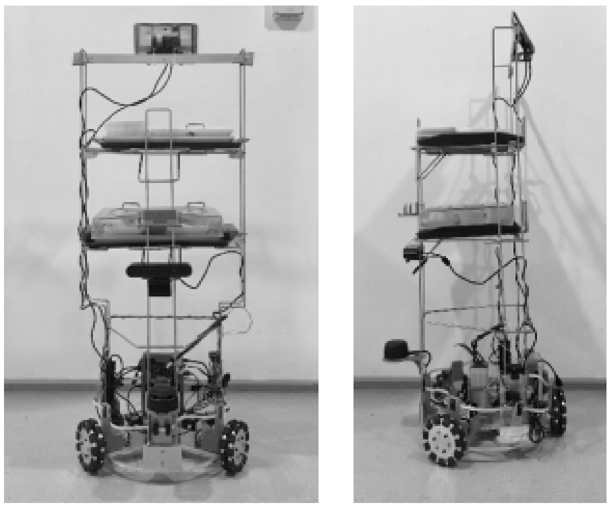


Figure 4: Serveur Robot

2.3 OSI Model

The OSI model is a theoretical framework for explaining networking system functions. This aids with the interoperability of various devices and apps. It uses a 7 layers design and can be further divided into 2 sub layers as seen in Figure 5 to represent how data is shown to us humans (top) and to a machine (bottom) [31]. Currently, Serveur's functionalities are primarily found in the hardware layer. Manipulation of data is necessary under the software layer that will control and instruct the robot where to go next in order to achieve service automation. To ease the implementation of OSI, several web standards or protocols has been made [32]. Which are used by programming languages to control data and send messages to clients. Websocket protocol is one of the examples of this.

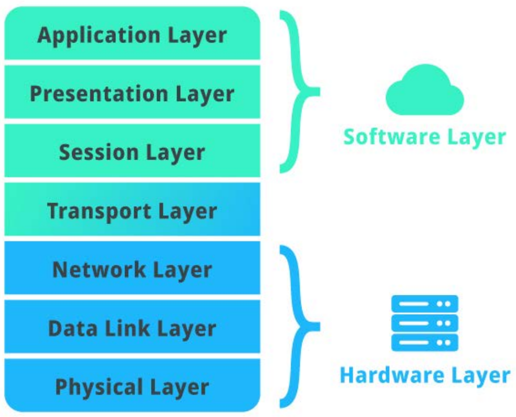


Figure 5: OSI Model [31]

2.3.1 Tech Stack

Tech Stack, also known as Solution Stack, is a list of software that a developer used to create a single software that meet customer needs. A tech stack can be divided into 3 segments Frontend, Backend, API. Frontend refers to how a user interact to the application. Backend refers to what will happen to a data once entered to the system. API refers to any additional software that integrates to the application to lessen complexity of a tech stack [33]. The difficulty of different tech stack differs from each other due to its programming languages and experience. The tech stack consisting of Flutter, ROS, Python and MariaDB will be tested and analyzed if suitable for robot implementation.

2.3.2 WebSocket

WebSocket is a computer networking technology that uses a single TCP connection to communicate. It is an Application Layer protocol that allows full-duplex communication between the server and the client. The handshake is also established using TCP under the Transport Layer. Websockets, unlike the HTTP, utilizes the "ws:" prefix in their address. It enables data to be changed without having to reload the page or interface. It is primarily utilized in real-time applications such as games that need the usage of the internet, stock websites, and other similar applications. It also predominantly uses JSON messages for communication [34]. This protocol is great in a restaurant setting which requires real-time updates.

2.4 Centralized Network

Centralized Network is an architecture that is built on a single server that does all of the hard lifting. Other devices can connect to the server and submit queries to the central server rather than completing them themselves. Applications, data storage/analysis, and utilities are all examples of this [35]. This concept is mostly deployed to data centers and cloud which offers freemium. A localized deployment will be used through a Raspberry PI in this project to bypass issues such as subscription.

2.4.1 Raspberry PI

Raspberry Pi is a microcontroller that functions as a minicomputer and permits many I/O connections to form an electromechanical device. It is widely used in IoT applications and uses Python as its foundation programming language [36]. It runs in Linux-based kernel. For this project PI-top 4 was used. It is a modular design which includes Raspberry PI 4, a battery, programmable screen and buttons and a heat sink with fan [37]. Its reusability and ease of use helps to host the localized server with the use of the websocket protocol.

2.4.2 Python

Python is a free, dynamically semantic, interpreted, object-oriented high-level scripting language. Because of its high-level built-in data structures, as well as dynamic typing and dynamic binding, it is suited for Rapid Application Development and as a scripting or glue language for integrating existing components. Python's syntax is brief and straightforward and it promotes legibility, which reduces software upkeep costs [38]. It is used in building websites, automating task and data analysis [39]. In this project, server and client scripts using websocket protocol will be created to send and collect messages to and from the UI and Serveur respectively. Websocket runs asynchronously to achieve multiple clients/server.

2.4.3 MariaDB SQL Database

MariaDB is an open-source relational database management system (DBMS) that may be used as a replacement for the popular MySQL database. MariaDB was formed as a software clone of MySQL in 2009 in response to Oracle Corp.'s acquisition of MySQL. It supports programming language such as Python, C++, Java and others for connection, sending and retrieving data [40].

A relational database is a form of database that stores and provides access to related data items. The relational model is the cornerstone of relational databases. It is a straightforward and apparent way of expressing data in tables. In a relational database, each row in a table is a record with a unique ID called the key. The data attributes are kept in the table's columns, and each record typically has a value for each attribute, making it easy to create relationships between data points. [41].

A database can be used as the state holder for Serveur to inform the UI its location and status without the need to connect to the robot. Since the data is localized, relationships between them can be established.

2.5 ROS

ROS is an open-source software that helps to build robot applications [42]. It comprises of tools and libraries for several applications such as hardware abstraction, 2-way communication between devices, simulation environments and etc. It runs on Unix-based operational systems such as Linux and Mac. It mainly uses Python and C++ to build topics and nodes. In addition, XML is used to launch several nodes together in one script.

Topics are unidirectional named buses that are transmits messages over different nodes to exchange information [42]. It uses subscribe-publish model; an asynchronous communication that are used in modern cloud architectures which facilitates event-driven applications [43]. Figure 6 shows how a topic provide and receive messages. Publisher provides messages to subscribers and the message does not need to come from one publisher as it can accommodate multiple publishers. Thereafter, the topic will combine the message received before sending it to each subscriber.

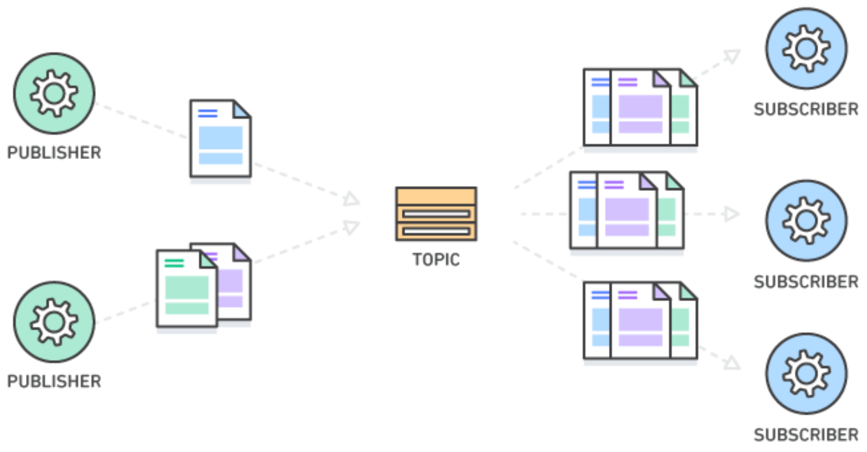


Figure 6: ROS Topic Structure [43]

Nodes are a process that works out the computation and algorithm of the robot. They transmit and receive messages through publishing and subscribing to Topics respectively [44]. Through that, a collective graph is formed and can be visualized through rqt_graph also called ROS Graph an example as seen in Figure 7 [45]. Working with Nodes makes it easier to debug and make the code less complex [44].

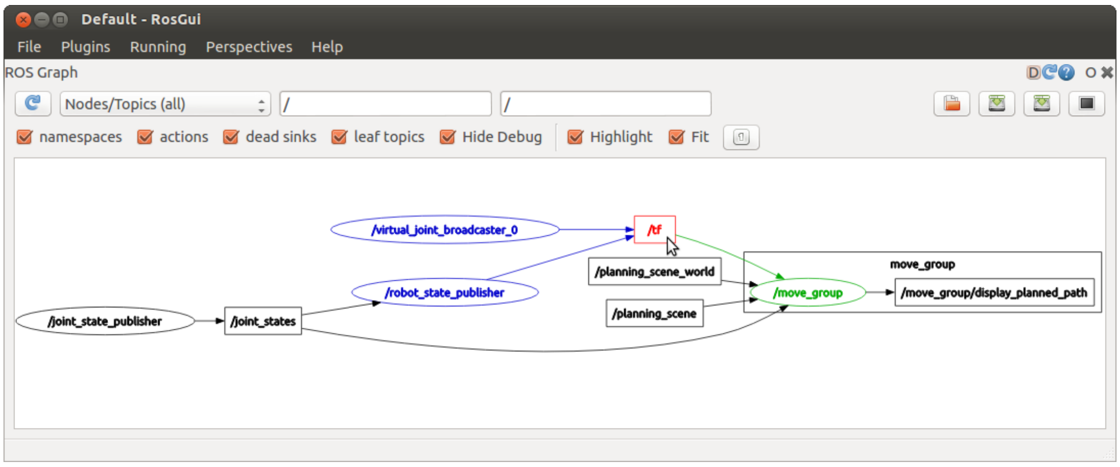

Default - RosGui
File Plugins Running Perspectives Help
ROS Graph
Nodes/Topics (all)
namespaces actions dead sinks leaf topics Hide Debug Highlight Fit
/virtual_joint_broadcaster_0
/tf
/planning_scene_world
move_group
/robot_state_publisher
/move_group
/move_group/display_planned_path
/joint_state_publisher
/joint_states
/planning_scene

Figure 7: ROS Node Graph [44]

ROS also allows simulation with Gazebo and RVIZ software. Gazebo is a powerful robot simulator that calculates physics, generates sensor data, and provides user-friendly interfaces. It is used by both business and academia [46]. RVIZ on the other hand, is a robot, sensor, and algorithm visualization software tool in 3D. It allows you to observe how the robot sees the world whether its real or simulated [47]. These software allows testing and debugging before implementation. Which makes it easier to find faults within the system.

Making use of Topics, Nodes and Simulations enables smooth implementation of OSI and various software to the Servour robot. In addition, rosbridge a package that makes ROS topics and services available as JSON messages through websockets [48]. Which exposes ROS topics and nodes for applications to manipulate.

2.6 Flutter

Flutter is a UI, Google Open-Source framework for creating beautiful and fast application. It allows the app to be built natively across numerous platforms, IOS, Android, Windows and Web from a single code repository. Not only that, it can also be incorporated with an existing application to increase productivity. With recent update of Flutter, Flutter 3, it is now capable of building apps on Linux environment as well [49]. It uses Dart as its programming language. Dart is a JavaScript-like language that prefers the use of OOP. It has a great community and documentation to help new and veteran developers to learn and develop applications. It also has various packages that help developers produce applications at a faster rate [50].

Flutter has a large array of packages to suit someone's needs. However, ROS-Flutter packages are in scarcity. It is because Flutter was made 2017 [51] which is relatively new compared to popular programming languages like Python and C++. There are 3 open-source Flutter-ROS packages available at the moment. These packages aside from ros_lib [52] does not implement ROS with websocket which make it a great choice. However, due to its incomplete library, the web_socket_channel, a package that enables websocket connections [53], will be used instead to connect to the server and Serveur. In this project, Flutter framework will be used to create a UI for the robot and kitchen staffs to activate Serveur to deliver food to the customers.

3. Conceptual Design

This section shows how the system were designed to add on to the existing Serveur robot. Refer to [54] for the full code of the Serveur, UI and Server.

3.1 Environments

In mechatronics-based designs, having several environments enables to minimize error and meet quality within the time frame. Having several environments helps as this project must be done within the 6-month timeframe. MBD is an approach that separates the system into segments to prove each of them can work and evaluate that the feasibility of an idea. Making use of MBD, the environment was separated into 2 parts: SIL and HIL [55].

Figure below shows how the devices was connected through SIL and HIL. This setup was used interchangeably to check and debug the system.

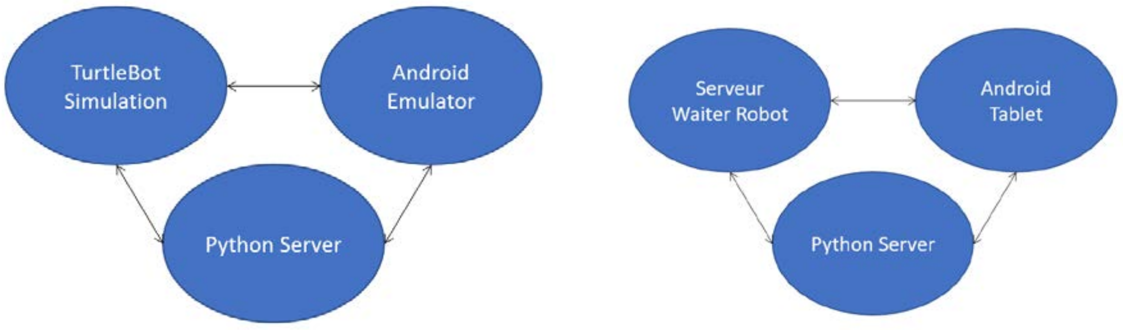


Figure 8: SIL (LEFT) and HIL (RIGHT)

3.2 System Architecture

An internet-less set-up was adapted to bypass IP restrictions imposed by NYP and SIT and to control the IP addresses of the devices. System architecture consists of 3 segments as seen in Figure 9. The first segment is the Software level, it shows what software is running in the background of the devices at the layer below. It also shows how each device communicates with each other through the network via websocket. The second segment is the Hardware level consisting of the Serveur, Raspberry PI, Tablets and Router. The third segment is the User Layer, consisting of Owner, Staffs and Customer. Do note that "Orders UI" was not done in this project.

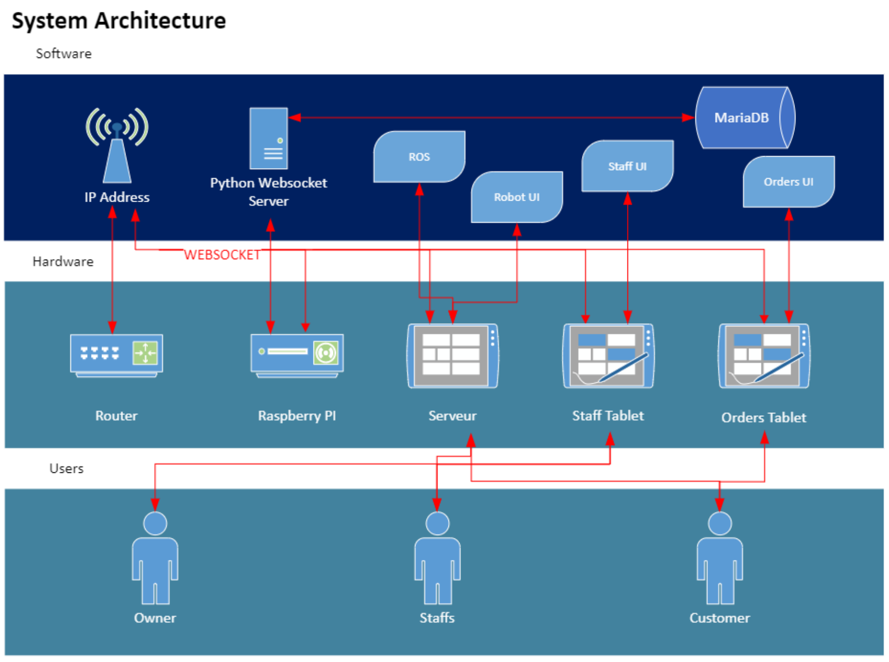


Figure 9: System Architecture

3.3 Centralized Network Design

Localized deployment of centralized network consists of server and clients as seen in Figure 10 below. Server controls and holds the information of the Waiter Robots. User interface requests server information about the Waiter Robot before they connect. Once connected, the interface can tell the robot several instructions directly. Waiter Robot also acts as a server but instead of control data, it acts as a node to provide data to Server and User Interface.

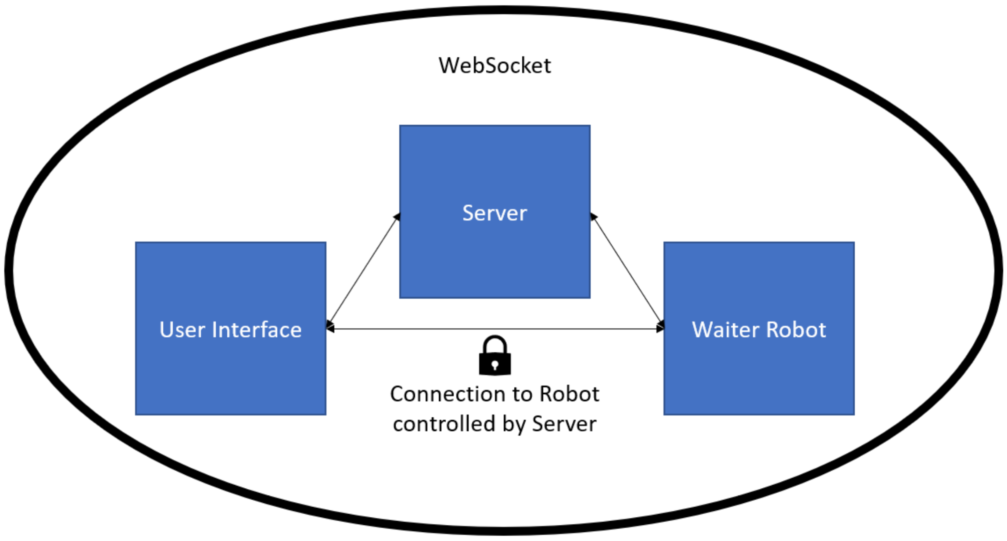


Figure 10: Centralized Network Design

3.4 Features to Add

Features to be added onto Serveur to allow the restaurant implementation can be divided into 2 parts Manual and Automatic Control. Manual Control consists of remote access to control the robot similarly to Japan's Dawn robot café [12], calibration of robot to localized robot to the map, instructing the robot goal and ability to create waypoints. Automatic control has scheduler to allow batch deliveries.

3.4.1 Manual Control

Under manual control, remote access requires the robot to provide a picture stream and a velocity control topic to allow the user to control the robot like controlling a character in a video game. For Calibration and Instructing Goal to robot, a design like RVIZ works to localize and instruct the robot to the map given by the robot. Waypoint creation is another feature that will be implemented. This is to save the current location and orientation of the robot with respect to the map to the database. Furthermore, each waypoint is added with a name in order for the scheduler to identify the table location on the map. This data will then be used as the goal for the robot to use in automatic control and "Go-To" function which is a clickable list to move the robot manually to the waypoint.

3.4.2 Automatic Control

Automatic control relies on the server to control its next instructions. A set of instructions following the flowchart in the Figure 11 can be used to allow the server if its clear to give orders to the robot. Server will check if it is connected to the robot. If yes, it will wait until the user activate the robot. Thereafter, it will check if there is an outstanding job it will go to the table location saved in the database. Otherwise, it will go to its standby or home position. This series of checks and instructions creates a scheduler for the robot to follow once activated. The loop check of outstanding job and home position allows the robot to perform batch deliveries.

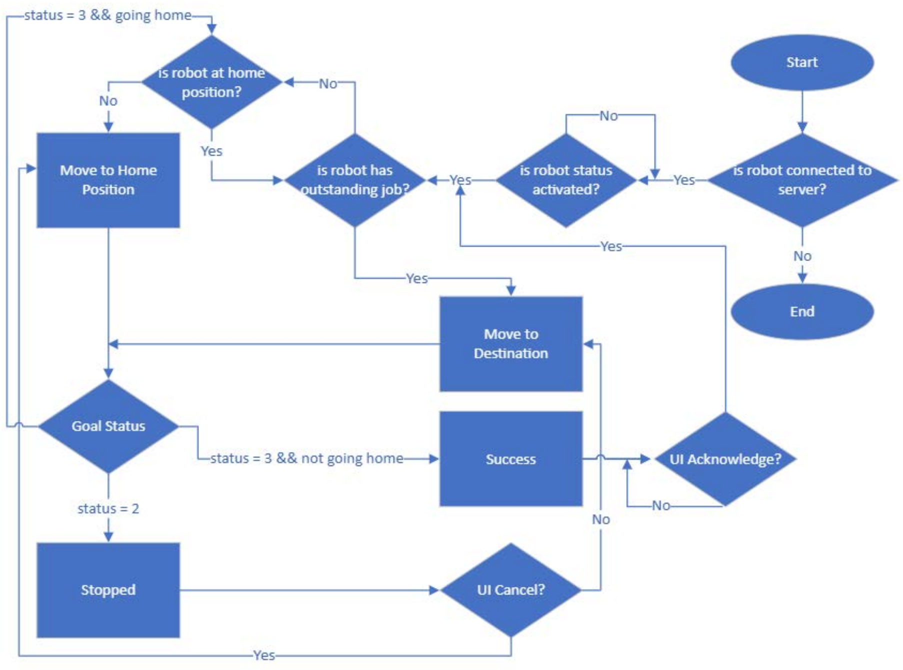


Figure 11: Scheduling Flowchart

Though it is called automatic control, it does need the human interaction to fulfill some instruction to complete a delivery to the table. Its first interaction is when activating the “Auto” mode. It is required to segregate manual control and automatic control. Second interaction is when the kitchen staff or human waiters instruct the robot to deliver food. Lastly, is the interaction between the customer and the robot to acknowledge the delivery of the food to the table. These interactions process as a state of the robot. The state is then become order status where it can be combined with the goal status from the robot making the robot status. Robot status informs and locks the UI and Server to a specific instruction available to the status.

The current way of instructing robot to go is to enter the table location which was added in Manual Control. In addition, a new way was added instead of entering the table location enter the food to be delivered.

3.5 Messages

Using WebSockets means using JSON format. JSON is a lightweight format for interchanging data over the internet [56]. It is formatted in key value pairs. Different programming languages uses different terms to identify key value pairs after JSON has been deserialized. In flutter/dart it is called a ‘map’ and in python it is called ‘dict’.

3.5.1 Flutter App with ROS and Server

Flutter favors OOP. So, instantiating data through classes makes it easier to display it. Quicktype, a tool that helps to change JSON to objects class of several type of programming languages [57], was used to eliminate boiler plate codes to save time. Figure 12 is how receiving data from the robot and server was structured to change JSON to object stream.

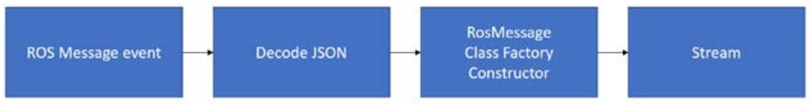


Figure 12: From ROS or Server to Flutter

For a duplex communication, a way to send message back to the robot is required to issue topic subscription, teleoperation, initial pose and goal assignment. It is structured as seen in the Figure 13.

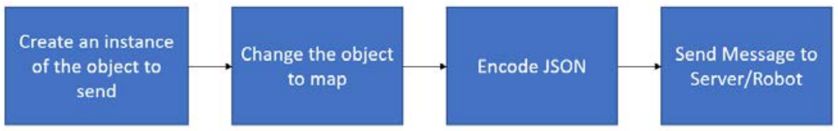

Create an instance of the object to send
Change the object to map
Encode JSON
Send Message to Server/Robot

Figure 13: From Flutter to ROS or Server

3.5.2 Server with ROS and Flutter App

Compared to flutter, python can work with objects or dictionary which for making the server is entirely to developer preference. For this project, dictionary was used instead of objects to eliminate boiler plate codes and make sending and receiving data simple. Figure 14 shows the how the JSON data is manipulated before storing to the database or sending to the robot/app.

