



Palestine Polytechnic University
College of Information Technology and Computer Engineering

Object Finder Robot

Team Members:

Hamza Sameeh Dwaik

Moayad Amjad Hrebat

Motaz Iyad Natsheh

Supervisor:

Dr. Amal Al-Dweik

Mr. Wael AL Takrouri

Hebron - Palestine

January-2023

Acknowledgment

In the name of "Allah", the most beneficent and merciful who gave us strength, knowledge and helped us to get through this project. For those who deserve our thanks the most, our parents, we are indebted to you for the rest of our lives for your unconditional love and support. We know that thanks is not enough and there are not enough words to describe how thankful we are. To our families and friends, thank you for your endless encouragement all our lives and especially during the completion of this project. We would like to thank our supervisors of this project, Dr. Amal Al-Dweik and Mr. Wael AL Takrouri help us and advice during this project.

We also thank our faculty and Professors at the College of Information Technology and Computer Engineering for their hard work and support to the students

Abstract

We frequently spend a lot of time looking for objects that we don't remember where we put them. Consequently, technological assistance for users. Some individuals find the process of looking for a specific object to be boring or annoying. This is particularly difficult for people with special needs and when there are several objects that are challenging for a person to identify all at the same time. A solution we suggest is an intelligent mobile robot that will interact with a mobile device to detect objects according to user request by using artificial intelligence YOLO and ROS algorithm. The purpose of this system is to find objects specified by the user and return the object's coordinates within the generated map.

In this project, we achieved results, which is detecting the object in the places where we inserted the robot. The speed of detecting the object was good, but the accuracy was less. We were also able to calculate the distance between the robot and the object, but not with high accuracy.

Keywords: Object Detection, Simultaneous Localization and Mapping, SLAM, Mobile Robot.

الخلاصة

غالباً ما نقضي الكثير من الوقت في البحث عن أشياء لا نتذكر أين وضعناها وبالتالي يعتقد بعض الأفراد ان عملية البحث عن شيء معين تكون مملة أو مزعجة ، وصعبة بشكل خاص للأشخاص ذوي الاحتياجات الخاصة وعندما يكون هناك العديد من الأشياء التي يصعب على الشخص التعرف عليها جميعها في نفس الوقت الحل هو نقترح روبوتاً ذكياً متنقلاً يتفاعل مع تطبيق على الهاتف المحمول للبحث عن شيء معين حسب طلب المستخدم باستخدام خوارزمية الذكاء الاصطناعي YOLO وبيئة ROS . الغرض من هذا النظام هو العثور على الكائنات المحددة من قبل المستخدم و إرجاع إحداثيات الكائنات وعرضها على الخريطة في تطبيق الهاتف.

في هذا المشروع ، حققنا نتائج وهي اكتشاف الكائن في الأماكن التي تم ادخال الروبوت فيها. حيث كانت سرعة اكتشاف الكائن جيدة ولكن كانت الدقة أقل. تمكنا أيضاً من حساب المسافة بين الروبوت والشيء ،ولكن ليس بدقة عالية.

الكلمات المفتاحية : تحديد الأشياء ، في وقت واحد ،الموقع ، رسم الخرائط ،روبوت محمول.

Table of Contents

List of Figures	vi
List of Tables	vii
List of Acronyms	viii
1 Introduction	1
1.1 Overview	1
1.2 Motivation	1
1.3 Project Objectives	1
1.4 Short Description Of The System	2
1.5 Problem Statement	2
1.6 List Of Requirements	2
1.6.1 Functional Requirement:	2
1.6.2 Nonfunctional Requirement:	3
1.7 Overview Of The Rest Of Report Sections	3
2 Background	4
2.1 Overview	4
2.2 Theoretical Background	4
2.2.1 YOLO (You Only Look Once) Algorithm	4
2.2.2 ROS (Robot Operating System)	6
2.2.3 Turtlebot2 Motion	8
2.2.4 Mapping Algorithm	8
2.3 Literature Review	9
2.4 The System Components And Design Options	11
2.4.1 Hardware Components and Options:	11
2.4.2 System Software Components:	19
3 System Design	23
3.1 Overview	23
3.2 Detailed Description Of The System	23
3.3 System Diagrams	24
3.3.1 System Block Diagram	24
3.3.2 Schematic Diagram	26
3.4 Pseudo-Code	29

3.5	Adjustments	29
4	System Implementation	31
4.1	Overview	31
4.2	Hardware Implementation:	31
4.3	Software Implementation:	33
4.3.1	Operating System	33
4.3.2	Installing needed packages	33
4.3.3	OpenCV implementation	33
4.3.4	Object detection implementation	33
4.3.5	YOLO Implementation	34
4.3.6	ROS implementation	34
4.3.7	ROS algorithms	34
4.4	Mobile Application Implementation	35
4.4.1	MQTT Protocol	35
4.4.2	User Interface	36
4.5	Implementation issues and challenges	38
5	Validation and Testing	39
5.1	Overview	39
5.2	Hardware Testing	39
5.2.1	Testing DC Motor In Kobuki	39
5.2.2	Testing Kinect XBOX 360	39
5.3	Software testing	40
5.3.1	OpenCV testing	40
5.3.2	Testing Of YOLO Algorithm	40
5.3.3	Software Testing Tables	41
5.3.4	Mobile Application Testing	42
5.4	System Validation	43
6	Conclusion and Future work	44
6.1	Conclusion	44
6.2	Future work	44
	References	43

List of Figures

2.1	YOLO Methodology [4]	5
2.2	Bounding Box [4]	6
2.3	ROS communication between nodes[5]	7
2.4	Figure 2.4 Raspberry Pi 3[8]	13
2.5	Xbox Kinect [11]	16
2.6	TurtleBot 2[12]	19
3.1	System Block Diagram	24
3.2	Schematic diagram for DC Motor(kobuki robot) with Raspberry Pi	26
3.3	Schematic diagram for Xbox Kinect Camera with Raspberry Pi	27
3.4	Schematic diagram for LIDAR Sensor with Raspberry Pi	28
3.5	from raspberry pi to Dell laptop	29
3.6	CPU and Memory usage of Raspberry Pi	30
3.7	URG LiDAR and XBOX Kinect	30
4.1	System Components	32
4.2	Connect XBOX 360 Kinect Sensor to Kobuki	32
4.3	Gmapping Result	35
4.4	Mobile Application Implementation	37
5.1	OpenCV Testing	40
5.2	Testing YOLO Result	41
5.3	MQTT Setting	42
5.4	Topic in MQTT	43

List of Tables

2.1: comparison between previous objects detection-based projects	10
2.2: List of Microcontroller options	12
2.3: List of sensors options	14
2.4: List of Depth cameras options	15
2.5: List of robot options	17
2.6: List object detection algorithms options	19
2.7: Comparison list for mobile applications Language	21
5.3 Software testing Table	39

List of Acronyms

CPU	Central Processing Unit
DC	Direct Current
FOV	Filed Of View
FRCN	Fast Region-Convolutional Neural Network
OpenCV	Open-Source Computer Vision
OS	Operating System
mAP	mean Average Precision
MQTT	Message Queuing Telemetry Transport
RAM	Random Access Memory
R-FCN	Region-based Fully Convolutional Network
RGB	Red Green Blue
RVIZ	ROS visualization
ROS	Robot Operating System
SBC	Single Board Computer
SLAM	Simultaneous Localization and Mapping
USB	Universal Serial Bus
VGA	Video Graphics Array
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Overview

In our project, we are going to develop a robot that will interact with a mobile to find objects according to user selections. The purpose of this system is to search objects specified by the user that are sometimes difficult for humans to find. The proposed system is supposed to save time and efforts.

This chapter presents a general idea about the project, overview, motivation and importance, objectives, short description of the system, problem statement, list of functional and nonfunctional requirements, and an overview of the rest of report sections.

1.2 Motivation

One of the most important motivations for the project is to help people and specially those with special needs in particular to search for objects that exist in closed or limited areas. It can be used for many different jobs and functions that may be too boring, difficult or dangerous for a human to do.

1.3 Project Objectives

The system aims to:

1. Learn to use the ROS and YOLO algorithm.
2. Using Artificial Intelligence methodologies to enable the robot to search for specific objects.
3. Generate map for the searching area.
4. Determine the coordinates of the desired object within the generated map. .

1.4 Short Description Of The System

At first, The robot generates a map of the place where it is located using the mapping algorithm. Then, instructions are given to the robot using a mobile application to search for certain objects. After that, the robot is able to detect the selected object such as: a ball, or remote,...etc, by a special camera equipped with it. It performs the object identification and localization of each object using ROS and YOLO algorithm. The robot should avoid any obstacles while it is searching and navigating in the targeted area. Each time the robot finds the aimed object , it sends a message to the mobile application pointing the coordinates.

1.5 Problem Statement

Looking for an object we do not remember where it is left occurs frequently, and it considered as a frequent problem regardless of age [1]. One survey reported that people waste 2.5 days a year looking for misplaced objects [2] .Thus, technological support to assist users in finding lost objects is demanded some people consider searching for a specific object boring or annoying. This is more demanding with special needs and in causes when there are a large number of objects that is difficult for a person to identify all at the same time. When a person searches for a specific purpose, his mind becomes blurred and focus is less, which makes it more difficult and he needs a proper solution to search for the object.

1.6 List Of Requirements

Some of the Functional and Nonfunctional Requirements for our system are:

1.6.1 Functional Requirement:

The system should be able to:

- Generate the place map.
- Navigate a space safely by avoiding obstacles.
- Find coordinates of searched objects.

1.6.2 Nonfunctional Requirement:

The system should be able to:

- User friendly and easy to be used.
- Acceptable response time.
- Safe (no damage occurs to the searched objects).

1.7 Overview Of The Rest Of Report Sections

The next chapters of our report will be as the following: Chapter 2 (Background), introduces the theoretical background and literature review, design options (hardware components and software component), Chapter 3 (Design), introduces the detailed conceptual description of the system, detailed design, structural diagrams, block diagrams, and any necessary information about the design.

Chapter 2

Background

2.1 Overview

This chapter introduces the theoretical background of our project, literature review, the system components, and design options.

2.2 Theoretical Background

Our current world is witnessing a very wide scope of the use of robots. It has many uses in several fields such as medical, military, and technological fields. In the following subsections, we will introduce YOLO Algorithm, ROS, and mapping algorithms which are the main that will be used in our developed system and algorithms.

2.2.1 YOLO (You Only Look Once) Algorithm

YOLO is a distinct kind of object detection algorithm than region-based algorithms [4]. The Bounding boxes are rectangles that mark objects on an image. There are multiple formats of bounding boxes annotations. Class probabilities for these boxes are predicted by a single neural network in YOLO. It is proposed as an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once and it can detect 80 different objects such as a Chair, Table, Person.

YOLO Methodology

First of all, the algorithm divides the image into N grids, each with an equal dimensional regions of $S \times S$. Each of these N grids is responsible for the detection and localization of the object it contains. These grids predict bounding box coordinates relative to their cell coordinates as shown in figure 2.1 [4].

