

**ROBOT SURVEILLANCE FOR CONNECTED HOME INTEGRATION**

**BY**

**YUEN KOK HEI**

**A REPORT**

**SUBMITTED TO**

**Universiti Tunku Abdul Rahman**

**in partial fulfillment of the requirements**

**for the degree of**

**BACHELOR OF INFORMATION TECHNOLOGY (HONOURS) COMPUTER**

**ENGINEERING**

**Faculty of Information and Communication Technology**

**(Kampar Campus)**

**JAN 2024**

## REPORT STATUS DECLARATION FORM

**Title:** Robot Surveillance for Connected Home Integration  
\_\_\_\_\_  
\_\_\_\_\_

**Academic Session:** 202401

I YUEN KOK HEI  
**(CAPITAL LETTER)**

declare that I allow this Final Year Project Report to be kept in  
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

*Hui*

\_\_\_\_\_  
(Author's signature)

Verified by,

*Kheng*

\_\_\_\_\_  
(Supervisor's signature)

**Address:**

33, JALAN VILLA PERMAI,  
TAMAN VILLA PERMAI  
BUTTERWORTH, PNG

Dr. Teoh Shen Khang

Supervisor's name

**Date:** 25 APRIL 2024

**Date:** 26 April 2024

<b>Universiti Tunku Abdul Rahman</b>			
Form Title : <b>Sample of Submission Sheet for FYP/Dissertation/Thesis</b>			
Form Number: <b>FM-IAD-004</b>	Rev No.: <b>0</b>	Effective Date: <b>21 JUNE 2011</b>	Page No.: <b>1 of 1</b>

**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

**UNIVERSITI TUNKU ABDUL RAHMAN**

Date: 25 APRIL 2024

**SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS**

It is hereby certified that YUEN KOK HEI (ID No: 20ACB02830 ) has completed this final year project entitled “ Robot Surveillance for Connected Home Integration ” under the supervision of DR TEOH SHEN KHANG (Supervisor) from the Department of Computer and Communication Technology, Faculty of Information and Communication Technology , and DR NORLIANA BINTI MUSLIM (Moderator) from the Department of Computer and Communication Technology, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.


Yours truly,

YUEN KOK HEI  
(*Student Name*)

\*Delete whichever not applicable

## DECLARATION OF ORIGINALITY

I declare that this report entitled “**ROBOT SURVEILLANCE FOR CONNECTED HOME INTEGRATION**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature :  \_\_\_\_\_

Name : \_YUEN KOK HEI \_\_\_\_\_

Date : \_25 April 2024\_\_\_\_\_

## **ACKNOWLEDGEMENTS**

I would like to express thanks and appreciation to my supervisor, Dr. Teoh Shen Khang and my moderator, Dr. Norliana Binti Muslim who have given me this bright opportunity to engage in a Robotic project. Besides that, they have given me a lot of guidance in order to complete this project. When I was facing problems in this project, the advice from them always assists me in overcoming the problems. Again, a million thanks to my supervisor and moderator.

Other than that, I would like to thank my friends and family, for their unwavering love, support, and continuous encouragement throughout this journey.

## **ABSTRACT**

Rapid technological advancements have led in the era of the connected home, where the integration of sensors and smart devices has provided innovative solutions to enhance safety and convenience. In response to this trend, the project "Robot Surveillance for Connected Home Integration" proposed an inventive approach to home surveillance. The project utilized a TurtleBot3 prototype robot programmed to autonomously navigate to specific locations within the home, triggered by sensor inputs. Upon arrival at the sensor-triggered area, the robot captured and transmitted a photo of the scene to the homeowner's mobile device. Leveraging the versatility and customization capabilities of the TurtleBot3 robotics platform, this project aimed to create an intelligent surveillance system that combined automation, real-time communication, and mobility. The robot was equipped with various sensors, a camera, and a Raspberry Pi for communication and processing. A central control system managed the robot's movements, allowing it to respond to sensor triggers, compute suitable navigation paths, and avoid obstacles seamlessly. Core functionalities of the system included real-time sensor monitoring, route planning, obstacle detection, image capturing, and remote communication with the homeowner's smartphone via the Telegram App. This report provided a comprehensive overview of the project, including the study of ROS2 (Robot Operating System 2), the TurtleBot3 platform, simultaneous localization and mapping (SLAM), autonomous navigation (Navigation2), camera setup, sensor transmissions, Telegram app integration, and graphical user interface design. Additionally, it presented experimental results and performance evaluations to assess the system's effectiveness in real-world circumstances. The project's key contribution was to provide homeowners with a cost-effective remote smart monitoring solution that smoothly fit into existing connected home ecosystems. The "Robot Surveillance for Connected Home Integration" project combined robotics and smart technology, making homes safer, smarter, and more connected. The system highlighted how robotics could improve home security and automation by leveraging TurtleBot3's mobility, sensors, and real-time communications. This project served as a foundational framework for future developments in the field of connected home surveillance, with potential applications in various household and commercial sectors.

# TABLE OF CONTENTS

<b>TITLE PAGE</b>	<b>i</b>
<b>REPORT STATUS DECLARATION FORM</b>	<b>ii</b>
<b>FYP THESIS SUBMISSION FORM</b>	<b>iii</b>
<b>DECLARATION OF ORIGINALITY</b>	<b>iv</b>
<b>ACKNOWLEDGEMENTS</b>	<b>v</b>
<b>ABSTRACT</b>	<b>vi</b>
<b>TABLE OF CONTENTS</b>	<b>vii</b>
<b>LIST OF FIGURES</b>	<b>x</b>
<b>LIST OF TABLES</b>	<b>xi</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xii</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 Problem Statement and Motivation	1
1.2 Objectives	2
1.3 Project Scope and Direction	4
1.4 Contributions	5
1.5 Report Organization	6
<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>8</b>
2.1 Modelling a TurtleBot3 Based Delivery System for a Smart Hospital in Gazebo	8
<b>CHAPTER 3 SYSTEM METHODOLOGY/APPROACH</b>	<b>12</b>
3.1 System Design Diagram	12
3.1.1 System Architecture Diagram	20
3.1.2 Use Case Diagram and Description	22
3.1.3 Activity Diagram	24
<b>CHAPTER 4 SYSTEM DESIGN</b>	<b>25</b>
4.1 System Block Diagram	25

4.2	System Components Specifications	27
4.3	Circuits and Components Design	31
4.4	System Components Interaction Operations	34
<b>CHAPTER 5 SYSTEM IMPLEMENTATION</b>		<b>36</b>
5.1	Hardware Setup	36
5.2	Software Setup	40
5.3	Setting and Configuration	42
5.4	System Operation (with Screenshot)	45
5.5	Concluding Remark	53
<b>CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION</b>		<b>55</b>
6.1	Testing Setup and Result	55
6.2	Project Challenges	65
6.3	Objectives Evaluation	66
6.4	Concluding Remark	68
<b>CHAPTER 7 CONCLUSION AND RECOMMENDATION</b>		<b>69</b>
7.1	Conclusion	69
7.2	Recommendation	69
<b>REFERENCES</b>		<b>71</b>
<b>WEEKLY LOG</b>		<b>75</b>
<b>POSTER</b>		<b>81</b>
<b>PLAGIARISM CHECK RESULT</b>		<b>82</b>
<b>FYP2 CHECKLIST</b>		<b>86</b>



# LIST OF FIGURES

<b>Figure Number</b>	<b>Title</b>	<b>Page</b>
Figure 2.1	Current state machine structure	9
Figure 2.1.2	Application of Block Diagram	10
Figure 3.1	Prototyping Model	12
Figure 3.1.1.1	Hardware Architecture Diagram	20
Figure 3.1.1.2	Software Architecture Diagram	21
Figure 3.1.2	Use Case Diagram	22
Figure 3.1.3	Activity Diagram	24
Figure 4.1.1	System Architecture Diagram	25
Figure 4.3.1	Sensor 1 Setup	31
Figure 4.3.2	Sensor 2 Setup	32
Figure 4.3.3	Sensor 3 Setup	33
Figure 4.3.4	Camera Module Setup	34
Figure 5.1.1	Turtlebot3 Burger Model	36
Figure 5.1.2	Open CR Board	36
Figure 5.1.3	Raspberry Pi Model 3B+	37
Figure 5.1.4	DHT22 Sensor Module	38
Figure 5.1.5	MPU6050	38
Figure 5.1.6	LDR Module	39
Figure 5.1.7	Pi Camera Module	40
Figure 5.2.1	Installation of ROS2 Package	41
Figure 5.2.2	Installation of Dependent Gazebo	41
Figure 5.2.3	Installation of Dependent Cartographer	41
Figure 5.2.4	Installation of Dependent Navigation2	41
Figure 5.2.5	Installation of Turtlebot3	41
Figure 5.2.6	Setting up ROS Environment	41
Figure 5.3.1	Configuration of SBC	42
Figure 5.3.2	Command to configure WIFI Network Setting	42
Figure 5.3.3	Editing WIFI SSID and Password	43

Figure 5.3.4	Installing and updating the LDS02 Driver and Turtlebot3 Package	43
Figure 5.3.5	Exporting LDS MODEL to bashrc file.	43
Figure 5.3.6	Installing Packages on Raspberry Pi.	44
Figure 5.3.7	Setting up OpenCR Name.	44
Figure 5.3.8	Downloading the firmware and loader.	44
Figure 5.3.9	Uploading the firmware to OpenCR	44
Figure 5.4.1	SSH to Ubuntu (IP address of Raspberry Pi)	45
Figure 5.4.2	Successfully Bring up Turtlebot3.	46
Figure 5.4.3	Teleop_Keyboard to Control Turtlebot3	47
Figure 5.4.4	Command to launch SLAM Cartographer	48
Figure 5.4.5	Running SLAM Cartographer	48
Figure 5.4.6	Map.yaml	49
Figure 5.4.7	Command to launch Navigation2.	50
Figure 5.4.8	RVIZ Window	50
Figure 5.4.9	Estimating Initial Pose	51
Figure 5.4.10	LDS sensor data overlays on the map.	52
Figure 5.4.11	Set the destination of Turtlebot3	52
Figure 5.4.12	Turtlebot3 Navigation on RIVZ	53
Figure 6.1.1	Testing for Sensor Data Collection	56
Figure 6.1.2	Testing for MQTT Integration	57
Figure 6.1.3	Test Run Camera Node	58
Figure 6.1.4	Checking ROS Topic List	58
Figure 6.1.5	Subscribing to 'image_raw' Topic and View Image	59
Figure 6.1.6	Launching the Navigation2 with Existing Map	60
Figure 6.1.7	Successfully Navigated to Three Locations by Running Python Script	60
Figure 6.1.8	Displaying Sensor Data On GUI	61
Figure 6.1.9	Showing Sensor Data on Telegram App	62
Figure 6.1.10	Viewing Image on GUI by Clicking "View Image" Button	62
Figure 6.1.11	Viewing Image on Telegram	63
Figure 6.1.12	Reset Sensors Using Telegram Command	64
Figure 6.1.13	Successfully Reset Sensors	64

## LIST OF TABLES

<b>Table Number</b>	<b>Title</b>	<b>Page</b>
Table 2.1.1	TASK OBJECT FIELDS DESCRIPTION	9
Table 2.1.2	ROBOT BEHAVIOUR STATES DESCRIPTION	9
Table 4.2.1	Specifications of Robot	29
Table 4.2.2	Specifications of OpenCR	30
Table 4.2.3	Specifications of Raspberry Pi	31

## LIST OF ABBREVIATIONS

<i>CCTV</i>	Closed-Circuit Television
<i>ROS</i>	Robot Operating System
<i>LiDAR</i>	Light Detection and Ranging
<i>LDS</i>	Laser Distance Sensor
<i>GUI</i>	Graphical User Interface
<i>SLAM</i>	Simultaneous Localization and Mapping
<i>RVIZ</i>	ROS Visualizer
<i>TEB</i>	Timed Elastic Band
<i>AMCL</i>	Adaptive Monte Carlo Localization
<i>DBSCAN</i>	Density-Based Spatial Clustering of Application with Noise
<i>SBC</i>	Single Board Computer
<i>Open CR</i>	Open-source Control Module for ROS
<i>CPU</i>	Central Processing Unit
<i>GPIO</i>	General Purpose Input Output
<i>RAM</i>	Random Access Memory
<i>LED</i>	Light-Emitting diode
<i>HDMI</i>	High-Definition Multimedia Interface
<i>USB</i>	Universal Serial Bus
<i>MQTT</i>	Message Queuing Telemetry Transport
<i>DHT</i>	Digital Humidity and Temperature
<i>VCC</i>	Voltage Common Collector
<i>GND</i>	Ground
<i>SCL</i>	Serial Clock Line
<i>SDA</i>	Serial Data Line
<i>CSI</i>	Camera Serial Interface
<i>LDR</i>	Light Dependent Resistor
<i>IMU</i>	Inertial Measurement Unit
<i>MPU</i>	Motion Processing Unit.

# Chapter 1

## Introduction

This chapter serves as the foundation of the report, offering a comprehensive overview of the project's context, objectives, scope, contributions, and structure. Each subsection within this chapter plays a crucial role in providing readers with the necessary background to fully understand the project's importance and direction. By delving into the problem statement and motivation, determining clear objectives, defining the project's scope and direction, highlighting its contributions, and detailing the report's organization, this chapter ensures a seamless transition into the subsequent sections, facilitating a deeper understanding and engagement with the project's content.

### 1.1 Problem Statement and Motivation

In recent years, the rapid rise of smart electronic devices and the Internet of Things (IoT) has ushered in the era of the connected home, offering homeowners extraordinary levels of convenience, comfort, and efficiency. However, within this technological transformation, one critical issue remains largely overlooked: home surveillance. Homeowners relying on traditional surveillance systems may encounter increasing risks of security breaches, intrusions, and property damage. These vulnerabilities stem from the inherent limitations of conventional systems, which are often static, small-scale, and operate independently of the broader ecosystem of smart home devices. Consequently, they lack the versatility and comprehensive coverage necessary to effectively safeguard the home environment.

The problem of home surveillance within a connected home is multifaceted. Firstly, current systems fail to accommodate the dynamic and ever-changing nature of modern households. Fixed cameras and motion detectors may be sufficient in certain areas but lack the intellectual capacity and flexibility to provide comprehensive home surveillance. Additionally, most of these systems operate in isolation, disconnected from other smart devices and sensors that could enhance their performance and responsiveness to homeowners' needs.

Secondly, increasing concerns about data privacy and security make it more difficult to adopt traditional surveillance systems. Many residents are uncomfortable with the notion of living in a home under constant surveillance, fearing the potential disclosure or misuse of their sensitive information. Thus, there is an urgent need for alternative surveillance solutions that afford residents the desired level of security while respecting their autonomy and privacy.

In response to these challenges, the project "Robot Surveillance for Connected Home Integration" proposes an innovative solution based on robotics to revolutionize home surveillance in connected environments. The project aims to address the prevailing inadequacies of home security by replacing outdated, immobile camera systems with more flexible and cost-effective surveillance robots.

This project seeks to develop a mobile, intelligent, and responsive surveillance system that harnesses robotics and automation to enhance safety within smart homes. For example, envision receiving a notification on your smartphone indicating that the motion detector in the living room has detected unusual activity. With this system, users can effortlessly access real-time images captured by a TurtleBot3 robot with just a few taps, irrespective of their location, providing tangible visual confirmation and valuable insights.

This advancement in home surveillance reflects the dynamic nature of our contemporary world, where change is constant, accessibility is paramount, and innovation is embraced. It embodies concepts such as the dynamic nature of modern life, ease of access, and creativity, underscoring the adaptability and ingenuity of technology in addressing evolving safety needs.

To bring this vision to fruition, the project will undertake a comprehensive analysis of every aspect of the proposed system, encompassing its execution, methodology, and evaluation. By harnessing technologies such as Navigation2 and the ROS2 operating system, the project aims to redefine the fundamental concept of home surveillance in the connected home era. Ultimately, the integration of robotics, real-time sensors, and seamless communication sets the stage for safer, smarter, and more secure homes.

## **1.2 Objectives**

The "Robot Surveillance for Connected Home Integration" project defined a wide range of strategic goals meticulously developed to enable the creation and implementation of a responsive and intelligent surveillance system specifically designed for the dynamic and networked environment of the modern connected home. These objectives served as the foundation of the project, reflecting a commitment to revolutionizing home security and situational awareness:

1. Develop and implement the robotic system: At the core of the project was an ambition to create a practical and versatile surveillance system based on the adaptable TurtleBot3 platform. The primary goal was to design, build, and operate a robotic

system capable of autonomously navigating the constantly evolving environment of the household when triggered by sensor inputs. The system's efficiency was based on its ability to seamlessly adapt to the dynamic nature of a home environment while providing precise and reliable real-time surveillance functionalities.

2. Achieve autonomous navigation: The overall vision included autonomy as a key component in the development of a robust path planning algorithm and obstacle avoidance mechanism for TurtleBot3 robot to move freely within the entire house area. In order to reach this vision, the robot had to be able to avoid obstacles, find optimal paths, and react quickly to sensor information, thus making sure that the robot became an integral part of the connected home.
3. Integrate real-time sensor data: For the project to be successful, it was crucial that a variety of sensors were accurately placed regarding their positions so as to watch and respond actively to change in environmental patterns at home. Strategically positioned sensors provided data that enabled the home system to detect and react to events in real-time, enhancing its understanding of its precise location. The sensor network was tailored for detecting motion and monitoring, for example, door or window statuses, boasting unparalleled sensitivity and effectiveness in surveillance.
4. Image capture and transmission: Setting up reliable mechanisms to capture real-time images at specific locations within the connected home was essential for this project. These captured images served as crucial records of events, enabling homeowners to evaluate situations and determine suitable actions. This feature enabled rapid responses, safeguarding against unforeseen threats within the familiar surroundings of the user's home.
5. Integrate communication with user's smartphone: Integrating communication with the user's smartphone via Telegram involved enabling the surveillance system to send notifications, real-time updates, photos, and commands to the homeowner's smartphone using the Telegram messaging app. When the surveillance system detected a security event or triggered a specific action, such as capturing an image or encountering an obstacle, it sent a notification to the homeowner's smartphone via the Telegram app. This notification provided real-time information about the event, allowing the homeowner to respond promptly. The integration with Telegram enabled two-way communication between the surveillance system and the homeowner's smartphone. This meant that not only did the surveillance system send notifications

and updates to the homeowner, but the homeowner could also interact with the system by sending commands or requesting information directly through the Telegram app.

### **1.3 Project Scope and Direction**

The scope of the project 'Robot Surveillance for Connected Home Integration' encompassed the development of an innovative surveillance system that leveraged robotics, sensors, and real-time communication to enhance home security and convenience within connected home environments. The project focused on designing, implementing, and evaluating a prototype surveillance system using the TurtleBot3 robot platform as the primary hardware component. The project's scope included several key components. Firstly, hardware integration involves merging various components such as sensors like motion detectors, temperature and humidity sensors, light sensors, a camera module, and a few Raspberry Pis for communication and processing into the TurtleBot3 robot platform. These components served as the foundation for the surveillance system, enabling it to detect security events, capture images, and communicate with the homeowner's smartphone.

Software development is another critical aspect, covering the implementation of algorithms and software modules for autonomous navigation, sensor data processing, real-time communication, and user interface design. This result in developing algorithms for obstacle avoidance, route planning, image processing, and integration with the Telegram messaging app for homeowner communication.

Furthermore, the project included integration with ROS2 and Navigation2, which involved exploring and leveraging their capabilities for robot control, navigation, and localization within indoor environments. This integration enables the TurtleBot3 robot to autonomously navigate to specific locations triggered by sensor inputs and effectively avoid obstacles in its path.

User interface design was crucial for providing homeowners with intuitive control and monitoring capabilities for the surveillance system. A graphical user interface (GUI) will be developed to allow users to access live camera feeds, receive notifications, and send commands to the robot remotely via the Telegram messaging app.

Moreover, extensive testing and evaluation were conducted to assess the performance, reliability, and effectiveness of the surveillance system in real-world scenarios. This includes evaluating factors such as responsiveness, accuracy, user-friendliness, and conducting field tests to validate its functionality within various home environments.



The project's direction was focused on creating an affordable, adaptable, and user-friendly surveillance solution that seamlessly integrated with existing connected home systems. By leveraging robotics, sensors, and real-time communication, the goal is to overcome the limitations of traditional static surveillance setups, offering homeowners enhanced security, convenience, and peace of mind.

Key objectives included developing an intelligent surveillance system that combined robotics, sensors, and smart devices. Algorithms for autonomous navigation and obstacle avoidance were implemented specifically for the TurtleBot3 robot. Furthermore, the project aimed to integrate sensor inputs, real-time communication, and user interface design to provide a cohesive user experience.

Another important objective was to enable remote monitoring and control through the Telegram messaging app, allowing homeowners to stay connected and in control from anywhere. The effectiveness and performance of the surveillance system were rigorously evaluated in real-world scenarios to ensure its reliability and suitability for various environments. In summary, the project seeks to establish a foundation for future advancements in connected home surveillance, with potential applications spanning both residential and commercial settings.

#### **1.4 Contributions**

The project presented an innovative surveillance system that used robotics, sensors, and real-time communication to improve home security and convenience in connected home environments. By combining these technologies, the project provided an innovative approach to home surveillance that addressed the shortcomings of traditional static surveillance systems.

The project advanced the field of robotics by developing algorithms and techniques for autonomous navigation and obstacle avoidance in indoor environments. Enabling the TurtleBot3 robot to navigate autonomously to specific locations triggered by sensor inputs while avoiding obstacles in its path represented a significant advancement in mobile robotics. Moreover, the project contributed to robotics software development by studying and leveraging the capabilities of ROS2 (Robot Operating System 2) and Navigation2 for robot control, navigation, and localization. Integrating with these frameworks demonstrated the potential for connectivity as well as communication among those involved with robotics.

The project also helped to advance real-time communication and remote monitoring by integrating with the Telegram messaging app. This allowed homeowners to receive notifications, view live real-time photos, and control their surveillance system remotely, increasing convenience and peace of mind.

In addition, the project contributed to user interface design by creating a graphical user interface (GUI) that allowed for intuitive control and monitoring of the surveillance system. Designing a user-friendly interface made the surveillance system more accessible and usable for homeowners.

Furthermore, by utilizing ready-to-use hardware components and open-source software frameworks, the project helped to develop cost-effective smart monitoring solutions. This approach reduced the barrier to entry for homeowners looking to improve their home security with connected surveillance systems.

These contributions collectively represent a significant step forward in the field of connected home surveillance, with potential implications for both residential and commercial applications.

## **1.5 Report Organization**

This report is organized into 7 chapters:

### **CHAPTER 1: INTRODUCTION**

This chapter offers an overview of the project, beginning with the presentation of the problem statement and the motivation behind the project's initiation. It describes the objectives set to guide the project's course and defines the scope and direction of the project. Additionally, it outlines the contributions of the project to the domain of connected home surveillance. Lastly, an overview of the report's organization is provided to offer a guide for the reader through the subsequent chapters.

### **CHAPTER 2: LITERATURE REVIEW**

This chapter scrutinizes existing literature pertinent to connected home surveillance systems, robotics in home automation, real-time communication protocols, autonomous navigation algorithms, and user interface design in home surveillance systems. By surveying prior work in these areas, insights are gained into the current state-of-the-art technologies and methodologies that informed the project.

### CHAPTER 3: SYSTEM METHODOLOGY/APPROACH

This chapter portrays the methodology and approach employed in the development of the surveillance system. It presents the system design diagram, which encompasses the system architecture, use case diagram, and activity diagram, to elucidate the overall structure and functionality of the system.

### CHAPTER 4: SYSTEM DESIGN

This chapter looks into the specifics of the system design, commencing with the system block diagram that outlines the high-level components and their interactions. Specifications are provided for each system component, encompassing hardware and software, and the design of circuits and components is elaborated upon. Moreover, interaction operations among system components are discussed to ensure seamless functionality.

### CHAPTER 5: SYSTEM IMPLEMENTATION

Focused on the practical implementation of the surveillance system, this chapter describes the hardware setup, software setup, and configuration process. It offers a step-by-step guide to operating the system, supplemented with screenshots for clarity. Furthermore, any implementation challenges encountered during the process are discussed.