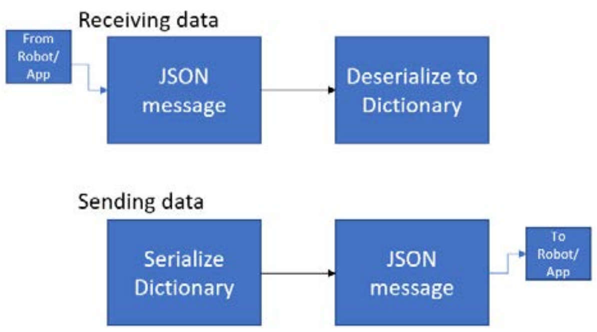


Figure 14: Python when Receiving (TOP) and Sending (BOTTOM)

3.6 App State Management

In any reactive application, state management is a crucial component to be aware of as this dictate whether the user will be able to go to a certain screen with the necessary data given by the user ie. authentication. Furthermore, user experience will be mostly derived from this management.

Flutter has an ample number of ways to manage app state. But for this project, 2 simple ways to manage the state was enough for accomplishing the task of showing data to the user. Firstly, ‘setState’ is a function of a stateful widget in flutter. If called, it re-renders screen. It is mostly used if there is an animation or new incoming data to show to the user. However, it is bounded to that certain widget tree. So, if the tree has been disposed or routed to a new screen the data will be lost. In other programming languages, global variables are used instead to bypass logics while having the data required. However, in flutter it is advised not to create global variables that are mutable as other widget may depend on that variable thus making the app prone to errors. So, to combat the mutable variables that required in multiple screens/widget tree Provider was created by the flutter community as one of many ways to manage state. Provider was the second way used in this project to manage the state of the application. Provider resides above the ‘MaterialApp’, the widget that manages page routes, design and navigation, to provide data from one tree to another tree [49].

3.7 Database Relationship

Using relational database, unlocks relational features that will take several boiler plates codes if it was done manually at the server side. Creating database relationships can be daunting for beginners, but as long as there are no duplicate columns across multiple tables the goal of having database relationships has been reached. The purpose of having a database relationship is to simplify the updating of the database as it does not know that a data is related to another unless specified. Refer to Appendix C for the database tables and relationships.

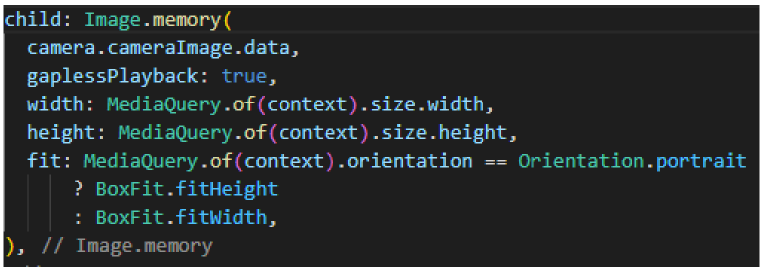
3.8 Consuming ROS Message in Flutter

After parsing the messages into objects and passed to stream, what is left is show its instance on the screen. This is to show information to the user about the robot.

3.8.1 Video

ROS videos are just a stream of images. Its message is structured in Unsigned Integer List type and one iteration represents one image. To show the video to Flutter UI, the ‘Image’ library was used because it can decode and render images from unsigned list of integers to normal image as long as the file type is one of the standards that we used today ie. jpeg, png, etc. The Serveur already has a topic that provides stream of images in jpeg. However, the TurtleBot simulation code did not come with a camera topic. Thus, the laptop's built-in camera was used to simulate the robot's camera, but the image type is not one of the standard file types. To change to the standard types the OpenCV package was used to transform the image to jpeg.

After establishing the image type, the rosbridge package encodes the image message to base64 encoding before sending to make sure the message was pass through correctly and with lesser size. So, in Flutter, the image message was decoded first before passing the unsigned integer list to 'Image.memory' widget as seen in Figure 15.

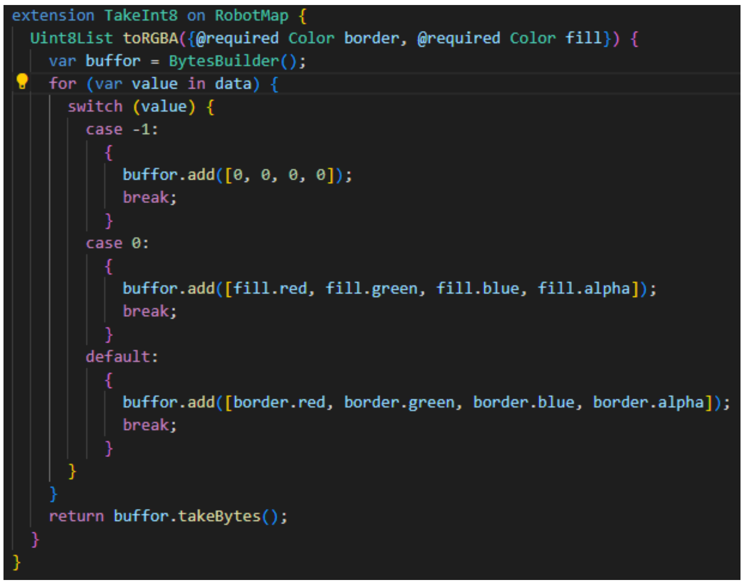


```
child: Image.memory(
  camera.cameraImage.data,
  gaplessPlayback: true,
  width: MediaQuery.of(context).size.width,
  height: MediaQuery.of(context).size.height,
  fit: MediaQuery.of(context).orientation == Orientation.portrait
      ? BoxFit.fitHeight
      : BoxFit.fitWidth,
), // Image.memory
```

Figure 15: Decoding Video Images

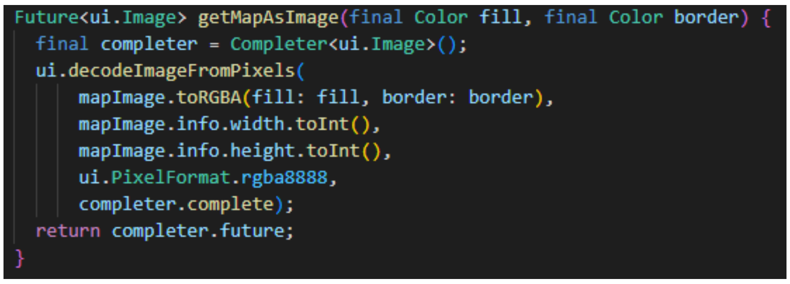
3.8.2 Map

Rendering robot map is not as straightforward as rendering video. Data published by '/map' topic is in Signed Integer List type instead of Unsigned in video. To show the map to Flutter UI, a new Unsigned List is created every time the topic sends a message. The list is created at a function as seen in Figure 16 which decodes the topic message values of -1 and range of 0 to 100. Thereafter, the decoded list contains representations of the map in pixel format which will be passed to a 'Future' function to parse the list to an image as seen in Figure 17.



```
extension TakeInt8 on RobotMap {
  Uint8List toRGBA({@required Color border, @required Color fill}) {
    var buffor = BytesBuilder();
    for (var value in data) {
      switch (value) {
        case -1:
          {
            buffor.add([0, 0, 0, 0]);
            break;
          }
        case 0:
          {
            buffor.add([fill.red, fill.green, fill.blue, fill.alpha]);
            break;
          }
        default:
          {
            buffor.add([border.red, border.green, border.blue, border.alpha]);
            break;
          }
      }
    }
    return buffor.takeBytes();
  }
}
```

Figure 16: Signed List to Unsigned List



```
Future<ui.Image> getMapAsImage(final Color fill, final Color border) {
  final completer = Completer<ui.Image>();
  ui.decodeImageFromPixels(
      mapImage.toRGBA(fill: fill, border: border),
      mapImage.info.width.toInt(),
      mapImage.info.height.toInt(),
      ui.PixelFormat.rgba8888,
      completer.complete);
  return completer.future;
}
```

Figure 17: Decode the List to Image

Subsequently, canvas was used to show the map image instead of ‘Image’ library because map requires more data to show useful information to the user. It is a widget in flutter that can display custom animation / drawings to user that are not possible using other widgets.

3.8.3 AMCL Pose

AMCL pose is the topic that shows the robot’s estimated location and orientation on the map [45]. Using the canvas, the position of the robot can be located using the ‘position’ data from the topic divided by resolution of the map to get the pixel distance of the robot from origin. Orientation on the other hand is shown by using the ‘orientation’ data from the topic to create Quaternion equivalent in radians using the ‘Quaternion’ library. Thereafter, paint the canvas at that pose

3.8.4 Scan

Scan topic represents the scanned objects scanned by a Lidar or any scanner that provide scan data. To show it on the canvas, iterate through the list of ranges and find each item’s cartesian coordinate using the formula below where θ is the initial angle, θ_i is the increment angle and i is the current iteration. Thereafter, paint the canvas at each cartesian coordinates.

$$x = \frac{distance}{resolution} * \cos(\theta + \theta_i * i)$$

$$y = \frac{distance}{resolution} * \sin(\theta + \theta_i * i)$$

3.8.5 Particle Cloud

Particle cloud is a set of pose estimates maintained my AMCL [45]. The further the particle is the more the AMCL unsure of the location of the robot on the map. To show it to canvas, the implementation of AMCL Pose to canvas was used with a loop to iterate through the set of particle cloud. Thereafter, arrow shaped icon was used to paint for indication of the direction of the particle cloud.

3.8.6 Waypoints

Waypoints are just saved AMCL pose in the database. To display it, only its location was used to pinpoint its position on the map. It is also divided by resolution similarly to AMCL pose to know it pixel distance. Thereafter, a diamond shaped icon was painted at the location to differentiate waypoints to robots. In addition, the waypoint name was shown for the users to distinguish different waypoints.

3.9 Consuming ROS or Flutter Message in Server

In the message received, there must be a key value pair that instruct the UI, Robot or Server as to what to do with the message. Rosbridge already uses a protocol to communicate to clients while the server does not. Thus, to communicate, Table 1 shows how the server accepts messages.

Table 1: Server Message Protocol

Instruction	**Value of key ‘clientmsg’**
Subscription	‘data’
Create job / waypoint	‘create’
Control Robot	‘control’
Control Finished	‘ready’
Food Collected	‘next’
If stopped, to Continue	‘continue’
If stopped, to Cancel	‘cancel’
List of Food Delivered	‘order’

The server does not need to render anything visually. Therefore, to perform the instruction it just needs to carry out specific CRUD functions to the database via queries.

3.10 Robot Navigation in App

For mobile robots, navigation is the core foundation to carry out tasks. For a restaurant setting, a semi-automated mobile robot that waits for a goal to be posted is required. Therefore, the app is required to send message to the robot to instruct it of the next work to do.

In a rigid body in 3-dimentional space, the object has 6 degrees-of-freedom, position x, y and z and orientation Ψ, Φ and θ. But for land robots like Serveur and TurtleBot, they are constrained in 2-dimentional space which means the degrees-of-freedom reduced into 3. They are x, y and θ.

3.10.1 Teleoperation

Serveur runs in omni-direction, meaning its ICR is in the center of its body. Making the robot able move in the x, y linear and z-angular direction. Therefore, it requires a controller that sends commands in all the direction available. TurtleBot on the other hand, is a differential-typed robot which means its maneuverability are only in the x-linear and z-angular direction. Therefore, a single joystick that controls the maneuverability is enough to control the robot.

3.10.2 Calibration/Goal Posting

RViz implements allocating initial pose/goal by using the mouse cursor location and direction it was dragged into then converted to the location and direction relative to the map frame before posting the message to the ROS topic. Its execution of initial pose and goal are intuitive and practical. Taking reference from it and porting it to the flutter app will make the app easy to calibrate the robot to the map according where is the robot in real life and assign goal.

To get the cursor location with respect to the map, given that the map origin is at the center, the screen origin must be same as the map origin. The RenderBox class can be used for locating the origin to center. Thereafter, minus cursor location provided by the touch callback from origin to find the location at the screen frame. Afterwards, change it to map frame by multiplying to the ratio of map size and screen size. Then divided by zoom scale given by the TransformationController class to know its equivalent at true scale. Lastly multiplied by map resolution for location equivalent in meters.

After locating the x and y cartesian coordinates, what is left is acquiring the orientation θ. It can be achieved using the cursor location at screen frame minus by another cursor location provided by touch callback. Thereafter find the arc tangent value of 2 points. Lastly, use the 'Quaternion' library to construct the quaternion equivalent of the arc tangent angle for the θ orientation.

3.10.3 Waypoints

There are 2 ways to make use of waypoint data they are manual and automatic. In manual, "Go-To" function, a clickable list, was used to show the created waypoint's names. To move the robot, click the specific list item then the app will send message to the robot to move the saved position. In automatic, issue task was used to give the robot tasks. It consists of a slidable pending orders list showing the current orders that the restaurant have and a virtual cart. To add order to the cart, slide an item from the pending order list and it will be added to the virtual cart. The virtual cart visualizes the order that the robot is going to deliver. Then a confirm button to send the virtual cart orders to the server to create list of tasks in the database for the robot to follow. Thereafter, the server will follow the automatic scheduling flowchart.

4. Methodology

4.1 Development Setup

During development, the SIL setup was used to conceptualize the ideas and was tested within the said setup to eliminate bugs before incorporating it to the hardware. It consists of a development computer, Aftershock MX Pro laptop, running Visual Studio Code with Flutter Development Kit for creating UI, Android Studio for Android emulator and MySQL Workbench for remote connection to database. In addition, another computer, Lenovo Z40-70, running ROS was used to run the TurtleBot simulation and the in-built camera was used to simulate the camera from the Serveur. As for the server, the PI-Top hosts the server and the database to provide WebSocket connection and data. Furthermore, the map used was the default map provided by the TurtleBot package to do mini test of features added.

4.2 Test Setup

The HIL setup was used to test the whole system. PI-Top as the server and database. Chuwi Tablet as the Staff UI. Serveur as the Waiter Robot equipped with ROS and Robot UI. An additional Intel® NUC portable PC was used to run RVIZ to offload payload from Serveur. In addition, software called VNC server was downloaded to all previously mentioned devices except Chuwi Tablet for remote desktop control. Control was done using the VNC viewer that was downloaded at Aftershock MX Pro laptop. A Singtel Aztech wireless router was used to act as an access point for transferring messages via WebSocket protocol and remote desktop control. In addition, to emulate the centralized network design without 3rd party API.

SIT-NYP LVL 6 was used as the testing location for the setup. A fire emergency floorplan was procured to be used as the map for the robot. However, the map was not scaled properly. Adobe Illustrator was used to clean the image before passing it to a MakeROSMap.py script. The script scales the map according to the distance between 2 points in horizontal and vertical entered by the user. The floor was measured before the readings was passed on to the script. Refer to Appendix E for the testing location and ROS map equivalent.

4.3 Assumptions and Limitations

The assumption made were that the Serveur navigation stack can navigate as close to the goal as possible. In addition, can deliver food and drinks without spilling it. The limitation of this setup is the latency of connections to router as it can affect the speed of data being transferred which can cause video delay during testing. Orders will come from updating the database manually instead of having a UI to choose a food menu.

4.4 Test Methodology

The test objective is to test each feature added to Serveur robot and test if it can perform the intended work as designed. Firstly, the Teleoperation with the video was tested to see what the robot sees when moving. Secondly, the Calibration and Goal Posting was tested and validated using RVIZ. This includes validation of ROS data consumed by the app and server. Not only that, but a visual check was also done if Serveur reaches the location specified on the map using the tablet. Thirdly, waypoint creation was tested and validated by whether it can save the current AMCL pose to the database and display it to the screen and move to the location through the use of Go-To and scheduler. Lastly, the scheduler was tested in 4 segments: single order, batch order, ‘stop and cancelled’ order and ‘stop and continue’ order to test the server handling of different type of order situation.

5. Results

The test was successful. Features added worked as intended with some bugs along the way. Most of them were fixed leading to successfully reaching the goal of implementation as a proof of concept. The Staff UI and Robot UI application built using Flutter adequately fulfills necessary features needed to control a Waiter Robot. In addition, the Server provides and stores data as intended.

Looking at the backend side of things, the centralized network design with the use of WebSocket has enabled the 2-way communication between Server, UI and Serveur. Following how messages structured for Flutter App with ROS and Server and Server with ROS and Flutter App for receiving and sending messages between devices have allowed data to be consumed properly. Which leads to having a correctly built database relationship. Furthermore, the app state management provider and ‘setState’ have proven to be enough to handle incoming and outgoing data from the UI. Lastly, the ways to consume ROS message was correctly implemented and shows the data at its intended location or use except for scan where it shows correctly on some occasions.

On the front-end side of things, the teleoperation of the robot is responsive and able to move the robot for all the direction available for an omni-directional robot. Having right controller controlling the x and y linear direction of the robot and left controller controlling the z-angular direction as seen in the figure below. In addition, whatever that the robot camera can see is projected to the tablet with some delay due to latency. Furthermore, a mini map at the top left corner visualizes the robot location on the map.

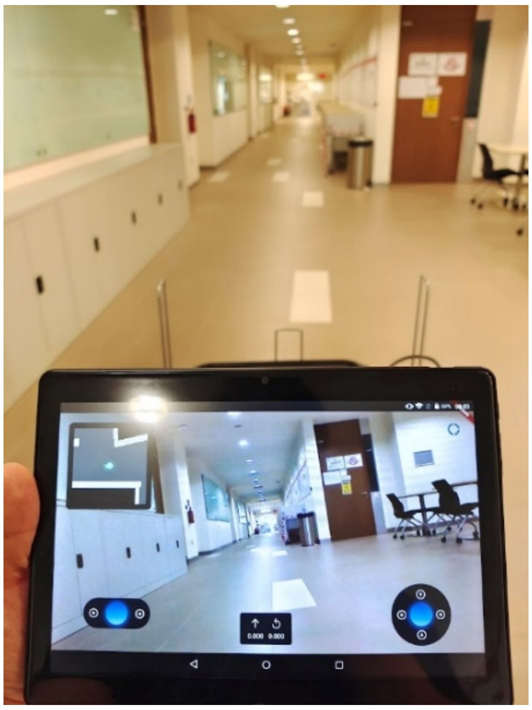
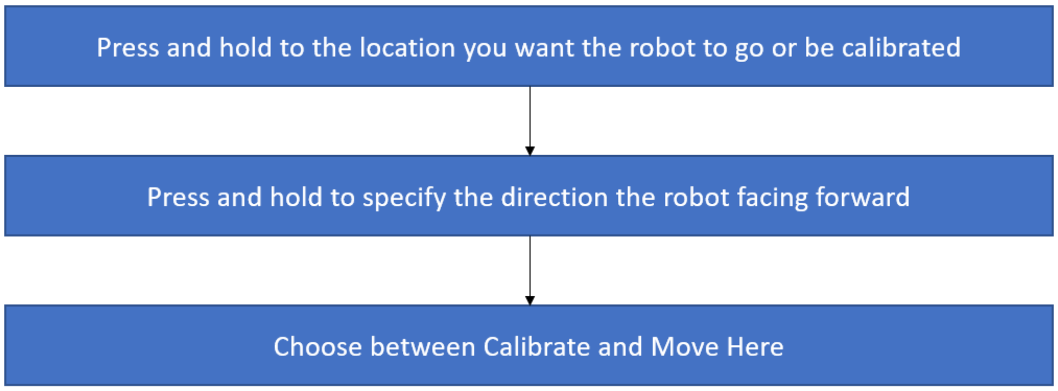


Figure 18: Teleoperation

Moving to Calibration, by following the calibration/goal posting steps as seen in Figure 19 and choosing "Calibrate" the robot calibrate itself as seen in the Figure 20. Where initially (left) when the robot starts it does not know its starting point unless specified. After the calibration steps the robot can localize itself on the map (right) and enables the goal posting feature.


Press and hold to the location you want the robot to go or be calibrated
Press and hold to specify the direction the robot facing forward
Choose between Calibrate and Move Here
Figure 19: Calibration/Goal Posting Steps

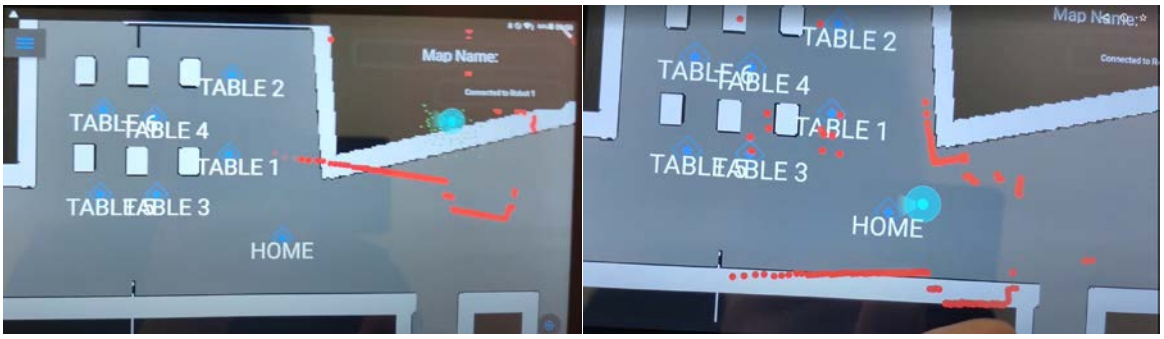


Figure 20: Calibration Before (Left) and After (Right) in the UI

Likewise for goal posting following the calibration/goal posting steps as seen in Figure 19 and choosing “Move Here” the robot can move to any location on the map. The UI works perfectly if the robot is running on the straight path without rotating. It starts to deviate the visual representation of the scan topic once it starts to rotate the robot. It appears to be a visual bug from the Flutter app as RVIZ on the left displays the correct scan of the robot.

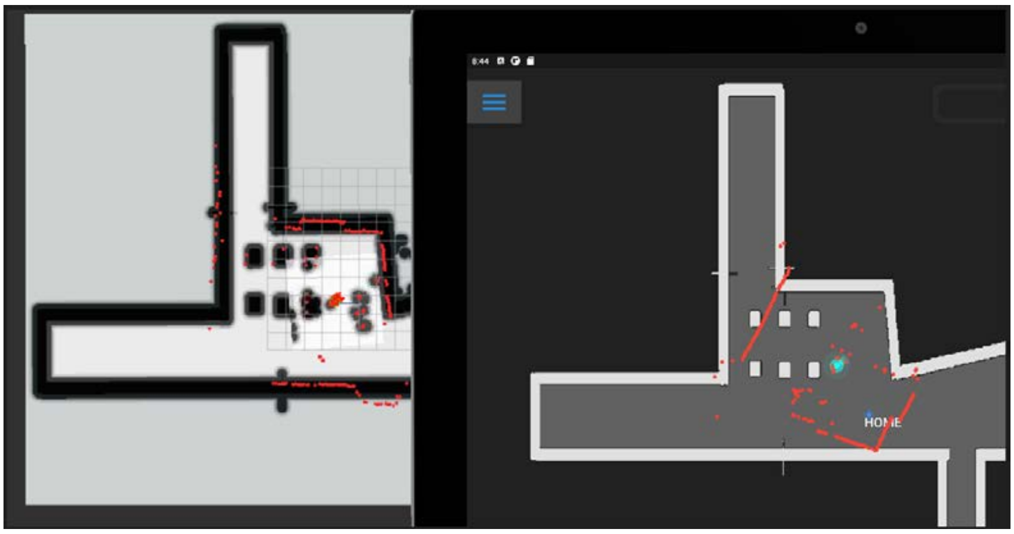


Figure 21: Goal Posting scan difference between RVIZ and Staff UI

Shifting towards Waypoint Creation, the feature itself saves the correct robot orientation and position despite the scan display irregularities. It was proven to work by using the scheduling and manually though the “Go-To” table location feature as seen in the figure below as it can move to the location specified by the waypoint accurately to some extent. The accuracy of the goal depended on how the robot was calibrated and the robot’s odometry error.