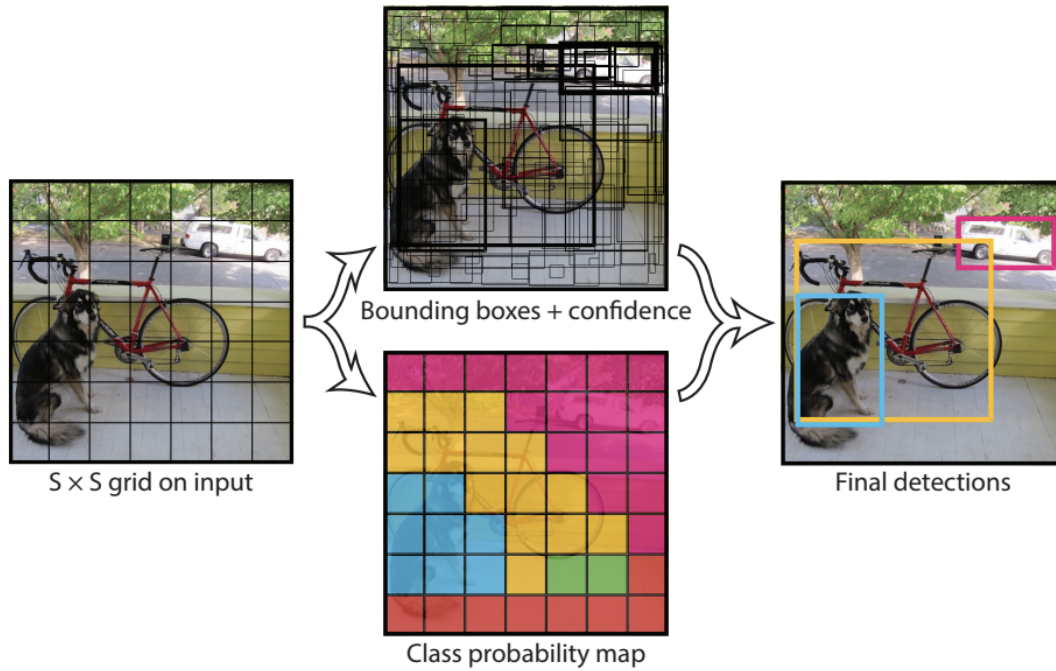


Figure 2.1: YOLO Methodology [4]

Each bounding box in the image consists of the following properties:

- Width (b_w).
- Height (b_h).
- Class (for example, person, car, etc.) and this is represented by the letter c .
- Bounding box center (b_x, b_y).
- P_c : it is probability of the existence or non-existence of the object.

To illustrate the above properties Figure, shows an example for the bounding box that has been represented by a yellow outline:

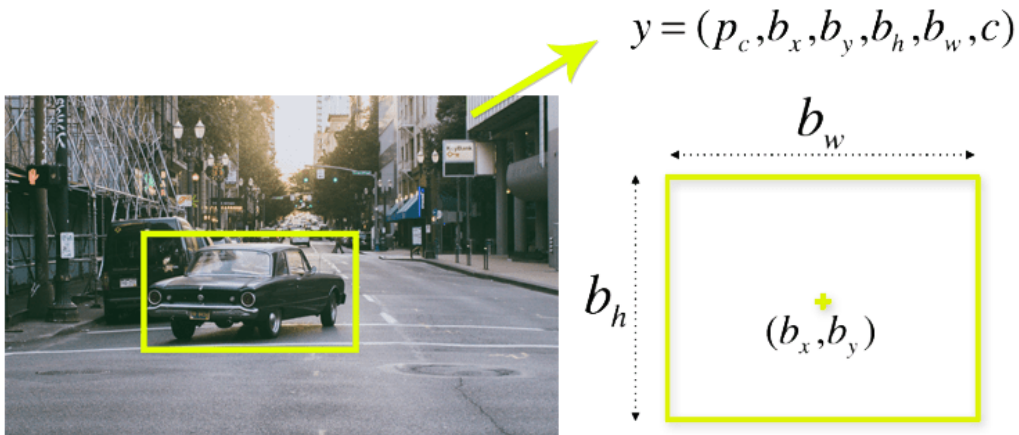


Figure 2.2: Bounding Box [4]

It is suitable to be used in the project for the following reasons [5]:

1. High accuracy: it provides accurate results with minimal errors.
2. It is a pre-trained model.
3. Learning capabilities: it learns object representations and uses them to detect objects.

2.2.2 ROS (Robot Operating System)

ROS is an open-source, and it is a software platform that provides libraries and tools to help software developers create robotics applications. It provides hardware abstractions, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is similar in some respects to robot frameworks[3], programming languages used in ROS python and c++.

ROS Methodology

ROS consists of a code and tools that help you run your project code and do the required task. ROS is designed to be a loosely coupled system where a process is called a node and every node should be responsible for one task. Nodes communicate with each other using message passing via logical channels called topics. Each node can send or get data from the other nodes using the publish/subscribe model. Software in ROS is organized in packages. A package might contain ROS nodes, a ROS-independent library, a dataset, configuration files, a third-party piece of software, or anything else that logically constitutes a useful module. The goal of these packages is to provide this useful functionality in an easy-to-consume manner so that software can be easily reused. In general, ROS packages follow a "Goldilocks" principle: enough functionality to be useful, but not too much that the package is

weight and difficult to use from other software[7]

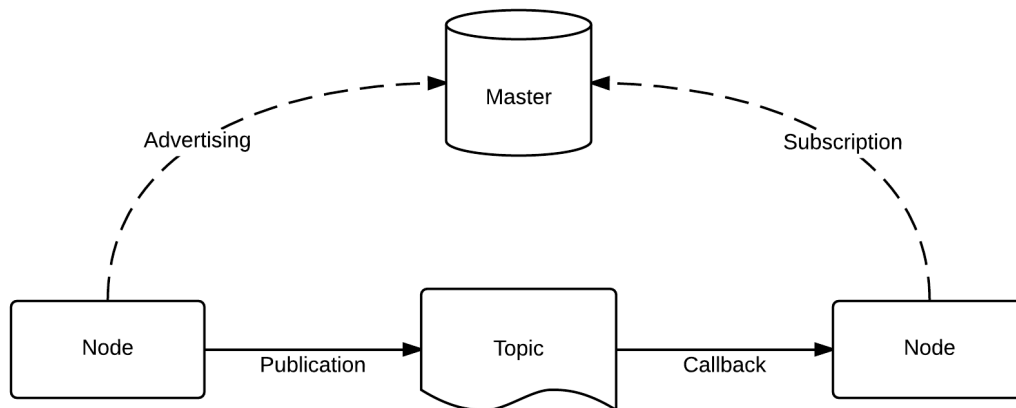


Figure 2.3: ROS communication between nodes[5]

ROS has multiple versions, currently there are two main versions that are supported[21]:

- ROS Noetic Ninjemys has released in May 23rd, 2020.
- ROS Melodic Morenia has released in May 23rd, 2018.
- ROS Kinetic Kame release stopped support in April, 2021 but you can use it.

The tools that are used in ROS[22]:

1. Rviz: rviz (short for “ROS visualization”) is a 3D visualization software tool for robots, sensors, and algorithms. It enables you to see the robot’s perception of its world (real or simulated). The purpose of rviz is to enable you to visualize the state of a robot. It uses sensor data to try to create an accurate depiction of what is going on in the robot’s environment.

2. Gazebo: is a 3D robot simulator. Its objective is to simulate a robot, It gives a close substitute to how your robot would behave in a real-world physical environment. It can compute the impact of forces (such as gravity).

ROS is used in project for the following reasons:

1. ROS is a language-agnostic. it enables a Python node to easily communicate with C++ node. It supports reusability and possibilities of co-working. It contains many libraries that allow user to use other languages (because ROS has mainly targeted C++ and Python) [17].

2. ROS has great simulation tools. such as Rviz and Gazebo that enable the unreal run of robot.
3. It can control multiple robots.
4. It doesn't take much space and resources.
5. It is an open-source project with a permissive license.

2.2.3 Turtlebot2 Motion

The turtlebot2 have a tow parts to drive linear and angular speed The linear speed refers to the speed at which the robot moves forward or backward, while the angular speed refers to the speed at which the robot rotates or turns. These speeds can be controlled through the use of commands sent to the robot motors. TurtleBo2t rotates around its own axis.

The x, y, and z directions refer to the standard Cartesian coordinate system, with x being the horizontal axis, y being the vertical axis, and z being the axis perpendicular to the xy plane. To control the linear and angular speed of the TurtleBot in the x, y, and z directions, you would need to use a more complex system of controls. The TurtleBot is primarily designed to move in the x and y directions (forwards and backwards, and left and right) using its wheels, and to rotate around its own axis (the z-axis) using its onboard motors. It is not designed to move in the z-direction, as it does not have the ability to fly or jump.

2.2.4 Mapping Algorithm

Mapping is a technique used for the purpose of making a map, updating it and to estimate the position of the robot to map the environment. There are many ROS packages which can be used, such as Gmapping.

Gmapping

Path Planing Techniques

Gampping is one method that can be used for path planning in robotics. In this approach, the robot constructs a map of its environment using sensors or other means, and then uses this map to plan a path to the desired destination. Gampping algorithms can be used to generate paths that are optimal in some sense, such as being the shortest or fastest path, or paths that minimize the risk of collision with obstacles. Gampping algorithms can also be used to plan paths that avoid certain areas or follow specific routes.

1) A*Planning Algorithm

It is a tracking algorithm that determines the path from the current location of the robot to a specific goal point while avoiding obstacles and giving greater priority to the goals that are closer with lower costs[15].

2) D*Planning Algorithm

The A* algorithm assumes that the entire environment is known, but there may be moving obstacles. To include this, D* has been proposed, as it aims to plan the effective course of the unknown and dynamic environments[15].

Mapping Methodology

The map generation is carried out by first, generating the environment and importing turtlebot in Gazebo. The mapping process is implemented using the gmapping package. To use gmapping the robot model must provide odometry. It is the use of data from motion sensors to estimate change in position over time. It is used in robotics by some legged or wheeled robots to estimate their position relative to a starting location. This method is sensitive to errors due to the integration of velocity measurements over time to give position estimates. Rapid and accurate data collection, instrument calibration, and processing are required in most cases for odometry to be used effectively.

2.3 Literature Review

In this section we will discuss some projects similar to the idea of our project.

1.Object Detection And Tracking System [14]

The main idea of that project this [14] is, to design a system for detecting and tracking objects based on their color. Its mechanism of action is to take pictures of the object continuously through the camera that is interfaced to the Raspberry Pi. When it is detected, the robot tracks the target. In this project, the object was only the ball and detection of object is not fast. In our project, there will be more than one object to be detected and the used algorithms makes the object detection faster.

2. Autonomous Wheelchair Project [15]

The main idea of the project is to design a wheelchair capable of overcoming obstacles, in crowded environments without bumping into objects. The user's job was to enter commands via the mobile phone, either by voice (file name location such as the kitchen, go ahead) or by touch locate on map or control the movement of the chair such as moving forward. After the user enters the command and with the help of the stored map, the chair can locate it on the map, locate the target, then select the optimal path while avoiding obstacles. In [15], the wheelchaired finds and determines the way to go it. In our project, the robot defines a way to go and find objects automatically.

3. Sanitizer Spider Robot [16]

The main idea of the project is to design and implement a sanitizer spider robot using Raspberry Pi by using some sensors to sense by the camera. The camera captures an image for the suspected objects to be infected. Then, it moves in the affected area, and it can measure the distance of objects for example, Chair or Table after that it determines the desired object and then sterilizes it. The main similarity between our project and the "Sanitizer Spider Robot", is the use of some algorithms, the most important of which is the YOLO algorithm, as there is a similarity in some hardware components such as the camera and some sensors. The difference between our work and listed previous works is presented in Table 2.1:

Table 2.1: Comparison Between Previous Objects Detection-Based Projects

Project	Processor unit	Sensors	Alogorithm	No. of target objects to search
Object detection and tracking system [14]	Arduino Uno and Raspberry Pi 3 Model B	Sensor Existing in Raspberry pi camera	KMeans	One
Autonomous Wheelchair Project [15]	Raspberry Pi 4 Model B	LIDAR	A* Planning and D* Planning	No objects need to detect because it need to locate desired location
Sanitizer Spider Robot [16]	Raspberry Pi 3 Model B	Ultrasonic	YOLO	Two

2.4 The System Components And Design Options

This section introduces designs options and system components.