### CHAPTER 6: SYSTEM EVALUATION AND DISCUSSION

In this chapter, the performance of the surveillance system is evaluated through systematic testing and analysis. Performance metrics are defined, and the testing setup and results are described. Any challenges faced during testing are discussed, and an evaluation of the extent to which the objectives were achieved is provided. Finally, concluding remarks on the system's effectiveness and potential improvements are made.

### CHAPTER 7: CONCLUSION AND RECOMMENDATION

This final chapter summarizes the key findings and conclusions drawn from the project. It highlights the contributions of the project and provides recommendations for future research or improvements to the surveillance system. Serving as a culmination of the project's efforts, this chapter offers closure to the report.

# Chapter 2

## Literature Review

### 2.1 Modelling a TurtleBot3 Based Delivery System for a Smart Hospital in Gazebo

The paper explains the creation of a "smart hospital" environment where robots are tasked with various assignments across multiple stations situated in diverse locations, navigating through a dynamic, object-filled environment. Equipped with a distance sensor (LIDAR), the robots utilize detected object signals for self-localization. Tasks are allocated to specific robots through a centralized interface, allowing for autonomous navigation to designated stations within the building to fulfill assigned tasks. To evaluate the system, a hospital map is generated within a Gazebo simulation. The workflow of the developed system enables users to assign tasks to robots via a graphical user interface, specifying task type, station location, task priority, robot ID, and, optionally, task completion time. A task manager oversees the assignment of tasks to robots, prioritizing them based on their urgency. Robots are assigned individual sets of tasks, displayed in the GUI by the task manager. The system supports two primary types of tasks: waiting and autonomous movement to stations [1].

Field	Description
id	Unique identifier
priority	An integer value ranging from 0 to 10 that defines the task execution sequence.
isDone	Boolean value will be set to true when task is executed
isCurrent	Boolean value will be set to true when task is being started to execute
goalId	The station's unique identifier on a map
taskType	Specifies what the robot should do.
robotName	Identifies the robot task that is assigned
waitTime	The time the robot should wait after completing the task is an optional value.

Table 2.1.1 TASK OBJECT FIELDS DESCRIPTION

State	Description
-------	-------------

NAVIGATE_TO_GOAL	Robot autonomously navigates to the point specified in task
EXECUTING_GOAL	Robot executes goal (i.e., waits of task time is specified)
WAIT_FOR_GOAL	Robot stands still until goal is received

Table 2.1.2 ROBOT BEHAVIOUR STATES DESCRIPTION

During task execution, the robot enters a nested state machine with transitions triggered by the move base package's "success," "preempted," or "aborted" statuses, enabling the description of more complex robot behaviors in the future [3], [4].

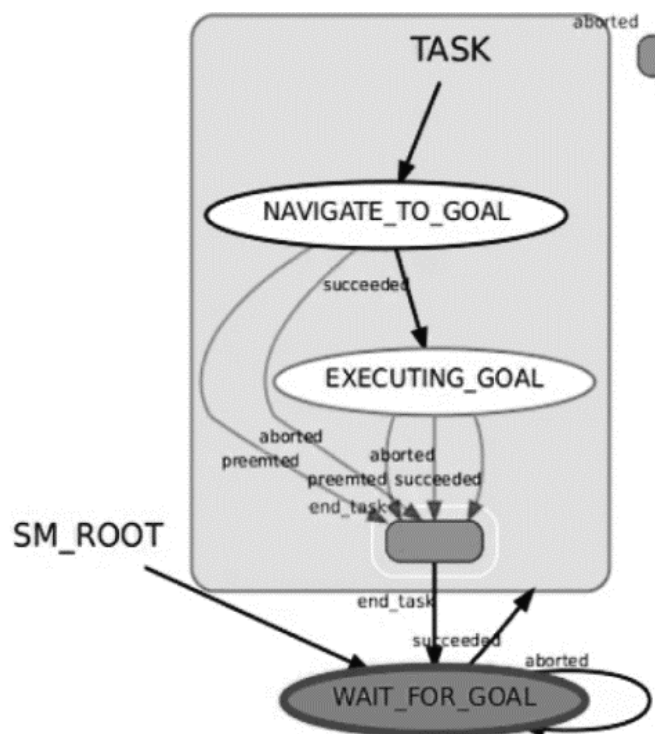


Figure 2.1.1 Current state machine structure

The paper elucidates the utilization of the ROS integrated move\_base package for route planning, localization, and SLAM within the robot's navigation module. The TEB local planner is employed to optimize the initial trajectory generated by the global planner [9]. Robot localization is achieved through the AMCL method, while dynamic obstacle avoidance is facilitated by the costmap\_converter ROS package. To represent obstacles on a map, the

costmap\_converter package transforms points from costmap\_2D into geometric primitives [10]. Furthermore, the CostmapToPolygonsDBSMCCH plugin utilizes the DBSCAN algorithm to convert occupied cells into convex polygons [11], [12].

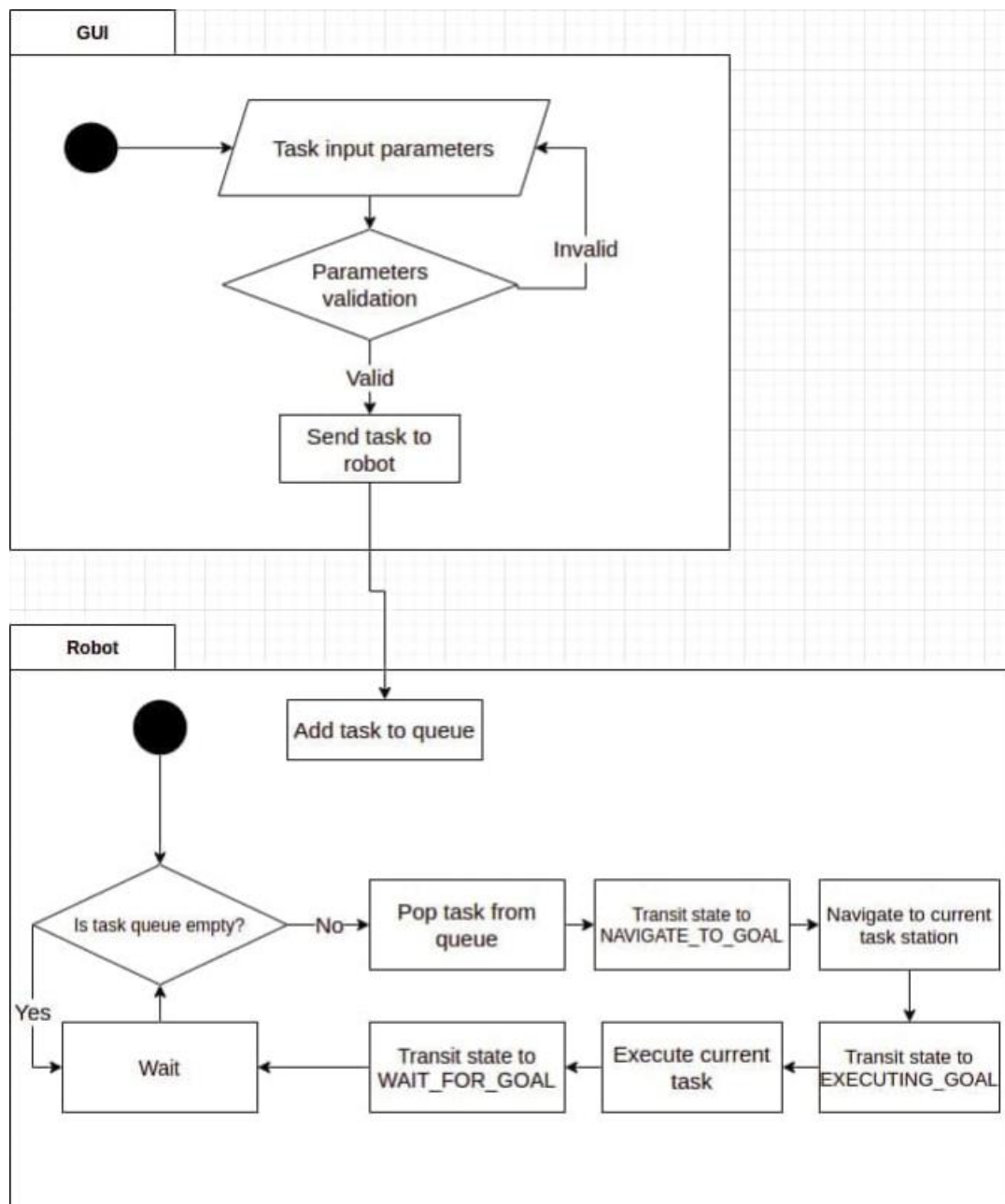


Figure 2.1.2 Application Block Diagram

Experiments conducted with the Turtlebot3 Burger and Turtlebot3 Waffle robots in a virtual hospital setting involved robot localization using the AMCL method and virtual map production using the mapping ROS package [13][14]. The simulation included stationary barriers such as furniture or trash, stations, and other moving objects. Equipped with a 360-

degree LiDAR sensor, the robots maneuvered and performed tasks within the hospital environment. Tasks were assigned through a user interface, with robots initiating tasks from random locations on the virtual map. Upon task completion, robots transitioned to the EXECUTING\_TASK state, then returned to the WAIT\_FOR\_GOAL state to await the next task assignment. The system maintained a history of completed tasks, station locations, and the task queue.

### **Limitation of Previous Study**

The first limitation is the system's lack of flexibility, potentially rendering it unable to handle unexpected situations or tasks beyond its predefined parameters. For instance, if a new task arises that the system is not programmed to handle, it may fail to execute it properly. The second limitation concerns the system's reliance on a pre-built map. This dependency necessitates map reconstruction when the environment changes or when robots are deployed in new locations, a process that can be time-consuming and impractical. The third limitation revolves around the need for precise localization. Accurate and reliable robot localization is essential for task execution; however, inadequate localization may hinder navigation and task accuracy. Finally, the system's limited range of tasks is a significant constraint. To address this, researchers suggest enhancing robots with more sophisticated manipulation and interaction capabilities to broaden their task collection.

### **Proposed Solution**

This project proposes the development of a more flexible system capable of adapting to new tasks and situations. Introducing advanced localization techniques such as SLAM (Simultaneous Localization and Mapping) is recommended to enhance robot localization accuracy. Additionally, expanding the range of tasks that robots can undertake by integrating new sensors and technologies will contribute to overcoming current limitations and enhancing system capabilities.

# Chapter 3

## System Methodology

### 3.1 System Design Diagram

The methodology employed in this project is the prototyping model. The Prototyping Model involves the iterative development, evaluation, and refinement of prototypes until a satisfactory solution is achieved. It serves to gather customer feedback and provides a scaled-down replica of the final product. Typically, a prototype represents an early and basic version of the final system, often lacking refined features, limited functionality, lower reliability, and sometimes ineffective performance compared to the actual software. However, it lays the groundwork for the finished software or system. This approach is particularly useful when the project's requirements are not fully understood at the outset. The Prototyping Model heavily relies on user feedback after each implementation iteration.

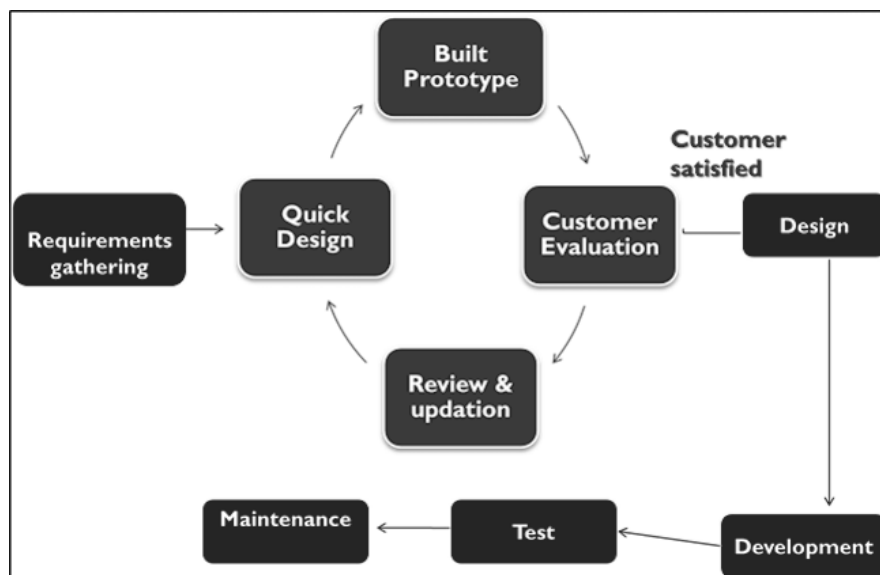


Figure 3.1 Prototyping Model

The prototyping model provides numerous benefits during the development process. First and foremost, it enables the early identification and clarification of project requirements [15]. By getting an initial version of the product in users' hands early on, valuable feedback can be collected to help refine the project's direction. This can help prevent the mistake of investing time and resources into creating a product that ultimately fails to meet user needs. Another advantage is the heightened level of user engagement and contentment [16]. Prototyping enables users to engage with the product and offer valuable feedback during the



development process [17]. This promotes a feeling of ownership and ensures that the end result meets their expectations [18]. In addition, prototyping can help identify errors and usability issues at an early stage [19]. By identifying these issues at an early stage, they can be resolved before they escalate into more costly and time-consuming challenges during the development process.

Although the prototyping model offers numerous benefits, it does have a few drawbacks [16]. One major issue to consider is the potential for it to be both time-consuming and costly, particularly if multiple prototypes are required [20]. Developing functional prototypes can be a time-consuming process that requires a significant allocation of resources. Additionally, if the requirements undergo significant changes, it may be necessary to discard and rebuild earlier prototypes. There is a potential danger of becoming overly attached to the initial prototype, which can hinder the ability to implement necessary changes in the future. Since users get to experience a working version early on, they might become focused on certain features or functionalities, even if they're not the best fit for the final product. Ultimately, the prototyping model may not be the best fit for every project, especially ones that have clearly defined requirements. If the requirements are clear from the beginning, a more conventional development approach might be more efficient.

The development of a robot surveillance system for connected home integration requires the use of several technologies and methods, including:

- Robot Operating System 2 (ROS2)
- Turtlebot3 Platform
- Python Programming Language
- Navigation 2 (Nav2)
- Raspberry Pi
- Real-Time Sensors
- PyQt5 Library

### **Robot Operating System 2 (ROS2)**

The Robot Operating System 2, or ROS2, is the essential software framework used in the "Robot Surveillance for Connected Home Integration" project. ROS2 is essential for optimising communication and coordination among the different components of a robot system [21].

ROS2 serves as a central hub that effectively coordinates conversations between different components of the robot [22]. These components are essential for the functioning of the robot. They work together seamlessly to ensure that the robot can detect events, capture images, process information, and move accordingly. ROS2 facilitates seamless communication between all these elements, enabling them to collaborate effectively in achieving the project's objective of home surveillance.

One advantage of ROS2 is its emphasis on a modular design approach. Functionalities are developed as separate packages, enabling the project to leverage pre-existing solutions for common tasks. As an example, ROS2 provides pre-existing packages for managing sensor data, robot navigation (utilising Navigation2), and real-time communication (which can be utilised to connect with the Telegram app). By utilising these existing functionalities, ROS2 helps to streamline the development process, saving precious time. The project can then prioritise its efforts on tailoring existing packages or developing specialised functionalities to meet the requirements of home surveillance.

Essentially, ROS2 serves as the software backbone of the robot [23]. It promotes effective communication between different components, streamlines the development process, and allows for seamless integration with different parts of the system. This enables the robot to function independently and offer efficient monitoring in a connected home setting.

### **TurtleBot3 Platform**

The TurtleBot3 platform serves as the central component of its physical robot. The TurtleBot3 robot is a pre-assembled mobile robot base equipped with wheels and motors. This removes the necessity of creating and building the robot's mechanical foundation from the ground up. This pre-built solution is a real time and resource-saver for any project. Additionally, the TurtleBot3 features a modular design approach [24]. This modularity enables convenient connection of different sensors, cameras, and computing components such as the Raspberry Pi [25]. This level of customization allows the project to develop a robot that is perfectly suited to its requirements. When it comes to home surveillance, you could consider incorporating a camera to capture images and maybe even adding extra sensors to improve the robot's detection abilities.

Additionally, the TurtleBot3 platform effortlessly integrates with a wide range of sensors. This feature is essential for the project because the robot heavily depends on sensors to detect events within the home environment.

At last, the TurtleBot3's wheels and motors enable it to navigate independently within a household. Being able to move around is crucial for the project, since the robot has to go to various places depending on sensor triggers. In addition, ROS2 (Robot Operating System 2) has the capability to utilize navigation packages such as Navigation2 to help the robot in mapping out secure and effective routes within the home setting.

### **Python Programming Language**

Using Python as the primary programming language would greatly benefit the project. Python is known for its versatility and readability, making it a popular choice among programmers [26]. This versatile language has a straightforward and succinct syntax, allowing it to be easily applied to different tasks in the project [27]. Here are a few possible uses:

1. **Processing Sensor Data:** Python is well-equipped to handle the data received from various sensors on the robot, including motion detectors and temperature sensors. It can then carry out the required calculations or analysis using this data.
2. **Robot Control Logic:** The project could potentially use Python code to establish the robot's behaviour by analysing sensor data. For example, when a sensor detects motion, Python code can be used to command the robot to move to that specific location.
3. **Image Processing:** Python libraries such as OpenCV can be integrated to handle image manipulation or analysis tasks.
4. **Communication with External Systems:** Python enables seamless communication between the robot and external systems such as the Telegram app. This can be utilised for sending notifications or images.

Python's seamless integration with ROS2 (Robot Operating System 2) is another factor that adds to its appeal. Given the project's use of ROS2 as its software framework, the widespread use of Python in developing ROS nodes is a major advantage. ROS nodes are small software programmes that communicate with each other within the ROS2 framework [28]. This enables the project to make use of pre-existing ROS functionalities written in Python and potentially create customised Python nodes for specific tasks within the robot's system. Python's ease of use and rapid prototyping capabilities make it perfect for projects that require iterative development cycles and frequent testing and refinement. It's a great choice for students and professionals alike. This is especially advantageous for a project of this nature, where testing and potential adjustments to functionalities are necessary during the development process.

Ultimately, Python emerges as a strong contender for the "Robot Surveillance for Connected Home Integration" project due to its adaptability, user-friendly syntax, and seamless integration with ROS2. Python is a powerful tool for developing the necessary functionalities for the robot to operate and communicate within a connected home environment.

## **Navigation 2 (Nav2)**

Navigation 2, commonly referred to as Nav2, is crucial for achieving this goal as it offers a full set of navigation tools within the ROS 2 (Robot Operating System 2) architecture.

Nav2 implemented in ROS 2:

Nav2 is a software framework developed on the foundation of ROS 2. ROS 2 functions as a central hub for communication, enabling the interchange of data between different software components on your robot. Nav2 utilises this communication architecture to manage many components of robot navigation.

Mapping and Localization:

One crucial role of Nav2 is to generate a map of the surroundings and accurately monitor the robot's position inside that map. Nav2 has the ability to be integrated with SLAM (Simultaneous Localization and Mapping) methods [33]. These algorithms enable the robot to construct a map of its environment while concurrently maintaining awareness of its position inside that map [34]. The map development process frequently relies on sensor data collected by the robot, such as LiDAR scanners and cameras [35]. If there is already a map of your home environment available, Nav2 can use it to determine the location of the robot. If the environment remains relatively static, the necessity for real-time map development is eliminated.

Path planning and movement:

Once the robot possesses a map and comprehends its precise location, Nav2 can strategize optimal and secure routes for the robot to traverse. Path planning algorithms take into account barriers, the robot's capabilities (such as speed constraints), and the desired destination in order to generate a viable path. It is crucial to acknowledge that Nav2 does not have direct control over the robot's movement. Conversely, it conveys the intended route to the robot's low-level control system. Subsequently, this control system converts those instructions into motor commands in order to propel the robot along the predetermined course.

## **Raspberry Pi**

Raspberry Pi is a small and flexible computer platform. The Raspberry Pi is a collection of compact single-board computers that were created in the United Kingdom by the Raspberry Pi Foundation. These computers, which are the size of a credit card, are widely recognised for their low cost, making them a favoured option among hobbyists, educators, and developers [36].

Single-board design refers to the use of a single circuit board to house all the necessary components of a device [37]. This design offers versatility, as it allows for the integration of many functionalities and features into a compact and efficient form [38]. The single-board design is a prominent characteristic of Raspberry Pi. A Raspberry Pi consolidates the CPU, RAM, and storage into a single circuit board, in contrast to conventional computers that have these components as independent entities. The small size of this design allows for easy portability, making it well-suited for tasks with limited area. The adaptability of Raspberry Pi enables its utilisation for many applications. For instance, it is a widely used platform for beginners to learn programming languages such as Python, thanks to its low cost and easy-to-use operating systems [39]. The Raspberry Pi's computational capabilities and its capacity to interface with a wide range of sensors and actuators render it well-suited for the development of robotic systems [40]. It is quite probable that this project, "Robot Surveillance for Connected Home Integration," utilised a Raspberry Pi as the primary processing unit for the robot. Raspberry Pi has the capability to be converted into a media centre, allowing for the streaming of videos and music. Raspberry Pi is an excellent platform for retro gaming fans due to its capability to simulate classic video game consoles. Lastly, Raspberry Pi is compatible with multiple operating systems, with Raspberry Pi OS (which is based on Debian) being the official option. These operating systems offer a recognisable graphical user interface for consumers to engage with the hardware. Furthermore, Raspberry Pi is available in various variants that offer varied levels of processing power, RAM capacity, and networking possibilities [41]. The selection of a model is contingent upon the particular requirements of the project. For example, your project may necessitate a model with enhanced computational capabilities to effectively manage sensor data and perform robot control duties.

### **Real-Time Sensor**

The real-time sensors are essential components in this project, serving as the visual and auditory perception system of the system. The sensors were seamlessly integrated and connected to Python for the purpose of acquiring and analysing data in real-time.

A camera can capture live photo feeds that can be utilised for an array of reasons. It plays a crucial role in capturing photos of intruders whenever sensors were triggered at certain location. Environmental sensors collect and provide information on different environmental conditions present in the home.

Temperature and humidity sensors: These sensors are used to monitor the surrounding temperature in order to maintain the most favourable living circumstances. In addition, they have the potential to identify fire threats by detecting abrupt increases in temperature.

Motion sensors: These sensors monitor any movement occurring inside the residence, which may suggest the presence of an unauthorised individual or abnormal behaviour. The system can establish a virtual security zone and activate alarms upon detecting unauthorised movement by strategically positioning motion sensors.

Light sensors: These are devices that measure and monitor the amount of light in the surrounding environment. The collected data can be utilised to activate automated lighting systems in response to occupancy, so enhancing energy efficiency and potentially dissuading potential invaders who may be deterred by a well-illuminated setting. The data obtained from these sensors that operate in real-time, which may include camera footage if relevant, is promptly analysed using the Python programming language.

This processing enables the system to conduct multiple essential tasks: Security Event Detection: The detection of abrupt movement by motion sensors or abnormal fluctuations in temperature readings may suggest the occurrence of a security breach. The system can be set to initiate alarms, dispatch notifications to homeowners, or engage other security actions based on these detections.

Environmental monitoring: This involves the collection of real-time data on temperature and light levels. This data is used to ensure that living conditions are comfortable and to potentially detect any problems that may arise. For example, the system has the capability to automatically modify thermostat configurations or initiate ventilation systems in response to temperature measurements.

Based on the analysed sensor data, the system is capable of making automatic decisions, enabling informed decision making. These decisions may involve initiating alerts, dispatching messages, implementing security measures, or modifying environmental controls to uphold optimal conditions inside the home environment.

## **PyQT5 Library**

PyQt5 is a comprehensive library that is constructed on the foundation of the Qt framework, offering a wide range of advanced features [29]. Developers can utilize Python syntax to design graphical user interfaces (GUIs) for desktop applications. Python developers will find PyQt5 straightforward to learn because it utilizes Python syntax [30]. PyQt5 effortlessly incorporates into current Python codebases, making it an ideal choice for projects that probably rely on Python for essential features.