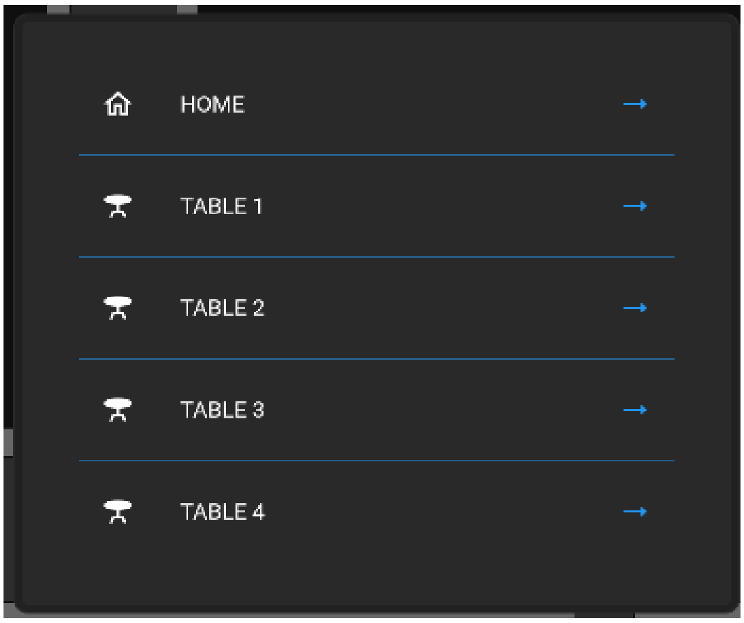


Figure 22: Go-To Feature

Now that the manual control has been covered, next is the Automatic Control. There are 3 interactions for a user to interact with the robot. Auto button, Issue task and Acknowledge button. Firstly, auto button successfully segregates manual control with auto control. This separation only allowed the server to be the only one in control of the robot's movements. Secondly, issue task a function that issues delivery orders to customers. It uses the food to be delivered as seed in Figure A7 instead of the widely used function of entering the table number. Together with the database it enables and separates orders that are in different tables so it could do multiple order deliveries and batch deliveries to the tables. Lastly, acknowledgment button successfully accepts that the robot has delivered the food which enables the robot to continue to its next task. Refer to Appendix A and B for UI screens.

The Automatic Control algorithm works like a charm with the loop check of outstanding job and home position successfully running batch table deliveries if there are more than one table to deliver into. In addition, by grouping the deliveries by table the customer virtual cart can display the food it delivered once it reaches the table enabling multiple order deliveries. A set of videos saved at [54] depicting the test results for batch and multiple deliveries and other manual controls. Within the algorithm, there is a cancel order to cancel the current order. But before cancelling order the robot has to stop its delivery. Thereafter, a prompt reconfirming the cancellation. By confirming, the function works and cancels the order. Then the order goes back to issue task as pending order and robot moves to its home position. On the other hand, asking the robot to continue, the robot will proceed to its delivery location completing the delivery process.

6. Discussion

The tech stack Flutter, ROS, Python and MariaDB works together with the help of WebSocket to create an application to control and use the robot for delivering food within the restaurant. Like in Dawn Café in Japan [12], having teleoperation opens the horizons for the owner and staffs. Waiter can remotely work from home if there is another virus outbreak. They can move the robot if it gotten stuck when they are not in the vicinity of the robot. With calibration and goal posting on the other hand, it allows the staffs to troubleshoot or control the robot visually aside from teleoperation. This creates an alternative if the person is not comfortable using teleoperation control. In addition, waypoints enable staffs to model their table arrangement without the supplier's help allowing creativity in the restaurant designs. Batch deliveries are no stranger to the current implementation to restaurants. As Keenbot shows that it can do said deliveries up to 4 tables. Which to say the application does par to the current implementation with added feature such as entering food instead of table to deliver food and customer virtual cart. The entering food feature only removes a notion a waiter has which is the link of which food is to which table. By delegating that concept to the server, the waiter would not need to think of which food goes to which table which helps in lowering the amount of work that the waiter has. The customer virtual cart adds visual aid for the customer as it shows what did the customer have ordered which has been delivered by the robot. Now, with the newly added features, the restaurant can customize their restaurant to fit their theme or occasion when they need be without the help of the supplier.

Another great point about the tech stack it is an open-source project. Being open source means free of charge, able to build applications quickly, securely and functionable with the backing of the community. It has allowed this project to be completed with no cost spent on software licenses while having the full access to its features. In addition, it enriches the learning for the developer through documentation online and engagement with the community. Furthermore, it provides higher quality, better reliability and greater flexibility due to its distributed peer review and transparency of process [58].

Flutter is a great alternative to JavaScript with ROS. With its recent update of Flutter 3.0, the supports of Linux desktop have made creating for the robot UI much easier. Compared to JavaScript where a developer needs to learn HTML and CSS to create an application for ROS, flutter only have one language to learn which is Dart. In addition, flutter build everything like a widget. That's why with time the developer can create multiple widgets that can be used in multiple projects that he/she might have which makes the development time much faster.

ROS, as a required framework for the robot, is an extensive software structure dedicated for robot commercialization and research. Packages that were used in Serveur on scratches the surface on the number of packages that were done by the developer and the community. It supports what would take months of work to do if it were to be done alone. With its combination of rosbridge, it extends it capabilities to other software such as Flutter to further support things such as implementation.

Python with MariaDB have created a server equipped with data storage mimicking a cloud server without the ability to connect outside the Wi-Fi network. Is enables scaling of the application to have what Pudu cloud have. Pudu cloud platform is a robust cloud intelligent service support platform for Pudu robots, providing smart catering, business management, automated operation and maintenance, scenario data gathering platform, and cloud intelligent service platform. [19]. Server with the automatic control, created universal status for the robot for the robot and human to understand. In addition, it is scalable in the amount of robot a restaurant has in possession. Basically, having a centralized server unlocks unlimited potential for waiter robot.

Focusing on the applications made using Flutter. It was built with objective of making applications to deploy the Serveur to the restaurant. So, it might not be appealing visually and easy to use but it does the work done that it was set up to do. In addition, the app itself is not even close to deployment stage. There are several things that are needed to accomplish before using this implementation idea.

6.1 Challenges

Creating this project was not without challenges. Firstly, I had no prior programming knowledge in Flutter (Dart), ROS and MariaDB. To fully comprehend and apply the frameworks, I had to go through a high learning curve. Furthermore, within the project's timetable, I had to incorporate and debug any issues that occurred throughout the project. Modules covered in my course of study, such as Real-Time Embedded Systems and Mechatronics Systems, aided in understanding and were employed throughout the project phase. As the Serveur utilizes ROS framework, I had to learn ROS from scratch and incorporate robotic theories to software that can be used by the server and the app created using Flutter. In addition, creating a miniature localized cloud system is something that I have not done before as well. The intricacies of asynchronous programming that governs a cloud platform or a real time gaming differs greatly from synchronous which was mostly what I have learnt throughout my course of study. It also created a learning curve to overcome to enable the server to function.

As a new aspiring developer learning Flutter, was taken to implement app state management and widget might not be the best way possible. There are still a lot of practices that recommended by various media to be implemented but were not applied. It is largely due to the scope of the project. This project differs from a traditional software development which requires team to finish and several milestones to finish. As mentioned before, this project is nowhere near deployment as it lacks security functions and robustness to handle the real environment.

As this project stems from several previous students' effort and knowledge, I expected the robot to be robust software-wise in implementation of ROS framework. However, it was the opposite. Firstly, the ROS packages were not arranged properly. Leaving a very bloated workspace and unfriendly to anyone trying to understand the project in the future. Secondly, lidar sensor does not match received orientation with the map visualization of the scan topic. This problem hindered most of the development as I am not diverse with ROS framework making the robot scan wrongly and creating wrong costmaps hindering path planning. Thirdly, is the scan topic again as it scans the robot supports as seen in Figure 24 and registering them as obstacle as seen left side of Figure 23. To solve this problem "laser_filters" package was used to filter the scans at the designated angle ignoring that range of angles. Thus, making the costmap clear of obstacle for the specified angle as seen in the right side of Figure 23. These problems are something I expected solved way before I took over as scan relies on the fundamental navigation of the robot. It was great the I got to learn and apply what I learnt on the spot. Nevertheless, this then begs the question of integrity and functionality of the previous student algorithms which is outside of this project scope seeing that the robot has these problems.

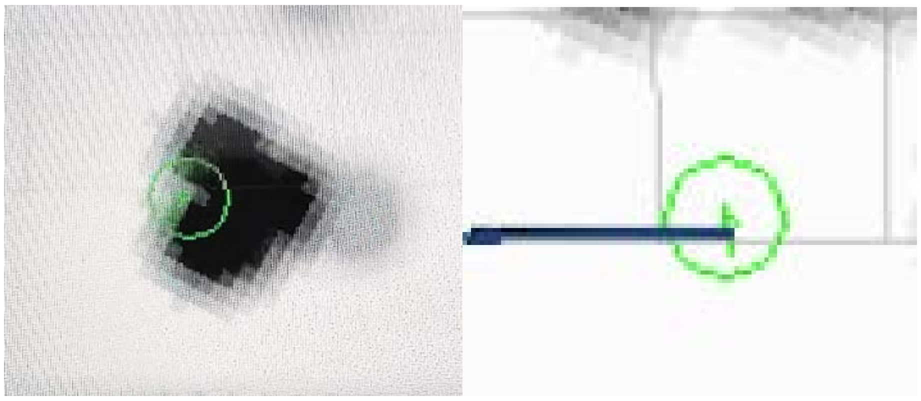


Figure 23: Lidar Registering Robot Legs as Obstacle Before (Left) and After (Right)

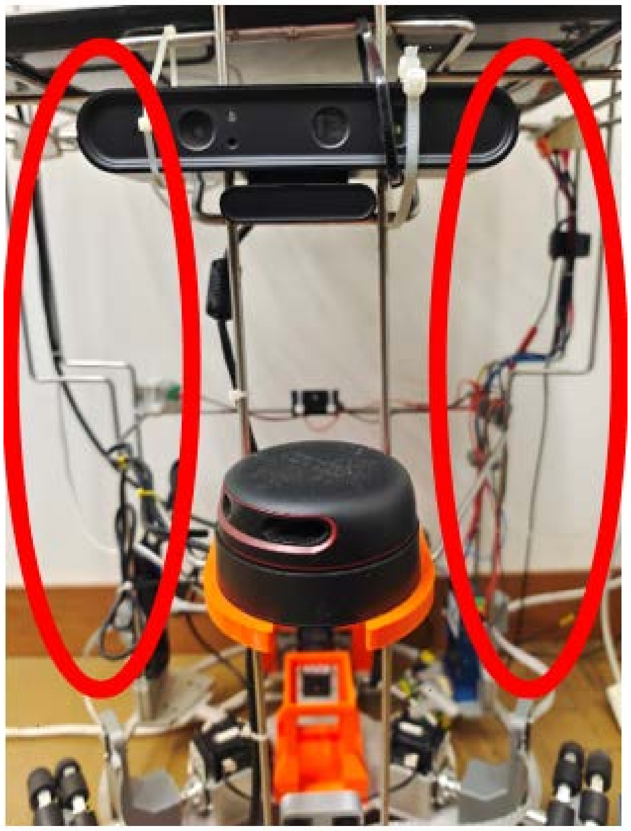
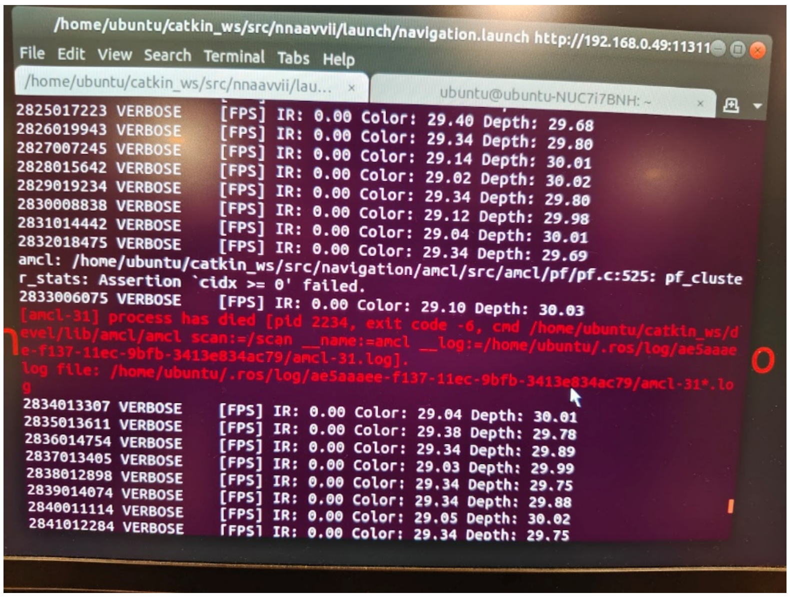


Figure 24: Robot Legs

Another challenge was processes failing unexpectedly as seen in Figure 25. This problem was very common when testing and requiring multiple resets of the whole system to try to make a function work. To minimize this occurrence, several processes were taken out leaving only the required ones to be running. This limits the functions available to tested thus leaving the obstacle avoidance out of this project's testing phase. An upgrade of PC is required for Serveur to function properly in the future.


/home/ubuntu/catkin_ws/src/nnaavvii/launch/navigation.launch http://192.168.0.49:11311
File Edit View Search Terminal Tabs Help
/home/ubuntu/catkin_ws/src/nnaavvii/lau...
ubuntu@ubuntu-NUC7i7BNH: ~

```
2825017223 VERBOSE    [FPS] IR: 0.00 Color: 29.40 Depth: 29.68
2826019943 VERBOSE    [FPS] IR: 0.00 Color: 29.34 Depth: 29.80
2827007245 VERBOSE    [FPS] IR: 0.00 Color: 29.14 Depth: 30.01
2828015642 VERBOSE    [FPS] IR: 0.00 Color: 29.02 Depth: 30.02
2829019234 VERBOSE    [FPS] IR: 0.00 Color: 29.34 Depth: 29.80
2830008838 VERBOSE    [FPS] IR: 0.00 Color: 29.12 Depth: 29.98
2831014442 VERBOSE    [FPS] IR: 0.00 Color: 29.04 Depth: 30.01
2832018475 VERBOSE    [FPS] IR: 0.00 Color: 29.34 Depth: 29.69
amcl: /home/ubuntu/catkin_ws/src/navigation/amcl/src/amcl/pf/pf.c:525: pf_cluste
r_stats: Assertion `cidx >= 0' failed.
2833006075 VERBOSE    [FPS] IR: 0.00 Color: 29.10 Depth: 30.03
[amcl-31] process has died [pid 2234, exit code -6, cmd /home/ubuntu/catkin_ws/d
evel/lib/amcl/amcl scan:=/scan __name:=amcl __log:=/home/ubuntu/.ros/log/ae5aaae
e-f137-11ec-9bfb-3413e834ac79/amcl-31.log].
log file: /home/ubuntu/.ros/log/ae5aaaee-f137-11ec-9bfb-3413e834ac79/amcl-31*.lo
g
2834013307 VERBOSE    [FPS] IR: 0.00 Color: 29.04 Depth: 30.01
2835013611 VERBOSE    [FPS] IR: 0.00 Color: 29.38 Depth: 29.78
2836014754 VERBOSE    [FPS] IR: 0.00 Color: 29.34 Depth: 29.89
2837013405 VERBOSE    [FPS] IR: 0.00 Color: 29.03 Depth: 29.99
2838012898 VERBOSE    [FPS] IR: 0.00 Color: 29.34 Depth: 29.75
2839014074 VERBOSE    [FPS] IR: 0.00 Color: 29.34 Depth: 29.88
2840011114 VERBOSE    [FPS] IR: 0.00 Color: 29.05 Depth: 30.02
2841012284 VERBOSE    [FPS] IR: 0.00 Color: 29.34 Depth: 29.75
```

Figure 25: Process Failing Unexpectedly

6.2 Future Works

First on the list of improvement is authentication. Authentication, whenever we interact with the internet nowadays there is always a page where we specify our identity so that we could access our data or do shopping using username and password. This keeps the non-users of application especially malicious ones away. The Robot controller app needs those as well and it could go in 2 ways of implementing it. Firstly, is through Google authentication. The creator of Flutter is Google thus implementing Google authentication is doable using Firebase. Firebase is a Google database software that allows Freemium subscription to its users. One might choose this way because of Google is trusted company with robust authentication scheme that ensures that the data is safe and protected. Another reason is convenience, as many people already have Google accounts thus using it for the app makes it easier for the user to log in in the app. The second way of implementing authentication is using the opensource way like what been used in this project. To implement this, it would require another server backend that uses HTTP to authenticate the user before giving access to the WebSocket server. Django and Flask opensource frameworks can fulfill such application. These are not the only frameworks that can fulfill the authentication but they are recommended as they run using python. Thereafter, the server can reside in the Raspberry PI together with the WebSocket server. Reason for opting this way is the localized database which ensures that data is within the restaurant and does not go over the internet.

Secondly, Encryption is another concept that needs to be implemented to the implementation idea before deploying it to the restaurants. As restaurants are filled with customers, data encryption is a must. Right now, the system architecture is communicating between devices in plain or decodable language. This makes it easy for anyone with malicious intent to harm anyone in the restaurant through the use of the waiter robot. To encrypt messages, wrap it using SSL. SSL is an encryption-based Internet security protocol. This ensures privacy from client to server. This makes the URL for WebSocket 'wss' and HTTP 'https'.

After authentication and encryption are done then configure firewall to allow control via internet enabling remote control of robot outside the restaurant. Because the data communicated between client and server are protected after encryption and authentication, the internet can then be used to enable control of robot safely and reliably.

Next on the list of improvements is data compression. Right now, the data are uncompressed and large. Data size scale proportionally to the size of the restaurant. Meaning with the staff UI and robot UI are not scalable data transfer wise. Rosbridge allows CBOR compression which makes the message smaller. To apply it, the messages need to be encoded to CBOR compression before sending it and decoded at its destination. This addition can incur an increase of development time. Because it requires to rebuild the code as decoding takes place after the data has been received when receiving and before sending then transferring data.

The application appearance isn't built with ease of use in mind. Thus, several parts of the application lacks on visual aid or control to make it easier to operate. A proposal of Quality-of-life improvements such as destination, path to take, and costmaps. Costmaps refers to the untraversable space on map in ROS. It is taken into account in path planning of the robot. This improves on the user experience for waiters. In addition, test its functionality and ease of use with different people preferably with experience in restaurant management for gathering feedback and further improving the capability of the application. As for the visual bug when goal posting the robot, one way to solve that is to use the package "laser_scan_matcher". Instead of using the raw value of scan given by the /scan topic, change it to an image and draw it to the canvas of the map to avoid the rotation offset given on the "Staff UI"

Aside from the software improvements there are several hardware improvements that can be done to the Serveur. First is design, there are several blind spots that hinders narrow area movement. In addition, its current flimsy design vibrates a lot. Furthermore, the electrical parts are exposed which is a hazardous when transporting food and beverages. Thus, an overhaul of the design is required. Secondly, adding a docking station to charge the battery of the robot. This is to allow the addition of charging behavior to the auto control. This makes the robot run fully semi-autonomous. After several improvements, incorporate the obstacle avoidance module and cocktail robot movement algorithm. Another improvement is to implement gmapping algorithm to make the robot create a map out of its scanned area and proceed to use that map as its work map.

7. Conclusion

In conclusion, following the OSI design the centralize network was able to make devices talk to each other using WebSocket allowing the integration of ROS with Flutter. Then the Staff UI and Robot UI application built using Flutter adequately fulfills necessary features needed to control a Waiter Robot. Aside from the scan visual bug, the application runs the implemented features such as teleoperation, calibration, goal posting, waypoint and automatic mode with scheduler as intended. This delegate some robot control to the users and allow them to customize the location that the robot will go without the help of the supplier. With the help of a centralized server a new deployment entering the food instead of the table and customer virtual cart was implemented. The deployment makes staffs think less of table increasing their productivity. Virtual cart adds visual aid for both customer and staffs increasing user experience. Furthermore, the universal status has restricted the control of the robot. Thus, preventing conflicting control between status.

The tech stack Flutter, ROS, Python and MariaDB being an opensource software, there were no expenditure spent on software licenses. Creating a project easy to scale and improve because of its full access to its features and the help of the opensource community. In addition, Flutter can be an alternative to JavaScript for developer as it is capable to display ROS messages to its desired output despite the implementation of app state management might not be the efficient. Furthermore, with time the developer can create multiple widgets that can be used in multiple projects that he/she might have which makes the development time much faster. Moreover, the combination of Python and MariaDB websocket server, created a similar environment to what Pudu or Softbank Robotics used in their implementation. This creates a scalable environment which in par with current implementation of waiter robots.

Before making this concept commercially available, several improvements must be made to make it secure and reliable. They are authentication, encryption, data compression, quality of life improvements and design overhaul of the mechanical parts to improve its capability and incorporate battery module to charge the battery. By accomplishing this and then this concept can be used on a real restaurant with realistic demands.

References

[1] E. GUIZZO and R. KLETT, "How Robots Became Essential Workers in the COVID-19 Response", *IEEE Spectrum*, 2022. [Online]. Available: https://spectrum.ieee.org/how-robots-became-essential-workers-in-the-covid19-response. [Accessed: 19- Jun- 2022].

[2] F. Loo and C. Sekkappan, "Notable Singapore restaurants, cafés and bars that have permanently closed", *Time Out Singapore*, 2022. [Online]. Available: https://www.timeout.com/singapore/restaurants/notable-restaurants-cafes-and-bars-permanently-closed. [Accessed: 19- Jun- 2022].

[3] H. Tan, "Restaurants v Covid-19: The fight for survival in Singapore's F&B scene", *The Straits Times*, 2022. [Online]. Available: https://www.straitstimes.com/life/food/restaurants-v-covid-19-the-fight-for-survival-in-singapores-fb-scene. [Accessed: 19- Jun- 2022].

[4] J. HENG, "Bigger role for service robots in pandemic", *Businesstimes.com.sg*, 2022. [Online]. Available: https://www.businesstimes.com.sg/government-economy/bigger-role-for-service-robots-in-pandemic. [Accessed: 19- Jun- 2022].

[5] "Food merchants and digital platforms in the era of Covid | Grab SG", *Grab SG*, 2022. [Online]. Available: https://www.grab.com/sg/blog/public-policy/food-merchants-and-digital-platforms-in-the-era-of-covid/. [Accessed: 19- Jun- 2022].

[6] W. TAN, "The rise of a robotic dawn in services industry", *TODAY*, 2017. [Online]. Available: https://www.todayonline.com/singapore/rise-robotic-dawn-services-industry. [Accessed: 19- Jun- 2022].

[7] M. Oitzman, “Relay+ is Savioke's new generation of Service Robot,” *Mobile Robot Guide*, 15-Dec-2021. [Online]. Available: https://mobilerobotguide.com/2021/12/15/relay-is-saviokes-new-generation-of-service-robot/. [Accessed: 01-Mar-2022].

[8] J. Lee, “Futuristic Haidilao With Robot Servers & Immersive Dining Experience opens at Marina Square Dec. 31, 2019,” *Mothership.SG - News from Singapore, Asia and around the world*, 2019. [Online]. Available: https://mothership.sg/2019/12/haidilao-marina-square-open/. [Accessed: 01-Mar-2022].

[9] S. PILLAI, "China's Keenon Robotics taps Singapore as launchpad, working with SoftBank", *Businesstimes.com.sg*, 2021. [Online]. Available: https://www.businesstimes.com.sg/garage/chinas-keenon-robotics-taps-singapore-as-launchpad-working-with-softbank. [Accessed: 19- Jun- 2022].

[10] "Singapore restaurant 'hires' robot waiters", *AsiaOne*, 2016. [Online]. Available: https://www.asiaone.com/singapore/singapore-restaurant-hires-robot-waiters. [Accessed: 19- Jun- 2022].

[11] T. Shimmura, R. Ichikari, T. Okuma, H. Ito, K. Okada and T. Nonaka, "Service robot introduction to a restaurant enhances both labor productivity and service

quality", *Science Direct*, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S221282712030425X. [Accessed: 19- Jun- 2022].

[12] E. SUGIURA, "Japan's 'work-via-robot' cafe helps disabled workers shine", *Nikkei Asia*, 2022. [Online]. Available: https://asia.nikkei.com/Business/Business-trends/Japan-s-work-via-robot-cafe-helps-disabled-workers-shine. [Accessed: 19- Jun- 2022].

[13] A. Dabral, A. Rawat, S. Juyal, S. Joshi and S. Pratap, "Robotic in Modern-Day Restaurants and its Impact on the Dining Experience", *Research Gate*, 2022. [Online]. Available: https://www.researchgate.net/publication/358877539_Robotic_in_Modern-Day_Restaurants_and_its_Impact_on_the_Dining_Experience. [Accessed: 19- Jun- 2022].

[14] M. Raza, "Introduction to the Gartner Hype Cycle", *BMC Blogs*, 2020. [Online]. Available: https://www.bmc.com/blogs/gartner-hype-cycle/. [Accessed: 19- Jun- 2022].