2.4.1 Hardware Components and Options:

We need processing unit to processes the data that will sent to a robot.it is considered as one of the essential parts of the project. The choice of the processor depends on specific characteristics such as:

- 1) Cost.
- 2) Sufficient memory.
- 3) Suitable size.
- 4) Speed.
- 5) Number of I/O pins.

We have studied the possible options, compared them, and choose the most suitable for our project. Which are presents in the Table 2.2.

Table 2.2: List of Processing Options

Processing Unit		
Requirements	Raspberry Pi 3 Model B 	Intel Galileo 
Lower cost	35\$[2022]	\$80[2022]
Suitable size	85.6*53.17 mm	15*15 mm
Power consumption	1.3-1.4 Watt	2.65 Watt
Relatively easier in programming	Yes	Yes
Number of pins is sufficient.	40 Pin	14 digital input/output Pins
RAM	1 GB	56 MB
CPU	1.4 GHZ	400 MHz

We chose the Raspberry Pi 3 Model B because it is the lowest price and available in the local market, below is a more technical description about it.

Raspberry pi 3

It is Single Board Computer (SBC) that can be connected to a computer monitor or TV, made by Raspberry Pi Foundation in the UK charity that aims to educate people in computing and create easier access to computing education specially for developing countries. Uses a keyboard and mouse and can be used easily by all ages. Also, can be learn how to programming in Python or Scratch languages. The system features 2 GB or 4 GB of RAM, plus a Micro USB port for power and micro-HDMI port for connecting to two displays. There are also USB 2.0 and USB 3.0 ports for connecting peripherals, as well as a Gigabit Ethernet port and a Wi-Fi and Bluetooth module for connecting to wired or wireless networks respectively.[8] and the operating system is Raspberry Pi OS.

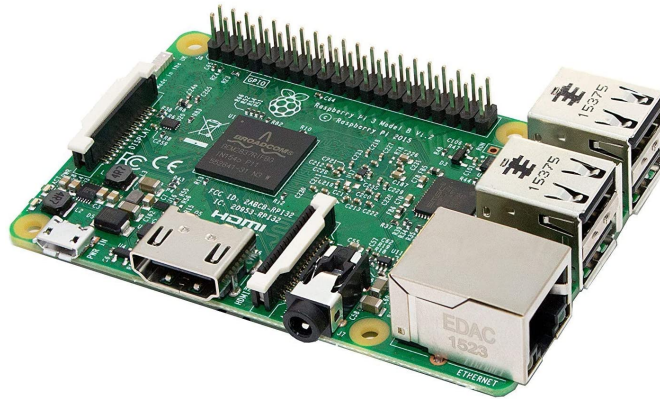





Figure 2.4: Figure 2.4 Raspberry Pi 3[8]

Obstacle Avoidance Sensor

We need a sensor to measure distance to avoid obstacles also it is needed by ROS Gamapping to generate maps for the environment. Table 2.3 presents the list of options of such sensors.

Table 2.3: List of Sensors Options



Requirements	HC-SR04 Ultrasonic sensor 	IR sensor 	URG-04LX-UG01 LIDAR sensor 
Detection range	Suitable to detect objects which are less than 1 meter away. Capable to detect objects within 5 mm more accurately. The object detection depends on shape, size and orientation.	More appropriate for targets which are closer than 10 mm. The maximum range 1-5 meters.	Extended detection of objects at a distance of up to 4 meters.
Frequency range	Operate from 20 kHz up to several GHz.	430 THz down to 300 GHz	200 THz
Interference from light sources (e.g. sunlight, fluorescent tubes etc.)	Unaffected	Affected	Affected
Cost	\$7.16	\$10	\$500-\$1000
Direction	Toward the target in front end [range is limited]	Toward the target in front end [range is limited]	The sensor for 360 degree detection and enables high speed scanning
Accuracy	Low	Low	High

We chose URG-04LX-UG01 LIDAR sensor, although it has a higher cost but compared to ultrasonic and infrared sensors, it is suitable for detecting fast-moving objects and it is good at detecting small objects. It is also able to detect 3D structures and it has a continuous beam that scans the environment.

Depth Camera

We need a camera to capture what resides in front of it and know what is moving in front of it.

Table 2.4: List of Depth Cameras Options

Requirements	Xbox 360 Kinect	ASUS Xtion 2
		
Cost	160\$	240\$
Filed Of View (FOV)	60°	52° (can be mounted in vertical position for 74°)
Depth Resolution	512x424	640x480 (actual is less than 320x240)
Multiple Sensors per single PC	Yes	Yes
External Power Supply	Required	Not required

We chose Xbox 360 Kinect because it has an infrared projector, infrared camera, and color camera. It's a great imaging tool, even for robots. It enhances the range and autonomous nature of robots and has a lower cost, and the depth resolution is better.

Xbox Kinect

The Kinect is a depth camera, and it contains three vital pieces that work together to detect your motion and create your physical image on the screen: an RGB color VGA video camera, a depth sensor, and a multi-array microphone. The camera detects the red, green, and blue color components as well as body-type and facial features. that can judge depth and distance to take photography to new levels. It uses the known speed of light to measure distance, and effectively calculates the amount of time it takes for a reflected beam of light to return to the camera sensor. In our project we will use Kinect camera and it use in Video games in Xbox device [11].



Figure 2.5: Xbox Kinect [11]

Mobile Robot

We need a robot to design the project and to link the components together and communicate with each other, which will be given instructions by the user through the mobile application. Table 2.5 presents the list of options of such robot.

Table 2.5: List of Robot Options

Requirements	TurtleBot 2 	Pioneer 3 
Cost	\$1400	\$6,278
Easy to Use	Easy	Easy
Reliable	Has highly reliable odometry and long battery hours and provides power for an external sensor and actuator. With the structure customized by a user's demand, it for development and research of various robots	Construction is durable and rugged. Easily handles the small gaps, minor bumping, jarring, or other obstacles that hinder other

Technical Support	Pioneer software and hardware is supported by product support team	Pioneer software and hardware comes fully documented with additional help available through by product support team
Size	Smaller size	Bigger size
Battery	2200 mAh	220000 mAh

TurtleBot 2

TurtleBot 2 is the world's most popular low cost, open-source robot for education and research. This second edition TurtleBot robot is equipped with a powerful Kobuki robot base and a trays for the installation of these components as shown in Figure 2.4 below. All components have been seamlessly integrated to deliver an out-of-the-box development platform. This robot officially proposed by Willow Garage to develop in the operating system dedicated to robotics (ROS) [12]. The Turtlebot2 have some components [19]:

- Kobuki Base is a low-cost mobile research base designed for education and research on state of art robotics. Kobuki provides power supplies for an external computer as well as additional sensors and actuators such as wheel encoder and Dc motor and motor driver. Its highly accurate odometry, amended by our factory calibrated gyroscope, enables precise navigation and contains 1x4S1P battery 2200 mAh, Battery charger, USB communication cable.
- Trays: It is used to install pieces on it.

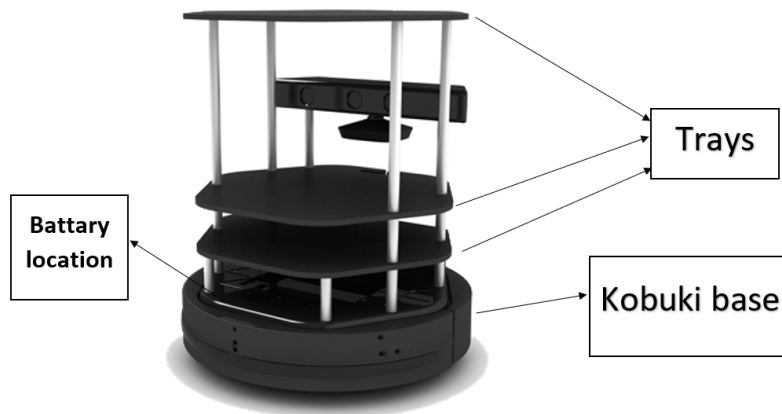


Figure 2.6: TurtleBot 2[12]

2.4.2 System Software Components:

Object detection algorithm

We need a algorithm for detecting objects. and It should have high quality specifications and easy to programming. Table 2.6: lists Object detection algorithms options.

Table 2.6: List Object Detection Algorithms Option.

Algorithm	R-FCN	YOLOV3	FRCN	YOLOV3-tiny
Detection speed	Not too fast	Fast	Not too fast	Faster
accuracy	High accuracy	High accuracy	Medium accuracy	Medium accuracy
It's a pre-trained	Yes	Yes	Yes	Yes
The mAP on (%) dateset	51.9	51.5	59.1	33.1

We chose YOLOv3-tiny algorithm although it has a medium accuracy but compared has a higher Speed in detection objects and higher accuracy and it is a pre-trained model and it can be trained easily.

Python Programming Language

Python is an open-source computer programming language and a high-level dynamically typed one that is among the most popular general-purpose programming languages. It is more quickly than other programming languages built in data structures. Python is combined with dynamic typing and dynamic binding which makes it has an easy structure that enhances readability and reduces the cost of code maintenance and debugging. Python programs is easy, while languages can pick up on Python very quickly. Also, beginners of use a python language find the clean syntax. and the indentation structure is easy to learn. Furthermore, it was supported by the OpenCV library that provides which are objects recognized on features in the researchers' projects [9].

OpenCV Library


Open CV is a library of Python bindings designed to solve computer vision problems. It is developed by intel and now supported by Willow Garage and it's free for both academic and commercial use. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of a multi-core processing [10].

TensorFlow

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow was developed by the Google Brain team for internal Google use in research and production. TensorFlow can be used in a wide variety of programming languages, most notably Python, as well as JavaScript, C++, and Java. This flexibility lends itself to a range of applications in many different sectors [20].

Mobile Application Language

Table 2.7: Comparison List for Mobile Applications Language

Mobile Application language	FLUTTER	KOTLIN	JAVA	REACT NATIVE
Supported Platforms	Android Jelly Bean, v16,4.1.x and IOS 8+	Android and IOs 8+	Android apps	Android 4.0.3+ Versions and ios 8+
Language Stack	Dart	JS and Native	Java works on JVM	JS and React.JS
Performance	Removed JS Bridging,enhanced App speed	Interoperable With Java and Java Virtual Machine	Fewer bugs	Higher Performance that of native app
Pricing	Open-source platform	Free of cost	Paid updates for JDKcost	React native is Open source
User interface	Easy-to-use interface	Remarkable user experiences	Flexible extensible ,and scalable	Impressive graphical user interface 

We chose flutter framework because it has a high performance and easy for beginners to programming and the User interface is easy to use as shown in Table 2.7 and the flutter have a feature called hot Reload while your application is running, you can make changes to the code and apply them to the running application. No recompilation is needed, and when possible, the state of your application is kept intact.

Flutter Framework

Flutter is an open-source framework released by Google in May 2017 for building beautiful, natively compiled, multi-platform applications from a single codebase and Flutter code compiles to ARM or Intel machine code as well as JavaScript, for fast performance on any device [13].

We will use flutter framework in our mobile application in project because a several reason [13]:

Reduced Code Development Time ,Similar to Native App Performance ,Simple Platform-Specific Logic Implementation and Open-source platform.

Chapter 3

System Design

3.1 Overview

In this chapter, we will explain the abstract block diagram of the system. Next, show the detailed description of the system and the detailed design for each component including its schematic diagram will be introduced. Finally, we will explain the schematic diagram of the hardware components of the system.

