ABSTRACT

ROS-Based Educational Mobile Robot MOBOT is an open source, educational mobile robot has been developed to be the first educational oriented project at Sana'a University - Faculty of Engineering - Mechatronics Engineering Department. It would be dedicated for the fields of Robotics and Artificial Intelligence Artificial Intelligence (AI) including the other related fields like *Mechatronnics System Design, etc.*

MOBOT is an out-door mobile robot intended for educational purposes for those learners concern about Robotics and AI study fields. It is based on Robotic Operating System (ROS). The robot relies on the Raspberry platform controller as the primary controller on which ROS was installed. It is the master controller over the other controller platforms (Arduinos)

The robot is able to navigate the environment around and create the map of the place where it is located. It uses **(!** ((**!**)LiDAR) laser scanner sensor to scan the environment for the robot to able to avoid obstacles.

ACKNOWLEDGEMENTS

We would like to express our deepest appreciation to all those who provided us the possibility to complete this project. A special gratitude we give to our supervisors in our our project MOBOT, Dr. Ghassan Ben Hammam and Eng. Mohammed Doba for their patient guidance, enthusiastic encouragement and useful critiques of this research work.

Furthermore we would also like to acknowledge with much appreciation the crucial role of technicians at the workshop department for their welcome and sympathy with us in helping us completing the mechanical part of the project. A special thank to technecian Ahmed Al-Shaibani for his endless support.

A special thank to be for the head of Mechatronics Enginnering Department represented by Dr. Eng. Abdulmalek Momin at Faculity of Engineering, Sana'a University. As well as, we introduce our thank and appreciation to all those doctors who guided us for the five years of study in Mechatronics Department

All thanks, appreciation and respect to be be given to Ms. Neven for her endless support in all levels.

Finally, we wish to thank our parents and relatives for their support and encouragement throughout my study.

Thanks for all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

LiDAR	Light Detecting And Ranging
ROS	Robotic Operating System
SLAM	Simultaneous Localization and Mapping
AI	Artificial Intelligence
URDF	Unified Robot Description Format
CPU	Central Processing Unit
PC	Personal Computer
USB	Universal Serial Bus
BIOS	Basic Input/Output System
CAD	Computer Aided Design
AIML	Artificial Intelligence Markup Language

LIST OF SYMBOLS

CHAPTER 1

INTRODUCTION

Robots are rapidly evolving generally like any other technology around us. There a variety of interesting types of robots that lead the way in the world of robotics and artificial intelligence. In our project, we focused on on of the most significant types of robots which is mobile robot. There are different types of robots divided is by application. Industrial robots, Domestic or household robots, Medical robots, service robots, military robots, Entertainment robots, space Robots and Hobby and competition robots. Between these types of robots, depending on the locomotion of the robot and the area on which it is functioning, some of them are stationary and some of them are autonomous. The autonomous are wheeled robots, legged robots, swimming robots and flying robots. see the following LiDAR Datasheet or see LiDAR Datasheet.pdf

Our Educational ROS-Based Autonomous Mobile Robot Mobot aims to provide students with the most important principles need to communicate theoretical aspects with the practical sections. In this project, we attempted to use the new robotic programming software, Robotic Operating System ROS. It is one of the of the latest trends used in robotics programming. Recently, it became the leading robotic programming software not only for general purpose robots, but also for many industrial robots like Kuka and ABB, Motoman and Robotica.

Mobot is a ROS-Based autonomous, open source robot platform. It is a compact four-wheeled robots equipped with a lidar, camera, IMU, and some functional sensors like Uterasonic and encoder. The Platform controller used in Mobot is Raspberry Pi3 +B which is a small compact computer running with linux environment operating system. It is shown in the figure below:

Although ROS is not operating system in the vein of Windows 10 or Linux, the software development platform provides a system of nodes that allows interprocesses to occur within the target's intelligent platform. These interprocesses allow the sharing of functional messages to occur within a robotic architecture. The architecture of a ROS system consists of five components: a ROS Master, nodes, publishers, subscribers, and topics.