[15] E. Roth, "Amazon is still struggling to make drone deliveries work", *The Verge*, 2022. [Online]. Available: https://www.theverge.com/2022/4/11/23020549/amazon-struggling-drone-deliveries-prime-air-bezos. [Accessed: 19- Jun- 2022].

[16] M. A. Figliozzi, C. Tucker, and P. Polikakhina, “Drone deliveries logistics, efficiency, safety and last mile trade-offs,” PDXScholar, 2018. [Online]. Available: https://pdxscholar.library.pdx.edu/cengin_fac/553/. [Accessed: 19-Jun-2022].

[17] H. Eißfeldt, “Acceptance of drone delivery is limited (not only) by noise concerns,” *eLib*, 20-Oct-2020. [Online]. Available: https://elib.dlr.de/136906/. [Accessed: 19-Jun-2022].

[18] A. Alamalhodaei, “Zipline raises $250M at $2.75B valuation to build out its instant logistics service,” *TechCrunch*, 30-Jun-2021. [Online]. Available: https://techcrunch.com/2021/06/30/zipline-raises-250m-at-2-75b-valuation-to-build-out-its-instant-logistics-service/. [Accessed: 19-Jun-2022].

[19] *Smart Delivery Robot-Pudu Robotics*, 2021. [Online]. Available: https://www.pudurobotics.com/. [Accessed: 19-Jun-2022].

[20] V. Rishivaran “Design Project - Final Report Obstacle Avoidance using AI”, 2021.

[21] J. Hwang, J. J. Kim, K.-H. Joo, and H. M. Kim, “The antecedents and consequences of brand authenticity in the restaurant industry: Robot Service Employees Versus Human Service Employees,” *Taylor & Francis*, 06-May-2022. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/10548408.2022.2061678. [Accessed: 19-Jun-2022].

[22] V. VTech, “Kids will be more tech-savvy than their parents by the time they are 10 years old,” *Kids Will Be More Tech-Savvy Than Their Parents by the Time They Are 10 Years Old*, 16-Oct-2020. [Online]. Available: https://www.prnewswire.com/news-releases/kids-will-be-more-tech-savvy-than-their-parents-by-the-time-they-are-10-years-old-301154064.html. [Accessed: 19-Jun-2022].

[23] “Robot web tools,” *Robot Web Tools*. [Online]. Available: http://robotwebtools.org/. [Accessed: 19-Jun-2022].

[24] SoftBank, “Keenbot,” *SoftBank*, 19-Jul-2020. [Online]. Available: https://apac.softbankrobotics.com/apac/sg/keenbot/. [Accessed: 19-Jun-2022].

[25] One-Stop Vegetarian, “Robot waiter in Singapore restaurant || Ananda Bhavan Vegetarian 24x7 || pocket-friendly food,” *YouTube*, 27-Feb-2021. [Online]. Available: https://www.youtube.com/watch?v=j0tug5Z7UnA. [Accessed: 19-Jun-2022].

[26] Marses, “Hospitality | marses,” *Marses Robotic Solution*, 2022. [Online]. Available: https://www.marses.systems/hospitality/. [Accessed: 19-Jun-2022].

[27] 1st Century Robot Co., Ltd, “Waiter Robot Manufacturer / Supplier,” *Henan 1st Century Robot Co., Ltd.*, 2022. [Online]. Available: https://www.1st-crobot.com/. [Accessed: 19-Jun-2022].

[28] A. Y. S. Wan, E. Foo, W. S. M. Lau, Z. Y. Lai, and H. H. Chen, “Waiter Bots for Casual Restaurants,” *Semantic Scholar*, 2019. [Online]. Available: https://pdfs.semanticscholar.org/d7b4/2c4cb6c2b12869b57b0d419538038ab14656.pdf. [Accessed: 02-Mar-2022].

[29] Wan Ash YS “Capstone Project Report Waiter Bots in Restaurant – Velocity Control”, 2019.

[30] Lin Mei L. “Final Report Waiter! Campaign Please!”, 2021.

[31] Mlytics Learning Center, “What is the OSI model?: Mlytics,” *Mlytics Learning Center*, 08-Oct-2021. [Online]. Available: https://learning.mlytics.com/the-internet/what-is-the-osi-model/. [Accessed: 19-Jun-2022].

[32] MDN, “The web and web standards - learn web development: MDN,” *Learn web development | MDN*, 2022. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/The_web_and_web_standards. [Accessed: 19-Jun-2022].

[33] "What is a Tech Stack? Choosing What Goes In Yours", *Heap*, 2022. [Online]. Available: https://heap.io/topics/what-is-a-tech-stack. [Accessed: 20- Jul- 2022].

[34] M. Ubl and E. Kitamura, “Introducing WebSockets: Bringing sockets to the web - HTML5 rocks,” *HTML5 Rocks - A resource for open web HTML5 developers*, 20-Oct-2010. [Online]. Available: https://www.html5rocks.com/en/tutorials/websockets/basics/. [Accessed: 01-Mar-2022].

[35] N-able, “The difference between centralized and decentralized networks: N-able,” *N-able*, 13-Apr-2021. [Online]. Available: https://www.n-able.com/blog/centralized-vs-decentralized-network#:~:text=A%20centralized%20network%20architecture%20is,%2C%20data%20storage%2C%20and%20utilities. [Accessed: 19-Jun-2022].

[36] “What is a Raspberry Pi?,” *Raspberry Pi*, 20-Aug-2015. [Online]. Available: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/. [Accessed: 01-Mar-2022].

[37] pi-top, “Top: PI-top [4],” *pi-top*, 2021. [Online]. Available: https://www.pi-top.com/products/pi-top-4. [Accessed: 19-Jun-2022].

[38] Python.org, “What is python? executive summary,” *Python.org*, 2022. [Online]. Available: https://www.python.org/doc/essays/blurb/. [Accessed: 19-Jun-2022].

[39] Coursera, “What is python used for? A beginner's guide,” *Coursera*, 2022. [Online]. Available: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python. [Accessed: 19-Jun-2022].

[40] J. Vaughan, “What is mariadb? - definition from whatis.com,” *SearchDataManagement*, 11-Jan-2018. [Online]. Available: https://www.techtarget.com/searchdatamanagement/definition/MariaDB. [Accessed: 19-Jun-2022].

[41] Oracle, “What is a relational database?,” *What Is a Relational Database | Oracle Singapore*, 2022. [Online]. Available: https://www.oracle.com/sg/database/what-is-a-relational-database/. [Accessed: 19-Jun-2022].

[42] “Robot operating system,” *ROS*. [Online]. Available: https://www.ros.org/. [Accessed: 01-Mar-2022].

[43] S. Davies, “WebSphere Business Integration Pub/Sub Solutions,” *Amazon*, 2004. [Online]. Available: https://aws.amazon.com/pub-sub-messaging/. [Accessed: 19-Jun-2022].

[44] The Robotics Back-End, “What is a ROS node? - the robotics back,” *The Robotics Back-End*, 28-Sep-2021. [Online]. Available: https://roboticsbackend.com/what-is-a-ros-node/. [Accessed: 19-Jun-2022].

[45] ros.org, “Wiki,” *ros.org*, 2018. [Online]. Available: http://wiki.ros.org/rqt_graph. [Accessed: 19-Jun-2022].

[46] L. Poubel, “Open source robotics: Getting started with Gazebo and Ros 2,” *InfoQ*, 07-May-2019. [Online]. Available: https://www.infoq.com/articles/ros-2-gazebo-tutorial/. [Accessed: 19-Jun-2022].

[47] automaticaddison, “What is the difference between rviz and gazebo?,” *Automatic Addison*, 27-Jun-2020. [Online]. Available: https://automaticaddison.com/what-is-the-difference-between-rviz-and-gazebo/. [Accessed: 19-Jun-2022].

[48] “Wiki,” *ros.org*, 2017. [Online]. Available: http://wiki.ros.org/rosbridge_suite. [Accessed: 01-Mar-2022].

[49] “Build apps for any screen,” *Flutter*. [Online]. Available: https://flutter.dev/. [Accessed: 01-Mar-2022].

[50] Perfecto, “What is the Flutter Framework and why you should learn it today: By perforce,” *Perfecto by Perforce*, 2022. [Online]. Available: https://www.perfecto.io/blog/what-is-flutter-framework. [Accessed: 19-Jun-2022].

[51] Flutter, “Adding interactivity to your flutter app,” *Flutter*. [Online]. Available: https://docs.flutter.dev/development/ui/interactive. [Accessed: 19-Jun-2022].

[52] C. Heidebrecht, A. Rymarz, and T. Whiting, “Roslib: Flutter Package,” *Dart packages*, 12-Jun-2019. [Online]. Available: https://pub.dev/packages/roslib. [Accessed: 19-Jun-2022].

[53] K. Moore, N. Weizenbaum, N. Bosch, J. Richman, D. Carew, T. Volkert, F. Yow, O. Sand, analogic, J. MacDonald, D. Tuppeny, P. Chalin, M. J. Rosenthal, N. T. Loi, L. Petersen, chirag729, L. R. H. Nielsen, A. Thomas, G. Georgiev, choong, and B. C. Ko, “Web_socket_channel: Dart package,” *Dart packages*, 26-Apr-2022. [Online]. Available: https://pub.dev/packages/web_socket_channel. [Accessed: 19-Jun-2022].

[54] K. Serrano, "Update your browser to use Google Drive, Docs, Sheets, Sites, Slides, and Forms - Google Drive Help", Drive.google.com, 2022. [Online]. Available: https://drive.google.com/drive/folders/1wQ65cGVqvIinjUUpMmcDL9yg-XR0gYxQ?usp=sharing. [Accessed: 27- Jul- 2022].

[55] Synopsys, “What is model-based design (MBD)? – how it works?,” *Synopsys*, 2022. [Online]. Available: https://www.synopsys.com/glossary/what-is-model-based-design.html. [Accessed: 19-Jun-2022].

[56] JSON, “Introducing json,” *JSON*. [Online]. Available: https://www.json.org/json-en.html. [Accessed: 19-Jun-2022].

[57] M. Probst, “Instantly parse JSON in any language,” *quicktype*. [Online]. Available: https://app.quicktype.io/. [Accessed: 19-Jun-2022].

[58] "Open Source Initiative", Open Source Initiative, 2022. [Online]. Available: https://opensource.org/. [Accessed: 27- Jul- 2022].