Advantages of PyQt5 Graphical User Interfaces:

**Extensive Widget Selection:** PyQt5 provides a wide array of pre-existing widgets, including buttons, sliders, text boxes, menus, and other components [31]. These widgets function as the fundamental components of the graphical user interface (GUI), enabling a user-friendly and dynamic user experience for homeowners who are interacting with the robot surveillance system.

**Customization:** PyQt5 offers an extensive selection of pre-built alternatives, while also enabling developers to construct new widgets, as necessary. This adaptability guarantees that the graphical user interface (GUI) is precisely tailored to meet the exact requirements of the project.

PyQt5 offers the significant benefit of cross-platform compatibility, allowing it to function on multiple operating systems [32]. PyQt5 allows for the deployment of a graphical user interface (GUI) application on several operating systems, including Windows, macOS, Linux, and certain embedded systems, thereby offering a flexible and adaptable solution.

## **Raspberry Pi**

Raspberry Pi is a small and flexible computer platform. The Raspberry Pi is a collection of compact single-board computers that were created in the United Kingdom by the Raspberry Pi Foundation. These computers, which are the size of a credit card, are widely recognised for their low cost, making them a favoured option among hobbyists, educators, and developers [36].

Single-board design refers to the use of a single circuit board to house all the necessary components of a device [37]. This design offers versatility, as it allows for the integration of many functionalities and features into a compact and efficient form [38]. The single-board design is a prominent characteristic of Raspberry Pi. A Raspberry Pi consolidates the CPU,

RAM, and storage into a single circuit board, in contrast to conventional computers that have these components as independent entities. The small size of this design allows for easy portability, making it well-suited for tasks with limited area. The adaptability of Raspberry Pi enables its utilisation for many applications. For instance, it is a widely used platform for beginners to learn programming languages such as Python, thanks to its low cost and easy-to-use operating systems [39]. The Raspberry Pi's computational capabilities and its capacity to interface with a wide range of sensors and actuators render it well-suited for the development of robotic systems [40]. It is quite probable that this project, "Robot Surveillance for Connected Home Integration," utilised a Raspberry Pi as the primary processing unit for the robot. Raspberry Pi has the capability to be converted into a media centre, allowing for the streaming of videos and music. Raspberry Pi is an excellent platform for retro gaming fans due to its capability to simulate classic video game consoles. Lastly, Raspberry Pi is compatible with multiple operating systems, with Raspberry Pi OS (which is based on Debian) being the official option. These operating systems offer a recognisable graphical user interface for consumers to engage with the hardware. Furthermore, Raspberry Pi is available in various variants that offer varied levels of processing power, RAM capacity, and networking possibilities [41]. The selection of a model is contingent upon the particular requirements of the project. For example, your project may necessitate a model with enhanced computational capabilities to effectively manage sensor data and perform robot control duties.

### 3.1.1 System Architecture Diagram

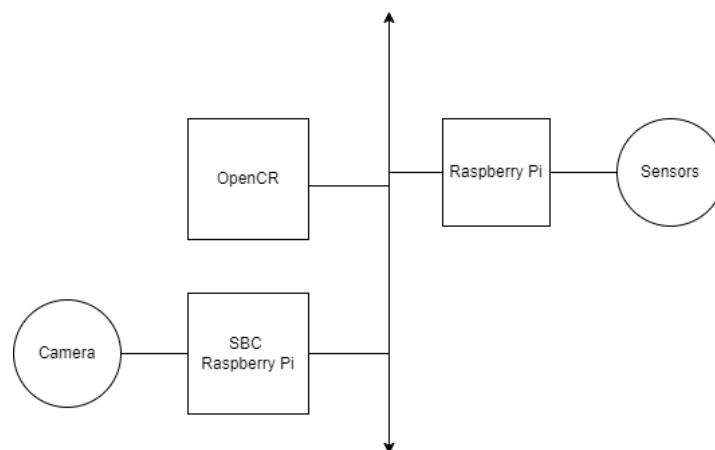


Figure 3.1.1.1 Hardware Architecture Diagram

Components:



- Raspberry Pi: A single-board computer serving as the processing unit for the system. One is attached to Turtlebot3 robot. Another three however serve as sensors integrator.
- Camera Module: This module, attached to the Raspberry Pi via the CSI (Camera Serial Interface) port, captures images or videos.
- OpenCR: The OpenCR platform has been specifically designed for ROS embedded systems, with the primary objective of offering a comprehensive open-source solution encompassing both hardware and software components.
- Sensors: Sensors are the inputs of the whole system.

Interactions:

1. Sensors → Raspberry Pi: Sensors keep sending sensor data through Raspberry Pi.
2. Raspberry Pi: The Raspberry Pi will connect to central control system to process the data.
3. Camera Module → Raspberry Pi (CSI Port): The camera module captures images or video data and sends it to the Raspberry Pi through the CSI port.
4. OpenCR: The board integrates multiple convenient functions into a single unit: Inertial Measurement Unit (IMU), power adaption, and servo control.

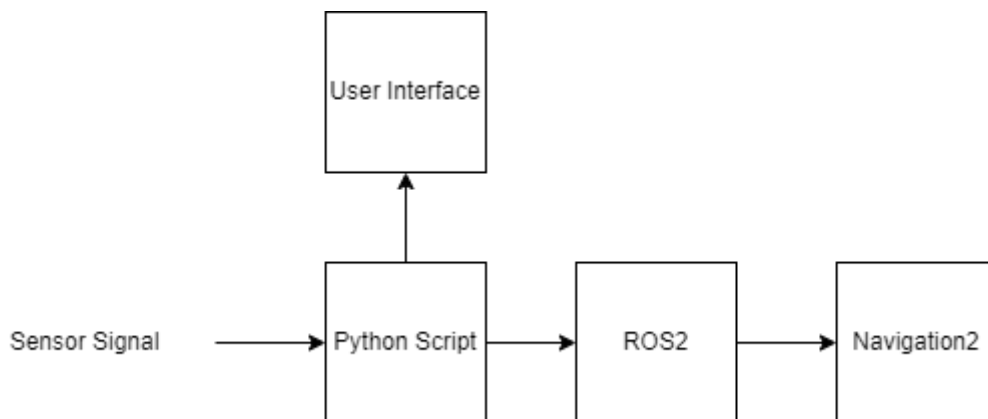


Figure 3.1.1.2 Software Architecture Diagram

Components:

- Sensor Signal: This block represents raw sensor data from various sensors such as LiDAR, camera, real-time sensors.

- Python Script: This block represents a Python script that is responsible for processing the raw sensor data. This processing involves tasks like filtering out noise, transforming data into a usable format, extracting logic for navigation.
- ROS2: ROS2 acts as the middleware, enabling communication between different parts of the navigation system. It provides a framework to create and integrate software modules for navigation.
- Navigation2: This block represents a ROS2 navigation package that is specifically designed for robot navigation. It likely uses the processed sensor data from the Python script to plan robot motion, avoid obstacles, and guide the robot towards its destination.

Interactions:

1. Sensor Signal → Python Script: Raw sensor data from the robot's sensors is sent to the Python script for processing.
2. Python Script → ROS2: The processed sensor data from the Python script is published to ROS2 topics.
3. ROS2 → Navigation2: Navigation2 subscribes to the relevant ROS2 topics to receive the processed sensor data.
4. Navigation2 → Navigation: Navigation2 communicates with ROS2 node to control the robot's movement based on the navigation plan.

### 3.1.2 Use Case Diagram and Description

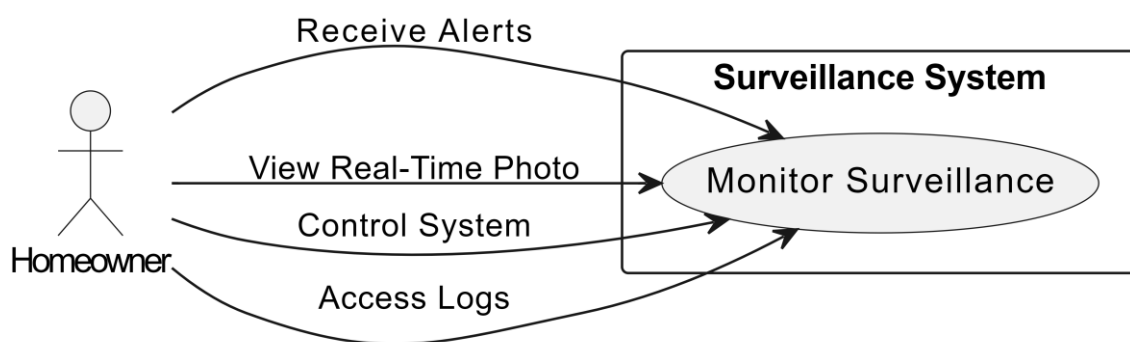


Figure 3.1.2 Use Case Diagram

Actor

- Homeowner: The person who interacts with the surveillance system to monitor their home.

Use Cases

Bachelor of Information Technology (Honours) Computer Engineering  
Faculty of Information and Communication Technology (Kampar Campus), UTAR

- **Receive Alerts:** The homeowner can receive alerts from the surveillance system when something is detected, such as motion or a change in temperature.
- **View Real-Time Photo:** The homeowner can view live photos from the surveillance system's camera to see what is happening in their home in real-time.
- **Monitor Surveillance:** The homeowner can monitor the surveillance system to see an overview of the system's status, such as sensor data or camera feeds (if applicable).
- **Access Logs:** The homeowner can access logs from the surveillance system to see a history of events that have been detected, such as motion detection events or temperature changes.
- **Control System:** The homeowner can control the surveillance system through the user interface. This might include functionalities like activating or deactivating the system, adjusting patrol routes (if applicable), or configuring sensor settings.

The use case diagram depicts that the homeowner can interact with the surveillance system in various ways to monitor their home environment. They can receive alerts, view live camera feeds (if applicable), monitor the system's overall status, and access historical data. Optionally, they can also control the system through the user interface.

### 3.1.3 Activity Diagram

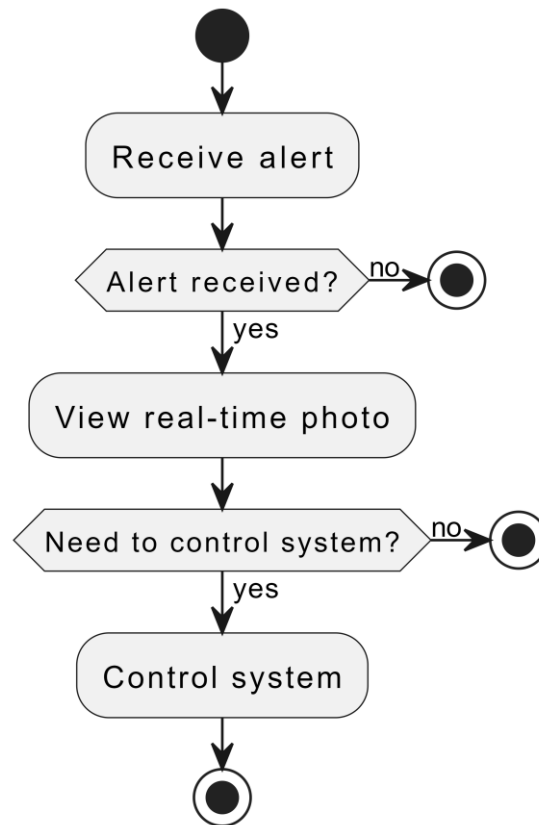


Figure 3.1.3 Activity Diagram

The activity diagram presented illustrates the sequential steps entailed in the operation of the surveillance system.

The procedure begins with receiving of an alert, which initiates the subsequent actions. Upon receiving the alert, the system proceeds to examine a real-time photograph in order to evaluate the situation.

After the observation of the photograph, the diagram illustrates an interval where the assessment of system control is conducted. If there is a need to manage the system, such as triggering a response to the alert, the system will then proceed to perform the required actions.

On the other hand, in the event that there is no necessity to exercise control over the system, the process comes to a halt at this moment, signifying that no additional measures are undertaken.

The above activity diagram offers a conscious visualization of the workflow encompassed in the process of addressing alerts within the surveillance system, which simplifies understanding of the system's functioning.

# Chapter 4

## System Methodology

### 4.1 System Block Diagram

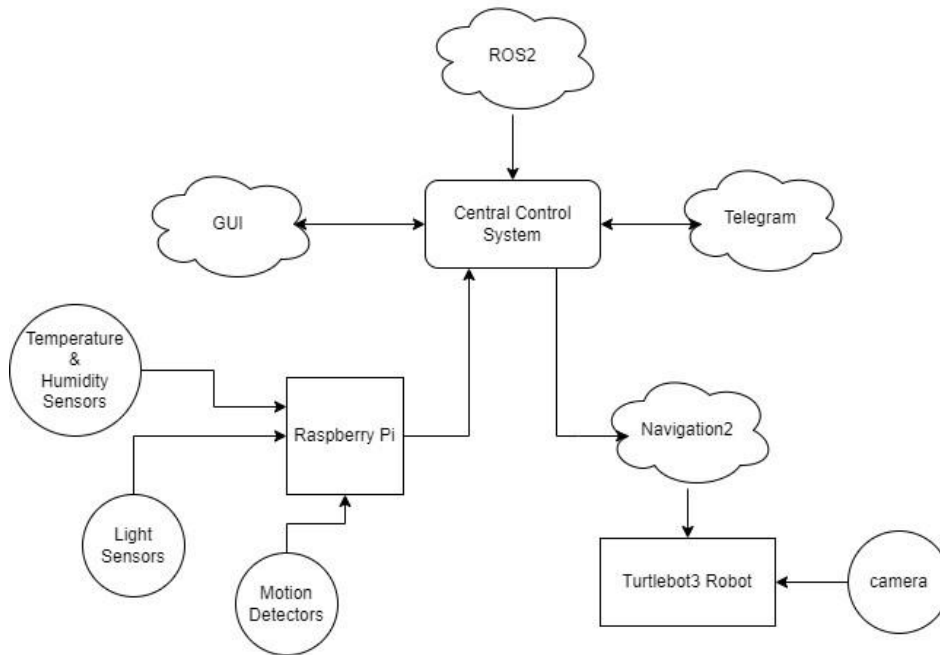


Figure 4.1.1 System Architecture Diagram

#### Central Control System:

The central control system acts as the brain of the operation, functioning as a central hub that oversees and controls all other components. It ensures the smooth integration of these components for optimal surveillance efficiency.

#### TurtleBot3 Robot:

The TurtleBot3 robot serves as the mobile platform for the surveillance system. This robot navigates the home environment using various sensors that collect information about its surroundings.

#### Sensor Devices:

The system incorporates various sensors into the TurtleBot3 robot, including:

- Motion detectors: These sensors detect any movement within the home, potentially indicating an intruder or unusual activity.
- Temperature and humidity sensors: These devices monitor the surrounding temperature to maintain ideal living conditions and potentially identify potential fire hazards.

- Camera: The camera provides live visuals for capturing photos and monitoring surroundings.

The data gathered by these sensors is crucial for the overall operation of the surveillance system.

### **Raspberry Pi:**

The Raspberry Pi acts as the central processing unit, the cognitive center within the robot. This microprocessor collects data from the diverse sensors on the TurtleBot3 robot. The system then analyzes the sensor data and communicates with other system components to trigger appropriate actions.

### **Real-Time Communication Interface (Telegram App):**

A cloud-based interface facilitates real-time communication between the surveillance system and the user's smartphone using the Telegram application. This interface enables bidirectional communication:

- The system can transmit notifications, alerts, and even visuals to the user's mobile device, keeping them informed about occurrences within their residence.
- Commands can be transmitted to the system using the Telegram app, potentially enabling remote control functionalities.

### **Graphical User Interface (GUI) (Optional):**

An optional Graphical User Interface (GUI) provides a user-friendly platform for interacting with the surveillance system. This interface may allow users to:

- Observe real-time video streams (if available) from the camera mounted on the robot.
- View notifications and alerts triggered by the system.
- Remotely control the system, including manual robot operation or adjusting surveillance configurations.

### **ROS2 (Robot Operating System 2):**

ROS2 acts as the central communication infrastructure for the entire system. It functions as middleware, facilitating smooth communication and coordination among all the various components. ROS2 offers a structured method for creating and incorporating software components for tasks such as robot navigation, sensor data analysis, and communication between various system elements.

### **Navigation2:**

The Navigation & Localization component is essential for ensuring the robot's efficient movement within the home environment. This component includes sub-components like Navigation2, which is responsible for planning safe routes for the robot to traverse while

avoiding obstacles. Through collaboration, these components enable the robot to operate autonomously and carry out its surveillance tasks with optimal efficiency.

These diverse elements work together seamlessly to create a highly efficient system. The central control system orchestrates the TurtleBot3 robot, equipped with sensors for data collection. The Raspberry Pi processes sensor data, while its real-time communication interface allows user interaction. An optional user-friendly GUI can further enhance the user experience. ROS2 ensures smooth communication among all components, and the Navigation & Localization module enables autonomous robot motion. The integration of these components allows the robot surveillance system to operate efficiently and maintain home security.

## 4.2 System Components Specifications

### Turtlebot3

Description	Specifications
Model	Turtlebot3
Maximum translational velocity	0.22 m/s
Maximum rotational velocity	2.84 rad/s (162.72 deg/s)
Maximum payload	15kg
Size (L x W x H)	138mm x 178mm x 192mm
Weight (+ SBC + Battery + Sensors)	1kg
Threshold of climbing	10 mm or lower
Expected operating time	2h 30m
Expected charging time	2h 30m
SBC (Single Board Computers)	Raspberry Pi
MCU	32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)
Actuator	XL430-W250
LDS(Laser Distance Sensor)	360 Laser Distance Sensor <a href="#">LDS-01</a> or <a href="#">LDS-02</a>

IMU	Gyroscope 3 Axis Accelerometer 3 Axis
Power connectors	3.3V / 800mA 5V / 4A 12V / 1A
Expansion pins	GPIO 18 pins Arduino 32 pin
Peripheral	UART x3, CAN x1, SPI x1, I2C x1, ADC x5, 5pin OLLO x4
DYNAMIXEL ports	RS485 x 3, TTL x 3
Audio	Several programmable beep sequences
Programmable LEDs	User LED x 4
Status LEDs	Board status LED x 1 Arduino LED x 1 Power LED x 1
Buttons and Switches	Push buttons x 2, Reset button x 1, Dip switch x 2
Battery	Lithium polymer 11.1V 1800mAh / 19.98Wh 5C
PC connection	USB
Firmware upgrade	via USB / via JTAG
Power adapter (SMPS)	Input: 100-240V, AC 50/60Hz, 1.5A @max Output: 12V DC, 5A

Table 4.2.1 Specifications of Robot



## OpenCR

Items	Specifications
Microcontroller	STM32F746ZGT6 / 32-bit ARM Cortex®-M7 with FPU (216MHz, 462DMIPS) <a href="#">Reference Manual</a> , <a href="#">Datasheet</a>
Sensors	<b>(Discontinued)</b> Gyroscope 3Axis, Accelerometer 3Axis, Magnetometer 3Axis (MPU9250) <b>(New)</b> 3-axis Gyroscope, 3-Axis Accelerometer, A Digital Motion Processor™ (ICM-20648)
Programmer	ARM Cortex 10pin JTAG/SWD connector USB Device Firmware Upgrade (DFU) Serial
Digital I/O	32 pins (L 14, R 18) *Arduino connectivity 5Pin OLLO x 4 GPIO x 18 pins PWM x 6 I2C x 1 SPI x 1
Analog INPUT	ADC Channels (Max 12bit) x 6
Communication Ports	USB x 1 (Micro-B USB connector/USB 2.0/Host/Peripheral/OTG) TTL x 3 ( <a href="#">B3B-EH-A</a> / DYNAMIXEL) RS485 x 3 ( <a href="#">B4B-EH-A</a> / DYNAMIXEL) UART x 2 ( <a href="#">20010WS-04</a> ) CAN x 1 ( <a href="#">20010WS-04</a> )
LEDs and buttons	LD2 (red/green) : USB communication User LED x 4 : LD3 (red), LD4 (green), LD5 (blue) User button x 2 Power LED : LD1 (red, 3.3 V power on) Reset button x 1 (for power reset of board) Power on/off switch x 1
Input Power Sources	5 V (USB VBUS), 5-24 V (Battery or SMPS) Default battery : LI-PO 11.1V 1,800mAh 19.98Wh Default SMPS : 12V 4.5A External battery Port for RTC (Real Time Clock) ( <a href="#">Molex 53047-0210</a> )
Input Power Fuse	125V 10A <a href="#">LittleFuse 0453010</a>
Output Power Sources	*12V max 4.5A( <a href="#">SMW250-02</a> ) *5V max 4A( <a href="#">5267-02A</a> ), 3.3V@800mA( <a href="#">20010WS-02</a> )
Dimensions	105(W) X 75(D) mm
Weight	60g

Table 4.2.2 Specifications of OpenCR

## Raspberry Pi b+

Description	Specifications
CPU:	BCM2837B0
CPU cores:	4
CPU speed:	1.4 GHz
RAM:	1 GB
Ethernet:	Yes
WiFi:	2.4 GHz 5 GHz 802.11b/g/n/ac
Bluetooth:	4.2
Bluetooth Low Energy:	Yes
HDMI:	Yes
Analog video:	Yes
SD socket:	MicroSD
Onboard regulators:	switching
Expansion header pins:	40
USB ports:	4
Mounting holes:	4
Dimensions <sup>2</sup> :	3.35" × 2.2" × 0.8"
Weight <sup>3</sup> :	50g

Table 4.2.3 Specifications of Raspberry Pi

### 4.3 Circuits and Components Design

For this project, three sensor stations were established to transmit sensor data over MQTT from Raspberry Pi to the user's PC or laptop.

#### Sensor Setup

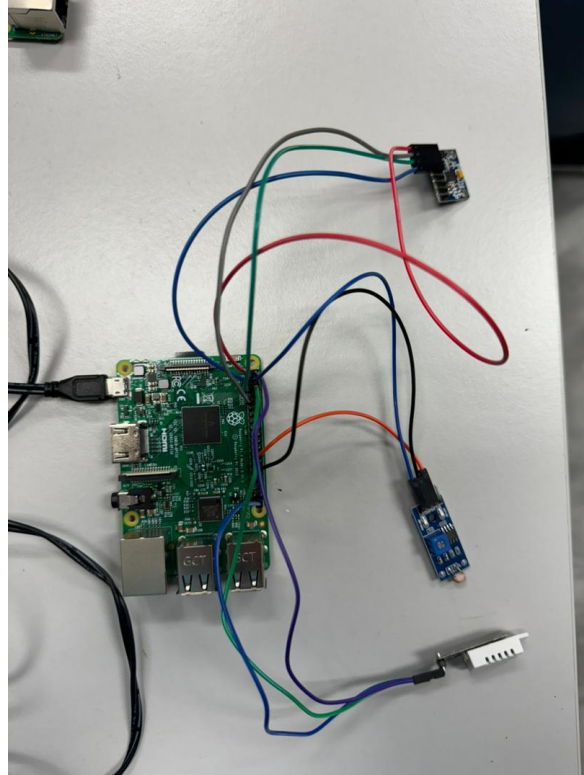


Figure 4.3.1 Sensor 1 Setup

DHT22 sensor module, MPU6050 motion tracking device, and LDR module were connected to the first Raspberry Pi (sensor1) as shown in the Figure 4.3.1 Sensor 1 setup.

#### **DHT22:**

- VCC – Connect to 5V
- Out - Connect to GPIO Pin 21
- GND - Connect to Ground

#### **MPU6050:**

- VCC - Connect to 5V
- GND - Connect to Ground
- SCL - Connect to SCL
- SDA – Connect to SDA

#### **LDR Module:**

- VCC - Connect to 5V
- DO - Connect to GPIO PIN 7
- GND – Connect to Ground

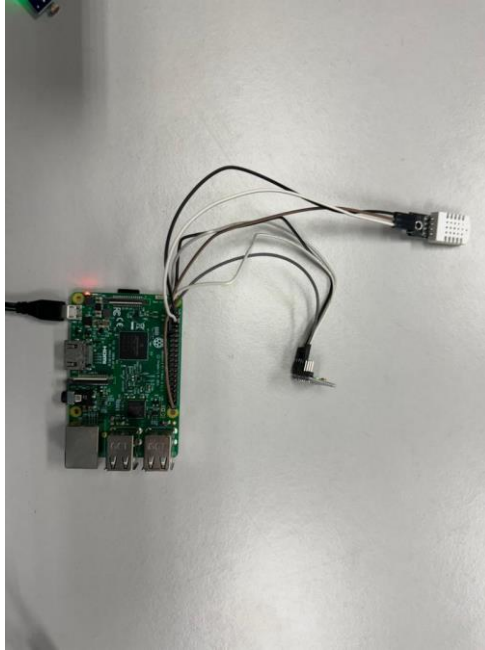


Figure 4.3.2 Sensor 2 Setup

DHT22 sensor module, and MPU6050 motion tracking device were connected to the second Raspberry Pi (sensor2) as shown in the Figure 4.3.2 Sensor 2 setup.