3.2 Detailed Description Of The System

We are going to design and implement an object finder robot using Raspberry Pi and based on a python environment. The system will use ROS, which is a flexible framework for writing robot software. ROS is a collection of tools, libraries, and conventions that aims to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms [3].

TurtleBot will be used since it is a low-cost, personal robot kit with an open-source hardware platform that has a mobile base and is supported by ROS and localization and mapping and vision and obstacle avoidance [12]. A depth camera is used as a special camera that can determine depth and distance to take image of new levels. It uses the know speed of light to measure distance, and effectively calculates the amount of time it takes for a reflected beam of light to return to the camera sensor.

In our project, we will use Xbox 360 Kinect camera. It contains three vital pieces that work together to detect your motion and create your physical image on the screen: an RGB color VGA video camera, a depth sensor, and a multi-array microphone.

3.3 System Diagrams

This section introduces two diagrams that represent block diagram and schematic diagram to understand the project concepts and design.

3.3.1 System Block Diagram

The general block diagram of the system is shown in Figure 3.1 As illustrated below, first, the user request the objects to be searched from turtlebot2, The Turtle-Bot2 starts explore the place and the map is generated by gmapping algorithm which it is programmed with it and simultaneously YOLO algorithm is running and detecting the known objects and classifies them with x and y coordinates of object in each frame captured by camera saves it in raspberry Pi. The LIDAR sensor sends a data indicating if there is an obstacle or not to the Raspberry pi. Finally, the Raspberry pi sends the result to the mobile application by MQTT protocol to display the existence or absence of the objects and if it exists, the mobile application displays the coordinates of the objects in the map.

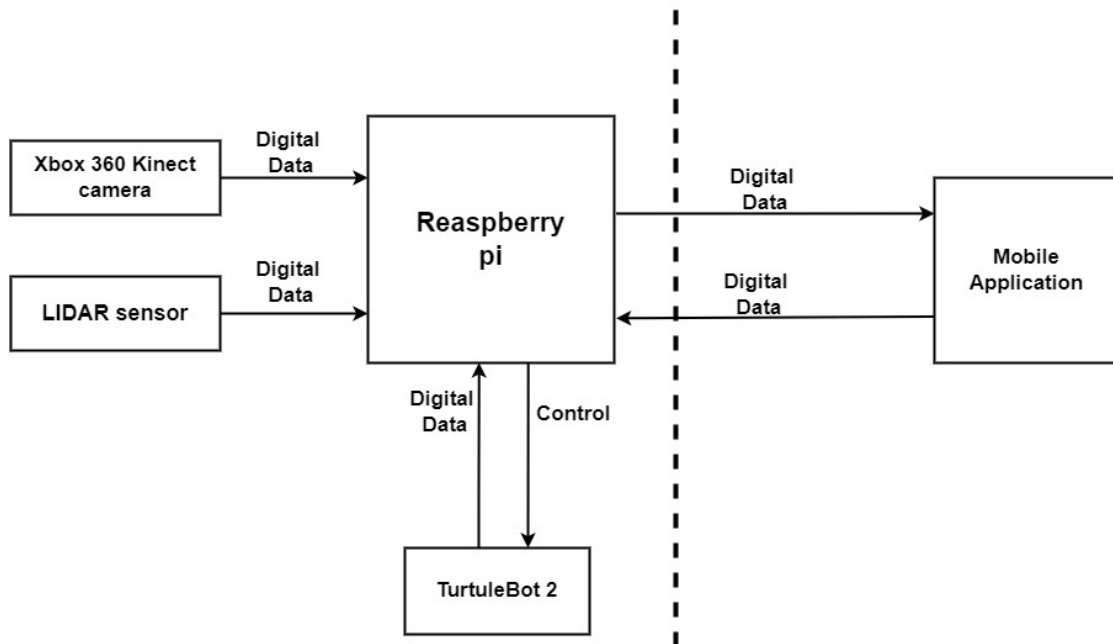


Figure 3.1: System Block Diagram

From the above discussions,

The main components of the system diagram are:

1. Raspberry pi: It is the main component of the system and The processing node of the mobile robot and it controls components connect to it for example: LIDAR Sensor, Xbox Kinect camera and motor driver of the mobile robot.

2. Xbox Kinect camera: It captures images of objects using a command from Raspberry Pi.

3. LIDAR Sensor: It checks if there is any object in front of the system in the range of about 4 meters for 240° degree detection then sends a signal to raspberry pi to give an order to the Xbox Kinect camera to capture an image.

4. TurtleBot2: It is one of the most important components of the project as it is the component on which will be implemented the project in it and will discovering the place and avoid obstacles so that all other components are linked to it.

5. Mobile Application: The user selects the object through the application and sends the data as input. The result of the search for an object is displayed as output in the mobile application.

3.3.2 Schematic Diagram

Figure 3.2 shows The connect between Raspberry Pi and DC motor(kobuki robot).

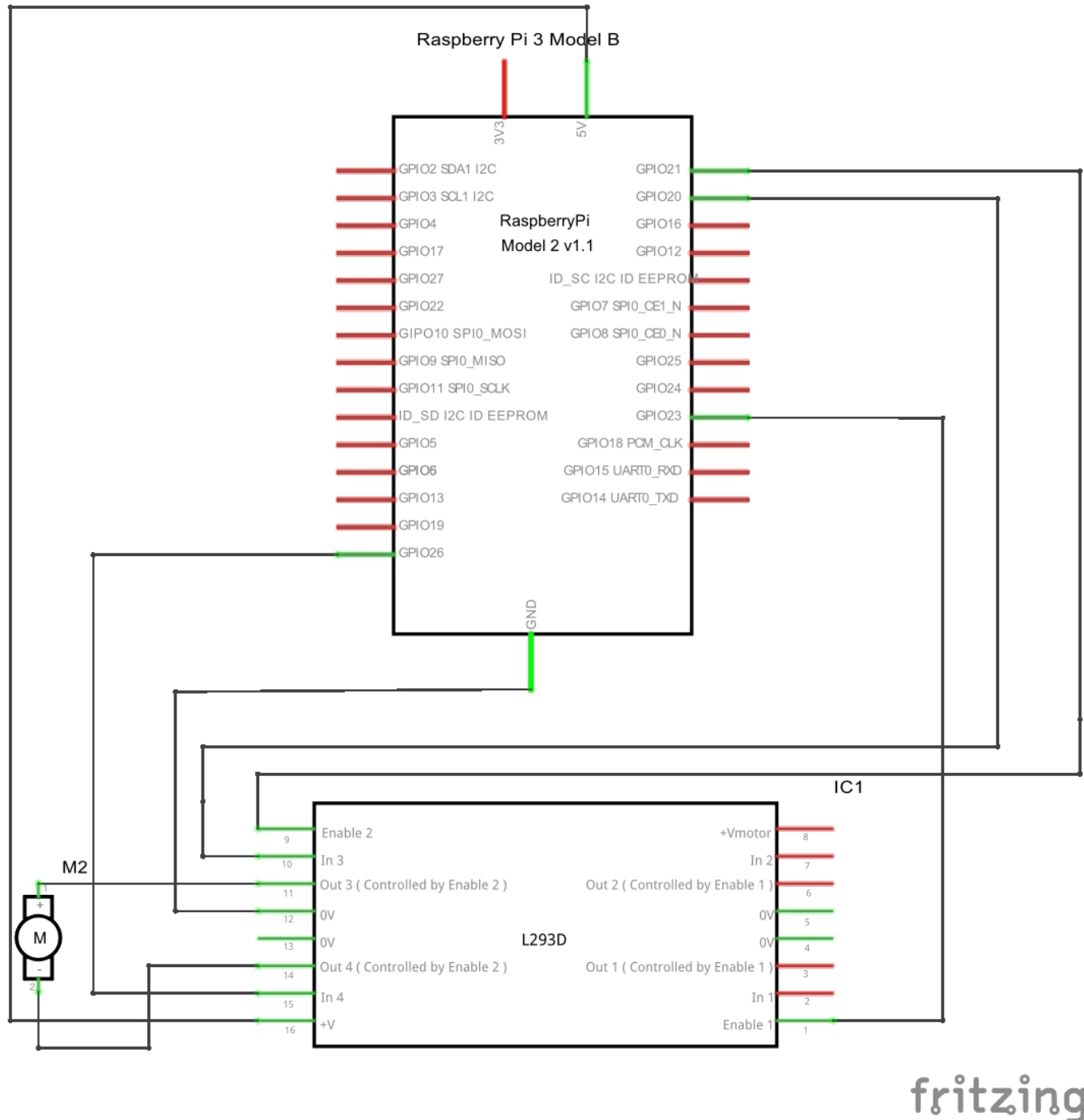


Figure 3.2: Schematic diagram for DC Motor(kobuki robot) with Raspberry Pi

We connect dc motor the pin 1 to Out3(Controlled by Enable2) and the pin 2 to Out4(Controlled by Enable2) in L293D motor driver and GND pin in Raspberry Pi to 0V(GND) in L293D motor driver and the vcc pin in Raspberry Pi to +V pin

in in motor driver then connect In3 and in4 pins to GPIO23 and GPIO20 pins in Raspberry Pi and Enable2 in motor driver pin to GPIO21 pin in Raspberry Pi.

Figure 3.2 shows The connect between Raspberry Pi and Xbox Kinect camera.

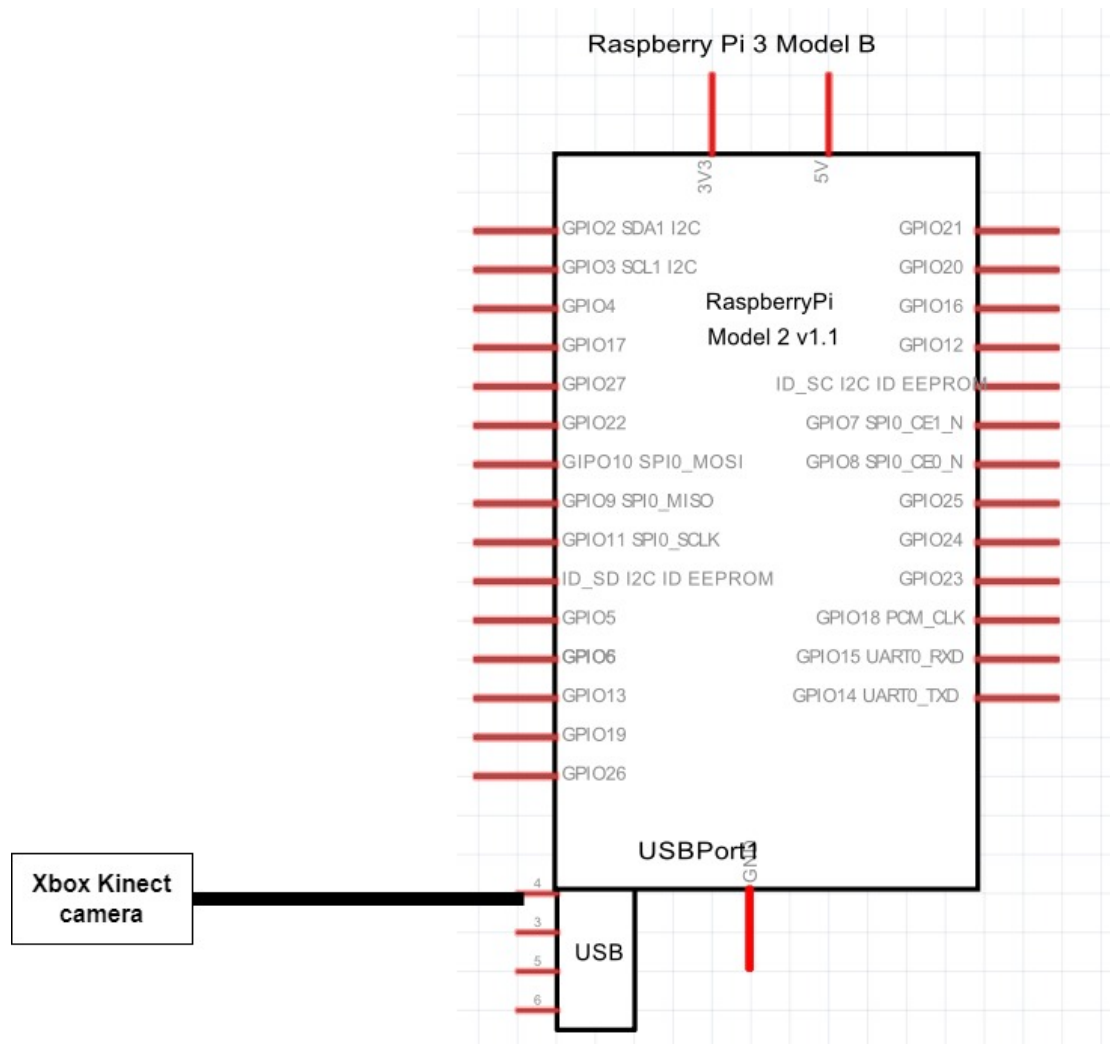


Figure 3.3: Schematic diagram for Xbox Kinect Camera with Raspberry Pi

We connect the XBOX Kinect camera to USB Port.