Appendices

Appendix A – Flutter App "Staff UI" Screens

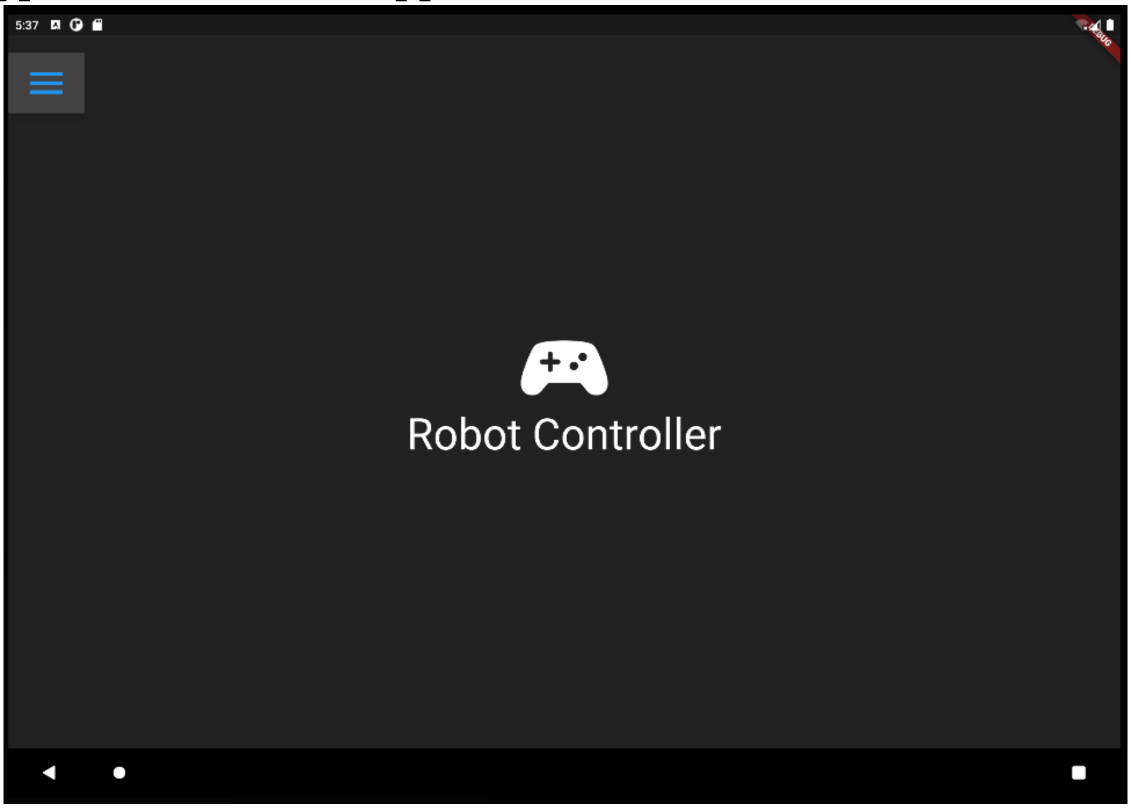


Figure A1: Home Screen

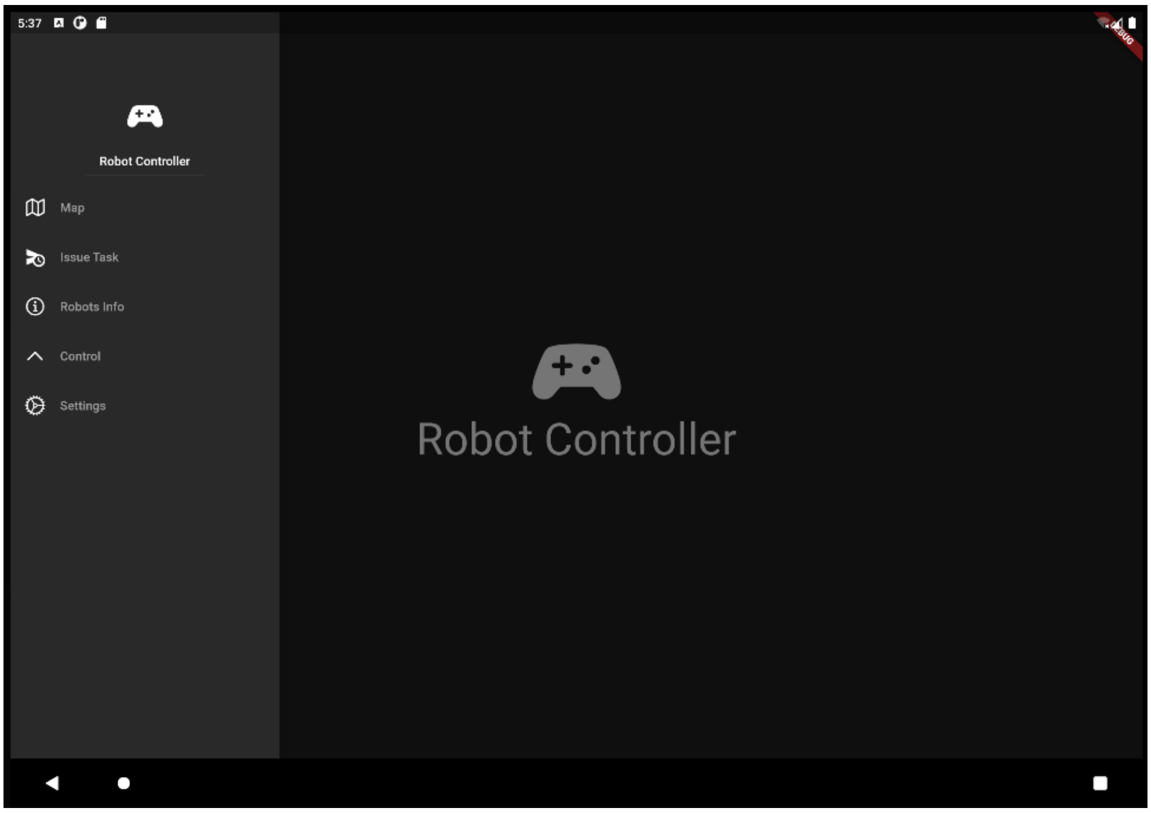


Figure A2: Initial Screen with Side menu

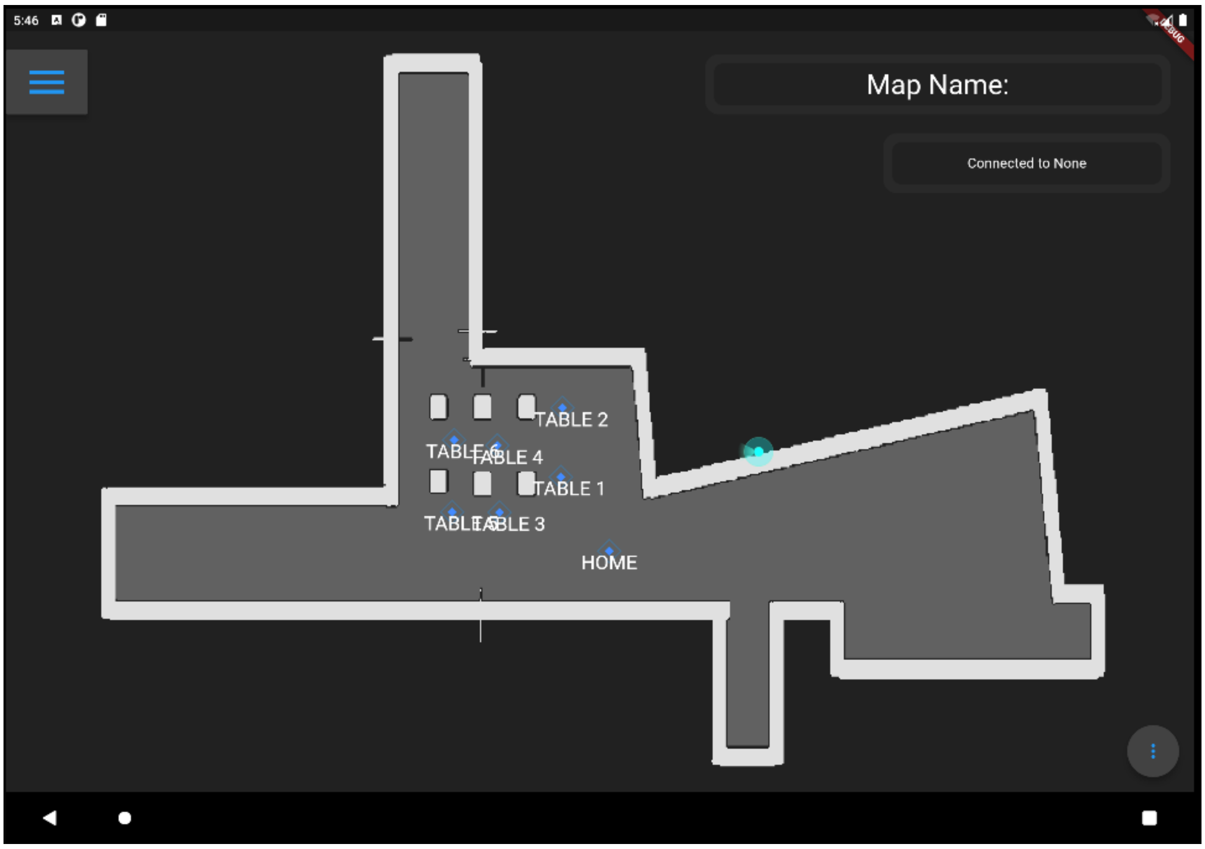


Figure A3: Map Screen Not Connected to Robot

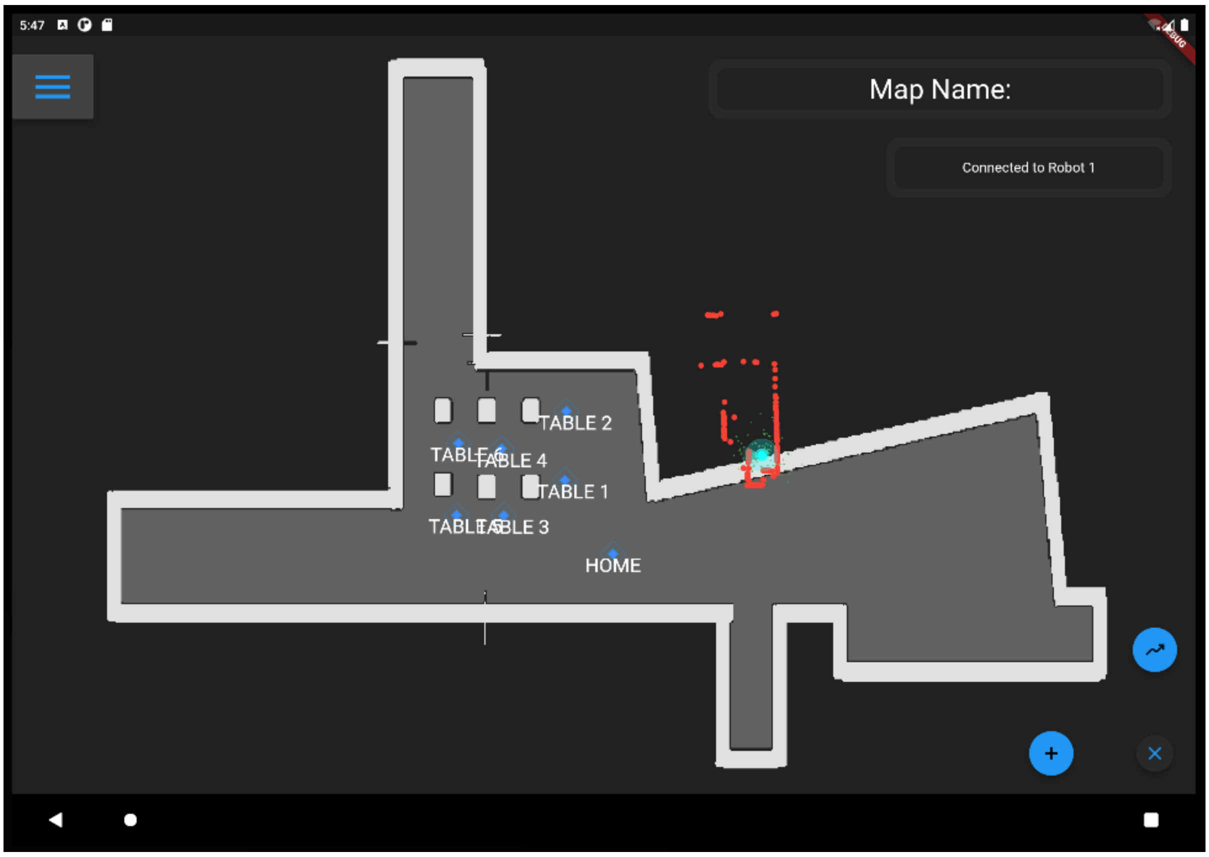


Figure A4: Map Screen Connected to Robot

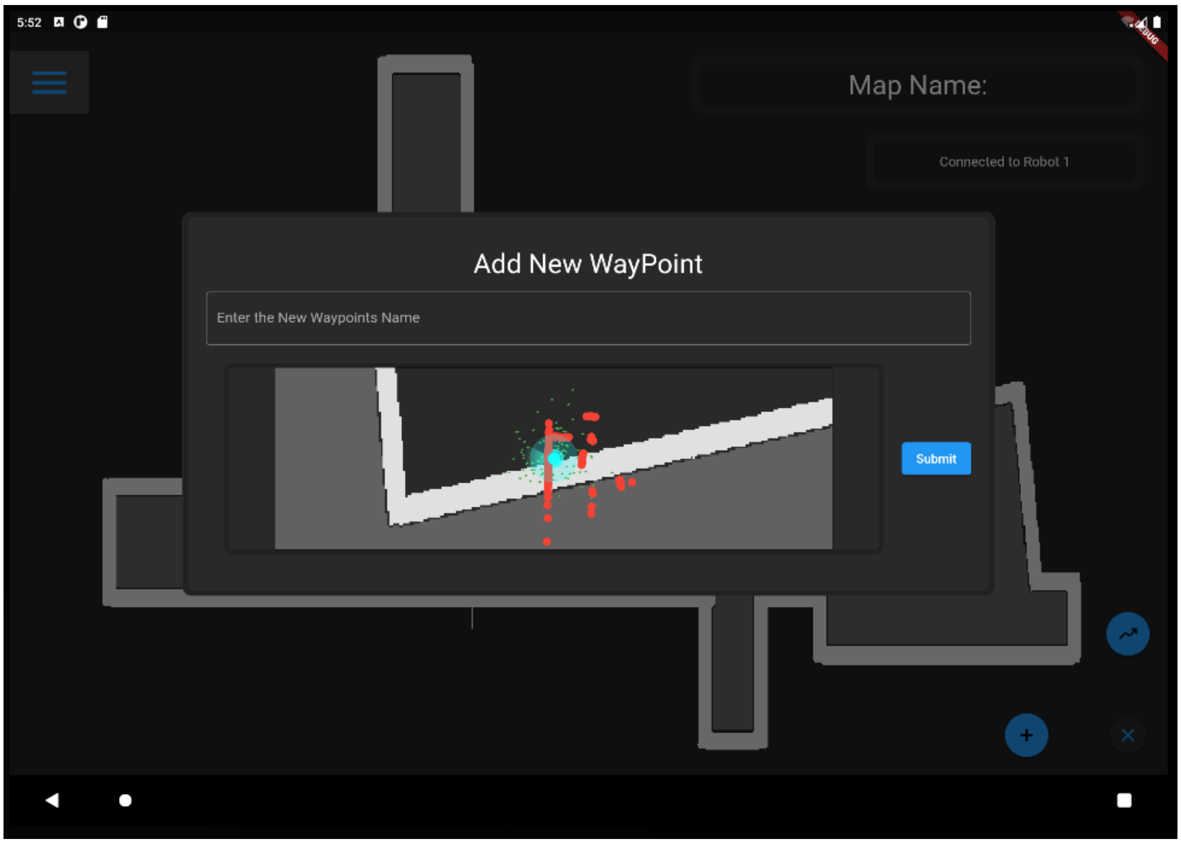


Figure A5: Map Screen Creating New Waypoint

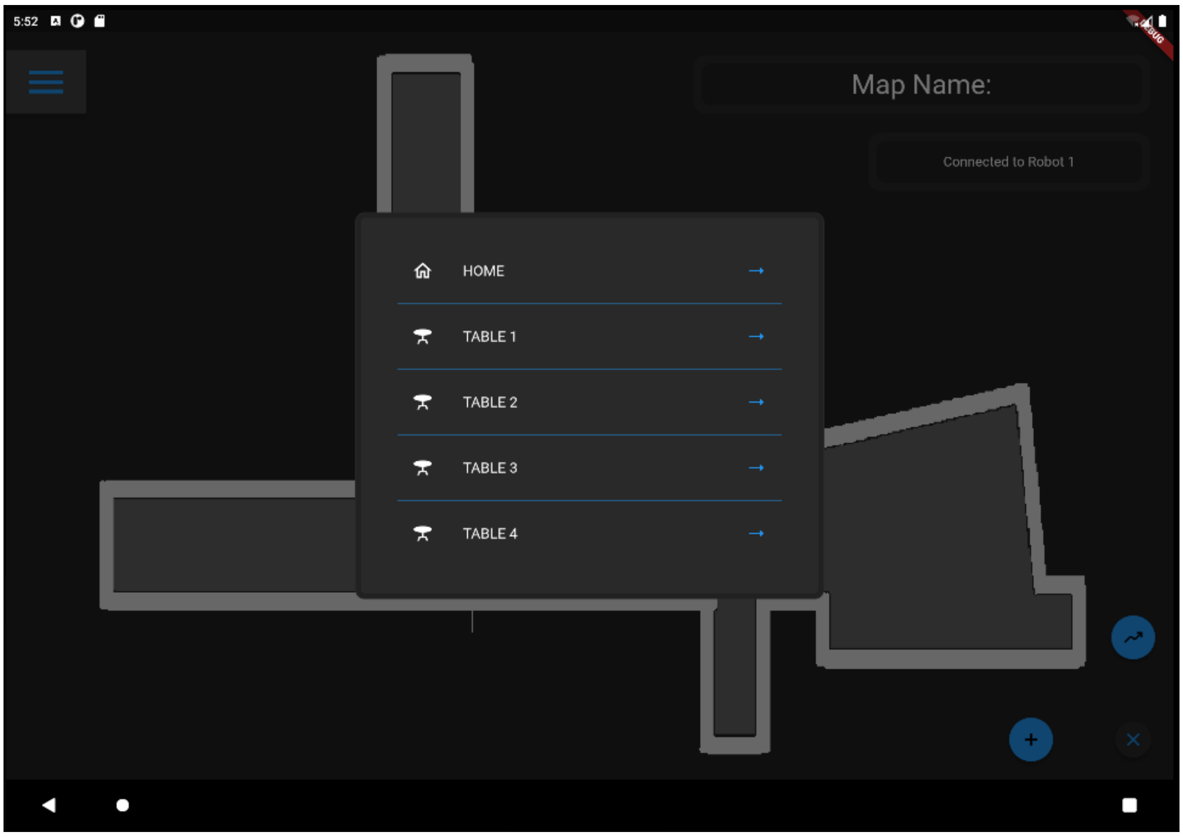


Figure A6: Map Screen Moving Robot Using Go-To Function

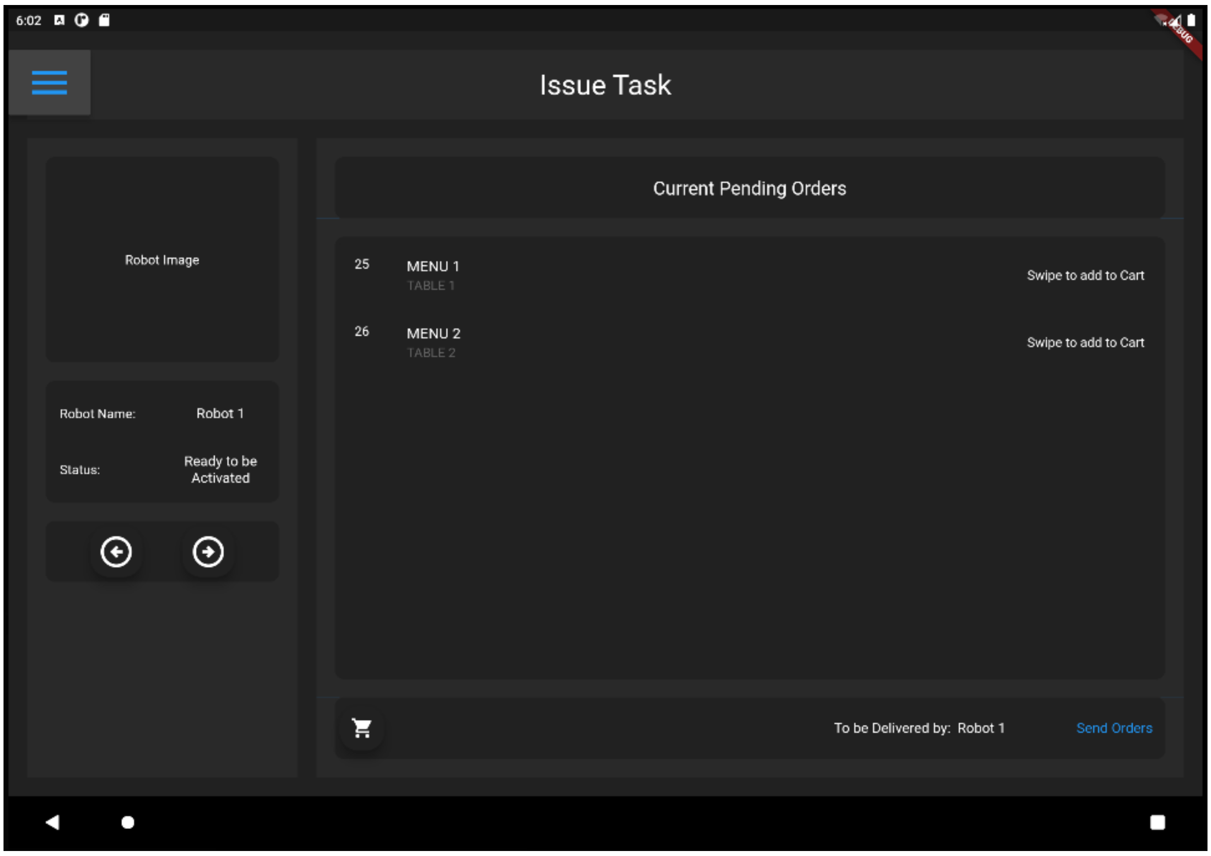


Figure A7: Issue Task Screen

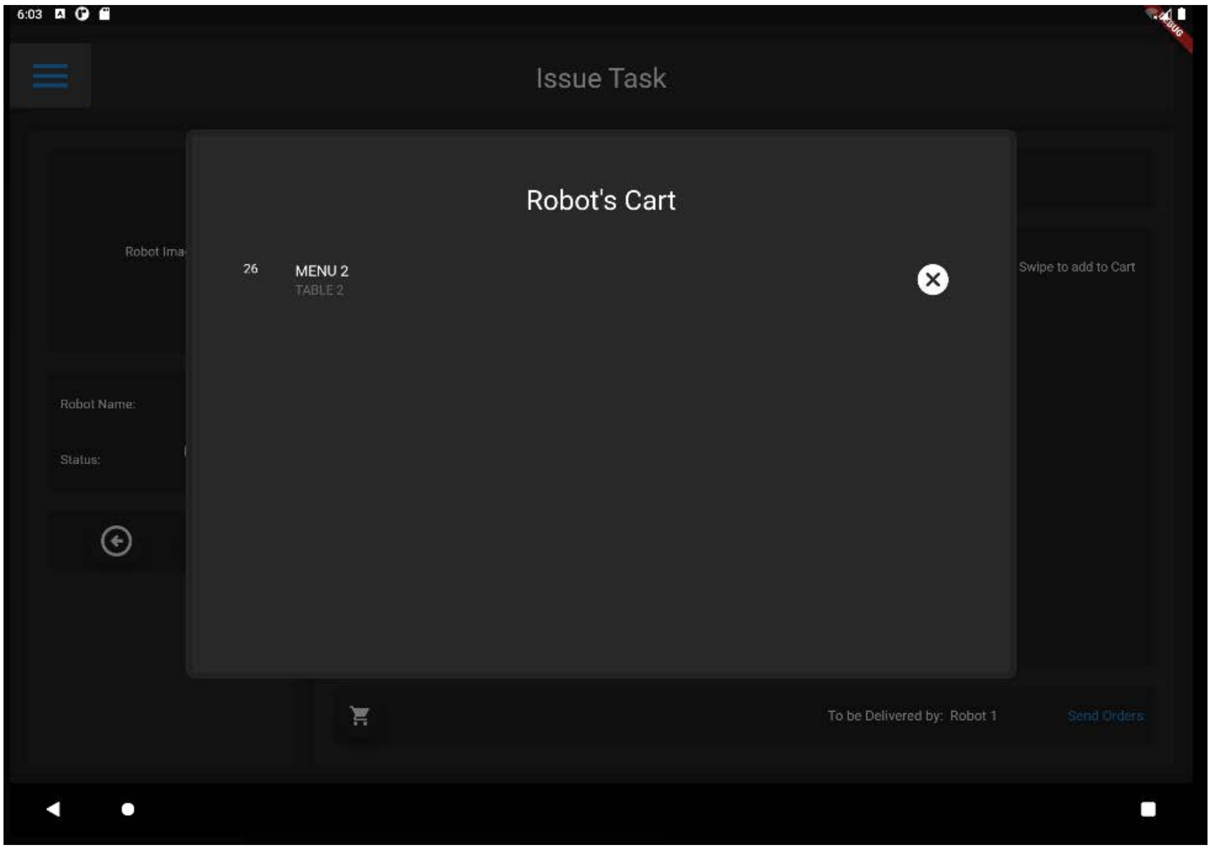
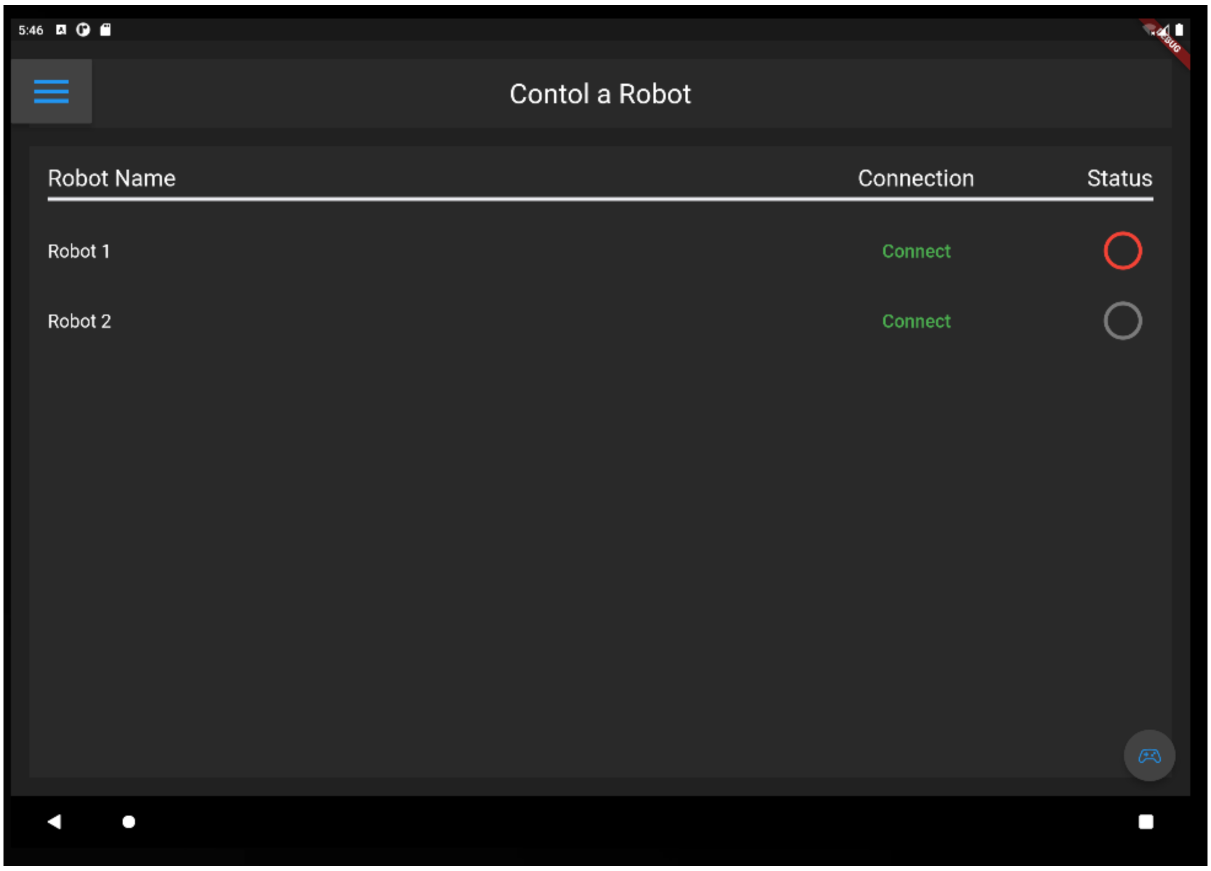


Figure A8: Issue Screen Cart Contents


Contol a Robot

Robot Name	Connection	Status
Robot 1	Connect	
Robot 2	Connect	

Figure A9: Control Robot Screen

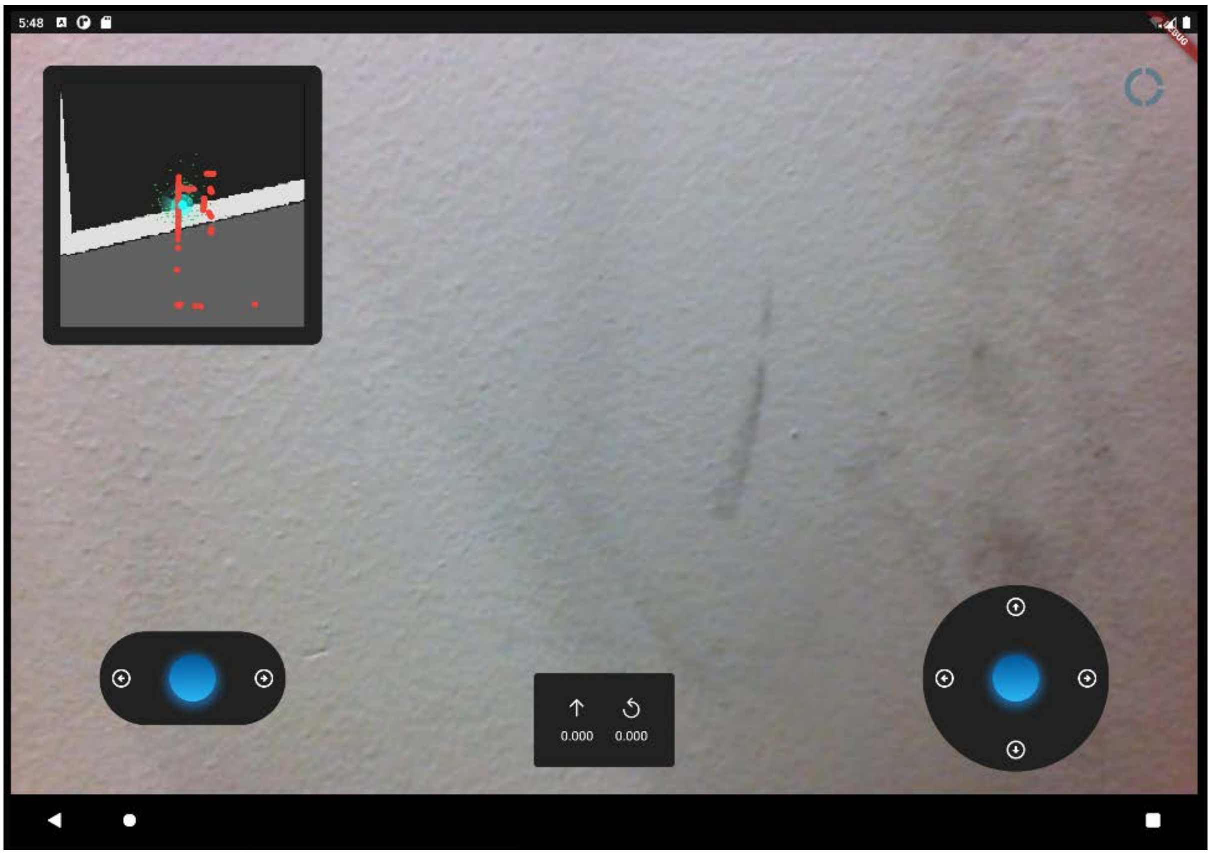
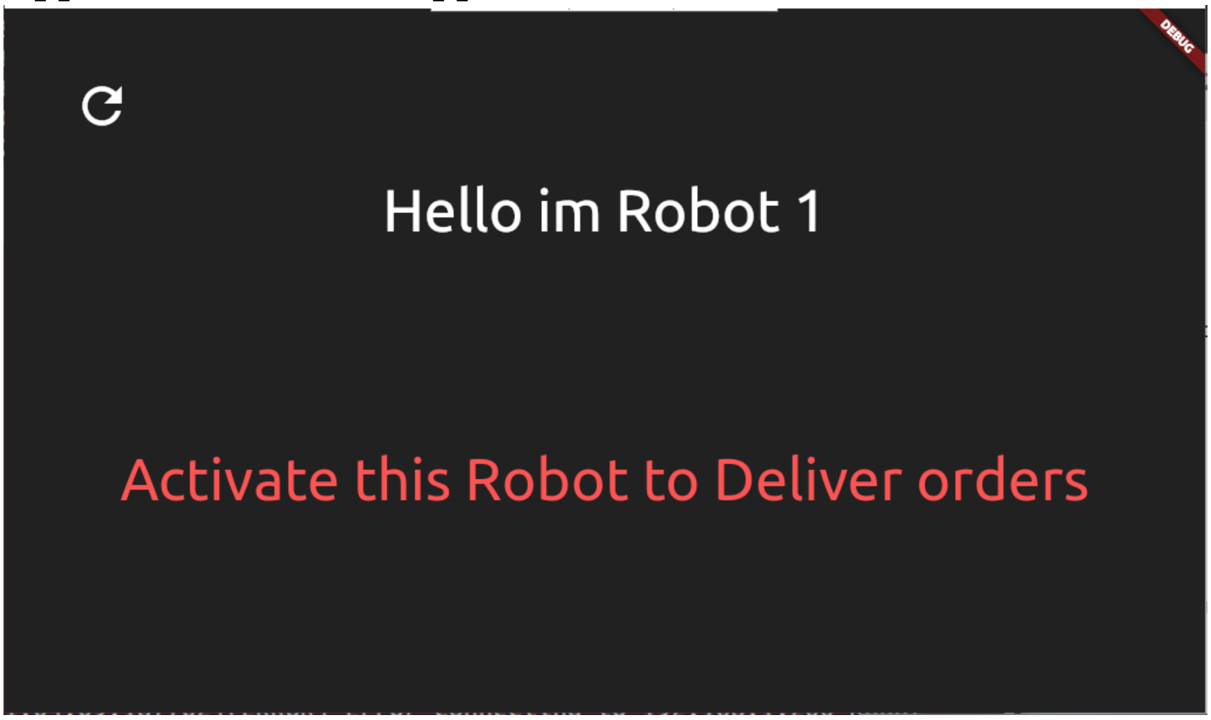


Figure A10: Controller Screen

Appendix B – Flutter App “Robot UI” Screens

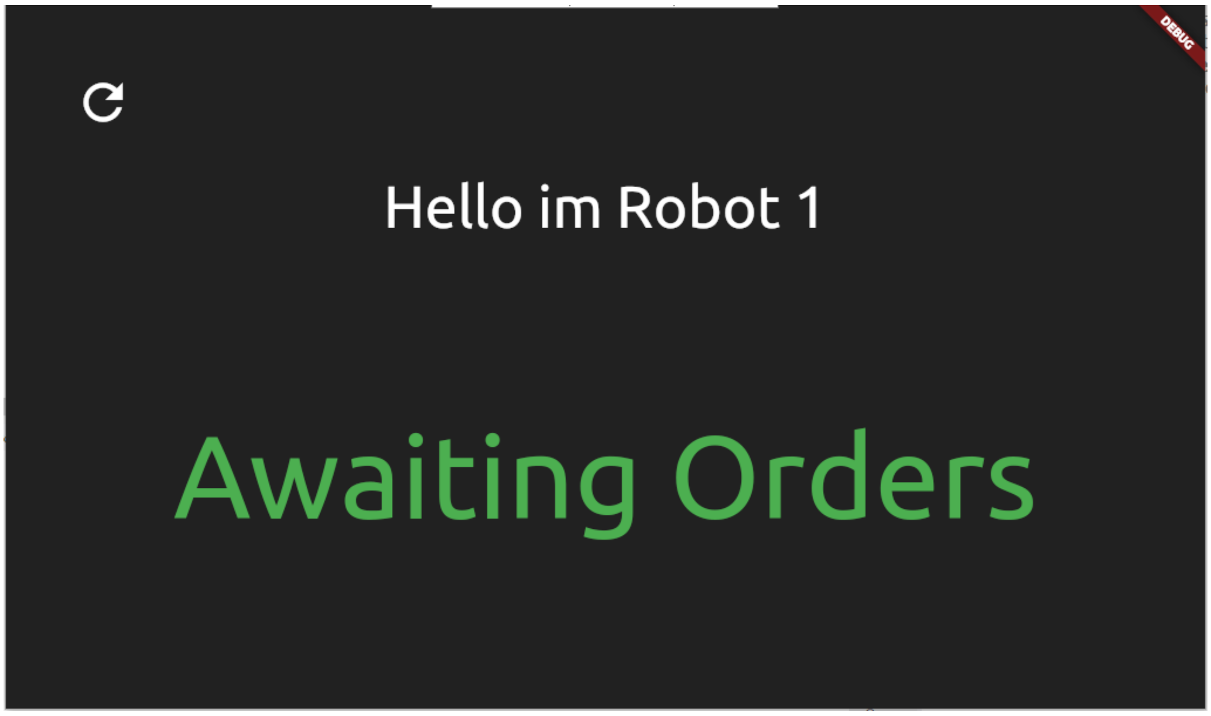


DEBUG

Hello im Robot 1

Activate this Robot to Deliver orders

Figure B1: Status "Ready"