**DHT22:**

- VCC – Connect to 5V
- Out - Connect to GPIO Pin 21
- GND - Connect to Ground

**MPU6050:**

- VCC - Connect to 5V
- GND - Connect to Ground
- SCL - Connect to SCL
- SDA – Connect to SDA

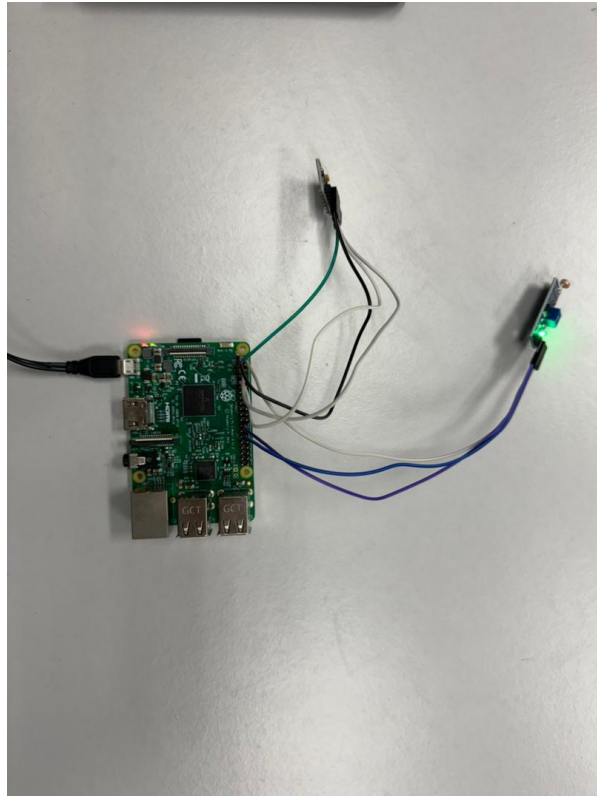


Figure 4.3.3 Sensor3 Setup

MPU6050 motion tracking device and LDR module were connected to the third Raspberry Pi (sensor3) as shown in the Figure 4.3.3 Sensor 3 setup.

**MPU6050:**

- VCC - Connect to 5V
- GND - Connect to Ground
- SCL - Connect to SCL
- SDA – Connect to SDA

**LDR Module:**

- VCC - Connect to 5V
- DO - Connect to GPIO PIN 7

GND – Connect to Ground

Camera Setup

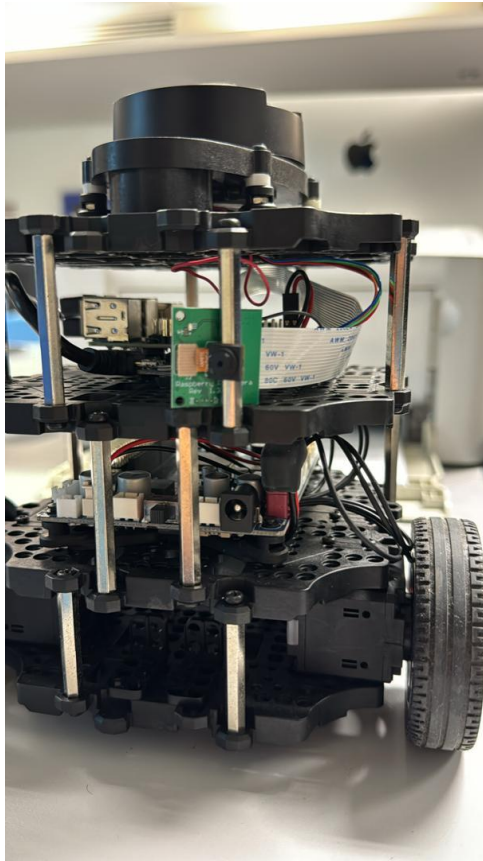


Figure 4.3.4 Pi Camera Module Setup

Pi Camera module was attached to the Raspberry Pi on the Turtlebot3 robot via the CSI (Camera Serial Interface) port.

#### 4.4 System Components Interaction Operations

In this project, the interactions between the system parts are very important for sending and receiving data without any problems. Three sensor stations are set up, and each one has a Raspberry Pi linked to different sensor modules. In addition, the TurtleBot3 robot has a camera section built in, which lets it keep an eye on the surroundings.

For each sensor station, including Sensor1, Sensor2, and Sensor3, the interaction operations involve collecting data from sensor modules and transmitting it over MQTT to the user's PC or laptop.

- **Sensor1 Setup:** This setup involves connecting a DHT22 sensor module, an MPU6050 motion tracking device, and an LDR module to the first Raspberry Pi (sensor1). The DHT22 sensor provides temperature and humidity data, the MPU6050 tracks motion, and the LDR module detects light intensity. Data from these sensors is collected and transmitted over MQTT to the central control system.

- **Sensor2 Setup:** The second Raspberry Pi (sensor2) is equipped with a DHT22 sensor module and an MPU6050 motion tracking device. Similar to Sensor1, Sensor2 collects temperature, humidity, and motion data, transmitting it over MQTT to the central control system.
- **Sensor3 Setup:** In the third sensor station, the Raspberry Pi (sensor3) is connected to an MPU6050 motion tracking device and an LDR module. This setup focuses on tracking motion and light intensity, with data transmitted over MQTT for analysis.

### **Camera Setup**

The camera module is an integral part of the TurtleBot3 robot, providing visual surveillance capabilities. The Pi Camera module is attached to the Raspberry Pi on the Turtlebot3 robot via the CSI (Camera Serial Interface) port. In order to capture real-time images, camera node is needed to be run so that the topic /image can be published and subscribed by ROS. After subscribing the image topic, it can capture and show real-time images, enhancing the overall surveillance system's effectiveness.

It is essential to take into consideration the communication protocol that is utilized for the transmission of data between the Raspberry Pi units and the central control system. This is in addition to the interactions that have been stated previously. In this project, the messaging capabilities of MQTT (Message Queuing Telemetry Transport) are leveraged because of its lightweight and efficient nature. This ensures that data may be transferred via network connections in a dependable way.

Additionally, the central control system which is user PC/laptop is also responsible for receiving, analysing, and storing the sensor data that is transmitted by the sensor stations. In order to successfully see and interact with the data that has been collected, it is necessary to develop appropriate data handling algorithms, data storage techniques, and user interface components in Python scripts.

# Chapter 5

## System Implementation

### 5.1 Hardware Setup

#### Turtlebot3 Burger

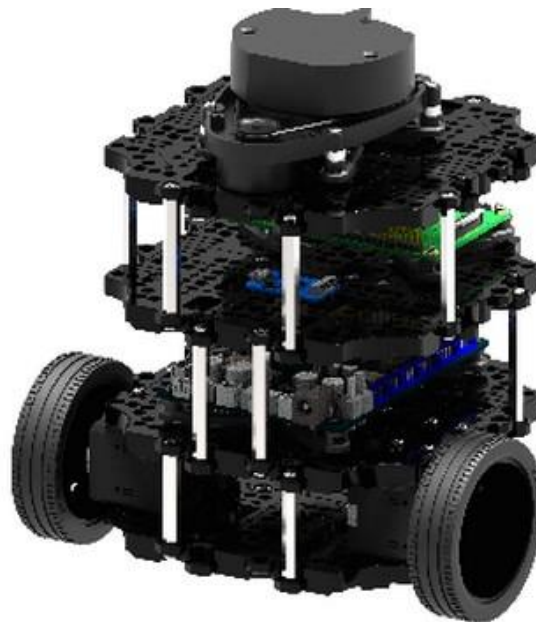


Figure 5.1.1 Turtlebot3 Burger Model

Turtlebot3 robot burger model as shown in figure 5.1.1 was used throughout the entire project.

#### OpenCR 1.0

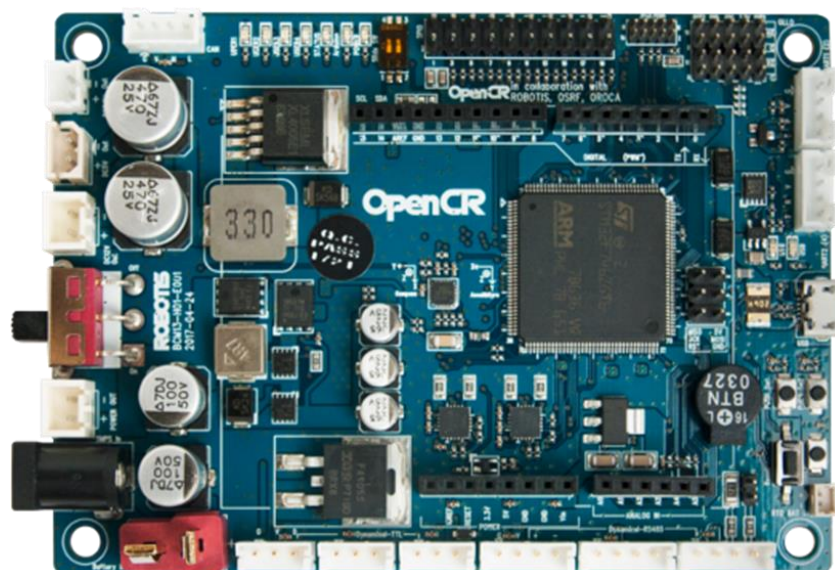


Figure 5.1.2 Open CR Board.



Open CR is a ROS embedded systems to provide completely open-source hardware and software. It was attached to the third layer from the top of Turtlebot3 robot.

### Raspberry Pi Model 3b+

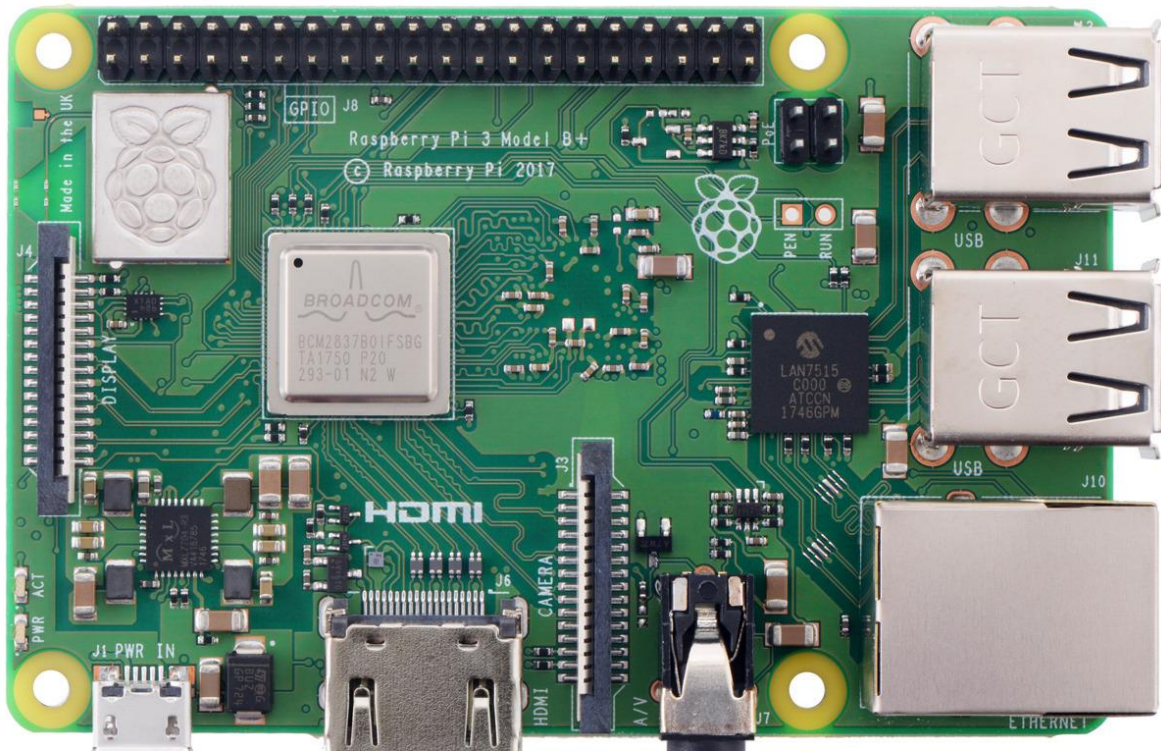


Figure 5.1.3 Raspberry Pi Model 3B+

The Raspberry Pi serves as the central processing unit in the TurtleBot3 robot. It is a single-board computer that processes sensor data obtained from the TurtleBot3's various sensors. These sensors include LiDAR for 3D mapping and obstacle avoidance, IMU (Inertial Measurement Unit) used to measure robot orientation and movement, depth sensor provides distance information about obstacles. Camera for collecting visual data about the environment. The Raspberry Pi also involves in processing the raw sensor data received from the TurtleBot3. This processing involve: filtering out noise from sensor readings, transforming data into a usable format for specific tasks like obstacle detection or path planning, and extracting relevant features from sensor data for decision making.

In this project, one Raspberry Pi model 3b+ was attached to second layer from the top of Turtlebot3 robot. A Pi camera module was attached to it to capture images. Another three Raspberry Pi model 3b+ was used to integrate with real-time sensors such as DHT22, MPU6050, and LDR module.

## DHT22 Sensor Module

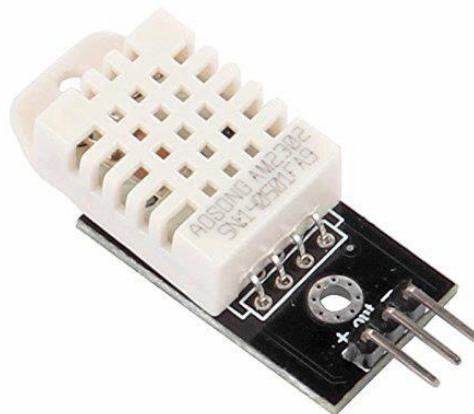


Figure 5.1.4 DHT22 Sensor Module

The DHT22 produces a calibrated digital signal as output. It employs a unique digital signal collection approach and humidity sensor technology, ensuring its dependability and stability. The sensor devices are coupled to an 8-bit single-chip computer. Each sensor of this model undergoes temperature compensation and calibration in a precise calibration chamber. The calibration coefficient is then stored as a program in OTP memory. When the sensor is in operation, it retrieves the coefficient from memory. The compact size, low power consumption, and long transmission distance of 20 meters make the DHT22 suitable for various challenging application scenarios.

## MPU6050 Motion Detector

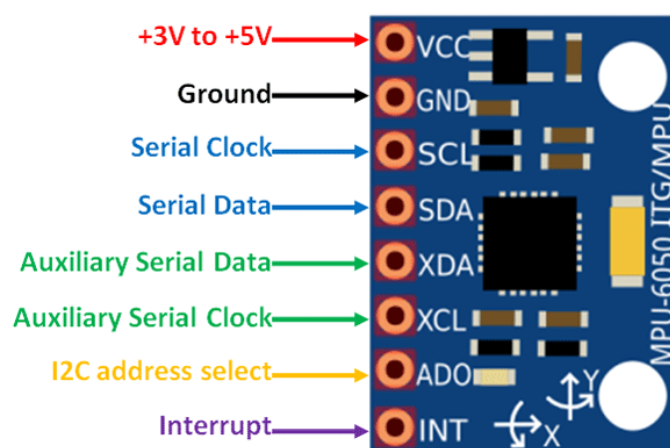


Figure 5.1.5 MPU6050

The MPU-6050™ parts are the first Motion Tracking devices ever made for smartphones, tablets, and wireless sensors that need to be low-power, low-cost, and high-performing.

The MPU-6050 has InvenSense's MotionFusion™ and run-time calibration firmware built in. This means that manufacturers don't have to choose, qualify, and integrate discrete devices at the system level in motion-enabled products, which can be expensive and time-consuming. This makes sure that the sensor fusion algorithms and calibration procedures give customers the best performance possible. The MPU-6050 devices have a 3-axis gyroscope, a 3-axis accelerometer, and a Digital Motion Processor™ (DMPTM) built in. The DMPTM handles complicated 6-axis MotionFusion algorithms. Through an extra master I<sup>2</sup>C bus, the device can talk to magnetometers or other sensors outside of it. This lets the device collect all of the sensor data without the main processor having to do anything at all [42].

### LDR Module

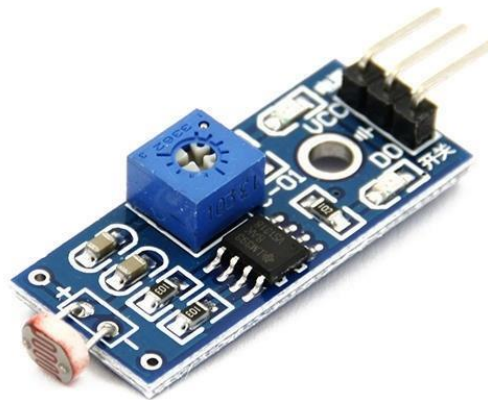


Figure 5.1.6 LDR Module

A Light Dependent Resistor, often known as an LDR, is a sort of passive electronic sensor that is utilised for the purpose of detecting light. As a result of being exposed to high levels of light intensity, it transforms into a variable resistor in the circuit. It is composed of two conductors that are separated by an insulator whose conductivity increases [43].

## Pi Camera Module

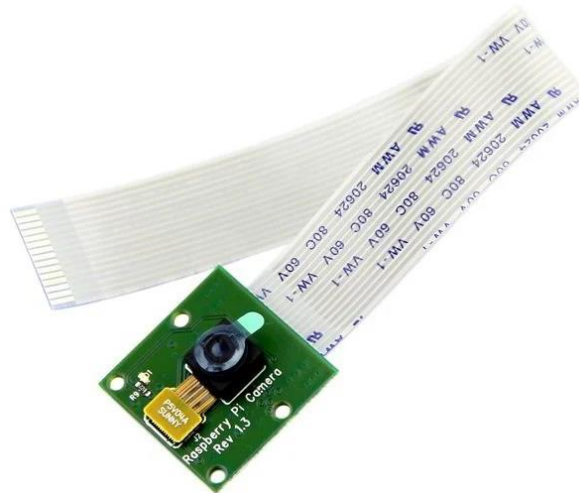


Figure 5.1.7 Pi Camera Module

The Pi Camera module is a camera that can capture both still images and high-definition video. The Raspberry Pi Board contains a CSI (Camera Serial Interface) interface, which allows users to directly connect the Pi Camera module. This Pi Camera module connects to the Raspberry Pi's CSI port via a 15-pin ribbon cable [44].

## 5.2 Software Setup

### PC setup

Ubuntu image was downloaded and installed on PC. Ubuntu version 20.04 LTS Desktop was used for this project. After done installation the Ubuntu OS, ROS2 package was installed using terminal on remote pc which was used for central control system. Foxy version of distro was used in this installation.

```
$ wget https://raw.githubusercontent.com/ROBOTIS-GIT/robotis_tools/master/install_ros2_foxy.sh
$ sudo chmod 755 ./install_ros2_foxy.sh
$ bash ./install_ros2_foxy.sh
```

Figure 5.2.1 Installation of ROS2 Package

Next, necessary dependencies were required to be downloaded: Gazebo, Cartographer, Navigation2.

```
$ sudo apt-get install ros-foxy-gazebo-*
```

Figure 5.2.2 Installation of Dependent Gazebo

```
$ sudo apt install ros-foxy-cartographer
$ sudo apt install ros-foxy-cartographer-ros
```

Figure 5.2.3 Installation of Dependent Cartographer

```
$ sudo apt install ros-foxy-navigation2
$ sudo apt install ros-foxy-nav2-bringup
```

Figure 5.2.4 Installation of Dependent Navigation2

Turtlebot3 was installed via Debian Packages.

```
$ source ~/.bashrc
$ sudo apt install ros-foxy-dynamixel-sdk
$ sudo apt install ros-foxy-turtlebot3-msgs
$ sudo apt install ros-foxy-turtlebot3
```

Figure 5.2.5 Installation of Turtlebot3

```
$ echo 'export ROS_DOMAIN_ID=30 #TURTLEBOT3' >> ~/.bashrc
$ source ~/.bashrc
```

Figure 5.2.6 Setting up ROS Environment

### 5.3 Setting and Configuration

#### SBC Setup

The Turtlebot3 SBC Image file was acquired and written into a microSD card using a card reader. Once the Raspberry Pi Imager was downloaded, the storage size and partition were configured.

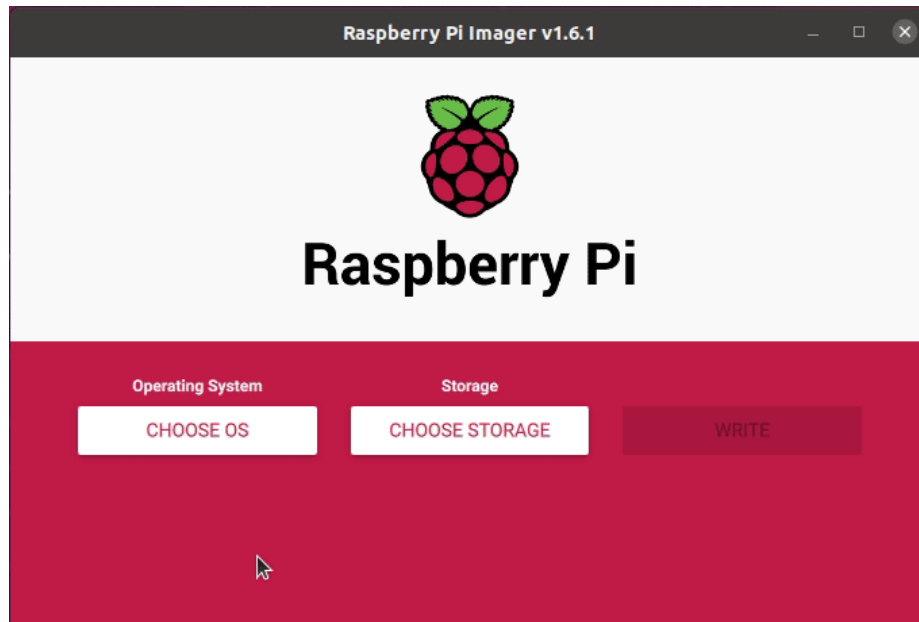


Figure 5.3.1 Configuration of SBC.

```
$ cd /media/$USER/writable/etc/netplan
$ sudo nano 50-cloud-init.yaml
```

Figure 5.3.2 Command to configure WIFI Network Setting

```

network:
  version: 2
  renderer: networkd
  ethernet:
    eth0:
      dhcp4: yes
      dhcp6: yes
      optional: true
  wifi:
    wlan0:
      dhcp4: yes
      dhcp6: yes
      access-points:
        WIFI_SSID:
          password: WIFI_PASSWORD

```

Figure 5.3.3 Editing WIFI SSID and Password

```

$ sudo apt update
$ sudo apt install libudev-dev
$ cd ~/turtlebot3_ws/src
$ git clone -b ros2-devel https://github.com/ROBOTIS-GIT/ld08_driver.git
$ cd ~/turtlebot3_ws/src/turtlebot3 && git pull
$ rm -r turtlebot3_cartographer turtlebot3_navigation2
$ cd ~/turtlebot3_ws && colcon build --symlink-install

```

Figure 5.3.4 Installing and Updating the LDS02 Driver and Turtlebot3 Package

```

$ echo 'export LDS_MODEL=LDS-02' >> ~/.bashrc
$ source ~/.bashrc

```

Figure 5.3.5 Exporting LDS MODEL to bashrc file.