Figure 3.4 shows The LIDAR Sensor with Raspberry Pi.

We connect pin 6(GND) in Lidar sensor to GND in Raspberry Pi and pin 1 vcc to pin 5v and pin2(Mode) and pin3 (PWR EN) to Usb port in Raspberry Pi.

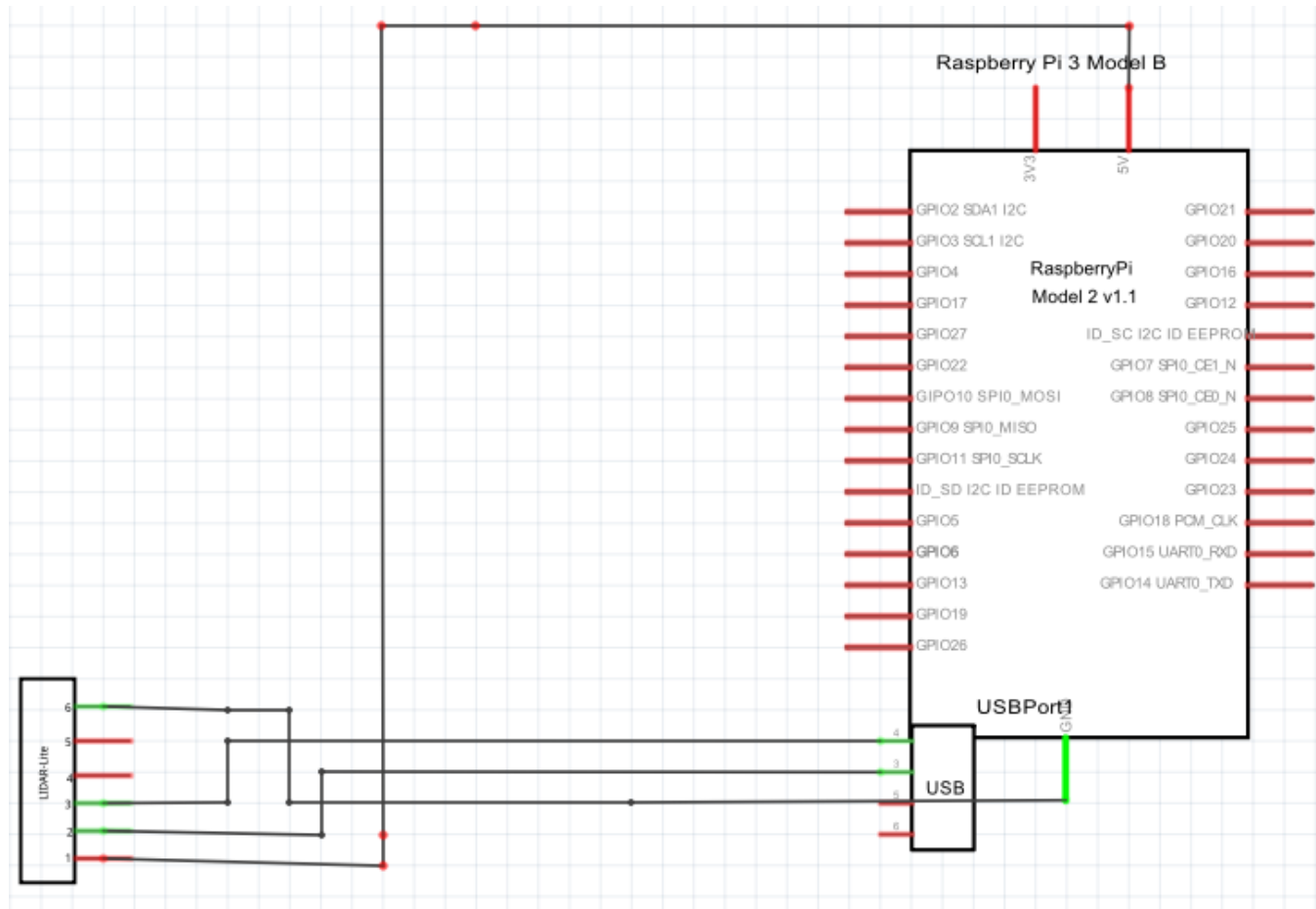


Figure 3.4: Schematic diagram for LIDAR Sensor with Raspberry Pi

3.4 Pseudo-Code

Algorithm 1 Finding The Object

```

procedure FINDING THE OBJECT
  INPUT   The user Request any object from turtlebot
  While:
    procedure (LIDAR sensor reading data using ROS SLAM algorithm to
    generated the map)
      procedure (Camera reading images using YOLO algorithm)
        if class == select object and probability >= thresholdvalue then
          distance between robot and obstacle equal 0.5m
          obstacle avoidance
          Mapping Map x and y coordinate detected object from YOLO with x and y
          coordinate in the gerated map
        OUTPUT The Raspberry Pi sends data to mobile application by
        MQTT to display it
  End While

```

3.5 Adjustments

When we started working on the project, we ran into some issues that forced us to make some changes:

1. Raspberry pi 3 model B

The Raspberry Pi 3 model B, which we intended to use in the project, has been replaced by a laptop Because the Raspberry Pi 3 model B a rather weak processor and a one gigabyte of memory, and when we ran the whole project, the processor consumption was 100%, and the memory was over 80% and Figure 3.6 shows this case , and therefore there is a not of responding in the system Significantly. This is because the YOLO algorithm requires a fairly powerful processor and consumes a large part of the memory. The mapping algorithm also consumes part of the processor and memory, so the two algorithms cannot be run at the same time. so we replaced Raspberry Pi .



Figure 3.5: from raspberry pi to Dell laptop

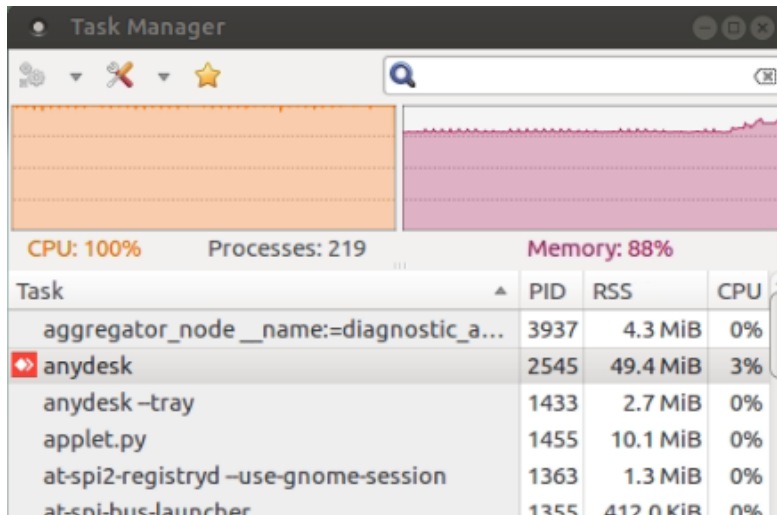


Figure 3.6: CPU and Memory usage of Raspberry Pi

2.URG-LIDAR Sensor

The URG-LIDAR sensor, which we had intended to use in the project, we encountered problems programming in the ability of the robot to avoid obstacles by LIDAR sensor, and to solve that we replaced it with a kinect camera XBOX 360 because it contains a laser to sense what is in front of it.



Figure 3.7: URG LiDAR and XBOX Kinect

Chapter 4

System Implementation

4.1 Overview

This chapter covers the software and hardware implementation of the project, as well as the various components and tools needed to construct the robot

4.2 Hardware Implementation:

The main component is the laptop which is connected with the other system components as follows:

1. We connected Turtlebot2(kobuki) to laptop by USB cable.
2. We connect XBOX 360 Kinect Sensor to laptop with adapter power supply Cable by USB and connected power supply with kobuki 12V/5A.

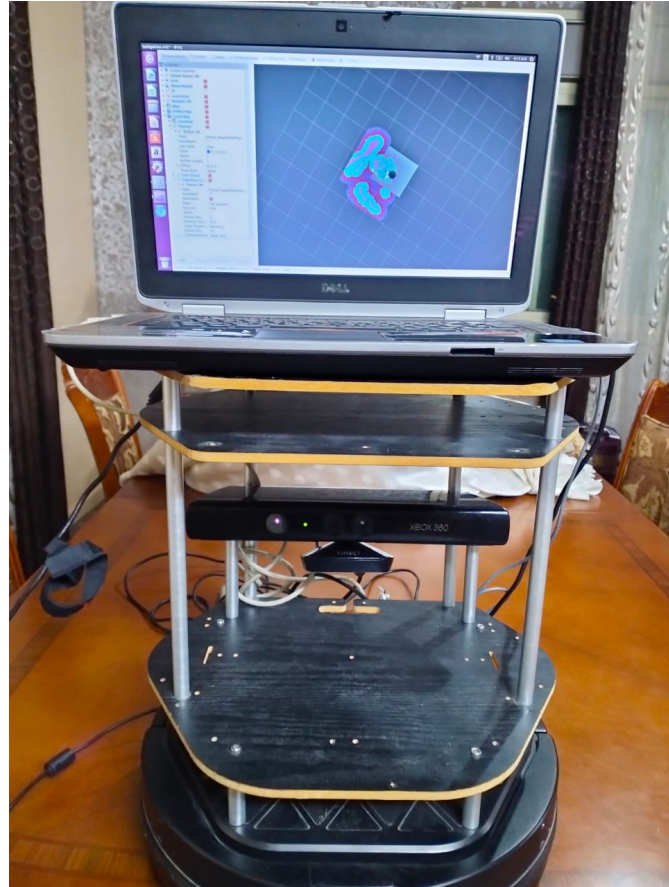


Figure 4.1: System Components

Figure 4.4 shows the Xbox 360 Kinect sensor to laptop with adapter power supply Cable by USB and connected power supply with kobuki 12V/5A



Figure 4.2: Connect XBOX 360 Kinect Sensor to Kobuki

4.3 Software Implementation:

4.3.1 Operating System

We installed Ubuntu 16.04 OS on laptop because the ROS kinetic release only supports Ubuntu 15.10, Ubuntu 16.04.