DEBUG

Hello im Robot 1

Awaiting Orders

Figure B2: Status "Awaiting"

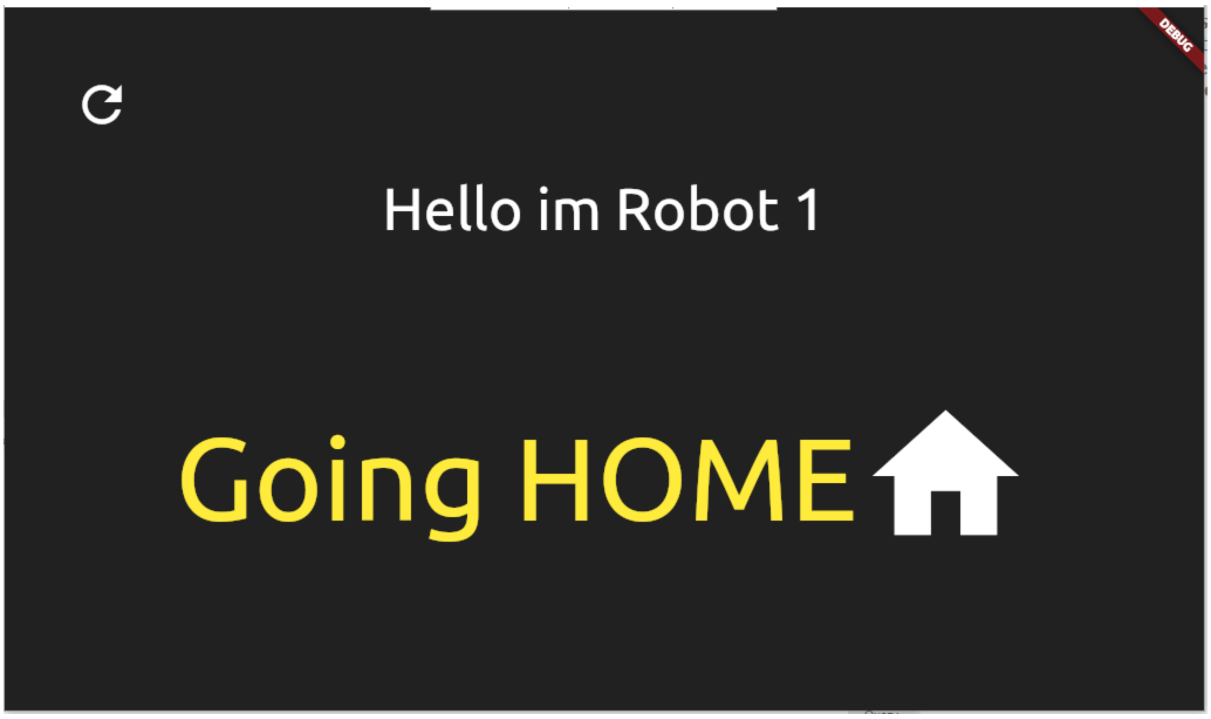


Figure B3: Status "Transit"

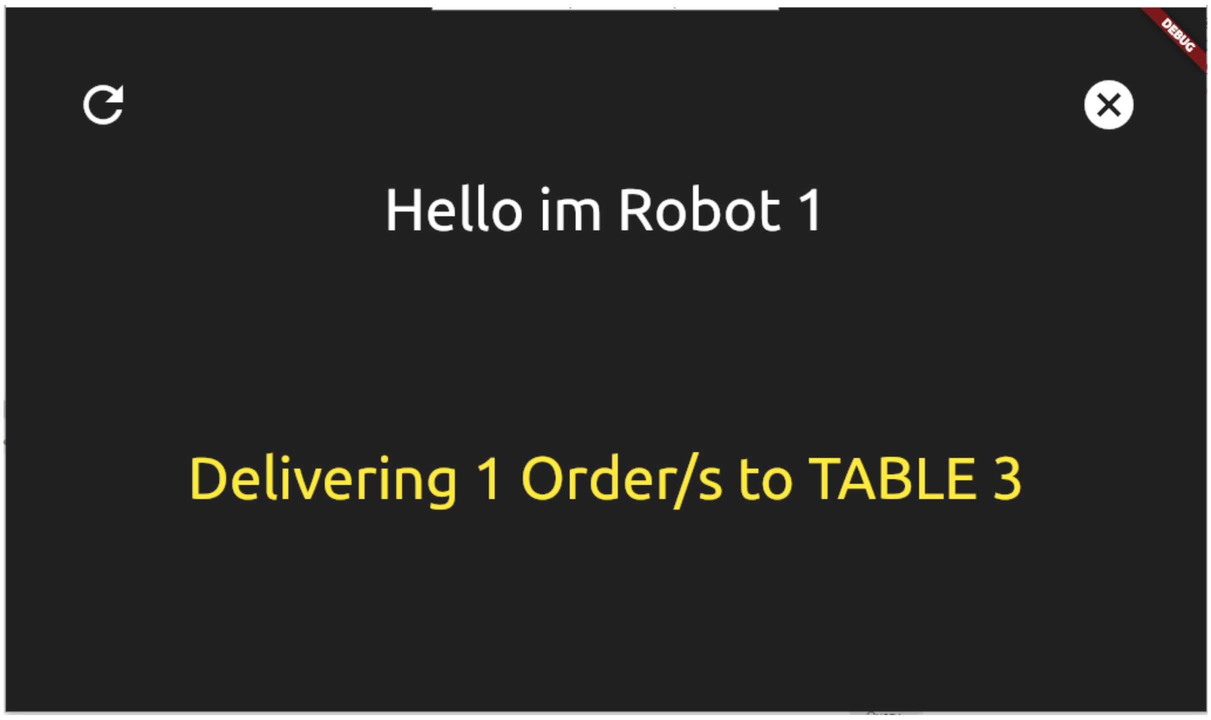


Figure B4: Status "Delivering"

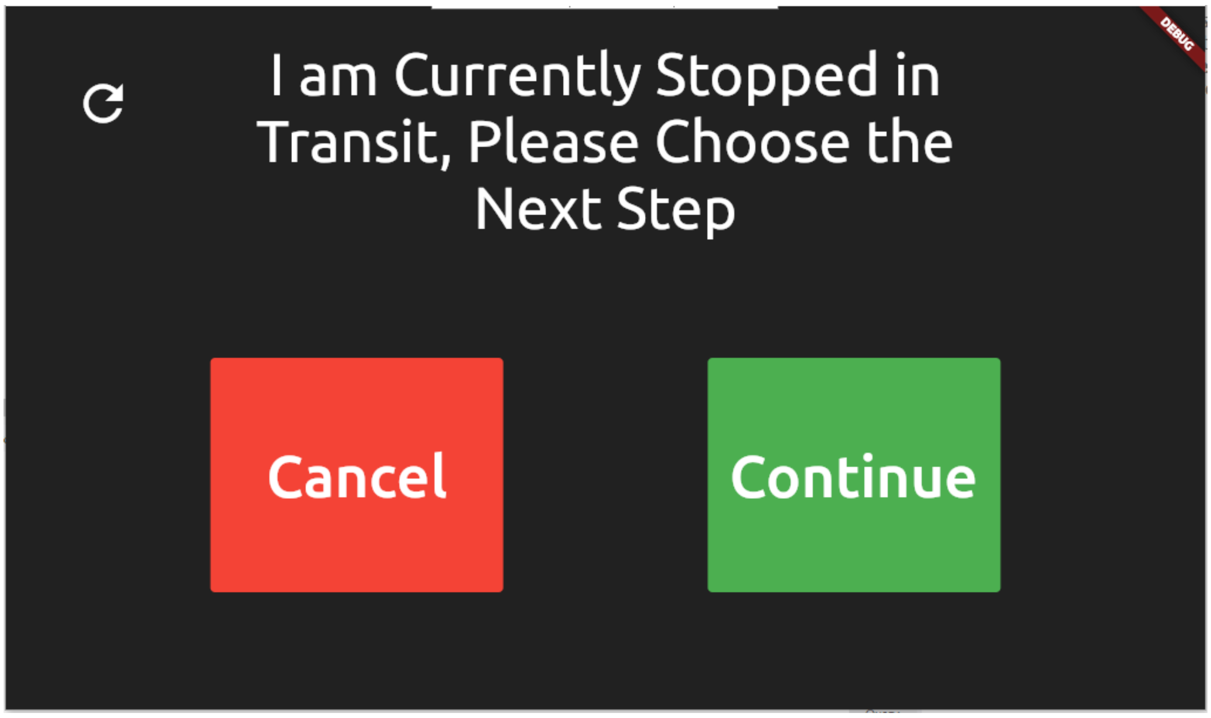

DEBUG
I am Currently Stopped in Transit, Please Choose the Next Step
Cancel
Continue

Figure B5: Status "Stopped"

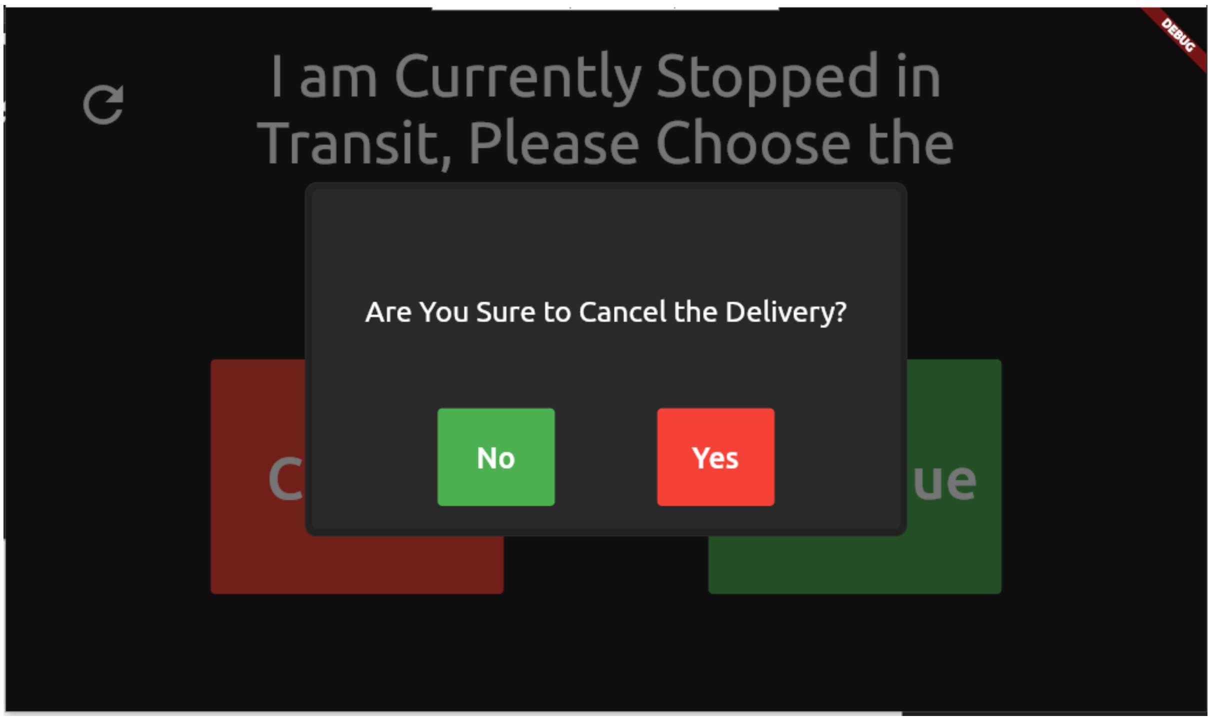

DEBUG
I am Currently Stopped in Transit, Please Choose the
Are You Sure to Cancel the Delivery?
No
Yes

Figure B6: Status "Stopped" Confirmation

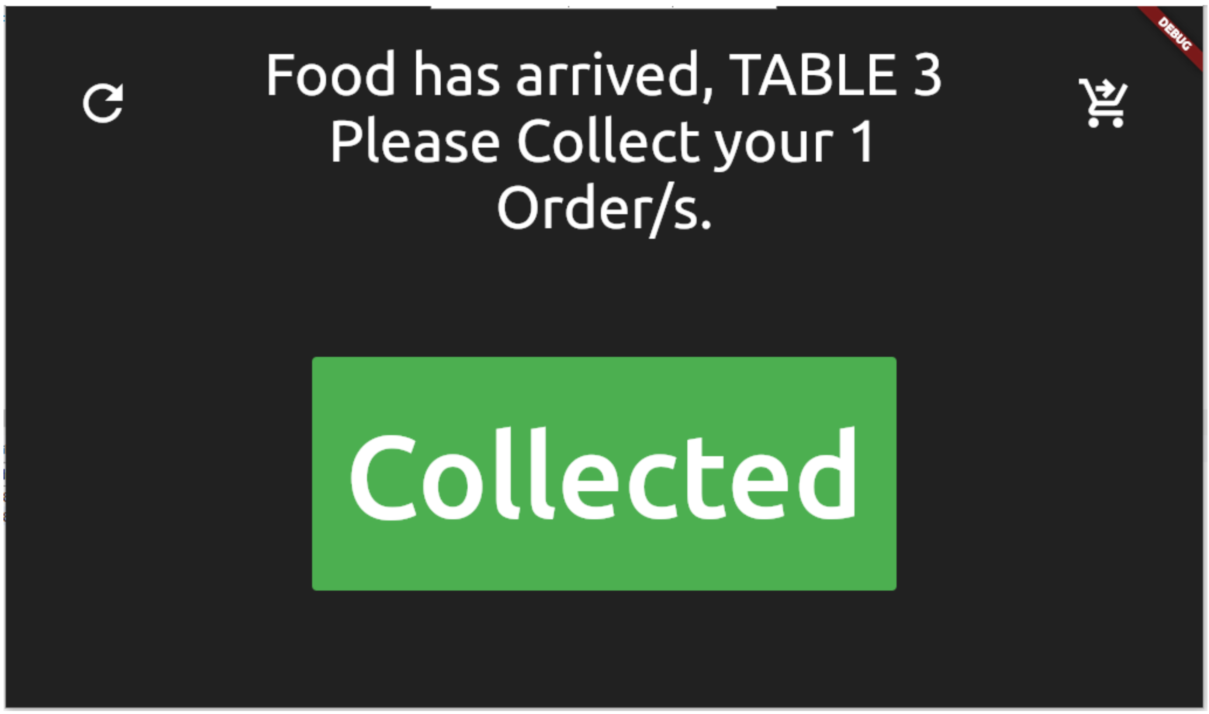
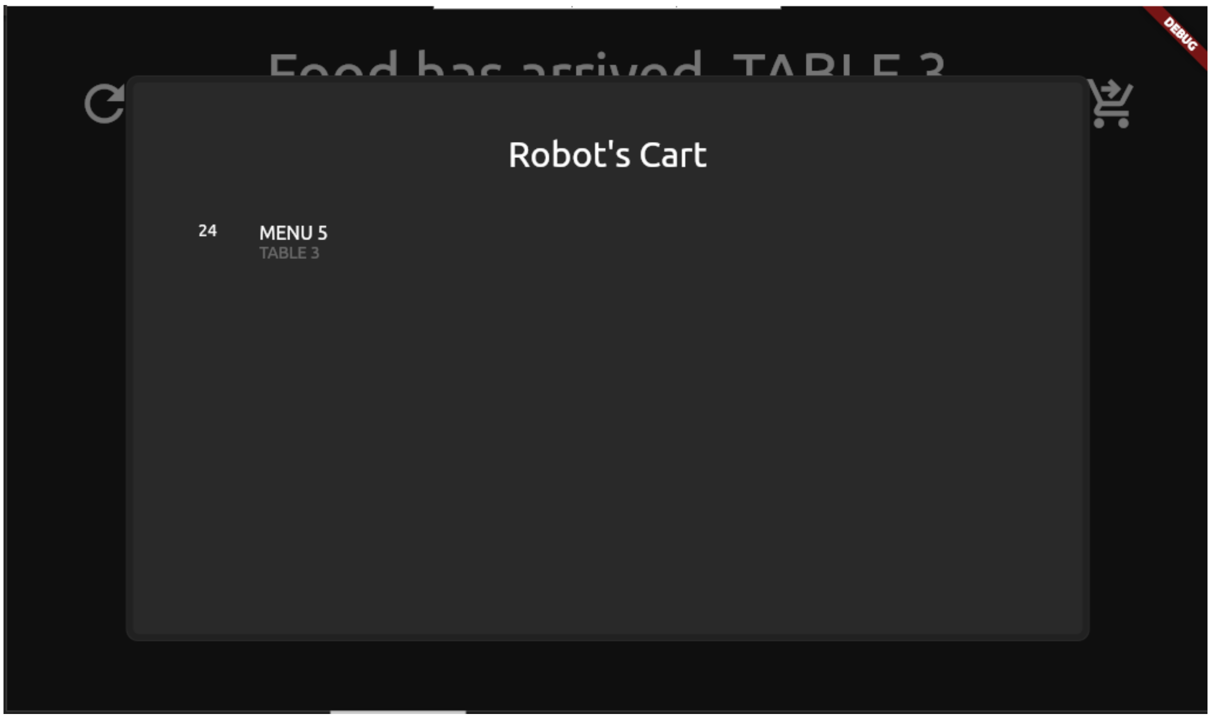

Food has arrived, TABLE 3 Please Collect your 1 Order/s.

Figure B7: Status "Delivered"


Robot's Cart

24 MENU 5
TABLE 3

Figure B8: Status "Delivered" Cart Item

Appendix C – Database Codes and Tables

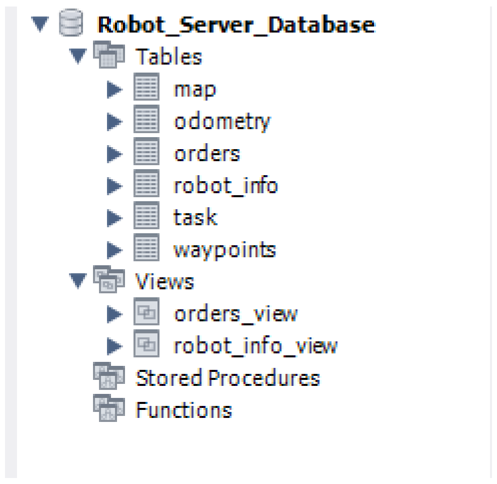


Figure C1: Database Tables

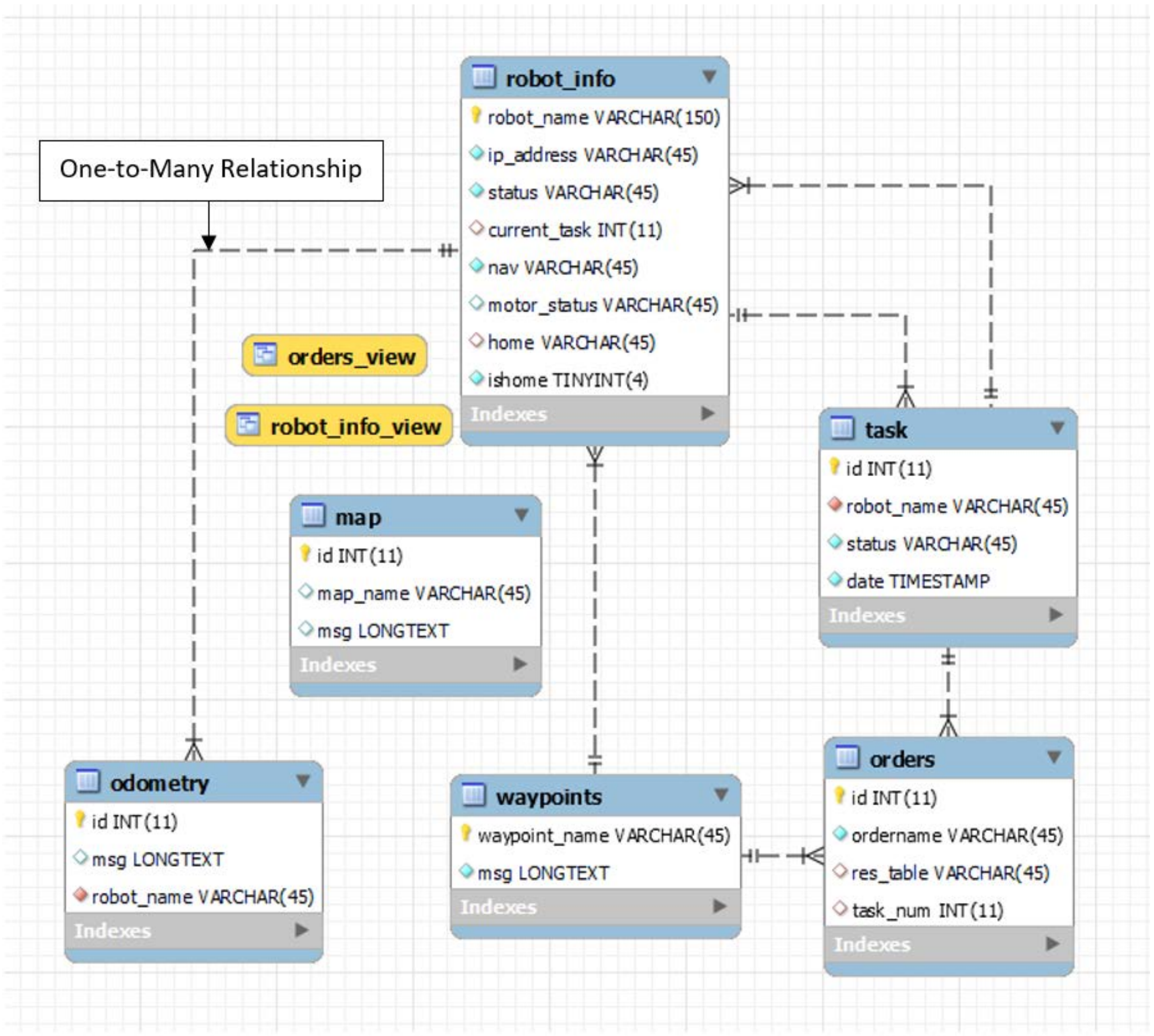


Figure C2: Tables Relationships

Orders View

```
VIEW `orders_view` AS
  SELECT
    `orders`.`id` AS `id`,
    `orders`.`ordername` AS `ordername`,
    `task`.`robot_name` AS `robot_name`,
    `orders`.`res_table` AS `res_table`,
    `task`.`status` AS `status`,
    `task`.`id` AS `task_id`
  FROM
    (`orders`
    LEFT JOIN `task` ON (`orders`.`task_num` = `task`.`id`))
  GROUP BY `orders`.`id`
```

Robot Info View

```
VIEW `robot_info_view` AS
  SELECT
    `robot_info`.`robot_name` AS `robot_name`,
    `robot_info`.`ip_address` AS `ip_address`,
    `robot_info`.`status` AS `status`,
    `robot_info`.`nav` AS `nav`,
    `robot_info`.`motor_status` AS `motor_status`,
    `robot_info`.`home` AS `home`,
    `orders_view`.`res_table` AS `destination`,
    COUNT(`orders_view`.`task_id`) AS `number_of_orders`
  FROM
    (`robot_info`
    LEFT JOIN `orders_view` ON (`robot_info`.`current_task` = `orders_view`.`task_id`))
  GROUP BY `robot_info`.`robot_name`
```