## OpenCR Setup

```
$ sudo dpkg --add-architecture armhf
$ sudo apt update
$ sudo apt install libc6:armhf
```

Figure 5.3.6 Installing Packages on Raspberry Pi .

```
$ export OPENCNCR_PORT=/dev/ttyACM0
$ export OPENCNCR_MODEL=burger
$ rm -rf ./opencnrcr_update.tar.bz2
```

Figure 5.3.7 Setting up OpenCR Name.

```
$ wget https://github.com/ROBOTIS-GIT/OpenCR-Binaries/raw/master/turtlebot3/ROS2/latest/opencnrcr_update.tar.bz2
$ tar -xvf ./opencnrcr_update.tar.bz2
```

Figure 5.3.8 Downloading the firmware and loader.

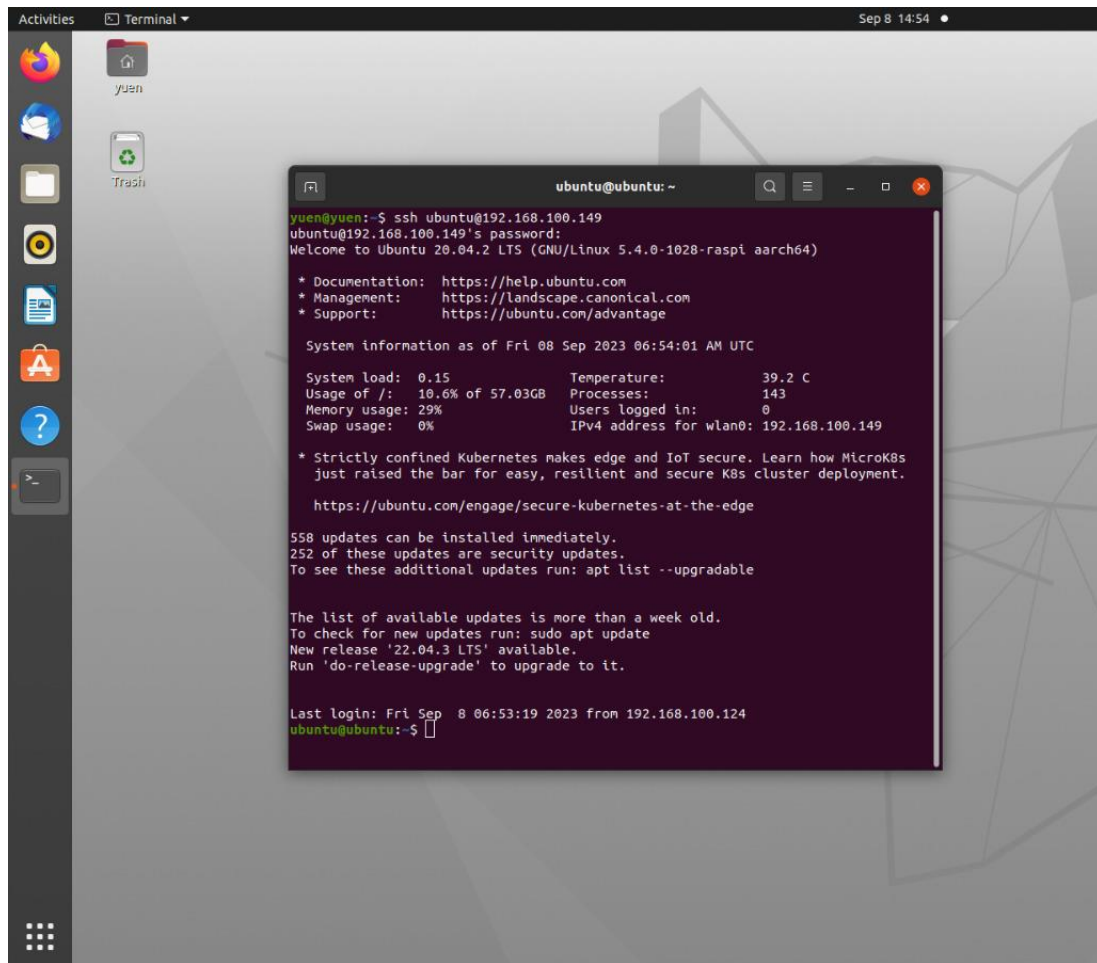
```
$ cd ~/opencnrcr_update
$ ./update.sh $OPENCNCR_PORT $OPENCNCR_MODEL.opencnrcr
```

Figure 5.3.9 Uploading the firmware to OpenCR



## 5.4 System Operation

### Bring Up



```
Activities Terminal Sep 8 14:54
yuen
Trash

yuen@yuen:~$ ssh ubuntu@192.168.100.149
ubuntu@192.168.100.149's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-1028-raspi aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Fri 08 Sep 2023 06:54:01 AM UTC

System load:  0.15          Temperature:   39.2 C
Usage of /:   10.6% of 57.03GB  Processes:    143
Memory usage: 29%          Users logged in: 0
Swap usage:  0%            IPv4 address for wlan0: 192.168.100.149

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.
   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

558 updates can be installed immediately.
252 of these updates are security updates.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Sep 8 06:53:19 2023 from 192.168.100.124
ubuntu@ubuntu:~$
```

Figure 5.4.1 SSH to Ubuntu (IP address of Raspberry Pi)

After successfully uploading the firmware, Turtlebot3 is required to bring up. On the remote pc, SSH have been used to remote access the Raspberry Pi which was installed on the Turtlebot3 hardware model.

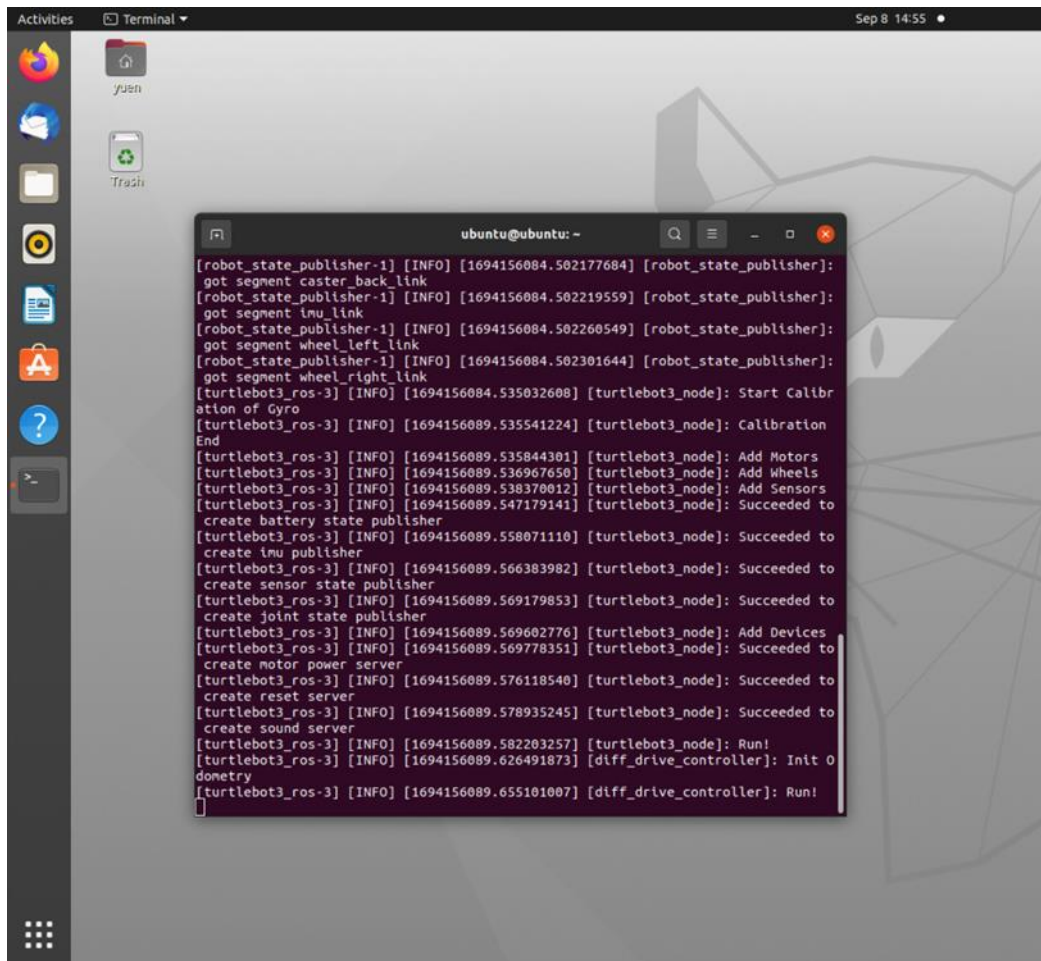
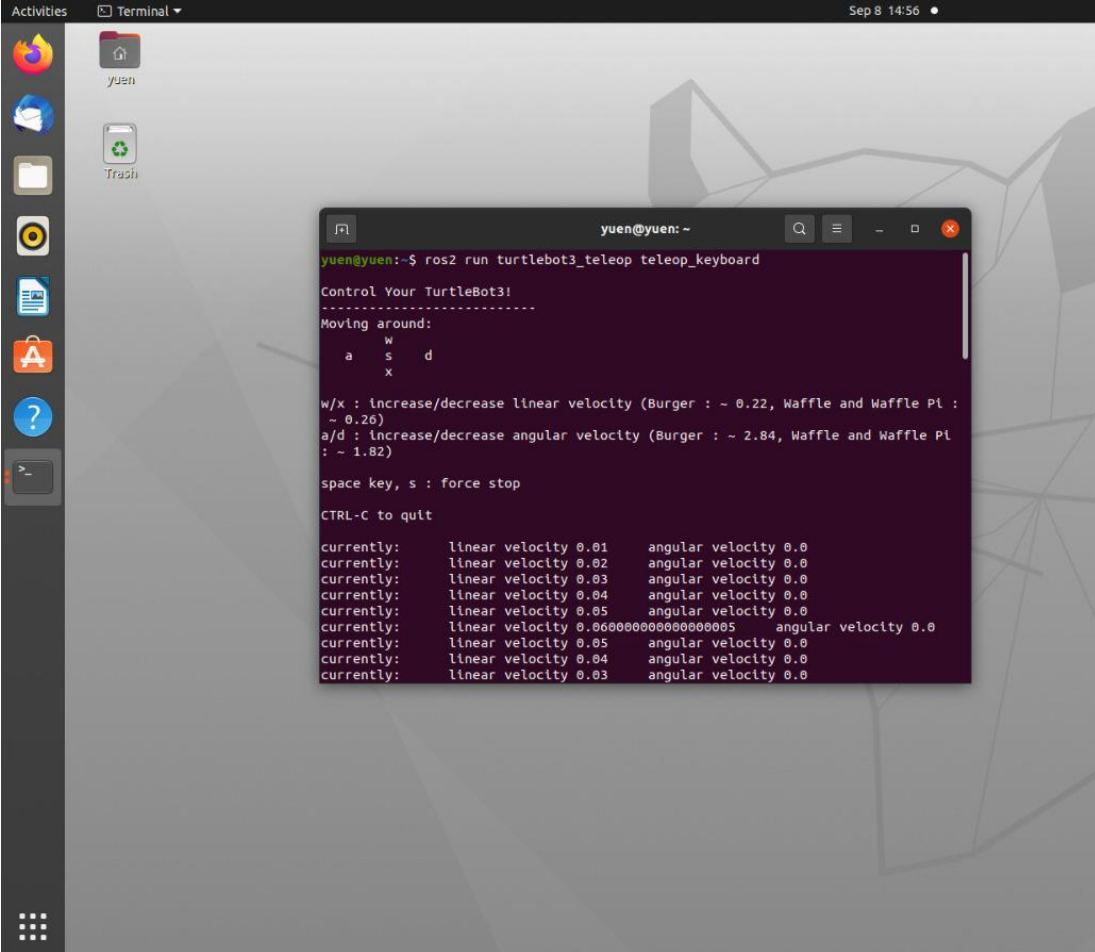


Figure 5.4.2 Successfully Bring up Turtlebot3.

## Basic Operation



```
Activities Terminal Sep 8 14:56
yuen
Trash

yuen@yuen: ~
yuen@yuen:~$ ros2 run turtlebot3_teleop teleop_keyboard

Control Your TurtleBot3!
-----
Moving around:
      w
    a   s   d
      x

w/x : increase/decrease linear velocity (Burger : ~ 0.22, Waffle and Waffle Pi :
~ 0.26)
a/d : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and Waffle Pi
: ~ 1.82)

space key, s : force stop

CTRL-C to quit

currently:   linear velocity 0.01   angular velocity 0.0
currently:   linear velocity 0.02   angular velocity 0.0
currently:   linear velocity 0.03   angular velocity 0.0
currently:   linear velocity 0.04   angular velocity 0.0
currently:   linear velocity 0.05   angular velocity 0.0
currently:   linear velocity 0.060000000000000005   angular velocity 0.0
currently:   linear velocity 0.05   angular velocity 0.0
currently:   linear velocity 0.04   angular velocity 0.0
currently:   linear velocity 0.03   angular velocity 0.0
```

Figure 5.4.3 Teleop\_Keyboard to Control Turtlebot3

Turtlebot3 can be controlled by various ways such as keyboard, RC-100 controller, PS3 joystick, XBOX 360 Joystick and so on. In my case, teleoperation node, keyboard has been used to control the Turtlebot3.

Key W and X: Increase and decrease linear velocity. Which are moving forward and backward.

Key A and D: Increase and decrease angular velocity. Which are turning left and right.

Key S to stop.

# SLAM

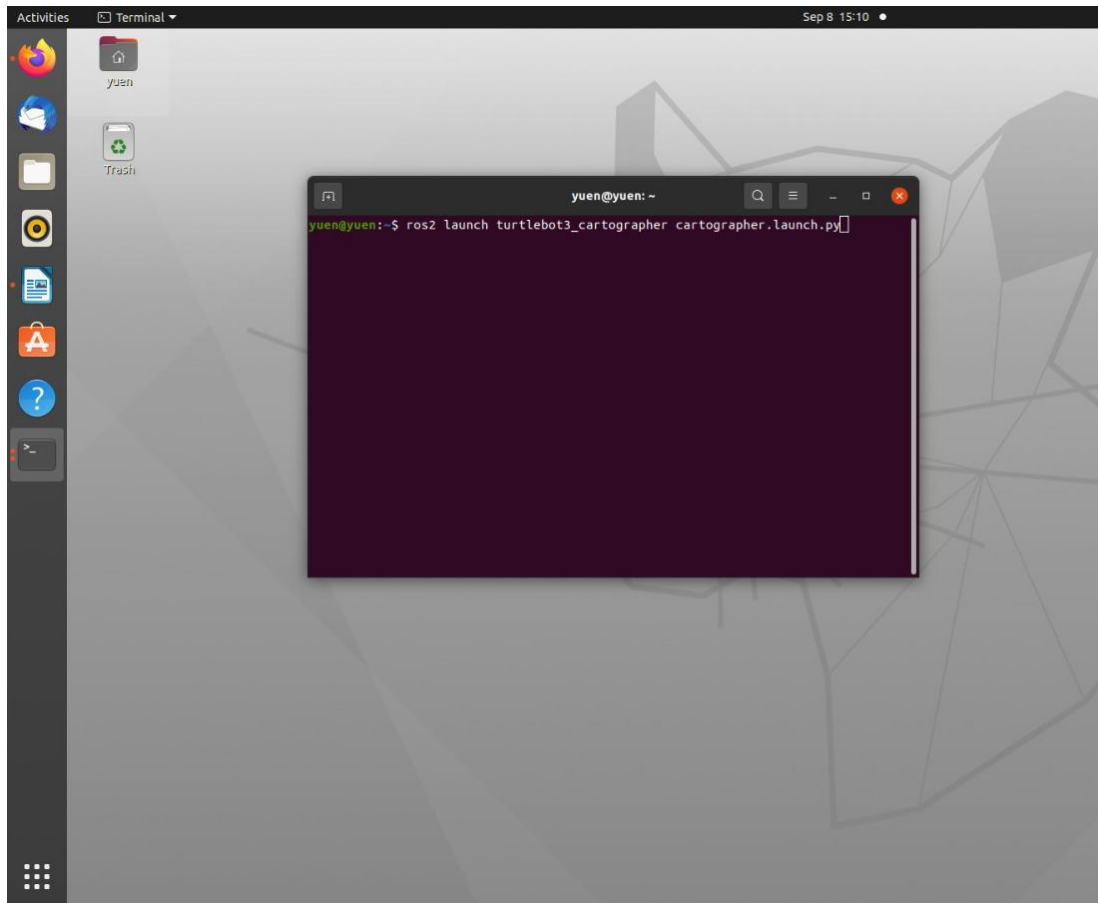


Figure 5.4.4 Command to launch SLAM Cartographer

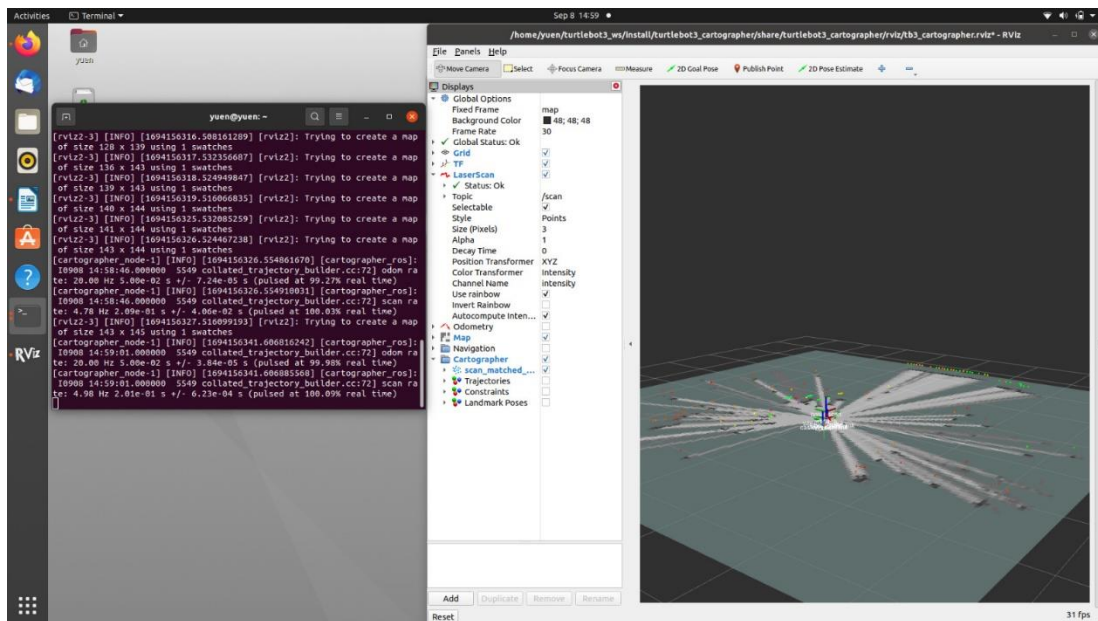


Figure 5.4.5 Running SLAM Cartographer

Ensuring the Turtlebot3 has been brought up, the SLAM node was launched. The cartographer is used as a default SLAM method.

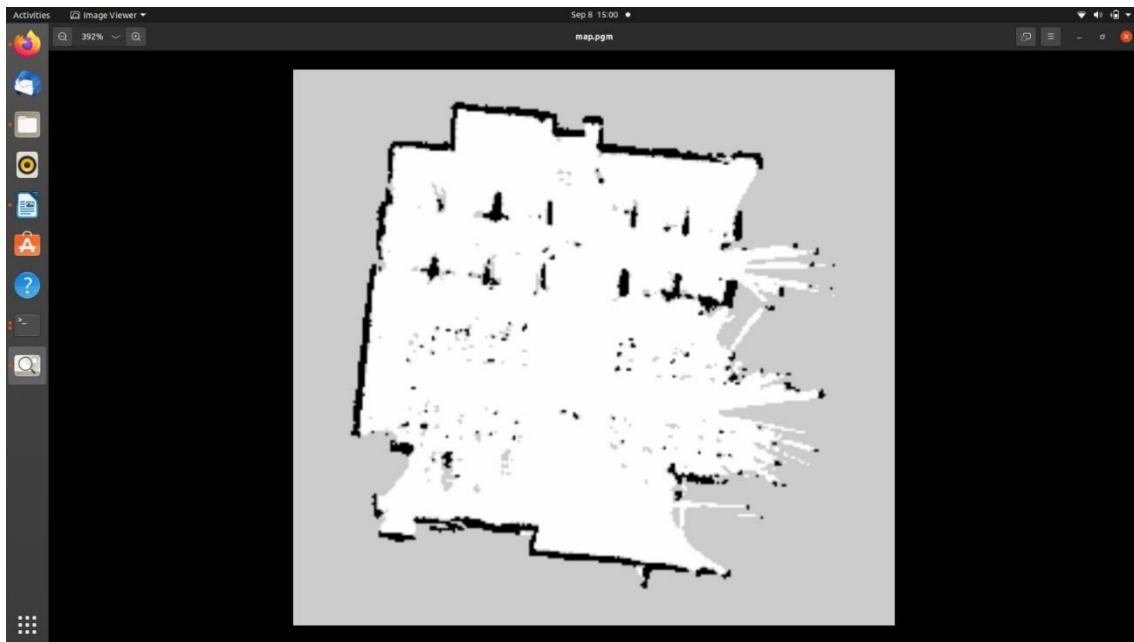


Figure 5.4.6 Map.yaml

Turtlebot3 will use teleoperation to explore an uncharted area of the map once the SLAM node is up and running. Teleoperation Node keyboard was used to control the Turtlebot3 for exploring and drawing the map. The map data is retrieved in the RVIZ window as the Turtlebot3 was moving.

After drawing the map by controlling the Turtlebot3 move around the room, `map_saver_cli` node was launched in the `nav2_map_server` package to create map files. The map was saved in the directory that hosts the `map_saver_cli` node.

## Navigation2

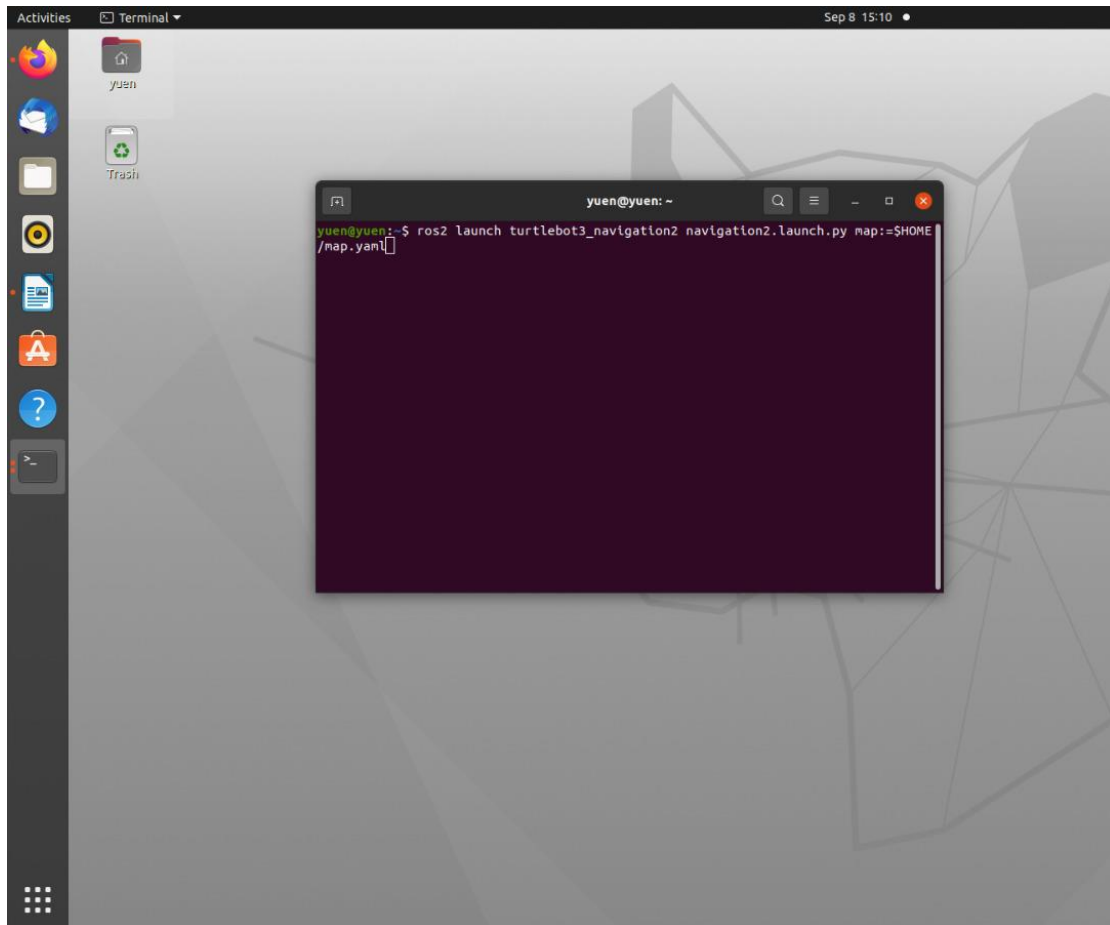


Figure 5.4.7 Command to launch Navigation2.