4.3.2 Installing needed packages

Raspbian comes with many useful pre-installed packages such as JDK, Python, and others, but we need to install some needed packages for our project. The most important one is the OpenCV library that will be used to provide object detection features. Many essential pre-installed packages, such as JDK, Python, and others, come with Raspbian, but we'll need to install a few more for our project. The OpenCV library, which will be utilized to offer object detection characteristics, is the most significant.

4.3.3 OpenCV implementation

OpenCV is a Python binding library for solving computer vision problems. The researchers followed the steps to install this library[23]:

- Install OpenCV dependencies on system.
- Download the OpenCV source from github .
- Setup suitable Python environment(python2.7)for our system.
- Install NumPy package on Python,This package is necessary to run OpenCV.
- Configured and compiled OpenCV on system.

4.3.4 Object detection implementation

We need an algorithm to implement object detection in our robot so that it can detect the bottle , remote and ball that need to find it. As for this project the YOLOV3-tiny algorithm was chosen for this project.

4.3.5 YOLO Implementation

The researchers followed the steps to implement YOLO[24]:

- Install the necessary dependencies, such as OpenCV and NumPy, on system.
- We Download the YOLO3-tiny.weight file and configuration YOLO3-tiny.cfg file and coco.names from the official website[24].
- Clone the Darknet repository from GitHub on system.

Darknet is an open-source framework that includes the YOLO object detection algorithm.

- Build Darknet on system by running the 'make' command in the Darknet directory.

4.3.6 ROS implementation

ROS provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more.

The researchers use Ros Kinetic release Because its compatibility with the Ubuntu 16.04 OS We use the follow the instructions in [25] to install ROS Kinetic.

4.3.7 ROS algorithms

The following are the algorithms that are responsible for fully directing the robot.

Gmapping

The map is drawn by the movement of the robot in a specific location, with distance sensors reading the distances between the robot and nearby obstacles, and during the drawing the robot evreything around is being discovered.

A.Inputs:

- 1.Distance sensors readings.
- 2.Encoders sensors readings.

B. Outputs:

The map

C.How to use Gmapping in Terminal(as commands):

Bring a TurtleBot2 to Create a Map.

```
$ roslaunch turtlebot_bringup minimal.launch.
```

```
$ roslaunch turtlebot_navigation gmapping_demo.launch.
```

Launch the rviz program.

```
$ roslaunch turtlebot_rviz_launchers view_navigation.launch.
```

To control the Turtlebot2 by keyboard.

```
$ roslaunch turtlebot_teleop keyboard_teleop.launch.
```

Gmapping result:

The map resulted using Rviz tool is shown in Figure 4.3

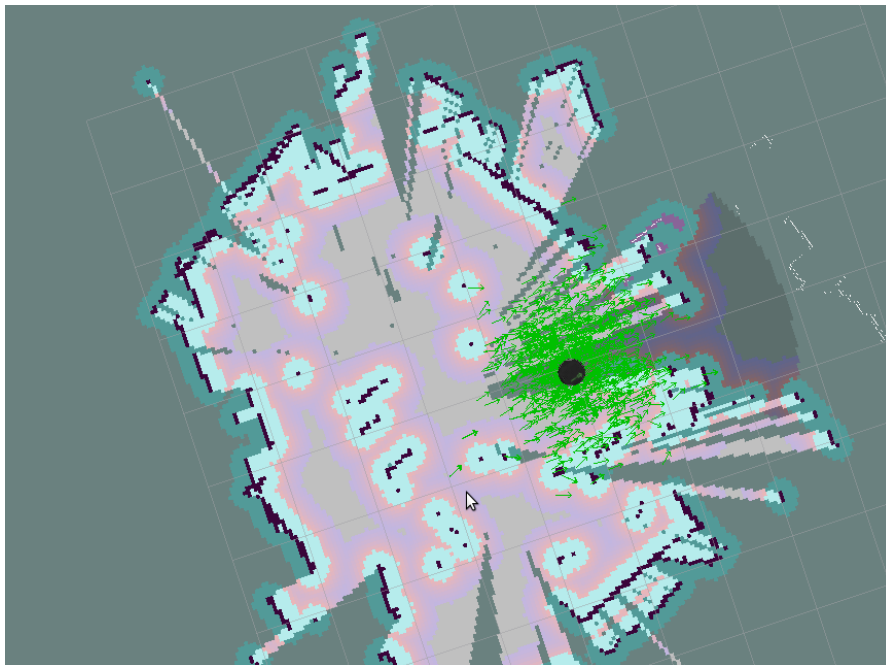


Figure 4.3: Gmapping Result

4.4 Mobile Application Implementation

4.4.1 MQTT Protocol

MQTT is a lightweight open messaging protocol that provides resource-constrained network clients with a simple way to distribute telemetry information in low-bandwidth environments. The protocol, which employs a publish/subscribe communication pattern, is used for machine-to-machine (M2M) communication[26].

We have used the following libraries in the pubspec.yaml file. Every pub package needs some metadata so it can specify its dependencies. Pub packages that are shared with others also need to provide some other information so users can discover them. All of this metadata goes in the package's pubspec: a file named pubspec. yaml that's written in the YAML language.

```
dependencies:
  flutter:
    sdk: flutter
  mqtt_client: ^9.3.1 #^7.2.1
  provider: ^5.0.0 #^4.1.3+1
```

The library mqtt-client is a plugin that is responsible for exchanging data between the application and the robot so that it achieves effectiveness and ease of dealing.

The library provider is a plugin that is an easy-to-use package which is basically a wrapper around the Inherited widgets that makes it easier to use and manage. It provides a state management technique that is used for managing a piece of data around the application[27].

4.4.2 User Interface

This section will introduce the interface that was designed in the application. There are main pages in our application. It contains an AppBar at the top of the application containing the name of our project, and contains two buttons: Connect and Disconnect. When we want to connect to an application to the laptop by MQTT protocol, you must press the Connect button and make sure that the phone is connected to the Internet and the MQTT protocol connection has been successful. It contains a place design for placing the map inside, and the coordinates of the objects are set on the map, and each object has a special color to distinguish it on the map. Such as a ball is red, a remote is green, and a bottle is yellow.

Figure 4.4 shows the Flutter Application Implementation.

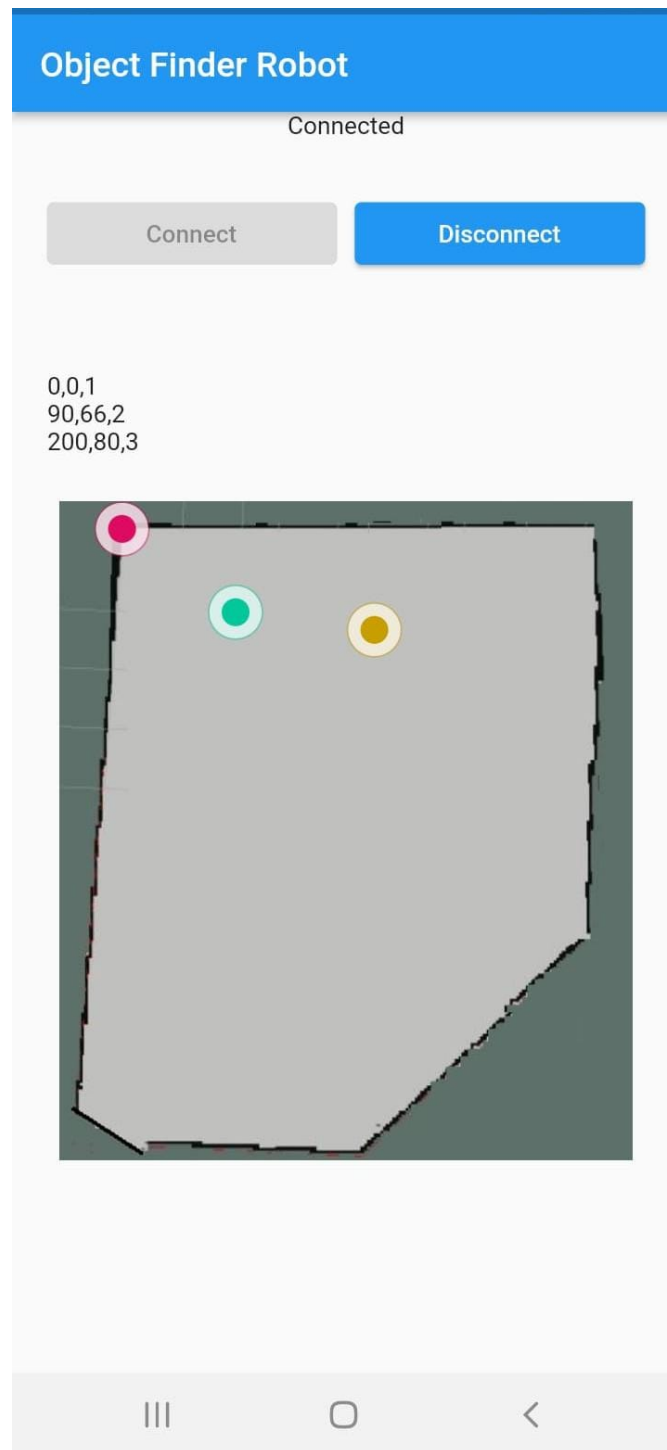


Figure 4.4: Mobile Application Implementation

This interface contains the following functionalities:

`OpenPainter({this.x_value, this.y_value}):`

This class obtains the coordinates that were sent from the robot and draws and set these coordinates in the form of points within the map created by the robot

each object have a color distinguish it.

MQTTManager: This class, its main function is the application is connected to mqtt protocol or not.

MQTTView: This class contains 2 buttons to display the status of the application connected or not and contains a map of the place to search for an object and contains a function to draw points on the map

4.5 Implementation issues and challenges

- At the beginning of the project implementation, we use the URG-LIDAR sensor to draw map the movement of the robot in a specific location and avoid the obstacles in front of it, as was planned in the implementation of the project, but we encountered problems in the ability of the robot to avoid obstacles, and we solved this issue by replacing it with Kinect XBOX 360.
- We faced problems in finding compatibility between downloading opencv library and tensorflow library on the version of the system that we have to run the YOLO algorithm, and to solve that replaced with opencv library and Darknet.
- When running the ROS environment and the YOLO algorithm in Raspberry Pi OS, we faced a problem that the system stopped responding, because the Ros environment uses more than half of the processing in cpu and the Yolo algorithm, when it runs, consumes a large amount of processing, and to solve that we replaced the Raspberry Pi 3 with a Laptop and when running the ros environment with the Yolo algorithm it was much better.
- We faced a problem in the process of transferring data between the Raspberry Pi and flutter application and to solve that we used the MQTT protocol to transfer data between the two parties.
- We faced a problem when using the Kinect Camera to identify the object using the Yolo algorithm and Gmapping algorithm to draw the map through it, but the camera port cannot be used for the two algorithms at the same time, so to solve that we used kinect Camera for the Gmapping algorithm to draw the map and used the webcam on the laptop for the Yolo algorithm to identify the object.

Chapter 5

Validation and Testing

5.1 Overview

This chapter explains the project component testing methodology and displays the project system implementation outcomes.

5.2 Hardware Testing

5.2.1 Testing DC Motor In Kobuki

The Wheels in kobuki was tested by connected to a laptop by usb cable And run the python script that was written for drive kobuki in linear motion or angular, in linear motion the robot drive forward or backward,in angular motion the robot dive to left or right with angular.

5.2.2 Testing Kinect XBOX 360

The Kinect XBOX 360 camera was attached to laptop with adapter power supply Cable by USB and connected power supply with Kobuki 12V/5A .
rosrun image_view image_view image_camera/rgb_image_color.