Appendix D – ROS Graph

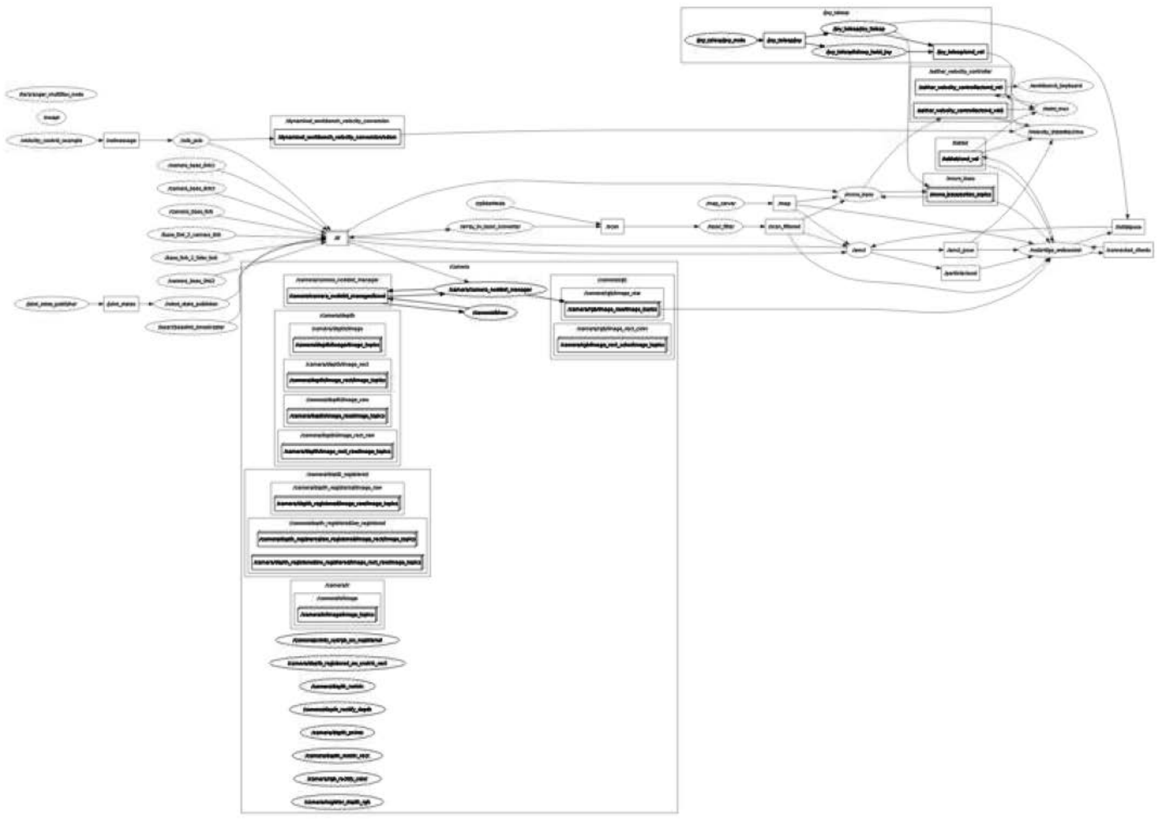


Figure D1: Full ROS Graph

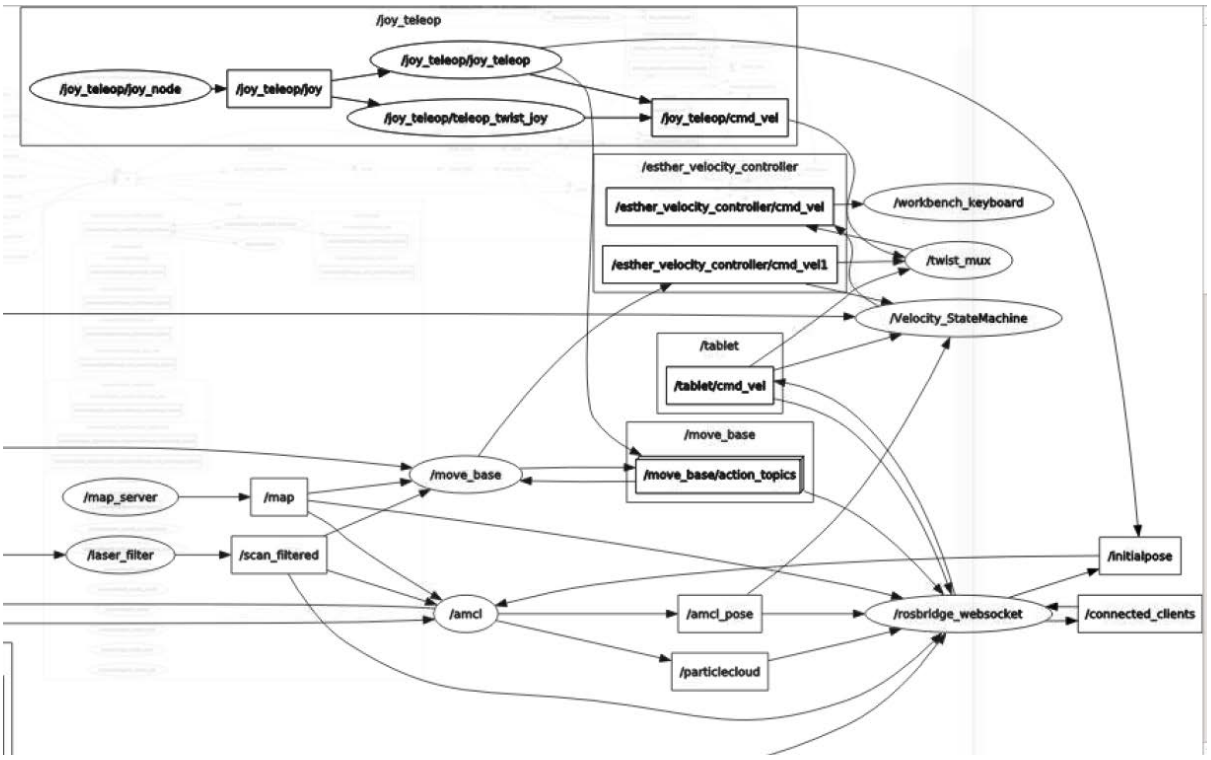


Figure D2: Top Right Side ROS Graph

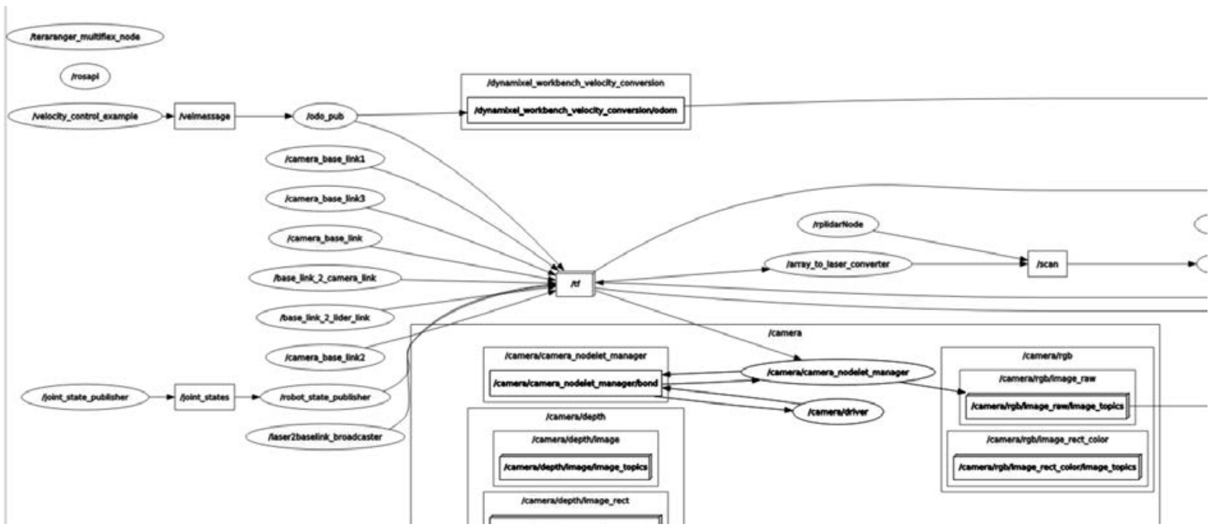

/teraranger_multiflex_node
/rosapi
/velocity_control_example
/velmessage
/odo_pub
/dynamixel_workbench_velocity_conversion
/dynamixel_workbench_velocity_conversion/odom
/camera_base_link1
/camera_base_link3
/camera_base_link
/base_link_2_camera_link
/base_link_2_lider_link
/camera_base_link2
/joint_state_publisher
/joint_states
/robot_state_publisher
/laser2baselink_broadcaster
/tf
/rplidarNode
/array_to_laser_converter
/scan
/camera
/camera/camera_nodelet_manager
/camera/camera_nodelet_manager/bond
/camera/camera_nodelet_manager
/camera/driver
/camera/depth
/camera/depth/image
/camera/depth/image/image_topics
/camera/depth/image_rect
/camera/rgb
/camera/rgb/image_raw
/camera/rgb/image_raw/image_topics
/camera/rgb/image_rect_color
/camera/rgb/image_rect_color/image_topics

Figure D3: Top Left Side ROS Graph

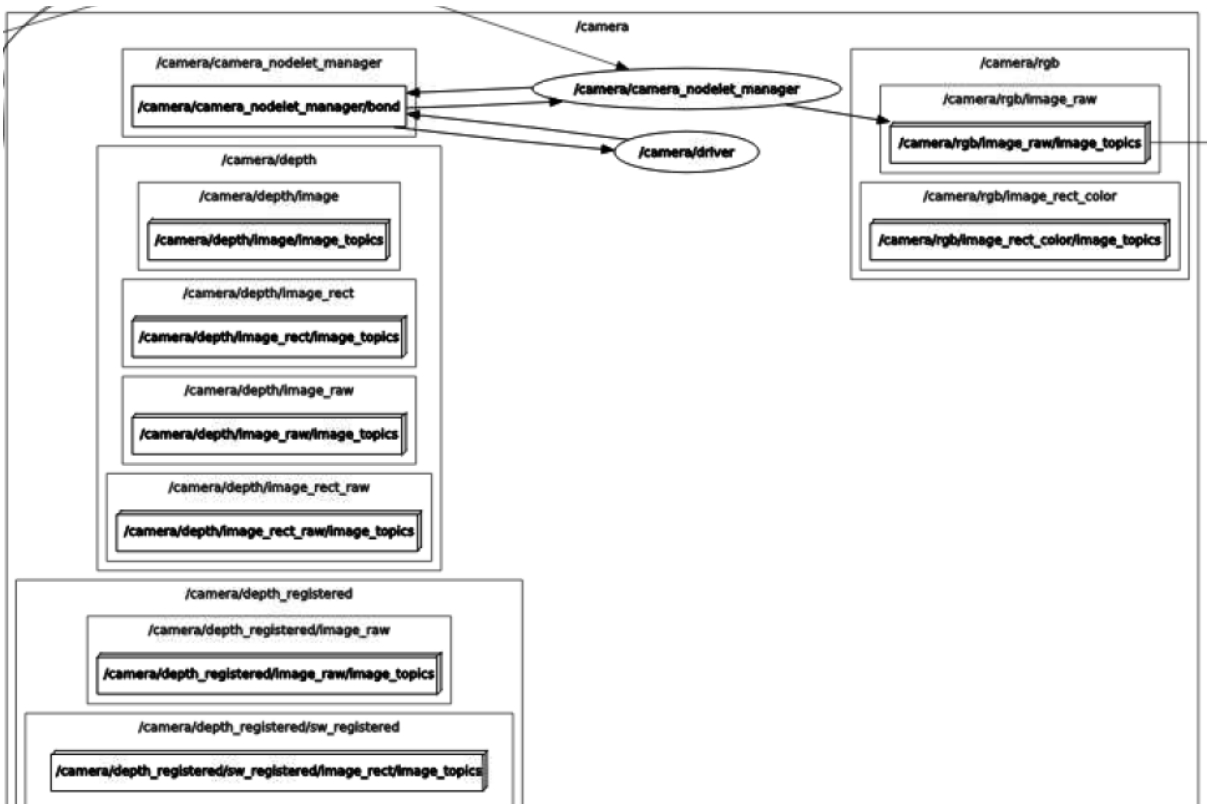

/camera
/camera/camera_nodelet_manager
/camera/camera_nodelet_manager/bond
/camera/camera_nodelet_manager
/camera/driver
/camera/rgb
/camera/rgb/image_raw
/camera/rgb/image_raw/image_topics
/camera/rgb/image_rect_color
/camera/rgb/image_rect_color/image_topics
/camera/depth
/camera/depth/image
/camera/depth/image/image_topics
/camera/depth/image_rect
/camera/depth/image_rect/image_topics
/camera/depth/image_raw
/camera/depth/image_raw/image_topics
/camera/depth/image_rect_raw
/camera/depth/image_rect_raw/image_topics
/camera/depth_registered
/camera/depth_registered/image_raw
/camera/depth_registered/image_raw/image_topics
/camera/depth_registered/sw_registered
/camera/depth_registered/sw_registered/image_rect/image_topics

Figure D4: Camera Node Top ROS Graph

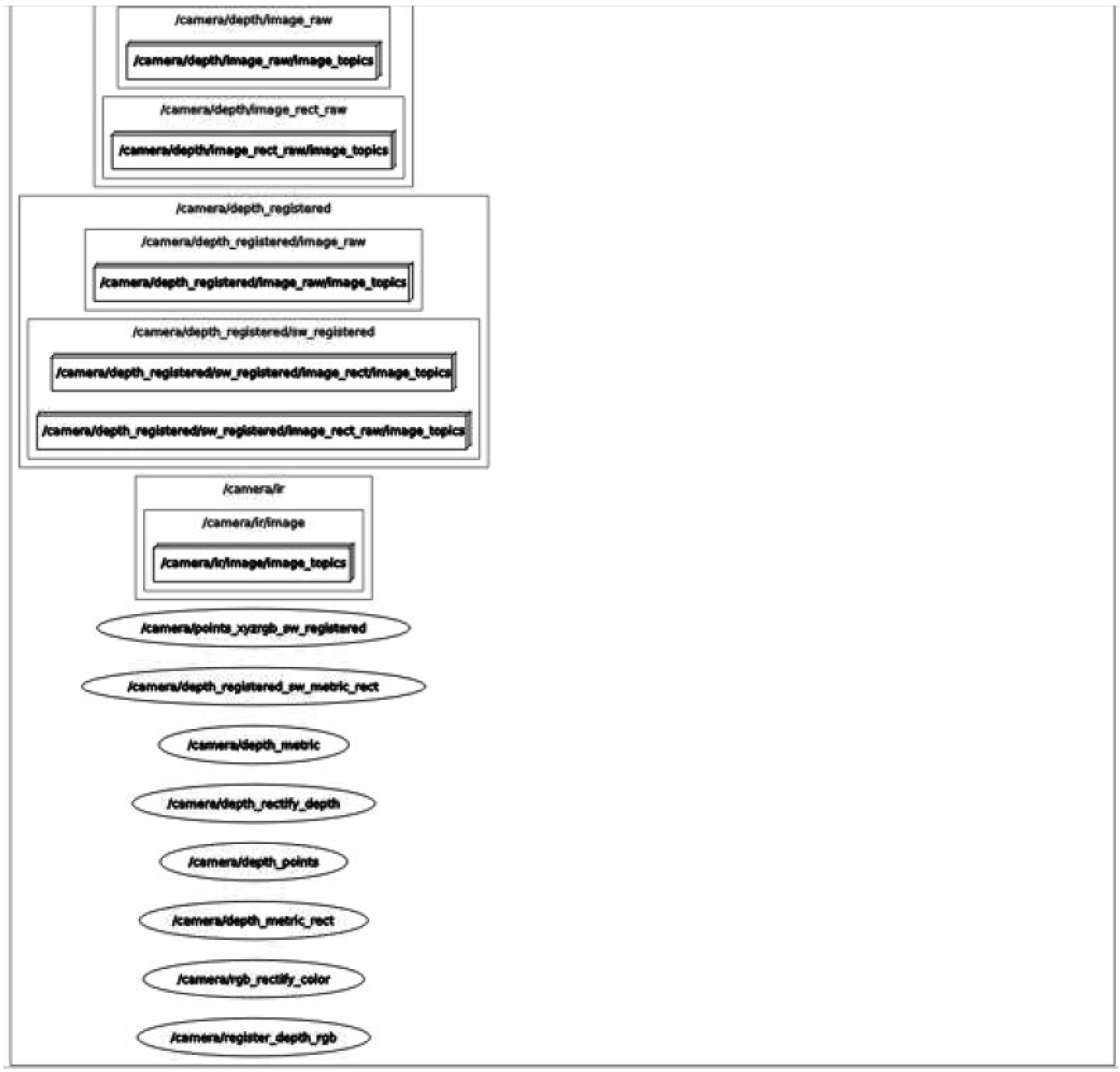

/camera/depth/image_raw
/camera/depth/image_raw/image_topics
/camera/depth/image_rect_raw
/camera/depth/image_rect_raw/image_topics
/camera/depth_registered
/camera/depth_registered/image_raw
/camera/depth_registered/image_raw/image_topics
/camera/depth_registered/sw_registered
/camera/depth_registered/sw_registered/image_rect/image_topics
/camera/depth_registered/sw_registered/image_rect_raw/image_topics
/camera/ir
/camera/ir/image
/camera/ir/image/image_topics
/camera/points_xyzrgb_sw_registered
/camera/depth_registered_sw_metric_rect
/camera/depth_metric
/camera/depth_rectify_depth
/camera/depth_points
/camera/depth_metric_rect
/camera/rgb_rectify_color
/camera/register_depth_rgb

Figure D5: Camera Node Bottom ROS Graph

Appendix E – Testing Location and ROS Map Equivalent

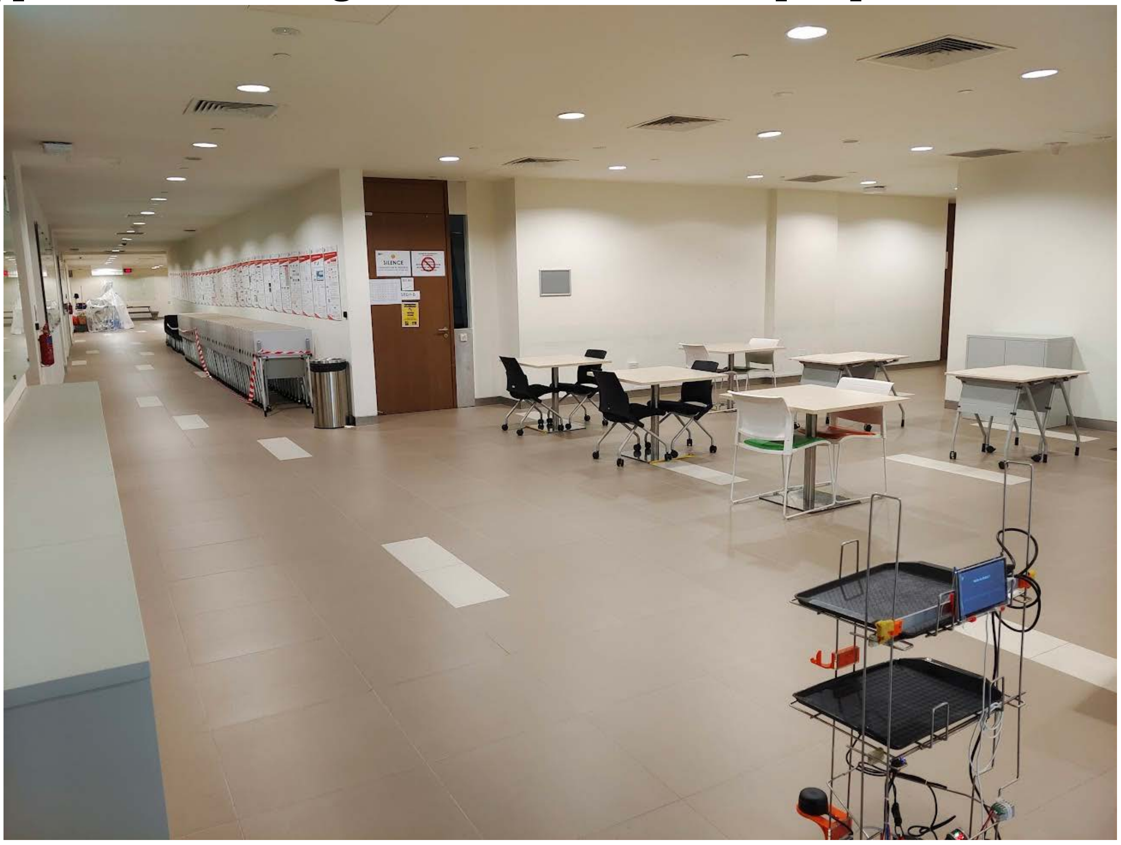


Figure E1: SIT@NYP Area Outside SR6C

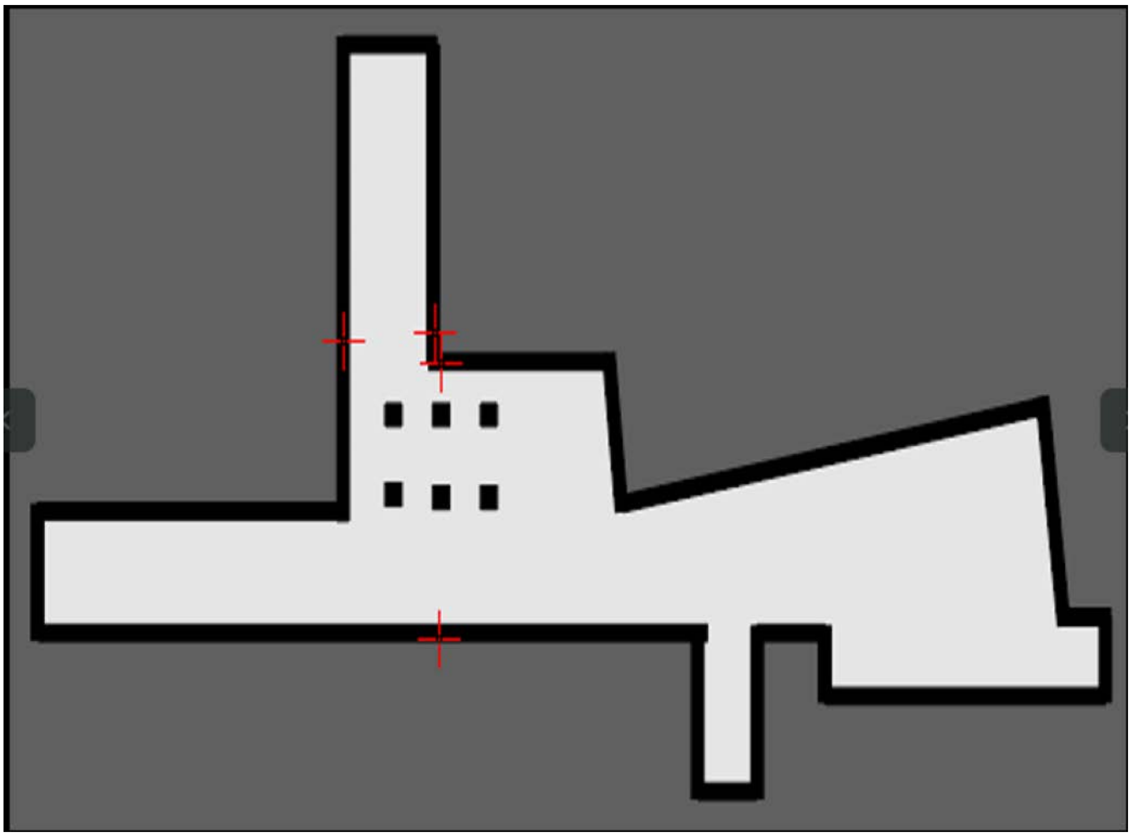


Figure E2: Scaled ROS Map Equivalent to SIT@NYP Area Outside SR6C

Appendix F – Gantt Chart

Activity / Duration (Week)		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Phase 1: Project Preparation																										
Define Project Scope		🟩								🟥	🟥	🟥														
	Aim of the project	🟩								🟥	🟥	🟥														
Project Research		🟩	🟩			🟩	🟩	🟩	🟩	🟥	🟥	🟥	🟩	🟩												
	Learn Progrramming Languages	🟩	🟩			🟩	🟩	🟩	🟩	🟥	🟥	🟥	🟩	🟩												
	Understand Communication Protocol	🟩	🟩			🟩	🟩	🟩	🟩	🟥	🟥	🟥	🟩	🟩												
Project Planning			🟩							🟥	🟥	🟥														
	Project Sequence		🟩							🟥	🟥	🟥														
Literature Review			🟩	🟩	🟩					🟥	🟥	🟥														
	Automation in Service Industry		🟩	🟩	🟩					🟥	🟥	🟥														
	Waiter/Service Robots		🟩	🟩	🟩					🟥	🟥	🟥														
	Demograpics vs Automation		🟩	🟩	🟩					🟥	🟥	🟥														
	RAISA model		🟩	🟩	🟩					🟥	🟥	🟥														
	OSI model		🟩	🟩	🟩					🟥	🟥	🟥														
	Stack Implementation		🟩	🟩	🟩					🟥	🟥	🟥														
Initial Project Plan				🟩	🟩	🟩				🟥	🟥	🟥														
	Documentation			🟩	🟩	🟩				🟥	🟥	🟥														
Phase 2: Develop and Implement the Environment																										
Flutter App and ROS						🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟦	🟦												
	Troubleshoot the Robot					🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟦	🟦												
	Robot Control					🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟦	🟦												
	Broadcasting Robot Info					🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟦	🟦												
	Render Map and Video information					🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟦	🟦												
	State Management					🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟦	🟦												
	Navigation					🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟦	🟦												
	Assignning Task					🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟦	🟦												
Server Setup						🟦	🟦			🟥	🟥	🟥	🟦	🟦												
	Server logic					🟦	🟦			🟥	🟥	🟥	🟦	🟦												
	Database					🟦	🟦			🟥	🟥	🟥	🟦	🟦												
	Async Tasks					🟦	🟦			🟥	🟥	🟥	🟦	🟦												
	Recursive connection to registered robots					🟦	🟦			🟥	🟥	🟥	🟦	🟦												
Applying Network Architechture						🟦	🟦	🟦		🟥	🟥	🟥														
	Establishing Network Protocol					🟦	🟦	🟦		🟥	🟥	🟥														
	Getting Available Ports or IP					🟦	🟦	🟦		🟥	🟥	🟥														
Phase 3: Project Completion																										
Conclusion of Project										🟥	🟥	🟥	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦				
	Testing and Evaluation									🟥	🟥	🟥	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦				
Project Poster										🟥	🟥	🟥										🟦	🟦	🟦		
	Documentation									🟥	🟥	🟥										🟦	🟦	🟦		
Project Final report										🟥	🟥	🟥											🟦	🟦	🟦	🟦
	Documentation									🟥	🟥	🟥											🟦	🟦	🟦	🟦
Project Presentation										🟥	🟥	🟥													🟦	🟦
	Documentation									🟥	🟥	🟥													🟦	🟦

🟥	Exam and Revision

Figure F1: Gantt Chart

Appendix G – Risk Assessment (Development @ NYP R407)

SINGAPORE INSTITUTE OF TECHNOLOGY	**DOCUMENT TITLE** Risk Assessment Form		**Page 1 of 6**
	EFFECTIVE DATE 20190904	**PROGRAMME TITLE** Risk Management Programme	**VERSION** 3.1

Inventory of Work Activities			
Reference Number: (please refer to S&H RA Repository for next running number)		Division	MDME
Title	Waiter		

Ref	Location	Process	Work Activity	Remarks
1	NYP Block R, level 4, room R407	Robot Code Development	Programming of Robot	
2			Simulation of written code	
3			Moving around the Lab	
4		Robot Maintenance	Rewire or beautify the existing electrical circuitry	
5		Robot's Battery Maintenance	Charging of Robot's Battery	
6			Battery Storage	
7		Robot Mobility Testing	Testing for detouring	
8			Testing for docking	
9			Testing robot's response time	
10		Speed Control Testing	Testing robot's braking distance	
11			Testing robot's load-carrying stability	

Note:
1. This form is to be completed before filling in the Risk Assessment Form.
2. The contents of the Process (column) and Work Activity (column) in the Inventory of Work Activities Form are to be copied over to the Process (row) and Work Activity (column), respectively, in the Risk Assessment Form.