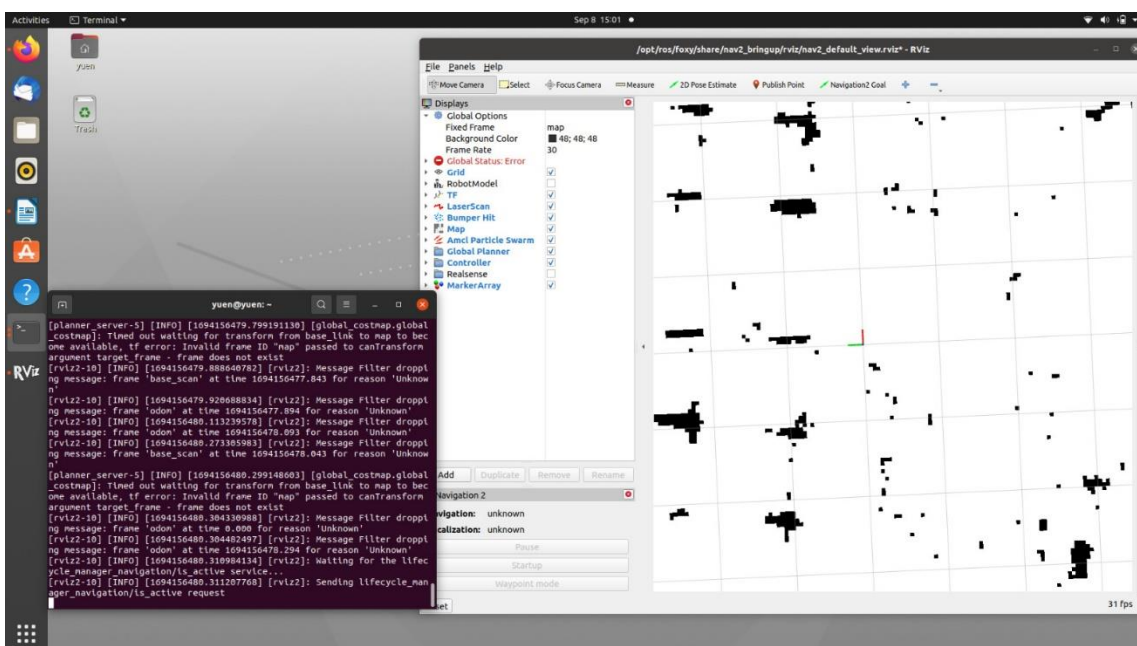


Figure 5.4.8 RVIZ Window

A robot can use the map and the navigation to move from its current pose to a planned goal pose. Ensuring the Turtlebot3 is brought up, the navigation node was launched. ROS2 uses Navigation2. Based on the global path planner, the robot will construct a route to the Navigation2 Goal. The robot then proceeds down the path. If a barrier is put in the way, Navigation2 will use a local path planner to get around it. The RVIZ window will show up.

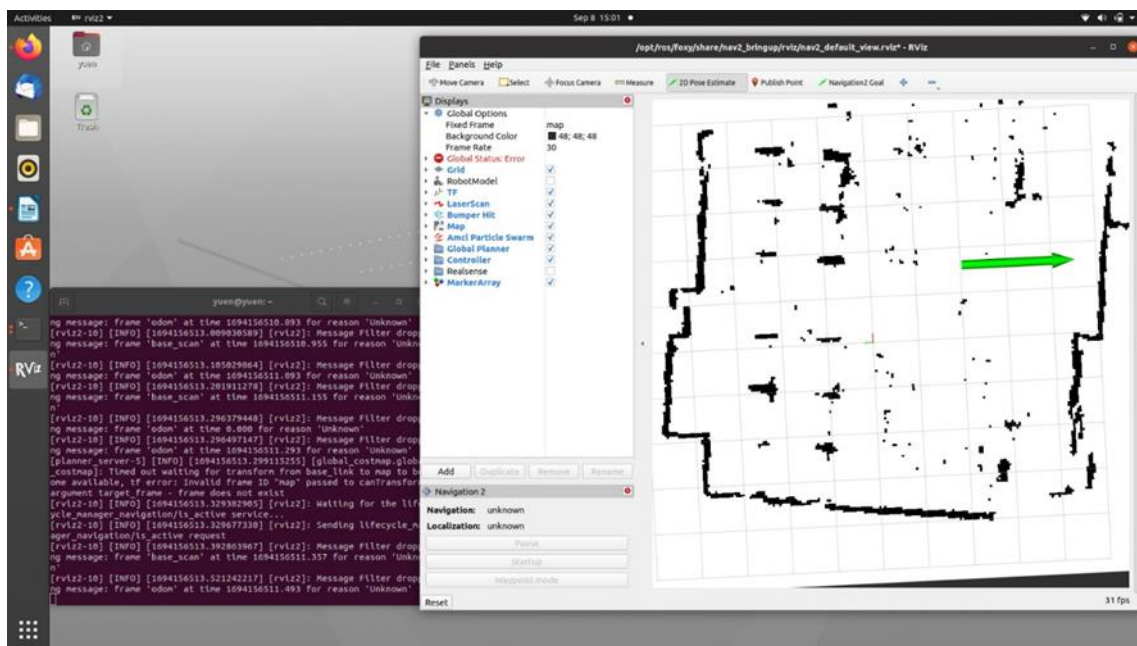


Figure 5.4.9 Estimating Initial Pose

Before running the navigation, initial pose estimation must be done as this will initialize the AMCL parameters. Turtlebot3 was correctly located on the map with LDS sensor that overlaps the map. The button 2D pose estimate was clicked in the RVIZ menu to estimate the Turtlebot3 initial pose. After the LDS sensor data overlaid on the map, the button Navigation2 Goal was clicked and then move the cursor and click on the map to set the destination of the Turtlebot3. The green indicates the direction where the Turtlebot3 will be facing.

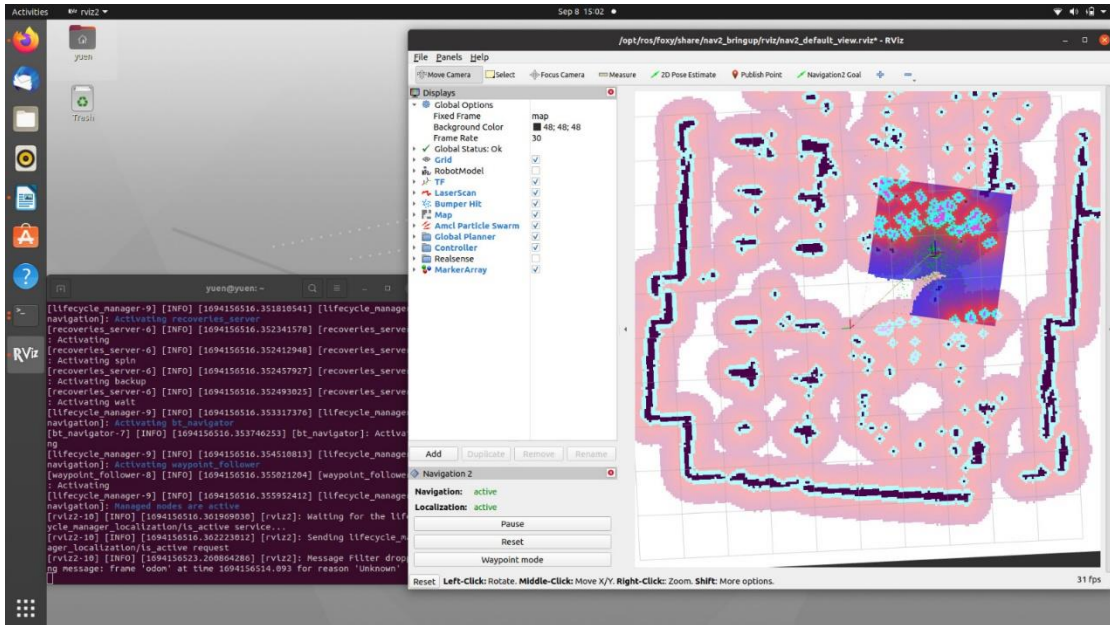


Figure 5.4.10 LDS sensor data overlays on the map.

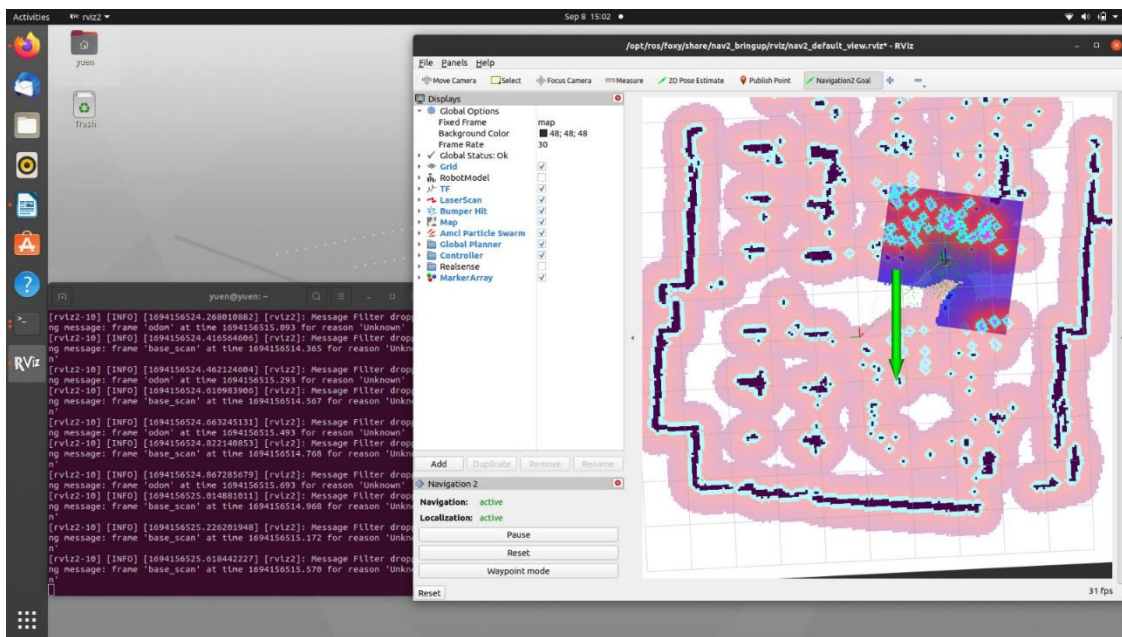


Figure 5.4.11 Set the destination of Turtlebot3.



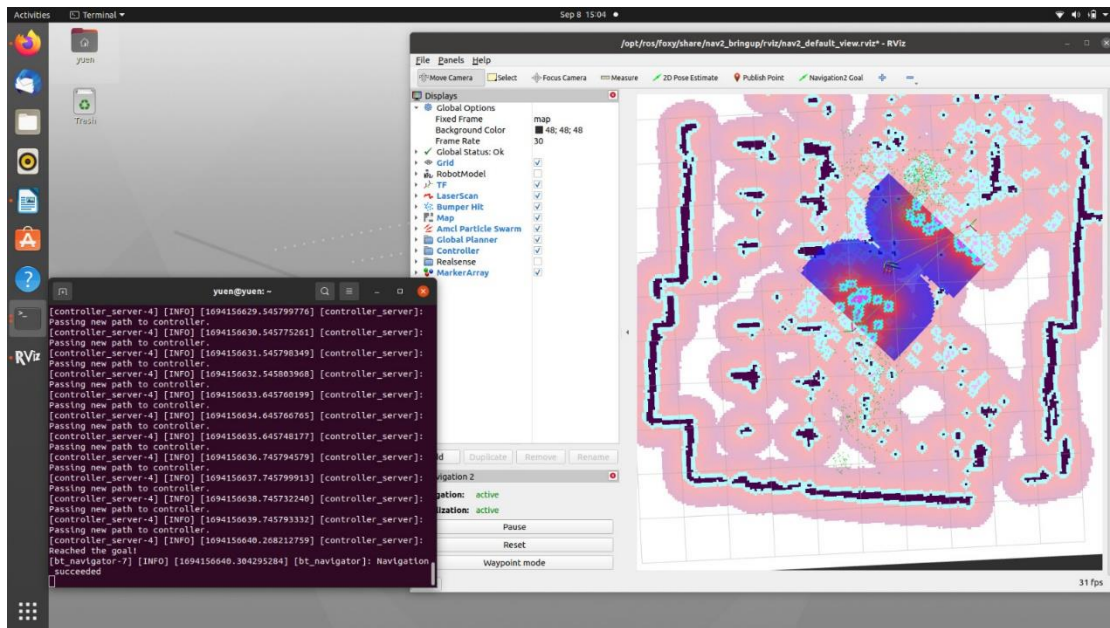


Figure 5.4.12 Turtlebot3 Navigation on RIVZ

## 5.5 Concluding Remark

Chapter 5 provides a thorough explanation of the hardware setup, software configuration, and system operation, offering a detailed account of the project's implementation phase. The hardware setup required the integration of different components, each playing a crucial role in the functionality of the TurtleBot3 robot. The Turtlebot3 Burger serves as the main platform, providing a solid foundation for attaching extra hardware components and carrying out navigation tasks with efficiency. Also included is the OpenCR 1.0, a ROS embedded system that provides open-source hardware and software support. Placed strategically on the third layer of the Turtlebot3 robot, the OpenCR allows for smooth communication and control among various system modules.

The Raspberry Pi Model 3b+ plays a crucial role in the hardware setup, acting as the central processing unit for the TurtleBot3 robot. With a range of sensors at its disposal, including LiDAR, IMU, depth sensor, and camera module, the Raspberry Pi proves essential in handling sensor data and carrying out important functions like detecting obstacles, planning paths, and making decisions. By incorporating three more Raspberry Pi Model 3b+ units, each linked to real-time sensors like the DHT22, MPU6050, and LDR module, the system's sensing capabilities are enhanced, allowing for thorough environmental monitoring and data collection. The sensor modules used include the DHT22, MPU6050, and LDR module, each offering unique functionalities for different sensing tasks. The DHT22 is well-known for its precise digital signal output, making it perfect for monitoring temperature and humidity in various

Bachelor of Information Technology (Honours) Computer Engineering  
Faculty of Information and Communication Technology (Kampar Campus), UTAR

environmental applications. The MPU6050 motion detector, with its advanced motion tracking capabilities, allows for precise movement detection and orientation tracking, essential for surveillance purposes. Additionally, the LDR module has the ability to detect changes in light intensity, improving environmental awareness and scene analysis, and allowing the system to adapt to different lighting conditions. The Pi Camera module is a versatile imaging solution that can capture both still images and high-definition video, complementing the sensor modules. Connected to the Raspberry Pi through the CSI interface, the Pi Camera module allows the TurtleBot3 robot to capture visual data and understand its environment more efficiently.

During system operation, various functionalities such as basic operation, SLAM, and Navigation2 are explored to demonstrate the capabilities of the TurtleBot3 robot. Through teleoperation using a keyboard interface, the robot's movement and mobility are tested, showcasing its real-time navigation and interaction skills with the environment. Additionally, SLAM techniques are used to create a map of the robot's surroundings and accurately determine its position within the environment. Navigation2 algorithms are then used to plan and execute optimal paths to predefined goals, demonstrating the robot's ability to navigate autonomously. Finally, the successful execution of the hardware setup, software configuration, and system operation marks an important milestone in achieving the project's goals.

# Chapter 6

## System Evaluation and Discussion

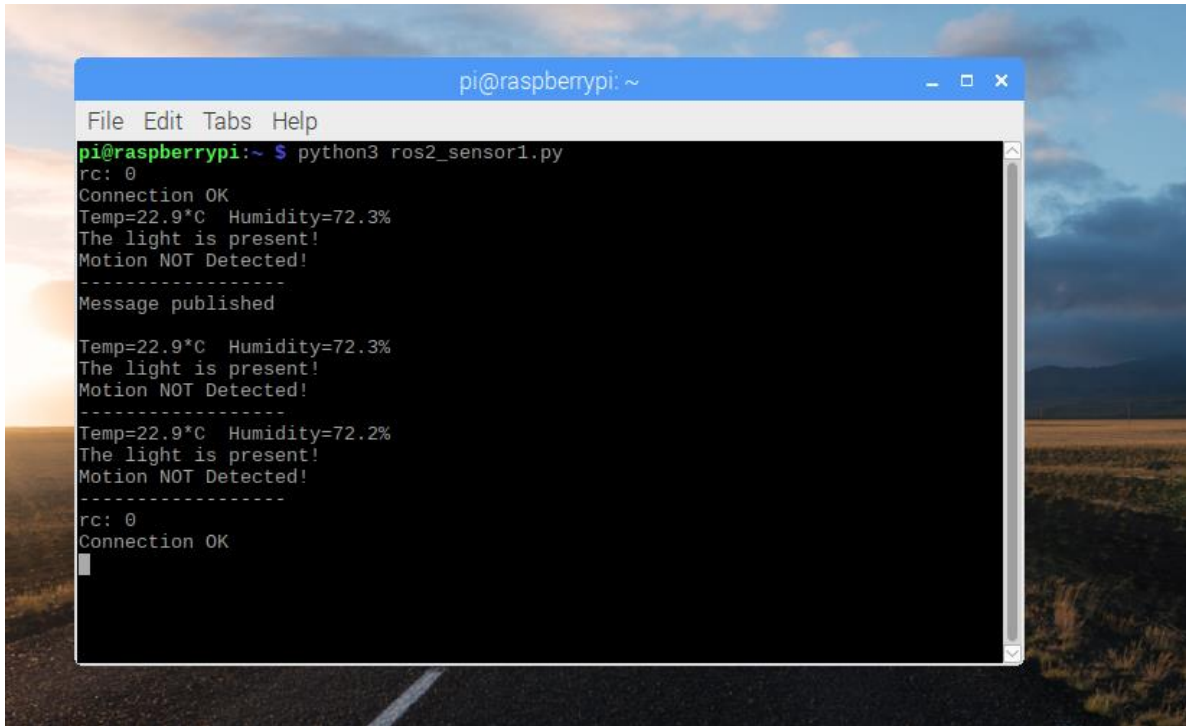
### 6.1 Testing Setup and Result

The testing setup was designed to validate the fulfilment of all criteria and requirements specified in the project specifications. The test scenarios were designed to encompass a diverse array of probable use cases and operational circumstances that the system may meet in real-world applications. The development of these scenarios was guided by the project requirements, objectives, and expected user demands. The tasks encompassed sensor data collection under diverse environmental conditions, including fluctuating temperatures and humidity levels, real-time image capture in varying lighting conditions, navigation in obstacle-laden environments, and system control operations such as remote monitoring and command execution. The testing arrangement included a wide range of scenarios to thoroughly evaluate the system's capabilities and robustness.

Key performance measures have been established for evaluating the system's performance in several aspects. These measurements included many aspects such as accuracy, reliability, speed, resource allocation, and user satisfaction. Every statistic was meticulously selected to correspond with precise project objectives and criteria, offering significant insights into the effectiveness and efficiency of the system. Testers can objectively evaluate the overall performance of the system and discover areas for improvement or optimisation by monitoring performance against established metrics.

#### **Test Case 1: Sensor Data Collection**

Evaluate the accuracy and reliability of temperature and humidity readings captured by the DHT22 sensor module in different environmental conditions (e.g., varying temperatures and humidity levels). At the same time, evaluate the effectiveness of motion detection and orientation tracking by the MPU6050 sensor module. Also, assess the sensitivity and responsiveness of the LDR module in detecting changes in light intensity across different lighting conditions (e.g., bright sunlight, dimly lit room).

A terminal window titled 'pi@raspberrypi: ~' with a menu bar (File, Edit, Tabs, Help) and window controls. The terminal output shows the execution of 'python3 ros2\_sensor1.py'. The output includes: 'rc: 0', 'Connection OK', 'Temp=22.9°C Humidity=72.3%', 'The light is present!', 'Motion NOT Detected!', '-----', 'Message published', 'Temp=22.9°C Humidity=72.3%', 'The light is present!', 'Motion NOT Detected!', '-----', 'Temp=22.9°C Humidity=72.2%', 'The light is present!', 'Motion NOT Detected!', '-----', 'rc: 0', and 'Connection OK'. The terminal is overlaid on a background image of a sunset over a field.

```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ python3 ros2_sensor1.py
rc: 0
Connection OK
Temp=22.9°C Humidity=72.3%
The light is present!
Motion NOT Detected!
-----
Message published
Temp=22.9°C Humidity=72.3%
The light is present!
Motion NOT Detected!
-----
Temp=22.9°C Humidity=72.2%
The light is present!
Motion NOT Detected!
-----
rc: 0
Connection OK
```

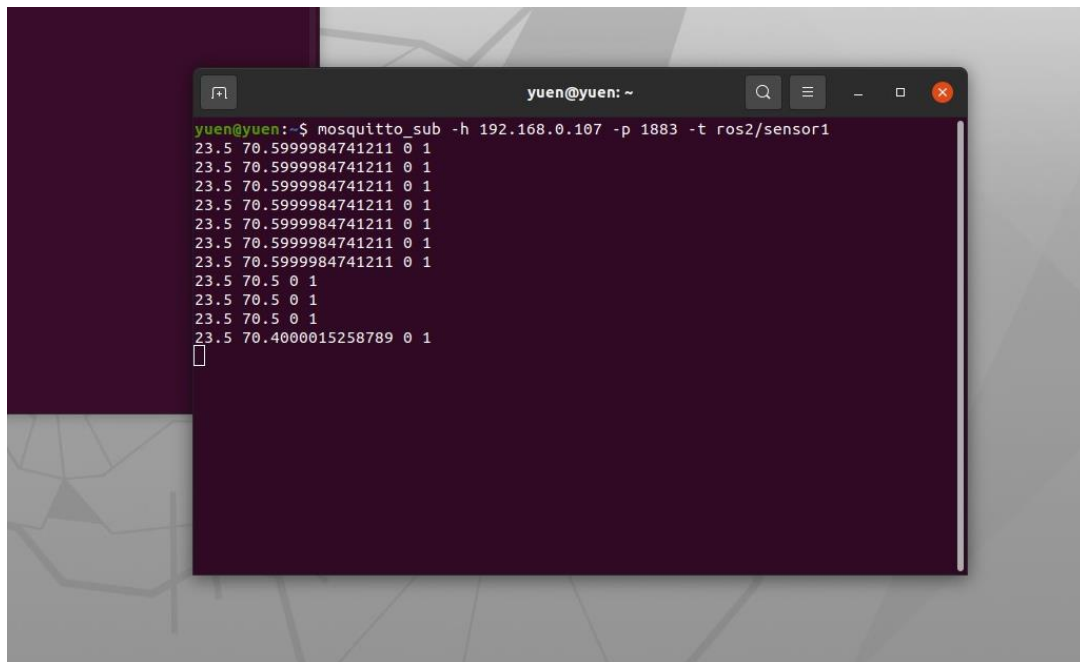
Figure 6.1.1 Testing for Sensor Data Collection

Test results:

All sensor data were accurately measured and detected.

### Test Case 2: MQTT Integration

Several test cases were used to assess MQTT performance, including the system's capacity to send sensor data effectively and efficiently. First, the Raspberry Pi's ability to send sensor data signals over MQTT to the user's PC or laptop was evaluated. This result in ensuring that sensor data was communicated reliably and in real time via the MQTT protocol, hence ensuring information delivery on time. Furthermore, the reliability and robustness of MQTT message delivery were assessed to guarantee that no messages are lost or duplicated during transmission.