Command was written to open live video stream, and we tested the sensor in Kinect camera by Gmapping algorithm and verify that when the robot moves, the map is drawn within the Rviz program as shown in figure 4.3.

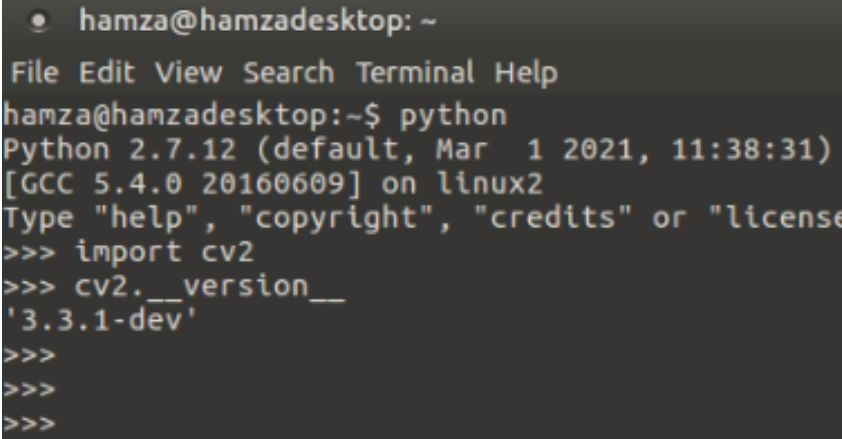
Autonomous Motion And Avoiding Obstacles

We tested a self-motion for the robot, robot and we checked if it was moving correctly, and we written python code to avoiding obstacles during movement and the python code is exist in Appendix B.

5.3 Software testing

5.3.1 OpenCV testing

To test if the OpenCV library was installed correctly. We tried the below in figure 5.1 show the commands in the python terminal and print the version of opencv 3.3.1 that mean it's installed correctly.

A screenshot of a terminal window with a dark background and light text. The prompt is 'hamza@hamzadesktop: ~'. The terminal shows the command 'python' being executed, which opens a Python 2.7.12 shell. The user enters 'import cv2' and then 'cv2.__version__', which outputs '3.3.1-dev'.

```
hamza@hamzadesktop: ~  
File Edit View Search Terminal Help  
hamza@hamzadesktop:~$ python  
Python 2.7.12 (default, Mar 1 2021, 11:38:31)  
[GCC 5.4.0 20160609] on linux2  
Type "help", "copyright", "credits" or "license"  
>>> import cv2  
>>> cv2.__version__  
'3.3.1-dev'  
>>>  
>>>  
>>>
```

Figure 5.1: OpenCV Testing

5.3.2 Testing Of YOLO Algorithm

YOLO algorithm that will be used to provide object detection. We downloaded a Github project of the YOLO algorithm. It contains files such as yolov3-tiny.cfg, yolov3-tiny.weights ,and coco.names. Then run a python script to detect objects that exist in front of the camera. Then detected objects were bound by a box and above each box there is a label which is the name of detected object figure 5.2 shows the output of YOLO algorithm.

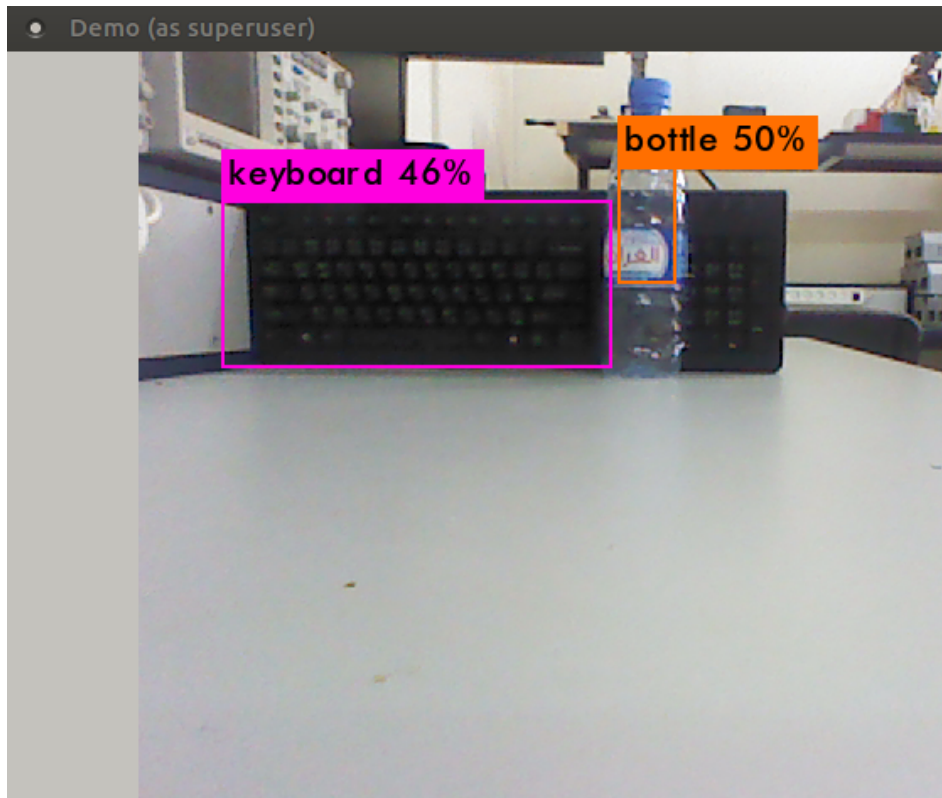


Figure 5.2: Testing YOLO Result

5.3.3 Software Testing Tables

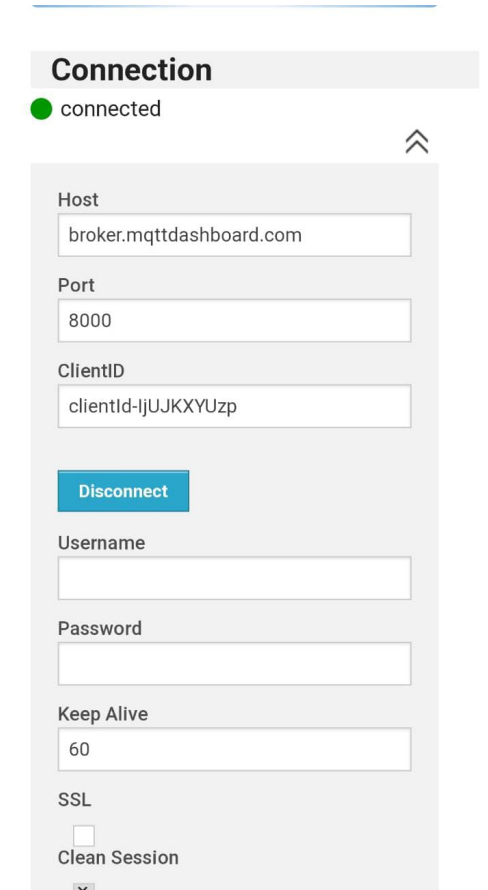
Table 5.3 shows the Software testing of system.

Table 5.3: Software Testing Table

Case	Expected Output	Obtained Output	Pass/Fail
Connect to the MQTT	Connect to the MQTT	Connect to the MQTT successfully	Pass
Get necessary data from Kinect sensor	Distance from robot and objects	Get distance successfully	Pass
Get necessary data from Robot	Get Coordinates of objects	Get Coordinates of object successfully	Pass
Object detection	Detect objects	Detect objects successfully	Pass

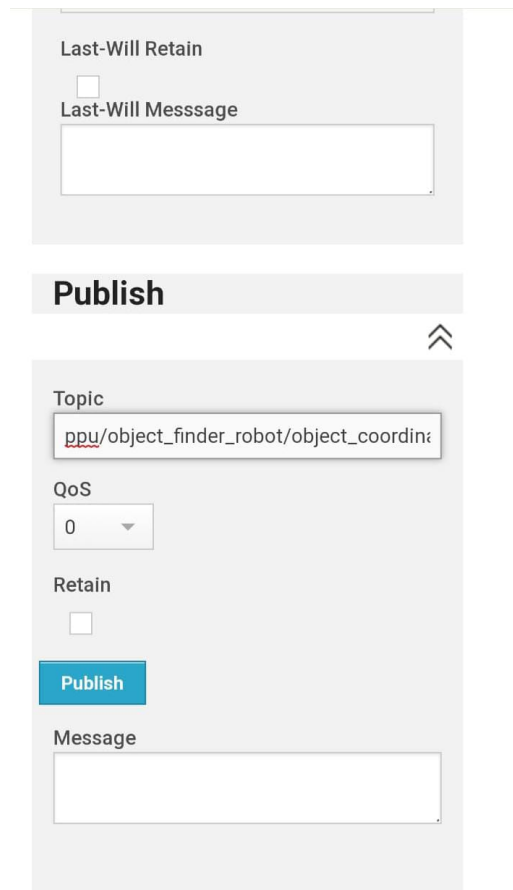
5.3.4 Mobile Application Testing

We linked the mobile application with the laptop by MQTT protocol, and verified sending and receiving data between them, and we written a Python code to send the data from laptop to application,we used the MQTT protocol free website to reserve host,port and topic as shown in figure 5.3 and figure 5.4.



The image shows a web interface for MQTT settings. At the top, there is a header "Connection" with a green dot and the text "connected". Below this, there is a list of configuration fields: "Host" (broker.mqttdashboard.com), "Port" (8000), "ClientID" (clientId-IjUJKXYUzp), "Disconnect" (button), "Username" (empty field), "Password" (empty field), "Keep Alive" (60), "SSL" (checkbox), and "Clean Session" (checkbox). There is also a small "x" icon at the bottom left of the settings panel.

Figure 5.3: MQTT Setting



The image shows a screenshot of an MQTT client interface. At the top, there is a section for 'Last-Will Retain' with a checkbox and a 'Last-Will Message' text area. Below this is a 'Publish' section with an upward arrow icon. The 'Publish' section contains a 'Topic' field with the text 'ppu/object_finder_robot/object_coordin:'. Below the topic field is a 'QoS' dropdown menu set to '0'. There is a 'Retain' checkbox which is unchecked. A blue 'Publish' button is located below the 'Retain' checkbox. At the bottom of the 'Publish' section is a 'Message' text area.

Figure 5.4: Topic in MQTT

5.4 System Validation

After each component is individually connected to the laptop and tested. We connect all the parts of the robot, and run the Python code check that the system is working correctly and the robot draws a map of where it is located correctly and identifies what is in front of it using the YOLO algorithm correctly and the coordinates of the objects are mapped correctly on the map within the mobile application.

Chapter 6

Conclusion and Future work

6.1 Conclusion

In this project, we have presented an approach to building a prototype intelligent mobile robot when it moves map is drawn . We created a communication between the smartphone and the laptop using mqtt protocol to send the objects coordinates from the robot to the mobile application The robot also can detect objects using Yolo algorithm and localize a certain object within its environment, and developed a mobile application to determine coordinates of objects in the map. It can be concluded that higher speed to detect objects in yolo algorithm could be achieved with using higher CPU and using GPU with microcontroller.