Ref	Location	Process	Work Activity	Remarks
12			Testing robot with varying velocity profiles	
13		Robot Testing	Lifting of Robot	

*add more rows if necessary.

** Examples: Contents will be automatically deleted when new content is typed into the rows.

Note:
1. This form is to be completed before filling in the Risk Assessment Form.
2. The contents of the Process (column) and Work Activity (column) in the Inventory of Work Activities Form are to be copied over to the Process (row) and Work Activity (column), respectively, in the Risk Assessment Form.

SINGAPORE INSTITUTE OF TECHNOLOGY	**DOCUMENT TITLE** Risk Assessment Form		**Page 3 of 6**
	EFFECTIVE DATE 20190904	**PROGRAMME TITLE** Risk Management Programme	**VERSION** 3.1

RISK ASSESSMENT

Reference Number				RA Leader:	Serrano Kurt Yves Espenilla	Approved by:	Dr Michael Lau
Title:	Waiter Robot Implementation			RA Team:		Signature:	
						Designation:	
Division:	MDME	Location:	NYP R407				
Last Review Date:		Next Review Date:				Date	

	Hazard Identification			Risk Evaluation				Risk Control						
Ref	Activity	Hazard	Possible injury / ill-health	Existing risk controls	S	L	RPN	Additional controls	S	L	RPN	Implementation Person	Due date	Remarks
				1) Robot Code Development										
1	Programming of Robot	Sitting in front of computer screen for long hours	Neck Aches	Take a short break every 1-2 hours to stretch muscles	2	3	6	N.A.	-	-	-	N.A.	-	-
2	Simulation of written code	Sitting in front of computer screen for long hours	Neck Aches	Take a short break every 1-2 hours to stretch muscles	2	3	6	N.A.	-	-	-	N.A.	-	-
3	Moving around the Lab	Fall/collision accident due to tripping over cables	Bodily injuries	Ensure good housekeeping with no loose cable on the floor	3	2	6	N.A.	-	-	-	N.A.		
				2) Robot Maintenance										
4	Rewire or beautify the existing electrical circuitry	Exposed Wires	Short Circuit	Use heat shrinking tubes or wire cutters as corrective measures	3	3	9	N.A.	-	-	-	N.A.	-	-

Likelihood / Severity	Rare (1)	Remote (2)	Occasional (3)	Frequent (4)	Almost Certain (5)
Catastrophic (5)	5 (M)	10 (M)	15 (H)	20 (H)	25 (H)
Major (4)	4 (M)	8 (M)	12 (M)	16 (H)	20 (H)
Moderate (3)	3 (L)	6 (M)	9 (M)	12 (M)	15 (H)
Minor (2)	2 (L)	4 (M)	6 (M)	8 (M)	10 (M)
Negligible (1)	1 (L)	2 (L)	3 (L)	4 (M)	(M)

	Hazard Identification			Risk Evaluation				Risk Control						
Ref	Activity	Hazard	Possible injury / ill-health	Existing risk controls	S	L	RPN	Additional controls	S	L	RPN	Implementation Person	Due date	Remarks
				3) Robot's Battery Maintenance										
5	Charging of Robot's battery	Overloading	Fire	Charge battery in a Li-Po Guard Bag	4	1	4	Keep a look out on the battery's charge	4	1	4	Kurt	-	-
6	Battery Storage	Fire	Burns	Keep the battery in the Li-Po Guard Bag when not in use. Store battery in an area that does not have direct sunlight.	4	1	4	N.A.	-	-	-	N.A.	-	-
				4) Robot Mobility Testing										
7	Testing for detouring	Obstructed Vision	Collision	Monitor robot closely during operation. Keep a lookout for obstacles.	2	3	6	Look for assistance to look out for blind spots and unexpected robot behaviours	2	1	2	Kurt	-	-
8	Testing for docking	Obstructed Vision	Collision	Monitor robot closely during operation. Keep a lookout for obstacles.	2	3	6	Look for assistance to look out for blind spots and unexpected robot behaviours	2	1	2	Kurt	-	-
9	Testing robot's response time	Obstructed Vision	Collision	Monitor robot closely during operation. Keep a lookout for obstacles.	2	3	6	Look for assistance to look out for blind spots and unexpected robot behaviours	2	1	2	Kurt	-	-
				5) Speed Control Testing										
10	Testing robot's braking distance	Obstructed Vision	Collision	Monitor robot closely during operation. Keep a lookout for obstacles.	2	3	6	Look for assistance to look out for blind spots and unexpected robot behaviours	2	1	2	Kurt	-	-

Likelihood / Severity	Rare (1)	Remote (2)	Occasional (3)	Frequent (4)	Almost Certain (5)
Catastrophic (5)	5 (M)	10 (M)	15 (H)	20 (H)	25 (H)
Major (4)	4 (M)	8 (M)	12 (M)	16 (H)	20 (H)
Moderate (3)	3 (L)	6 (M)	9 (M)	12 (M)	15 (H)
Minor (2)	2 (L)	4 (M)	6 (M)	8 (M)	10 (M)
Negligible (1)	1 (L)	2 (L)	3 (L)	4 (M)	(M)

	Hazard Identification			Risk Evaluation				Risk Control						
Ref	Activity	Hazard	Possible injury / ill-health	Existing risk controls	S	L	RPN	Additional controls	S	L	RPN	Implementation Person	Due date	Remarks
11	Testing robot's load-carrying stability	Obstructed Vision	Collision	Monitor robot closely during operation. Keep a lookout for obstacles.	2	3	6	Look for assistance to look out for blind spots and unexpected robot behaviours	2	1	2	Kurt	-	-
12	Testing robot with varying velocity profiles	Obstructed Vision	Collision	Monitor robot closely during operation. Keep a lookout for obstacles.	2	3	6	Look for assistance to look out for blind spots and unexpected robot behaviours	2	1	2	Kurt	-	-
	6) Robot Testing													
13	Lifting of robot for trolley mounting	Repetitive Motion	Back Strains	Exercise proper lifting posture. Ask for help if required.	3	3	9	N.A.	-	-	-	N.A.	-	-

*add/ delete rows if necessary.

** Examples: Contents will be automatically deleted when new content is typed into the rows.

Likelihood / Severity	Rare (1)	Remote (2)	Occasional (3)	Frequent (4)	Almost Certain (5)
Catastrophic (5)	5 (M)	10 (M)	15 (H)	20 (H)	25 (H)
Major (4)	4 (M)	8 (M)	12 (M)	16 (H)	20 (H)
Moderate (3)	3 (L)	6 (M)	9 (M)	12 (M)	15 (H)
Minor (2)	2 (L)	4 (M)	6 (M)	8 (M)	10 (M)
Negligible (1)	1 (L)	2 (L)	3 (L)	4 (M)	(M)

Level	Severity	Description
1	Negligible	Not likely to cause injury or ill-health.
2	Minor	Injury or ill-health requiring first-aid only (includes minor cuts and bruises, irritation, ill-health with temporary discomfort).
3	Moderate	Injury or ill-health requiring medical treatment (includes lacerations, burns, sprains, minor fractures, dermatitis and work-related upper limb disorders).
4	Major	Serious injuries or life-threatening occupational diseases (includes amputations, major fractures, multiple injuries, occupational cancers, acute poisoning, disabilities and deafness).
5	Catastrophic	Death, fatal diseases or multiple major injuries.

Level	Likelihood	Description
1	Rare	Not expected to occur but still possible.
2	Remote	Not likely to occur under normal circumstances.
3	Occasional	Possible or known to occur.
4	Frequent	Common occurrence.
5	Almost certain	Continual or repeating experience.

Risk score	Acceptability of risk	Recommended actions
Low 1-3	Acceptable	No additional risk control measures may be needed. Frequent review and monitoring of hazards are required to ensure that the risk level assigned is accurate and does not increase over time.
Medium 4-12	Tolerable	A careful evaluation of the hazards should be carried out to ensure that the risk level is reduced to as low as reasonably practicable (ALARP) within a defined time period. Interim risk control measures, such as administrative controls, may be implemented while long term measures are being established.
High 15-25	Not acceptable	High Risk level must be reduced to at least Medium Risk before work commences. There should not be any interim risk control measures and risk control measures should not be overly dependent on personal protective equipment. If practicable, the hazard should be eliminated before work commences. Management review is required before work commences.

Appendix H – Risk Assessment (Development @ Home)

Inventory of Work Activities			
Reference Number: (please refer to S&H RA Repository for next running number)		Division	MDME
Title	Waiter Robot Implementation		

Ref	Location	Process	Work Activity	Remarks
1	Kurt's Home	Robot Code Development	Programming of Robot	
2			Simulation of written code	
3			Moving around the House	

*add more rows if necessary.

** Examples: Contents will be automatically deleted when new content is typed into the rows.

Note:

3. This form is to be completed before filling in the Risk Assessment Form.
4. The contents of the Process (column) and Work Activity (column) in the Inventory of Work Activities Form are to be copied over to the Process (row) and Work Activity (column), respectively, in the Risk Assessment Form.

SINGAPORE INSTITUTE OF TECHNOLOGY	**DOCUMENT TITLE** Risk Assessment Form		**Page 2 of 3**
	EFFECTIVE DATE 20190904	**PROGRAMME TITLE** Risk Management Programme	**VERSION** 3.1

RISK ASSESSMENT

Reference Number				RA Leader:	Serrano Kurt Yves Espenilla	Approved by:	Dr Michael Lau
Title:	Waiter Robot Implementation			RA Team:		Signature:	
						Designation:	
Division:	MDME	Location:	Kurt's Home				
Last Review Date:		Next Review Date:				Date	

	Hazard Identification			Risk Evaluation				Risk Control						
Ref	Activity	Hazard	Possible injury / ill-health	Existing risk controls	S	L	RPN	Additional controls	S	L	RPN	Implementation Person	Due date	Remarks
				7) Robot Code Development										
1	Programming of Robot	Sitting in front of computer screen for long hours	Neck Aches	Take a short break every 1-2 hours to stretch muscles	2	3	6	N.A.	-	-	-	N.A.	-	-
2	Simulation of written code	Sitting in front of computer screen for long hours	Neck Aches	Take a short break every 1-2 hours to stretch muscles	2	3	6	N.A.	-	-	-	N.A.	-	-
3	Moving around the House	Fall/collision accident due to tripping over cables	Bodily injuries	Ensure good housekeeping with no loose cable on the floor	3	2	6	N.A.	-	-	-	N.A.		

*add/ delete rows if necessary.

** Examples: Contents will be automatically deleted when new content is typed into the rows.

Likelihood / Severity	Rare (1)	Remote (2)	Occasional (3)	Frequent (4)	Almost Certain (5)
Catastrophic (5)	5 (M)	10 (M)	15 (H)	20 (H)	25 (H)
Major (4)	4 (M)	8 (M)	12 (M)	16 (H)	20 (H)
Moderate (3)	3 (L)	6 (M)	9 (M)	12 (M)	15 (H)
Minor (2)	2 (L)	4 (M)	6 (M)	8 (M)	10 (M)
Negligible (1)	1 (L)	2 (L)	3 (L)	4 (M)	(M)

Level	Severity	Description
1	Negligible	Not likely to cause injury or ill-health.
2	Minor	Injury or ill-health requiring first-aid only (includes minor cuts and bruises, irritation, ill-health with temporary discomfort).
3	Moderate	Injury or ill-health requiring medical treatment (includes lacerations, burns, sprains, minor fractures, dermatitis and work-related upper limb disorders).
4	Major	Serious injuries or life-threatening occupational diseases (includes amputations, major fractures, multiple injuries, occupational cancers, acute poisoning, disabilities and deafness).
5	Catastrophic	Death, fatal diseases or multiple major injuries.

Level	Likelihood	Description
1	Rare	Not expected to occur but still possible.
2	Remote	Not likely to occur under normal circumstances.
3	Occasional	Possible or known to occur.
4	Frequent	Common occurrence.
5	Almost certain	Continual or repeating experience.

Risk score	Acceptability of risk	Recommended actions
Low 1-3	Acceptable	No additional risk control measures may be needed. Frequent review and monitoring of hazards are required to ensure that the risk level assigned is accurate and does not increase over time.
Medium 4-12	Tolerable	A careful evaluation of the hazards should be carried out to ensure that the risk level is reduced to as low as reasonably practicable (ALARP) within a defined time period. Interim risk control measures, such as administrative controls, may be implemented while long term measures are being established.
High 15-25	Not acceptable	High Risk level must be reduced to at least Medium Risk before work commences. There should not be any interim risk control measures and risk control measures should not be overly dependent on personal protective equipment. If practicable, the hazard should be eliminated before work commences. Management review is required before work commences.

Appendix I – Risk Assessment (Testing @ SIT@NYP Area Outside SR6C)

Inventory of Work Activities			
Reference Number: (please refer to S&H RA Repository for next running number)		Division	MDME
Title	Waiter		

Ref	Location	Process	Work Activity	Remarks
1	SIT@NYP Area Outside SR6C	Robot Code Development	Programming of Robot	
2			Simulation of written code	
3			Moving around the Lab	
4		Robot Maintenance	Rewire or beautify the existing electrical circuitry	
5		Robot's Battery Maintenance	Charging of Robot's Battery	
6			Battery Storage	
7		Robot Mobility Testing	Testing for detouring	
8			Testing for docking	
9			Testing robot's response time	
10		Speed Control Testing	Testing robot's braking distance	
11			Testing robot's load-carrying stability	

Note:

5. This form is to be completed before filling in the Risk Assessment Form.
6. The contents of the Process (column) and Work Activity (column) in the Inventory of Work Activities Form are to be copied over to the Process (row) and Work Activity (column), respectively, in the Risk Assessment Form.

Ref	Location	Process	Work Activity	Remarks
12			Testing robot with varying velocity profiles	
13		Robot Testing	Lifting of Robot	

*add more rows if necessary.

** Examples: Contents will be automatically deleted when new content is typed into the rows.

Note:

5. This form is to be completed before filling in the Risk Assessment Form.
6. The contents of the Process (column) and Work Activity (column) in the Inventory of Work Activities Form are to be copied over to the Process (row) and Work Activity (column), respectively, in the Risk Assessment Form.

RISK ASSESSMENT

Reference Number		RA Leader:	Serrano Kurt Yves Espenilla	Approved by:	Dr Michael Lau	
Title:	Waiter Robot Implementation	RA Team:		Signature:		
				Designation:		
Division: MDME	Location: SIT@NYP Area Outside SR6C					
Last Review Date:	Next Review Date:			Date		

	Hazard Identification			Risk Evaluation				Risk Control						
Ref	Activity	Hazard	Possible injury / ill-health	Existing risk controls	S	L	RPN	Additional controls	S	L	RPN	Implementation Person	Due date	Remarks
				8) Robot Code Development										
1	Programming of Robot	Sitting in front of computer screen for long hours	Neck Aches	Take a short break every 1-2 hours to stretch muscles	2	3	6	N.A.	-	-	-	N.A.	-	-
2	Simulation of written code	Sitting in front of computer screen for long hours	Neck Aches	Take a short break every 1-2 hours to stretch muscles	2	3	6	N.A.	-	-	-	N.A.	-	-
3	Moving around the Lab	Fall/collision accident due to tripping over cables	Bodily injuries	Ensure good housekeeping with no loose cable on the floor	3	2	6	N.A.	-	-	-	N.A.		
				9) Robot Maintenance										

Likelihood / Severity	Rare (1)	Remote (2)	Occasional (3)	Frequent (4)	Almost Certain (5)
Catastrophic (5)	5 (M)	10 (M)	15 (H)	20 (H)	25 (H)
Major (4)	4 (M)	8 (M)	12 (M)	16 (H)	20 (H)
Moderate (3)	3 (L)	6 (M)	9 (M)	12 (M)	15 (H)
Minor (2)	2 (L)	4 (M)	6 (M)	8 (M)	10 (M)
Negligible (1)	1 (L)	2 (L)	3 (L)	4 (M)	(M)

	Hazard Identification			Risk Evaluation				Risk Control						
Ref	Activity	Hazard	Possible injury / ill-health	Existing risk controls	S	L	RPN	Additional controls	S	L	RPN	Implementation Person	Due date	Remarks
4	Rewire or beautify the existing electrical circuitry	Exposed Wires	Short Circuit	Use heat shrinking tubes or wire cutters as corrective measures	3	3	9	N.A.	-	-	-	N.A.	-	-
10) Robot's Battery Maintenance														
5	Charging of Robot's battery	Overloading	Fire	Charge battery in a Li-Po Guard Bag	4	1	4	Keep a look out on the battery's charge	4	1	4	Kurt	-	-
6	Battery Storage	Fire	Burns	Keep the battery in the Li-Po Guard Bag when not in use. Store battery in an area that does not have direct sunlight.	4	1	4	N.A.	-	-	-	N.A.	-	-
11) Robot Mobility Testing														
7	Testing for detouring	Obstructed Vision	Collision	Monitor robot closely during operation. Keep a lookout for obstacles.	2	3	6	Look for assistance to look out for blind spots and unexpected robot behaviours	2	1	2	Kurt	-	-
8	Testing for docking	Obstructed Vision	Collision	Monitor robot closely during operation. Keep a lookout for obstacles.	2	3	6	Look for assistance to look out for blind spots and unexpected robot behaviours	2	1	2	Kurt	-	-
9	Testing robot's response time	Obstructed Vision	Collision	Monitor robot closely during operation. Keep a lookout for obstacles.	2	3	6	Look for assistance to look out for blind spots and unexpected robot behaviours	2	1	2	Kurt	-	-
12) Speed Control Testing														

Likelihood / Severity	Rare (1)	Remote (2)	Occasional (3)	Frequent (4)	Almost Certain (5)
Catastrophic (5)	5 (M)	10 (M)	15 (H)	20 (H)	25 (H)
Major (4)	4 (M)	8 (M)	12 (M)	16 (H)	20 (H)
Moderate (3)	3 (L)	6 (M)	9 (M)	12 (M)	15 (H)
Minor (2)	2 (L)	4 (M)	6 (M)	8 (M)	10 (M)
Negligible (1)	1 (L)	2 (L)	3 (L)	4 (M)	(M)

	Hazard Identification			Risk Evaluation				Risk Control						
Ref	Activity	Hazard	Possible injury / ill-health	Existing risk controls	S	L	RPN	Additional controls	S	L	RPN	Implementation Person	Due date	Remarks
10	Testing robot's braking distance	Obstructed Vision	Collision	Monitor robot closely during operation. Keep a lookout for obstacles.	2	3	6	Look for assistance to look out for blind spots and unexpected robot behaviours	2	1	2	Kurt	-	-
11	Testing robot's load-carrying stability	Obstructed Vision	Collision	Monitor robot closely during operation. Keep a lookout for obstacles.	2	3	6	Look for assistance to look out for blind spots and unexpected robot behaviours	2	1	2	Kurt	-	-
12	Testing robot with varying velocity profiles	Obstructed Vision	Collision	Monitor robot closely during operation. Keep a lookout for obstacles.	2	3	6	Look for assistance to look out for blind spots and unexpected robot behaviours	2	1	2	Kurt	-	-
				13) Robot Testing										
13	Lifting of robot for trolley mounting	Repetitive Motion	Back Strains	Exercise proper lifting posture. Ask for help if required.	3	3	9	N.A.	-	-	-	N.A.	-	-

*add/ delete rows if necessary.

** Examples: Contents will be automatically deleted when new content is typed into the rows.

Likelihood / Severity	Rare (1)	Remote (2)	Occasional (3)	Frequent (4)	Almost Certain (5)
Catastrophic (5)	5 (M)	10 (M)	15 (H)	20 (H)	25 (H)
Major (4)	4 (M)	8 (M)	12 (M)	16 (H)	20 (H)
Moderate (3)	3 (L)	6 (M)	9 (M)	12 (M)	15 (H)
Minor (2)	2 (L)	4 (M)	6 (M)	8 (M)	10 (M)
Negligible (1)	1 (L)	2 (L)	3 (L)	4 (M)	(M)

Level	Severity	Description
1	Negligible	Not likely to cause injury or ill-health.
2	Minor	Injury or ill-health requiring first-aid only (includes minor cuts and bruises, irritation, ill-health with temporary discomfort).
3	Moderate	Injury or ill-health requiring medical treatment (includes lacerations, burns, sprains, minor fractures, dermatitis and work-related upper limb disorders).
4	Major	Serious injuries or life-threatening occupational diseases (includes amputations, major fractures, multiple injuries, occupational cancers, acute poisoning, disabilities and deafness).
5	Catastrophic	Death, fatal diseases or multiple major injuries.

Level	Likelihood	Description
1	Rare	Not expected to occur but still possible.
2	Remote	Not likely to occur under normal circumstances.
3	Occasional	Possible or known to occur.
4	Frequent	Common occurrence.
5	Almost certain	Continual or repeating experience.

Risk score	Acceptability of risk	Recommended actions
Low 1-3	Acceptable	No additional risk control measures may be needed. Frequent review and monitoring of hazards are required to ensure that the risk level assigned is accurate and does not increase over time.
Medium 4-12	Tolerable	A careful evaluation of the hazards should be carried out to ensure that the risk level is reduced to as low as reasonably practicable (ALARP) within a defined time period. Interim risk control measures, such as administrative controls, may be implemented while long term measures are being established.
High 15-25	Not acceptable	High Risk level must be reduced to at least Medium Risk before work commences. There should not be any interim risk control measures and risk control measures should not be overly dependent on personal protective equipment. If practicable, the hazard should be eliminated before work commences. Management review is required before work commences.