A terminal window titled 'yuen@yuen: ~' with search, menu, and window control icons. The terminal shows the command 'mosquitto\_sub -h 192.168.0.107 -p 1883 -t ros2/sensor1' and its output: a series of 14 lines of sensor data. The first 10 lines have a timestamp of 23.5, a value of 70.5999984741211, and a status of 0 1. The next 3 lines have a timestamp of 23.5, a value of 70.5, and a status of 0 1. The final line has a timestamp of 23.5, a value of 70.4000015258789, and a status of 0 1.

```
yuen@yuen:~$ mosquitto_sub -h 192.168.0.107 -p 1883 -t ros2/sensor1
23.5 70.5999984741211 0 1
23.5 70.5999984741211 0 1
23.5 70.5999984741211 0 1
23.5 70.5999984741211 0 1
23.5 70.5999984741211 0 1
23.5 70.5999984741211 0 1
23.5 70.5999984741211 0 1
23.5 70.5999984741211 0 1
23.5 70.5 0 1
23.5 70.5 0 1
23.5 70.5 0 1
23.5 70.5 0 1
23.5 70.4000015258789 0 1

```

Figure 6.1.2 Testing for MQTT Integration

Test results:

MQTT communication between the Raspberry Pi and the central control system is successful. Test messages are reliably published and received, demonstrating robust communication.

Test Case 3: Camera Functionality:

To test the camera functionality, the system needs to ensure that the camera node is running and capable of publishing images to the 'image\_raw' topic. Additionally, it should be able to subscribe to this topic and display the captured images using 'rqt\_image\_view'.

```
ubuntu@ubuntu: ~
Last login: Thu Apr 25 03:07:07 2024 from 192.168.0.125
ubuntu@ubuntu:~$ ros2 run v4l2_camera v4l2_camera_node --ros-args -p image_size:
="[640,480]" -p camera_frame_id:=camera_link_optical
[INFO] [1714112784.568047332] [v4l2_camera]: Driver: bm2835 mmal
[INFO] [1714112784.568682081] [v4l2_camera]: Version: 328782
[INFO] [1714112784.568834271] [v4l2_camera]: Device: mmal service 16.1
[INFO] [1714112784.568926199] [v4l2_camera]: Location: platform:bcm2835-v4l2
[INFO] [1714112784.569015576] [v4l2_camera]: Capabilities:
[INFO] [1714112784.569157088] [v4l2_camera]:   Read/write: YES
[INFO] [1714112784.569247506] [v4l2_camera]:   Streaming: YES
[INFO] [1714112784.569396102] [v4l2_camera]:   Current pixel format: JPEG @ 1024x7
68
[WARN] [1714112784.569872672] [v4l2_camera]: Listing of frame intervals for non-
discrete image sizes is not currently supported
[WARN] [1714112784.570009903] [v4l2_camera]: Listing of frame intervals for non-
discrete image sizes is not currently supported
[WARN] [1714112784.570091790] [v4l2_camera]: Listing of frame intervals for non-
discrete image sizes is not currently supported
[WARN] [1714112784.570204239] [v4l2_camera]: Listing of frame intervals for non-
discrete image sizes is not currently supported
[WARN] [1714112784.570291220] [v4l2_camera]: Listing of frame intervals for non-
discrete image sizes is not currently supported
[WARN] [1714112784.570383513] [v4l2_camera]: Listing of frame intervals for non-
discrete image sizes is not currently supported
```

Figure 6.1.3 Test Run Camera Node

```
ubuntu@ubuntu: ~
Last login: Thu Apr 25 03:07:07 2024 from 192.168.0.125
ubuntu@ubuntu:~$ ros2 run v4l2_camera v4l2_camera_node --ros-args -p image_size:
="[640,480]" -p camera_frame_id:=camera_link_optical
[INFO] [1714112784.568047332] [v4l2_camera]: Driver: bm2835 mmal
[INFO] [1714112784.568682081] [v4l2_camera]: Version: 328782
[INFO] [1714112784.568834271] [v4l2_camera]: Device: mmal service 16.1
[INFO] [1714112784.568926199] [v4l2_camera]: Location: platform:bcm2835-v4l2
[INFO] [1714112784.569015576] [v4l2_camera]: Capabilities:
[INFO] [1714112784.569157088] [v4l2_camera]:   Read/write: YES
[INFO] [1714112784.569247506] [v4l2_camera]:   Streaming: YES
[INFO] [1714112784.569396102] [v4l2_camera]:   Current pixel format: JPEG @ 1024x7
68
[WARN] [1714112784.569872672] [v4l2_camera]: Listing of frame intervals for non-
discrete image sizes is not currently supported
[WARN] [1714112784.570009903] [v4l2_camera]: Listing of frame intervals for non-
discrete image sizes is not currently supported
[WARN] [1714112784.570091790] [v4l2_camera]: Listing of frame intervals for non-
discrete image sizes is not currently supported
[WARN] [1714112784.570204239] [v4l2_camera]: Listing of frame intervals for non-
discrete image sizes is not currently supported
[WARN] [1714112784.570291220] [v4l2_camera]: Listing of frame intervals for non-
discrete image sizes is not currently supported
[WARN] [1714112784.570383513] [v4l2_camera]: Listing of frame intervals for non-
discrete image sizes is not currently supported

yuen@yuen: ~$ ros2 topic list
/camera_info
/image_raw
/image_raw/compressed
/image_raw/compressedDepth
/image_raw/theora
/parameter_events
/rosout
yuen@yuen:~$
```

Figure 6.1.4 Checking ROS Topic List

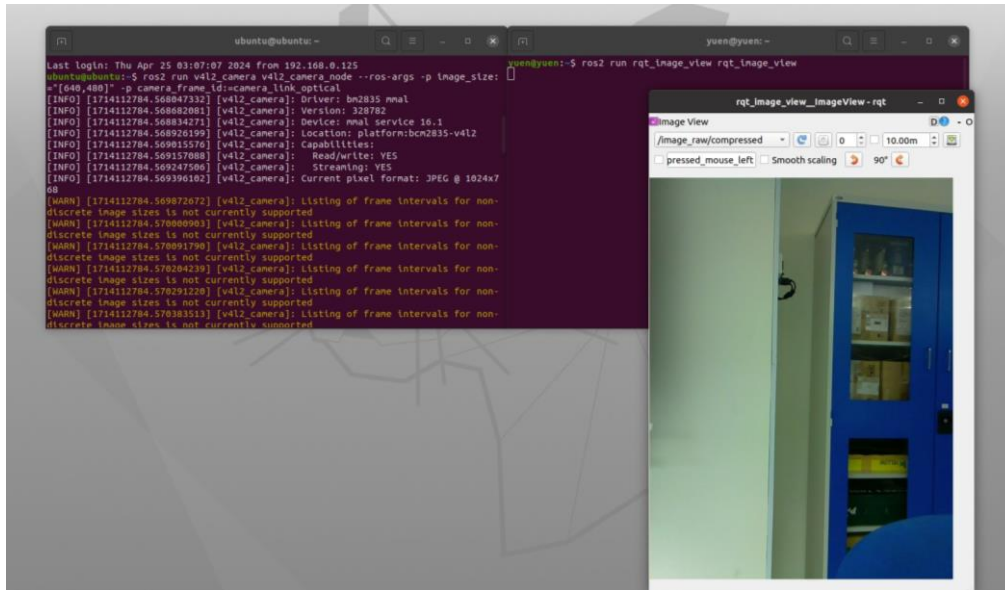


Figure 6.1.5 Subscribing to 'image\_raw' Topic and View Image

Test result:

The camera node was successfully running on the Raspberry Pi, as confirmed by checking the list of active ROS topics. The "rqt\_image\_view" tool successfully subscribes to the "image\_raw" topic and displays the captured images in real-time. This demonstrates that the system is capable of receiving and processing images from the camera node effectively.

#### Test case 4: Mapping and Navigation

Test the accuracy and reliability of SLAM (Simultaneous Localization and Mapping) algorithms by launching the drawn map on Navigation2 RVIZ. Evaluate the effectiveness of path planning algorithms in generating optimal navigation paths to three goals which were predefined in Python script.

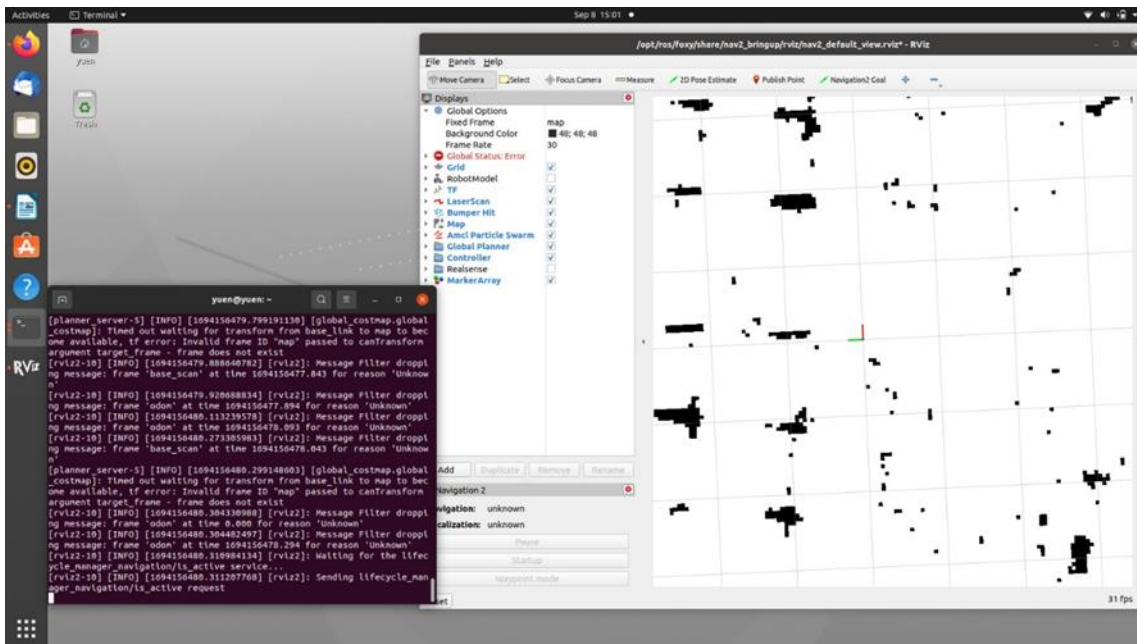


Figure 6.1.6 Launching the Navigation2 with Existing Map

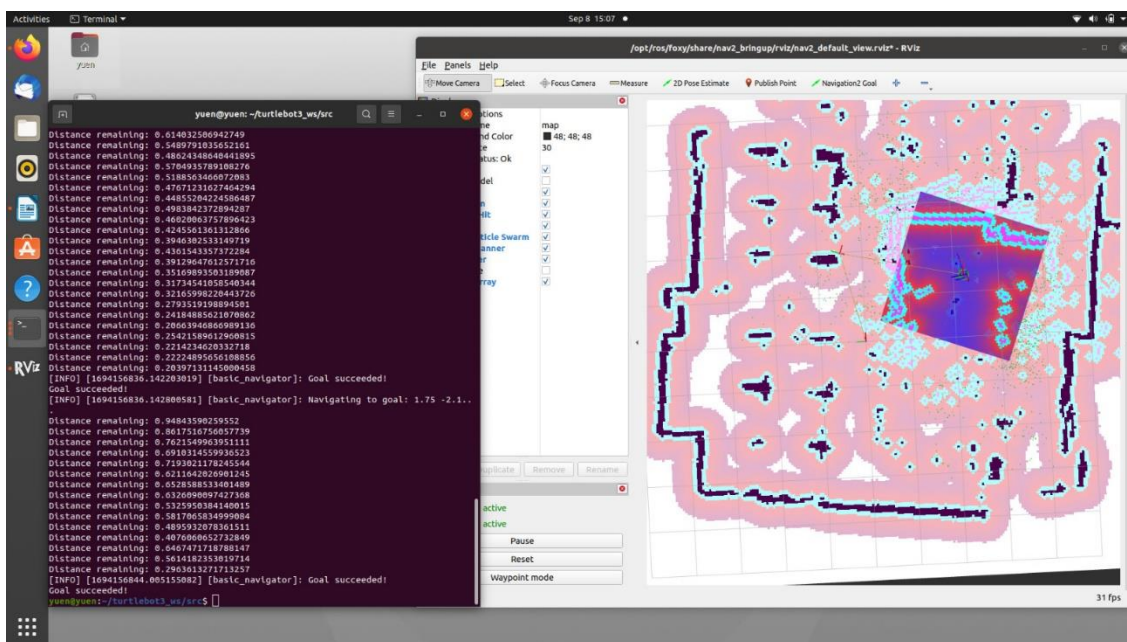


Figure 6.1.7 Successfully Navigated to Three Locations by Running Python Script

Test results:

SLAM was successfully used to draw the map, and it was able to navigate to selected areas independently on its own.



### Test case 5: System Control and Visualization

The system was able to integrate with the mobile Telegram app. The user can access sensor data and view photo on both the GUI and the Telegram app by running commands. To reset sensors, the user can also use the GUI's reset button or the /reset command on Telegram.



Figure 6.1.8 Displaying Sensor Data On GUI



Figure 6.1.9 Showing Sensor Data on Telegram App

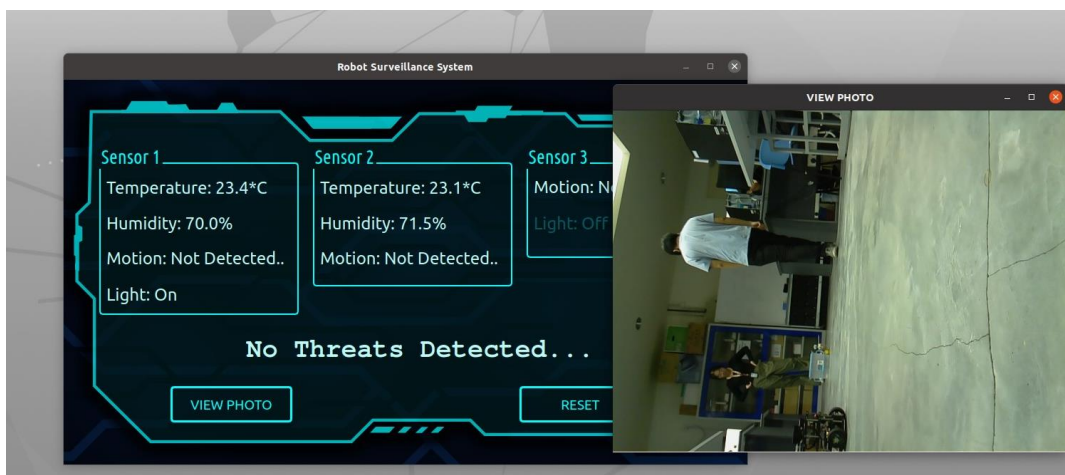


Figure 6.1.10 Viewing Image on GUI by Clicking “View Image” Button

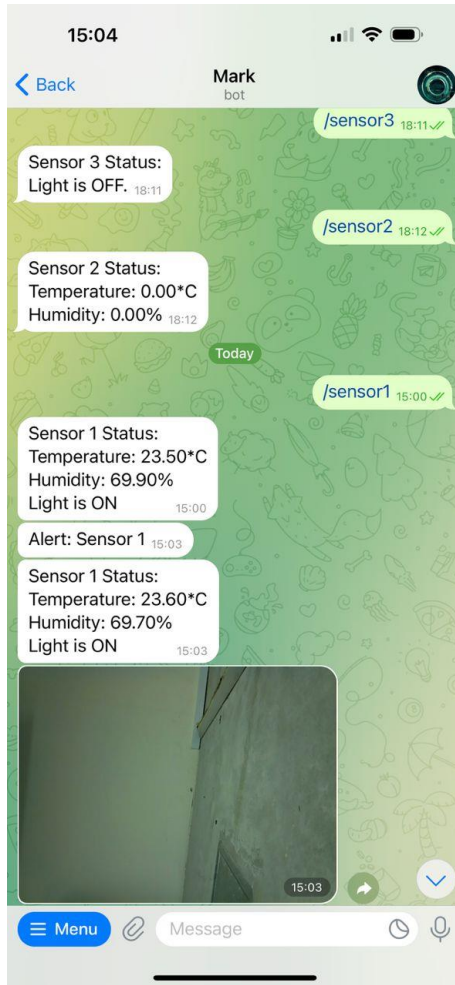


Figure 6.1.11 Viewing Image on Telegram



Figure 6.1.12 Reset Sensors Using Telegram Command

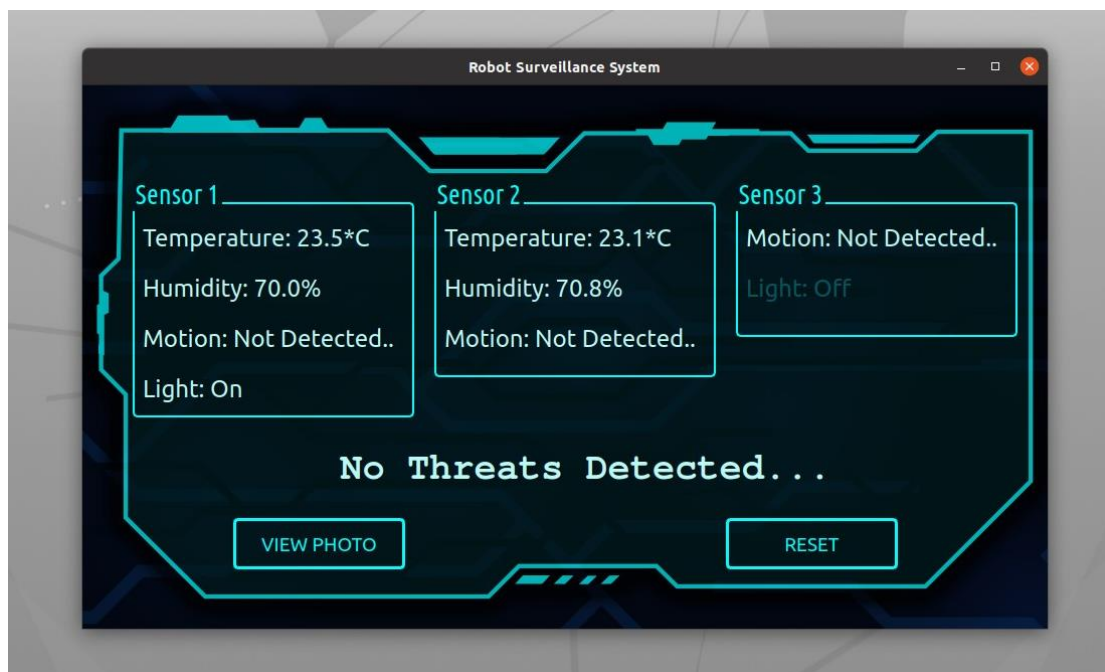


Figure 6.1.13 Successfully Reset Sensors

Test results:

The system successfully demonstrated integration with the mobile Telegram app. Sensor data was easily accessible through both the graphical user interface (GUI) on the PC or laptop and the Telegram app installed on a mobile device. The GUI displayed real-time sensor data accurately, providing users with immediate access to essential information about the environment. Similarly, commands sent via the Telegram app to request sensor data for example (/sensor1) were promptly processed. This ensured seamless communication between the system and the mobile platform.

Furthermore, the system's functionality for viewing images was thoroughly evaluated. Upon clicking the "View Image" button on the GUI interface, the captured image from the camera module was displayed correctly, indicating the system's ability to process and render visual data effectively. Additionally, images being sent and displayed on the mobile device without any issues. In terms of resetting sensors, both the GUI and the Telegram app provided convenient options for users to reset sensor data. Clicking the "Reset" button on the GUI interface effectively cleared sensor data from the display, demonstrating smooth operation and responsiveness. Similarly, sending a reset command (/reset) via the Telegram app resulted in successful sensor resets, with the system promptly clearing data and preparing for new inputs.

## **6.2 Project Challenges**

One major challenge that arose during project execution was the problem of navigation accuracy, especially when multiple TurtleBot3 robots were using the same network connection and operating in the same environment. The TurtleBot 3's navigation system uses a variety of sensors and algorithms to map its environment and conduct autonomous navigation. Nonetheless, a number of difficulties arise that may affect the precision of navigation when numerous robots are navigating in close areas.

Interference between robots is one of the most significant issues, particularly in circumstances where communication channels overlap. Multiple robots transmitting data and commands over the same network connection may cause signal interference, resulting in communication delays or packet loss. This interference can break the synchronization of robots and the central control system, resulting in inconsistencies in navigation commands and sensor data processing. Furthermore, in locations with limited space or complex layouts, many robots may unintentionally block each other's pathways, resulting in collisions or navigational problems. TurtleBot3's navigation algorithms are meant to avoid obstacles and determine ideal courses

using sensor data and prepared maps. However, the presence of additional robots can bring unforeseen obstructions or dynamic changes to the environment, causing a challenge to the robot's navigation accuracy.

To address these issues, precise coordination and synchronization among robots are required. Advanced communication protocols and collision avoidance tactics can assist reduce interference and disputes among robots. Furthermore, optimizing resource allocation and prioritizing key activities can boost overall system performance in multi-robot scenarios. Regardless of these issues, solving navigation accuracy in situations with several TurtleBot3 robots is critical to assuring the surveillance system's effectiveness and reliability. By recognizing and addressing these challenges, the project can realize its goals of seamless navigation and coordination among several robots in a connected home environment.

### **6.3 Objectives Evaluation**

#### **1. Develop and implement the robotic system:**

The project successfully created and put into place a reliable TurtleBot3-based surveillance system. For instance, by combining OpenCR 1.0 with the TurtleBot3 Burger model, the system showed flexibility and adaptability to operate in a variety of home settings. Moreover, the effective implementation of ROS2 as the fundamental software framework enabled smooth communication and coordination between system elements, guaranteeing effective functioning. The robotic system successfully avoided obstructions and reacted to sensor inputs as it moved around a variety of rooms in the connected home during testing. The system's adaptability to shifting ambient factors, such moving furniture or altering lighting levels, demonstrated its usefulness and dependability.

#### **2. Achieve autonomous navigation:**

The project made significant progress toward autonomous navigation capabilities for the TurtleBot3 robot by implementing robust path planning algorithms and obstacle avoidance techniques. Using sensor data and real-time feedback, the system successfully discovered ideal pathways and modified the path to avoid obstructions.

During testing, the TurtleBot3 robot successfully navigated through areas, constantly changing its path to find the best route to reach the designated locations.

#### **3. Integrate real-time sensor data:**

One of the major objectives of the project was the integration of real-time sensor data, which made it possible for the surveillance system to actively monitor and react to changes in the surrounding environment. It was possible for the system to efficiently gather and analyze data in order to improve situational awareness. This was accomplished by strategically putting sensors such as the DHT22, MPU6050, and LDR modules respectively. An accurate detection of motion events, monitoring of the state of the light, and measurement of environmental characteristics such as temperature and humidity were all achieved by the system throughout the testing phase. The system was able to activate suitable responses, such as collecting photographs or sending notifications to the user's smartphone, based on predetermined thresholds or criteria through the implementation of real-time analysis of sensor data.

#### 4. Image capture and transmission:

The project effectively implemented dependable mechanisms for capturing and sending live photos within the interconnected home environment. By combining the Pi Camera module with the Raspberry Pi, the system successfully acquired high-resolution photographs under different lighting circumstances and effortlessly communicated them to the central control system. During operational testing, the Pi Camera module successfully captured images of particular regions inside the house, offering visual verification of security incidents or changes in the environment. The photos were sent without the need of wires to the central control system, where they could be immediately accessed and examined, therefore improving the total capability to monitor and observe.

#### 5. Integrate communication with user's smartphone:

The integration of the surveillance system with the Telegram messaging app enabled smooth communication between the system and the user's smartphone, hence improving user satisfaction and system efficiency. Through the utilisation of Telegram, the system facilitated the transmission of notifications, real-time updates, images, and commands, hence delivering timely information to users and allowing for swift responses to security occurrences. During the testing phase, the system effectively transmitted notifications to the user's smartphone using Telegram in reaction to security events, such as detecting motion. Users can engage with the system by issuing orders or requesting information using the Telegram app, showcasing the system's ability to facilitate two-way conversation and improve user involvement.