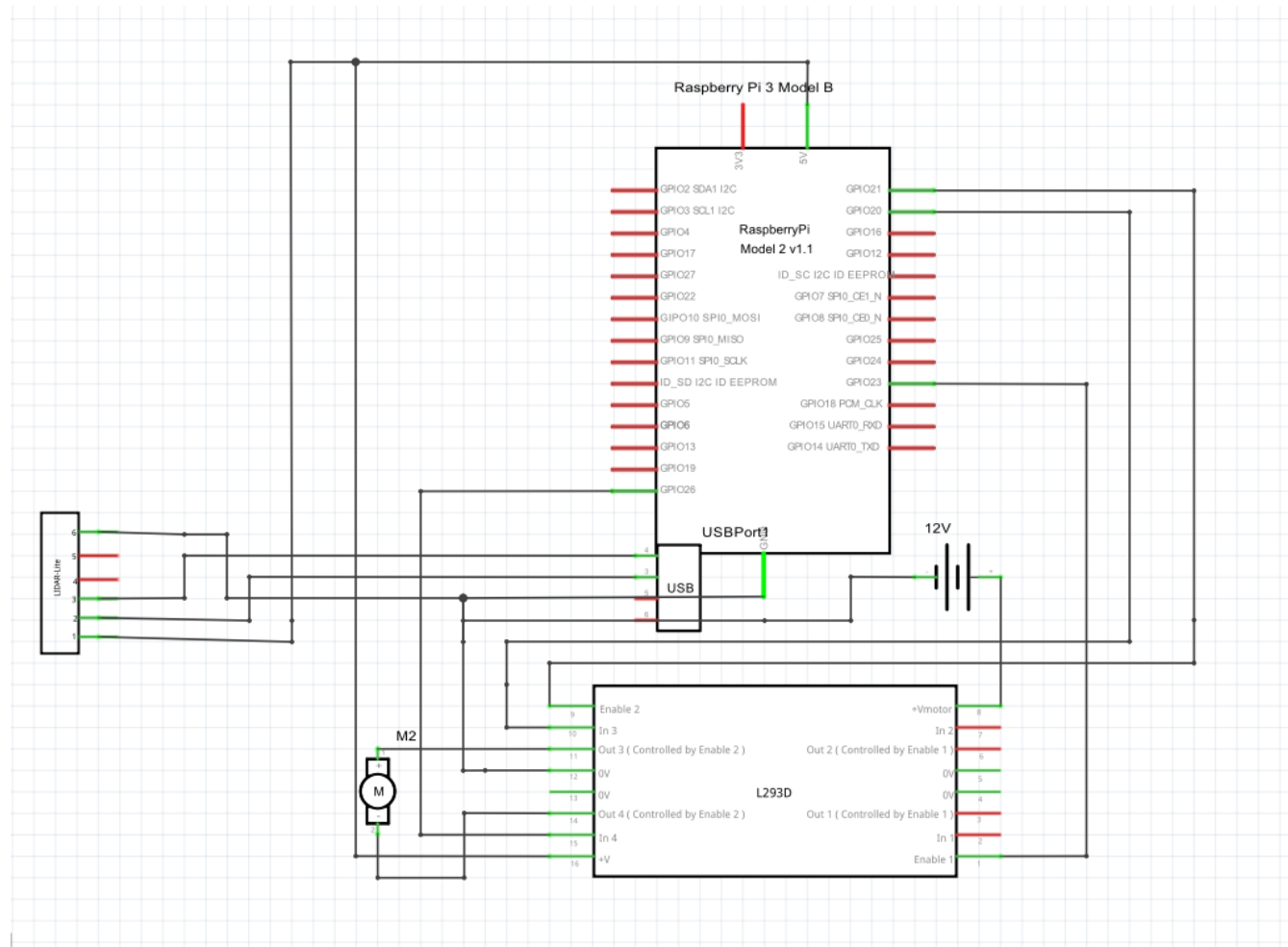
6.2 Future work

Some future works are suggested and recommended to improve the project:

1. Adding additional objects to be finding.
2. Attach a robotic arm to be able to bring the object.
3. Add feature of self charging using automatic docking for recharging when the battery is low by itself.

Appendix A

Schematic Diagram for the system



Appendix B

avoid_obstacle.py script

```
import rospy

from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal

class GoForwardAvoid():

    def __init__(self):

        rospy.init_node('nav_test', anonymous=False)

        rospy.on_shutdown(self.shutdown)

        self.move_base = actionlib.SimpleActionClient("move_base", MoveBaseAction)

        rospy.loginfo("wait for the action server to come up")

        self.move_base.wait_for_server(rospy.Duration(5))

        goal = MoveBaseGoal()

        goal.target_pose.header.frame_id = 'base_link'

        goal.target_pose.header.stamp = rospy.Time.now()

        goal.target_pose.pose.position.x = 3.0

        goal.target_pose.pose.orientation.w = 1.0

        self.move_base.send_goal(goal)

        success = self.move_base.wait_for_result(rospy.Duration(60))

        if not success:

            else:

                state = self.move_base.get_state()

                if state == GoalStatus.SUCCEEDED:

                    rospy.loginfo("Hooray, the base moved 3 meters forward")

if __name__ == '__main__':

    try:

        GoForwardAvoid()

    except rospy.ROSInterruptException:

        rospy.loginfo("Exception thrown")
```

drawing coordinate on the map Ganvas.dart

```
class OpenPainter extends CustomPainter {

  final x1_value;
  final y1_value;

  final x2_value;
  final y2_value;

  final x3_value;
  final y3_value;

  final x4_value;
  final y4_value;

  double new_orgin_x=50/455*330;//50 is x position start map
  double new_orgin_y=22/522*379;

  double x_factor=330/455;//330:widget img width & 455:original img width
  double y_factor=379/522;//379:widget img height & 522:original img height

  OpenPainter({this.x1_value, this.y1_value,this.x2_value, this.y2_value,this.x3_value, this.y3_value,this.x4_value,
  this.y4_value});

  @override
  void paint(Canvas canvas, Size size) {

    var whiteCircle = Paint()
      ..color = Color(0xbbFFFFFF)
      ..style = PaintingStyle.fill;
  }
}
```

connect to MQTT protocol MQTT_Connection.dart

```
void _configureAndConnect() {
  // ignore: flutter_style_todos
  // TODO: Use UUID
  String osPrefix = 'Flutter_iOS';
  if (Platform.isAndroid) {
    osPrefix = 'Flutter_Android';
  }
  manager = MQTTManager(
    host: "broker.hivemq.com",
    topic: "ppu/object_finder_robot/object_coordinate",
    identifier: osPrefix,
    state: currentAppState);
  manager.initializeMQTTClient();
  manager.connect();
}

void _disconnect() {
  manager.disconnect();
}

void _publishMessage(String text) {
  String osPrefix = 'Flutter_iOS';
  if (Platform.isAndroid) {
    osPrefix = 'Flutter_Android';
  }
  final String message = osPrefix + ' says: ' + text;
  manager.publish(message);
  _messageTextController.clear();
}
```


References

- [1] Rodney E Peters, Richard Pak, Gregory D Abowd, Arthur D Fisk, and Wendy A Rogers. May,2004. Finding lost objects: Informing the design of ubiquitous computing services for the home. Technical Report GIT-GVU-04-01. Georgia Institute of Technology.
- [2]“An article on the average number of days a person spends searching for a particular object?” May, 2017,. [Online]. Available: <https://www.prnewswire.com/news-releases/lost-and-found-the-averageamerican-spends-25-days-each-year-looking-for-lost-items-collectively-costing-us-households-27-billionannually-in-replacement-costs-300449305.html/>.
- [3] “Wiki,” *ros.org*. [Online]. Available: <http://wiki.ros.org/gmapping/>. [Accessed: 27-May-2022].
- [4] *Medium*. [Online]. Available: <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detectionalgorithms-36d53571365e>. [Accessed: 22-Dec-2022].
- [5] “Introduction to yolo algorithm for object detection,” *Section*. [Online]. Available: <https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/>. [Accessed: 21-Dec-2022].
- [6] “Real-time operating system,” *Wikipedia*, 19-Dec-2022. [Online]. Available: https://en.wikipedia.org/wiki/Real-time_operating_system. [Accessed: 25-Oct-2022].
- [7] “An introduction to robot operating system (ROS) - technical articles,” *All About Circuits*. [Online]. Available: <https://www.allaboutcircuits.com/technical-articles/an-introduction-to-robot-operating-system-ros/>. [Accessed: 11-Dec-2022].
- [8]“ “Raspberry,” *Wikipedia*, 08-Dec-2022. [Online]. Available: <https://en.wikipedia.org/wiki/Raspberry>. [Accessed: 27-Dec-2022].
- [9] “Welcome to Python.org,” Python.org. [Online]. Available: <https://www.python.org/>. [Accessed: 22-Dec-2022].
- [10] “Home,” OpenCV, 21-Dec-2022. [Online]. Available: <https://opencv.org/>. [Accessed: 27-Dec-2022].
- [11] “How does the xbox kinect work,” How It Works: Xbox Kinect. [Online]. Available: <https://www.jameco.com/Jameco/workshop/howitworks/xboxkinect.html>. [Accessed: 10-Nov-2022].

- [12] "A 'Getting started' guide for developers interested in robotics," *Learn TurtleBot and ROS*. [Online]. Available: <https://learn.turtlebot.com/>. [Accessed: 5-Jun-2022].
- [13] "Build apps for any screen," *Flutter*. [Online]. Available: <https://flutter.dev/>. [Accessed: 30Dec-2022].
- [14] Object detection and tracking system: Oraib Daas, Safa Shehada June-2017, Dr. Ashraf Armoush, Dr.Emad Natsheh ,[Online]
<https://repository.najah.edu/handle/20.500.11888/14334?show=full>.
- [15] Autonomous Wheelchair Project: Akram abu ayyash ,Islam warasna , June-2021 , Dr. mohammad aldesht ,[Online]
<https://scholar.ppu.edu/handle/123456789/38/browse?type=author&value=abu+ayyash%2C+Akram>.
- [16] Sanitizer Spider Robot: Bayan Karajat,Rana Awlad Mohammad, June-2021, Eng.Wael Takroui. [17]"Why use ROS"April, 2018, [Online]. Available:
<https://service.niryo.com/en/blog/8-reasons-use-rosrobotics-projects>.
- [18] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, may-2009., ROS: an open-source Robot Operating System.
- [19] "Online store for robotic products supported by Ros," *ROS Components*. [Online]. Available: <https://www.roscomponents.com/en/>. [Accessed: 27-Dec-2022].
- [20] "Tensorflow," *Wikipedia*, 18-Dec-2022. [Online]. Available: <https://en.wikipedia.org/wiki/TensorFlow>. [Accessed: 20-Oct-2022].
- [21] "Wiki," *ros.org*. [Online]. Available: <http://wiki.ros.org/>. [Accessed: 16-May-2022].
- [22] "Wiki," *ros.org*. [Online]. Available: <http://wiki.ros.org/ROS/CommandLineTools>. [Accessed: 25-Oct-2022].
- [23] A. Rosebrock, "Ubuntu 16.04: How to install opencv," *PyImageSearch*, 17-Apr-2021. [Online]. Available: <https://pyimagesearch.com/2016/10/24/ubuntu-16-04-how-to-install-opencv/>. [Accessed: 27-Dec-2022].
- [24] J. Redmon, *Installing darknet*. [Online]. Available: <https://pjreddie.com/darknet/install/>. [Accessed: 10-Apr-2022].
- [25] "Wiki," *ros.org*. [Online]. Available: <http://wiki.ros.org/kinetic/Installation/Ubuntu>. [Accessed: 12-Oct-2022].

[26] C. Bernstein, K. Brush, and A. S. Gillis, "What is MQTT and how does it work?," *IoT Agenda*, 27-Jan-2021. [Online]. Available: <https://www.techtarget.com/iotagenda/definition/MQTT-MQ-Telemetry-Transport>. [Accessed: 27-Dec-2022].

[27] "Flutter - provider package," *GeeksforGeeks*, 01-Feb-2021. [Online]. Available: <https://www.geeksforgeeks.org/flutter-provider-package/>. [Accessed: 27-Dec-2022].