## 6.4 Concluding Remark

Chapter 6 involved a comprehensive assessment and examination of the system to verify its performance and efficiency in achieving project goals. The testing arrangement was carefully planned to cover a broad range of scenarios, guaranteeing a thorough evaluation of the system's capacities in various operational conditions. The test cases were precisely planned to test the gathering of sensor data, integration of MQTT, functionality of the camera, mapping and navigation capabilities, system control, and visualisation. The system exhibited resilience, dependability, and effectiveness across a range of tasks, as evidenced by thorough testing. During the testing phase, the system demonstrated a high level of accuracy and dependability in collecting sensor data. It effectively detected changes in temperature, humidity, motion, and light intensity. MQTT integration enabled smooth communication and instantaneous data transmission between system components, guaranteeing efficient information delivery without any loss or duplication. The camera functionality was extensively assessed, and the system successfully demonstrated its capability to capture and send images in real-time. This allows for visual verification of security incidents or environmental alterations. The system completed testing to evaluate its mapping and navigation capabilities. It successfully utilised SLAM techniques to create maps and autonomously navigate to predetermined destinations. The project revealed challenges related to the accuracy of navigation in multi-robot scenarios, emphasising the necessity of exact coordination and synchronisation among robots to guarantee smooth operation.

Integration with the Telegram messaging app enabled user-friendly communication, allowing users to access sensor data, view images, and issue commands conveniently from their smartphones. The system's ability to provide real-time updates, notifications, and two-way communication enhanced user engagement and satisfaction.

Overall, the project successfully achieved its objectives, demonstrating the development and implementation of a practical and versatile surveillance system for connected home integration. By addressing challenges and optimizing system performance, the project lays the groundwork for revolutionizing home security and situational awareness in modern connected environments.



# Chapter 7

## Conclusion and Recommendation

### 7.1 Conclusion

Overall, the "Robot Surveillance for Connected Home Integration" project has made significant advancements in the field of home security and situational awareness. By adopting methodology in design, development, and testing, the project has effectively deployed an advanced surveillance system that can adjust to the constantly evolving characteristics of modern connected homes. Through the use of the adaptable TurtleBot3 platform, integration of live sensor data, and establishment of seamless connection with users through the Telegram messaging app, this project has established the groundwork for a revolutionary era of intelligent home security systems.

During the project, a wide range of objectives were pursued, each aimed at improving home security and user experience. Every part of the project, from the creation and construction of the robotic system to the accomplishment of autonomous navigation and integration of real-time sensor data, has been methodically completed to ensure the highest standards of performance and dependability. The testing phase confirmed and strengthened the system's skills in many situations, offering significant knowledge on its effectiveness and adaptability in real-life settings.

Although facing difficulties, such as the accurateness of navigation in scenarios involving multiple robots, the project has highlighted crucial areas for enhancement and offered significant suggestions for resolving them. The project aims to promote continued innovation and improvement in the field of home surveillance systems by focusing on precise coordination, optimisation of communication protocols, and research of advanced technologies.

### 7.2 Recommendation

In anticipation of the future, there are various suggestions that can be put out to further improve the project's influence and efficiency. Continuous testing and refining are necessary for optimising performance and addressing identified issues in the project. This may require optimising navigation algorithms, improving communication protocols, and perfecting sensor integration approaches.

While current sensors provide a good foundation for object detection, the actual potential is found in investigating advanced technologies. Machine learning and computer vision can

greatly improve the system's capabilities. By adding these techniques, the system might proceed beyond simple motion detection and into robust object identification. This would allow the system to distinguish between harmless motions and potential security threats, resulting in more targeted reactions and increased overall security.

Ongoing collaboration among stakeholders is vital for the success of effective robot surveillance systems. Continuously engaging with users and other interested parties enables us to gather valuable input on the system's functionality and evolving needs. By actively seeking feedback and understanding user expectations, the system can be customized to address real-world security concerns. Engaging with users in a proactive manner encourages creativity and helps to keep the system up-to-date and effective in addressing security challenges, both now and in the future.

The project should be developed with scalability and flexibility as key considerations, enabling seamless interaction with supplementary sensors, devices, and platforms. This will facilitate the system's ability to adjust to evolving circumstances and handle future growth. Security and Privacy Considerations: Due to the involvement of sensitive data gathering and transmission in the project, it is necessary to establish strong security and privacy measures to safeguard user information and prevent unauthorised access. This encompasses encryption mechanisms, access controls, and data anonymization techniques. By adopting these suggestions and persisting in their pursuit of innovation, the "Robot Surveillance for Connected Home Integration" initiative has the potential to lead to progress in home security and situational awareness. This, in turn, will ultimately improve the safety, convenience, and peace of mind of homeowners throughout the globe.

## REFERENCES

- [1]. R. Safin, R. Lavrenow, K. Hsia, E. Maslak, N. Schiefermeier and E. Magid, "Modelling a TurtleBot3 Based Delivery System for Smart Hospital in Gazebo," in *2021 International Siberian Conference on Control and Communications (SIBCON)*. IEEE, 2021, pp. 2.
- [2]. J. Bohren and S. Cousins, "The smach high-level executive [ros news]," *IEEE Robotics Automation Magazine*, vol. 17, no. 4, pp. 18–20, 2010.
- [3]. R. Mishra and A. Javed, "Ros based service robot platform," in *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*, 2018, pp. 55–59.
- [4]. S. Putz, J. Santos Sim ˆ on, and J. Hertzberg, "Move base flex a highly ' flexible navigation framework for mobile robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3416–3421.
- [5]. R. Lavrenov, F. Matsuno, and E. Magid, "Modified spline-based navigation: guaranteed safety for obstacle avoidance," in *International Conference on Interactive Collaborative Robotics*. Springer, 2017, pp. 123–133
- [6]. E. Magid, R. Lavrenov, and I. Afanasyev, "Voronoi-based trajectory optimization for ugv path planning," in *2017 International Conference on Mechanical, System and Control Engineering (ICMSC)*. IEEE, 2017, pp. 383–387.
- [7]. R. Yagfarov, M. Ivanou, and I. Afanasyev, "Map comparison of lidarbased 2d slam algorithms using precise ground truth," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2018, pp. 1979–1983.
- [8]. E. Mingachev, R. Lavrenov, T. Tsoy, F. Matsuno, M. Svinin, J. Suthakorn, and E. Magid, "Comparison of ros-based monocular visual slam methods: Dso, Idso, orb-slam2 and dynaslam," in *International Conference on Interactive Collaborative Robotics*. Springer, 2020, pp. 222–233.
- [9]. C. Rosmann, F. Hoffmann, and T. Bertram, "Integrated online trajec- ˆ tory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.
- [10]. D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 709–715.
- [11]. M. Ester, H.-P. Kriegel, J. Sander, X. Xu et al., "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.
- [12]. J.-S. Park and S.-J. Oh, "A new concave hull algorithm and concaveness measure for n-dimensional datasets," *Journal of Information science and engineering*, vol. 28, no. 3, pp. 587–600, 2012.

- [13]. B. Gerkey. Gmapping ros package web page. [Online]. Available: <http://wiki.ros.org/gmapping>
- [14]. B. P. Gerkey. Amcl ros package web page. [Online]. Available: <http://wiki.ros.org/amcl>
- [15]. Institute of Data, “Understanding the prototype model in software Engineering | Institute of Data,” *Institute of Data*, Dec. 12, 2023. <https://www.institutedata.com/blog/prototype-model-in-software-engineering/>
- [16]. B. X. Hiên, “All advantages and disadvantages of prototype model,” *BiPlus*, Mar. 23, 2023. <https://biplus.com.vn/advantages-and-disadvantages-of-prototype-model/>
- [17] “Prototype: Definition, types, and examples in design & development | Roadmunk,” *Roadmunk*. <https://roadmunk.com/glossary/prototype/>
- [18] Userlytics, “Prototype Testing | UserLytics Corporation,” *Userlytics Corporation*, Oct. 25, 2023. <https://www.userlytics.com/blog/prototype-testing/>
- [19] “Do the Advantages of Prototyping Outweigh Prototyping Disadvantages?,” Jan. 20, 2022. <https://www.rready.com/blog/advantages-of-prototyping-vs-prototyping-disadvantages-rd>
- [20] TheKnowledgeAcademy, “Prototype model in software Engineering: All you need to know.” <https://www.theknowledgeacademy.com/blog/prototype-model-in-software-engineering/>
- [21] S. Innovation, “From ROS to ROS 2: A comprehensive guide to the next generation robotics,” *Medium*, Nov. 16, 2023. [Online]. Available: <https://scalexi.medium.com/from-ros-to-ros-2-a-comprehensive-guide-to-the-next-generation-robotics-f93a4e2e5793>
- [22] A. Ademovic, “An Introduction to Robot Operating System: the ultimate robot application framework,” *Toptal Engineering Blog*, Mar. 03, 2016. <https://www.toptal.com/robotics/introduction-to-robot-operating-system>
- [23] “ROS2 Overview - University of Waterloo Robotics Design Team - Confluence.” <https://uwaterloo.atlassian.net/wiki/spaces/UWRT/pages/33964066033/ROS2+Overview>
- [24] R. Team, “TurtleBot 3,” *ROBOTS: Your Guide to the World of Robotics*, Dec. 19, 2023. <https://robotsguide.com/robots/turtlebot3>
- [25] Y. Name, “ROBOTIS e-Manual,” *ROBOTIS e-Manual*. <https://emanual.robotis.com/docs/en/platform/turtlebot3/features/>
- [26] S. Gruppetta, “Python Readability, the PEP 8 style guide, and Learning Latin,” *The Python Coding Book*, Nov. 12, 2023. <https://thepythoncodingbook.com/2021/10/11/python-readability-the-pep-8-style-guide-and-learning-latin/>
- [27] C. Staff, “What is Python used for? A beginner’s guide,” *Coursera*, Apr. 03, 2024. <https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python>

[28] SKG Labs, “A Beginner’s Guide to ROS 2 nodes and Messages - SKG Labs - Medium,” *Medium*, Jul. 28, 2023. [Online]. Available: <https://medium.com/@tetraengnrng/a-beginners-guide-to-ros-2-nodes-and-messages-3609176c3e84>

[29] “PyQt5 tutorial 2024: Create a GUI with Python and Qt.” <https://build-system.fman.io/pyqt5-tutorial>

[30] R. Python, “Python and PyQt: Building a GUI desktop Calculator,” Sep. 25, 2023. <https://realpython.com/python-pyqt-gui-calculator/>

[31] S. Abraham, “Mastering PyQt: Python bindings for QT Toolkit,” *NeoITO Blog*, Apr. 20, 2023. <https://www.neoito.com/blog/pyqt-an-overview/>

[32] SQLPad, “Python pyqt gui calculator - Embark on GUI programming journey with PyQt in Python Discover how to create intuitive cross-platform applications from simple windows to fully-functional calculator - SQLPad.io.” <https://sqlpad.io/tutorial/python-pyqt-gui-calculator/>

[33] N. Koukis, “See VI-SLAM and Nav 2 stack integration in action,” *Slamcore*, Mar. 22, 2023. <https://www.slamcore.com/news/vi-slam-and-nav-2-stack-integration-in-action-watch-our-latest-demo/>

[34] F. A. Cheein, N. López, C. Soria, F. A. Di Sciascio, F. L. Pereira, and R. Carelli, “SLAM algorithm applied to robotics assistance for navigation in unknown environments,” *Journal of Neuroengineering and Rehabilitation*, vol. 7, no. 1, Feb. 2010, doi: 10.1186/1743-0003-7-10.

[35] AZoAi, “Role of AI in robot localization and mapping,” *AZoAi*, Feb. 13, 2024. <https://www.azoai.com/article/Role-of-AI-in-Robot-Localization-and-Mapping.aspx>

[36] N. Heath, “What is the Raspberry Pi 4? Everything you need to know about the tiny, low-cost computer,” *ZDNET*, Oct. 02, 2020. [Online]. Available: <https://www.zdnet.com/article/what-is-the-raspberry-pi-4-everything-you-need-to-know-about-the-tiny-low-cost-computer/>

[37] “Exploring the World of Single board Computers with Raspberry Pi - FasterCapital,” *FasterCapital*. <https://fastercapital.com/content/Exploring-the-World-of-Single-board-Computers-with-Raspberry-Pi.html>

[38] “Raspberry Pi boards compared,” *Core Electronics*. <https://core-electronics.com.au/guides/compare-raspberry-pi-boards/>

[39] A. Silva, “3 popular programming languages you can learn with Raspberry Pi,” *Opensource.com*. <https://opensource.com/article/19/3/programming-languages-raspberry-pi>

[40] “Raspberry Pi as a media center,” *Adafruit Learning System*, Mar. 15, 2013. <https://learn.adafruit.com/raspberry-pi-as-a-media-center/overview>

[41] R. Pelayo, “What is Raspberry Pi? models, features, and uses,” Dec. 12, 2023. <https://www.nextpcb.com/blog/what-is-raspberry-pi-models-features-and-uses>

Bachelor of Information Technology (Honours) Computer Engineering  
Faculty of Information and Communication Technology (Kampar Campus), UTAR

[42] “MPU-6050 | TDK InvenSense,” *TDK InvenSense*, Mar. 13, 2024.  
<https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/>

[43] Robocraze, “What is the LDR Sensor?,” *Robocraze*, Nov. 17, 2022.  
<https://robocraze.com/blogs/post/what-is-the-ldr-sensor>

[44] “Pi Camera Module Interface with Raspberry Pi using Python | Raspb...,” © 2018  
*ElectronicWings*. <https://www.electronicwings.com/raspberry-pi/pi-camera-module-interface-with-raspberry-pi-using-python>

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Trimester 3 Year 3</b>	<b>Study week no.:4</b>
<b>Student Name &amp; ID: Yuen Kok Hei 20ACB02830</b>	
<b>Supervisor: Dr. Teoh Shen Khang</b>	
<b>Project Title: Robot Surveillance for Connected Home Integration</b>	

## 1. WORK DONE

In Week 4, I revised my Final Year Project (FYP) 1 report and reviewing over ideas and information I had learned in earlier trimesters. I also bought a webcam and sensors such as DHT22, MPU6050, LDR sensors online.

## 2. WORK TO BE DONE

Install and setup the webcam on Turtlebot3.

## 3. PROBLEMS ENCOUNTERED

Temporary none.

## 4. SELF EVALUATION OF THE PROGRESS

Revise on ROS2 knowledge and Python Programming.



Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Trimester 3 Year 3</b>	<b>Study week no.:7</b>
<b>Student Name &amp; ID: Yuen Kok Hei 20ACB02830</b>	
<b>Supervisor: Dr. Teoh Shen Khang</b>	
<b>Project Title: Robot Surveillance for Connected Home Integration</b>	

## 1. WORK DONE

Successfully setup a Raspberry Pi integrated with DHT22, MPU6050, and LDR sensors.

## 2. WORK TO BE DONE

Install and setup the webcam on Turtlebot3.

## 3. PROBLEMS ENCOUNTERED

Failed to setup webcam on Turtlebot3. Raspberry Pi did not detect the camera.

## 4. SELF EVALUATION OF THE PROGRESS

Mastered how to setup DHT22, MPU6050, and LDR sensors.

Some ROS packages online may not working, therefore I have to find alternative solutions.



Supervisor's signature



Student's signature



# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Trimester 3 Year 3</b>	<b>Study week no.:9</b>
<b>Student Name &amp; ID: Yuen Kok Hei 20ACB02830</b>	
<b>Supervisor: Dr. Teoh Shen Khang</b>	
<b>Project Title: Robot Surveillance for Connected Home Integration</b>	

## 1. WORK DONE

Successfully setup a Raspberry Pi integrated with DHT22, MPU6050, and LDR sensors. Failed to setup webcam on Turtlebot3. So, I decided to buy Pi Camera instead of webcam.

## 2. WORK TO BE DONE

Install and setup the Pi Camera on Turtlebot3.

## 3. PROBLEMS ENCOUNTERED

Failed to setup webcam on Turtlebot3. Raspberry Pi did not detect the camera.

## 4. SELF EVALUATION OF THE PROGRESS

Keep trying although I was facing bottleneck. Eventually, solutions will pop up one day.



Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Trimester 3 Year 3</b>	<b>Study week no.:10</b>
<b>Student Name &amp; ID: Yuen Kok Hei 20ACB02830</b>	
<b>Supervisor: Dr. Teoh Shen Khang</b>	
<b>Project Title: Robot Surveillance for Connected Home Integration</b>	

## 1. WORK DONE

All Raspberry Pis have been successfully set up with DHT22, MPU6050, and LDR sensors. The MQTT protocol was successfully configured to publish sensor data to the laptop. Pi Camera has been successfully installed on TurtleBot3, and the image\_raw topic can now be published.

## 2. WORK TO BE DONE

I need to implement codes to take photo from Pi Camera.

## 3. PROBLEMS ENCOUNTERED

Not sure what's the problem.

## 4. SELF EVALUATION OF THE PROGRESS

I have learned how to setup Pi Camera.



Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Trimester 3 Year 3</b>	<b>Study week no.:12</b>
<b>Student Name &amp; ID: Yuen Kok Hei 20ACB02830</b>	
<b>Supervisor: Dr. Teoh Shen Khang</b>	
<b>Project Title: Robot Surveillance for Connected Home Integration</b>	

## 1. WORK DONE

All Raspberry Pis have been successfully set up with DHT22, MPU6050, and LDR sensors. The MQTT protocol was successfully configured to publish sensor data to the laptop. Pi Camera has been successfully installed on TurtleBot3, and the image\_raw topic can now be published. I have implemented codes to take photo.

## 2. WORK TO BE DONE

I need to work on preparing the GUI and probably start my report.

## 3. PROBLEMS ENCOUNTERED

When I combined my navigation.py, sensors\_work.py, and camera\_node.py something went wrong.

## 4. SELF EVALUATION OF THE PROGRESS

I have learned how to take photo using Pi camera.



Supervisor's signature



Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year: Trimester 3 Year 3</b>	<b>Study week no.:12</b>
<b>Student Name &amp; ID: Yuen Kok Hei 20ACB02830</b>	
<b>Supervisor: Dr. Teoh Shen Khang</b>	
<b>Project Title: Robot Surveillance for Connected Home Integration</b>	

## 1. WORK DONE

I successfully to combine all the codes together in one script, and surprisingly it works. I also setup telegram bot father to integrate with the system. I also managed to design a GUI in time.

## 2. WORK TO BE DONE

Focus on finalizing FYP report.

## 3. PROBLEMS ENCOUNTERED

No problem encountered.

## 4. SELF EVALUATION OF THE PROGRESS

I have learned how to design a good GUI and also integrating with Telegram Bot Father.



Supervisor's signature



Student's signature

# POSTER

FACULTY OF INFORMATION  
COMMUNICATION AND  
TECHNOLOGY



## Robot Surveillance for Connected Home Integration

In our rapidly evolving world, the concept of the connected home has become a reality. This project, explores the fusion of robotics and smart homes to enhance security and convenience. Using the TurtleBot3 platform and advanced sensors, our system responds to sensor triggers by autonomously navigating and capturing real-time images for homeowners.



### MAIN OBJECTIVE

- Develop a responsive surveillance system using the TurtleBot3 robot to autonomously navigate homes in response to sensor triggers, enabling dynamic and real-time monitoring.



### PROPOSED METHOD

- Utilize the TurtleBot3 robot as the core platform, integrating sensors for environmental monitoring, and develop a centralized control system to enable autonomous navigation in response to sensor triggers, coupled with real-time surveillance features.
- system will be designed to provide flexible and responsive home monitoring within changing environments.

**Project Developer: Yuen Kok Hei**

**Project Supervisor: Dr Teoh Shen Khang**

# PLAGIARISM CHECK RESULT

## ROBOT SURVEILLANCE FOR CONNECTED HOME INTEGRATION.docx

### ORIGINALITY REPORT

<b>7</b> %	<b>4</b> %	<b>3</b> %	<b>3</b> %
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

### PRIMARY SOURCES

<b>1</b>	Ruslan Safin, Roman Lavrenov, Kuo-Hsien Hsia, Elena Maslak, Natalia Schiefermeier-Mach, Evgeni Magid. "Modelling a TurtleBot3 Based Delivery System for a Smart Hospital in Gazebo", 2021 International Siberian Conference on Control and Communications (SIBCON), 2021 Publication	<b>1</b> %
<b>2</b>	<a href="https://eprints.utar.edu.my">eprints.utar.edu.my</a> Internet Source	<b>1</b> %
<b>3</b>	Submitted to Universiti Tunku Abdul Rahman Student Paper	<b>1</b> %
<b>4</b>	Submitted to Liverpool John Moores University Student Paper	<b>&lt;1</b> %
<b>5</b>	<a href="https://ebin.pub">ebin.pub</a> Internet Source	<b>&lt;1</b> %
<b>6</b>	<a href="http://www.cse.iitd.ac.in">www.cse.iitd.ac.in</a> Internet Source	<b>&lt;1</b> %

7	"Web, Artificial Intelligence and Network Applications", Springer Science and Business Media LLC, 2019 Publication	<1 %
8	www.mdpi.com Internet Source	<1 %
9	docplayer.net Internet Source	<1 %
10	"Recent Advances in Information Systems and Technologies", Springer Science and Business Media LLC, 2017 Publication	<1 %
11	Submitted to Queen's University of Belfast Student Paper	<1 %
12	Submitted to University of Nebraska at Omaha Student Paper	<1 %
13	Submitted to The Robert Gordon University Student Paper	<1 %
14	patents.google.com Internet Source	<1 %
15	trepo.tuni.fi Internet Source	<1 %
16	Submitted to University of East London Student Paper	<1 %

17 Visakh Bobby, S. Praneel, Gokul Krishna S, Athuljith A S, Reeja S L. "EBICS: Even Blind I Can See, A Computer Vision Based Guidance System For Visually Impaired", 2023 International Conference on Control, Communication and Computing (ICCC), 2023  
Publication <1 %

18 sageadvices.com  
Internet Source <1 %

19 Advances in Intelligent Systems and Computing, 2014.  
Publication <1 %

20 lab.daiict.ac.in  
Internet Source <1 %

Exclude quotes On

Exclude matches < 8 words

Exclude bibliography On



<b>Universiti Tunku Abdul Rahman</b>			
<b>Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)</b>			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

<b>Full Name(s) of Candidate(s)</b>	Yuen Kok Hei
<b>ID Number(s)</b>	20ACB02830
<b>Programme / Course</b>	Bachelor of Information Technology (HONS) Computer Engineering
<b>Title of Final Year Project</b>	Robot Surveillance for Connected Home Integration

<b>Similarity</b>	<b>Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)</b>
<b>Overall similarity index: <u>7</u> %</b>  <b>Similarity by source</b> Internet Sources: <u>4</u> % Publications: <u>3</u> % Student Papers: <u>3</u> %	
<b>Number of individual sources listed of more than 3% similarity: <u>0</u></b>	
<b>Parameters of originality required and limits approved by UTAR are as Follows:</b> (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

*Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.*

\_\_\_\_\_  
Signature of Supervisor

Name: Dr. Teoh Shen Khang

Date: 26 April 2024

\_\_\_\_\_  
Signature of Co-Supervisor

Name: \_\_\_\_\_

Date: \_\_\_\_\_

Bachelor of Information Technology (Honours) Computer Engineering  
Faculty of Information and Communication Technology (Kampar Campus), UTAR



**UNIVERSITI TUNKU ABDUL RAHMAN**

**FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY  
(KAMPAR CAMPUS)**

**CHECKLIST FOR FYP2 THESIS SUBMISSION**

Student Id	20ACB02830
Student Name	Yuen Kok Hei
Supervisor Name	Dr. Teoh Shen Khang

<b>TICK (✓)</b>	<b>DOCUMENT ITEMS</b>
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
✓	Title Page
✓	Signed Report Status Declaration Form
✓	Signed FYP Thesis Submission Form
✓	Signed form of the Declaration of Originality
✓	Acknowledgement
✓	Abstract
✓	Table of Contents
✓	List of Figures (if applicable)
✓	List of Tables (if applicable)
	List of Symbols (if applicable)
✓	List of Abbreviations (if applicable)
✓	Chapters / Content
✓	Bibliography (or References)
✓	All references in bibliography are cited in the thesis, especially in the chapter of literature review
✓	Appendices (if applicable)
✓	Weekly Log
✓	Poster
✓	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
✓	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

\*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

*Hei*

(Signature of Student)

Date: 25 April 2024