1.1 Original aims of the project

1.2 The scope of the project

1.3 The structure of the report

chapter 1 introduces the project and the idea as well as the general view of the MOBOT robot. in chapter 2 it introduces the background related to similar autonomous mobile robots and the different technologies utilized in developing these robots. the chapter highlighted Lady-Bot Ros-Based Autonomous Mobile Robot which is belongs to first batch of Mechatronics department, Faculty of Engineering-Sana'a University. Chapter 3 is dedicated to Robotic Operating System ROS the software used to control the mobile robot. The aspects related to artificial intelligence were introduced in this chapter as well. Chapters 4 and 5 are the most important since it contains the design and the implementation of the project from the first up to the end. Chapter 6 was dedicated to discussing the results and evaluation. It is considered as a proof fro the success of the project. Chapter 7 talks about the future work to be curried on the robot since this is an educational open source. chapter 8 summarizes up the document. The rest of the document was dedicated for the Bibliography and the appendices.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

In this section we discuss the background of the project and the previous related studies that utilized ROS as a software controller platform. Through this section, we discuss quickly the theory of the our project, on which it depends and the technologies utilized to develop it. In section 2.1, the scope of our project has been discussed providinf the type of hardware and software controllers used in MOBOT robot. Section 2.2, the problems and constrains of the project have been discussed briefly. More details about this subject are discussed in the next chapters in sections where we faced theses difficulties. The privous study have been discussed in section 2.6.

2.1 Project Scope

The scope of the project will not be limited exclusively to the principles we have studied, instead, it is going to cover some of the technologies and techniques being applied and used these days in the industries and educational and search applications. For example, ROS is going to be used as the main operating system of the Mobile Robot. Robotics and Artificial Intelligence principles are going to be connected together to guarantee the intended performance of the Mobile Robot. The following points summarize the scope of the project

- **ROS-based Software controller**
- **Raspberry PI 3 +b platform**
- **Linux environment operating system**
- **DC Motor actuated Mobile Robot**
- **Robotics Principles such as Navigation, Mapping, Visualization, Localization, Search of target, Modeling and preparing URDF, Simulation of the mobile Robot via Gazipo (ROS-Based simulation Package)**
- **LiDAR sensor for navigation and 3D Mapping and connecting it with ROS as a node. Encoder sensors for the DC motors.**

2.2 The problem we solve by this project

Due to the shortage of the educational means for students in aspects related to practical subjects like Robotics and Artificial Intelligence, it is a good opportunity to exploit our work on this filed in our final project. Many of the principle of the theoretical cannot be understood by students if they are not connected to the practical part in the real world. Through ROS-Based Educational Mobile Robot, an added value to the department is going to be considered as one of the reliable means for educational purposes.

Another point, for those student who are working on projects in Robotics or any other system that has complexity in its input/output communication. It is very very difficult to make an effective coding that make the system operates properly. With ROS regardless of the sensors and the complexity of the system, it becomes fairly easier since it deals with every part in the system separately so called node and its code is built separately, the code of this node and the other nodes are communicating with each other through topics and messages.

2.3 Stakeholders within the problem area (Students)

As mentioned above in section 2.2, this educational open source project is going to be dedicated for students in Mechatronics Department in aspects related to robotics and Artificial Intelligence AI. Moreover, it is suitable to be generalized for other practical parts related the fields of study during the five years. Furthermore, the robot can be dedicated not only to Sana'a University, but also for some other universities have Mechatronics Engineering as a department.

2.4 MOBOT Theory Tools and Methods

for any robot there are a lot of components that incorporate together to do an intended task. The most important components are sensors and actuators. The more sensors and actuators, the more complex the controller is. This because there is a need for a complex communication scenario between these components. Furthermore, communication speed time is significant to enhance the response of the robot. due to the difficulty in programming robots in conventional programming softwares, ROS is very convenient for controlling any robot regardless it com-

plexity or number of its components. ROS is connecting nodes together, these nodes are pieces of software that can be written in C++ or Python programming languages. These nodes are dedicated for reading a sensor or controlling a motor and ultimately are connected together via a publish-subscribe protocol. Nodes communicate with each other through topics and service messages

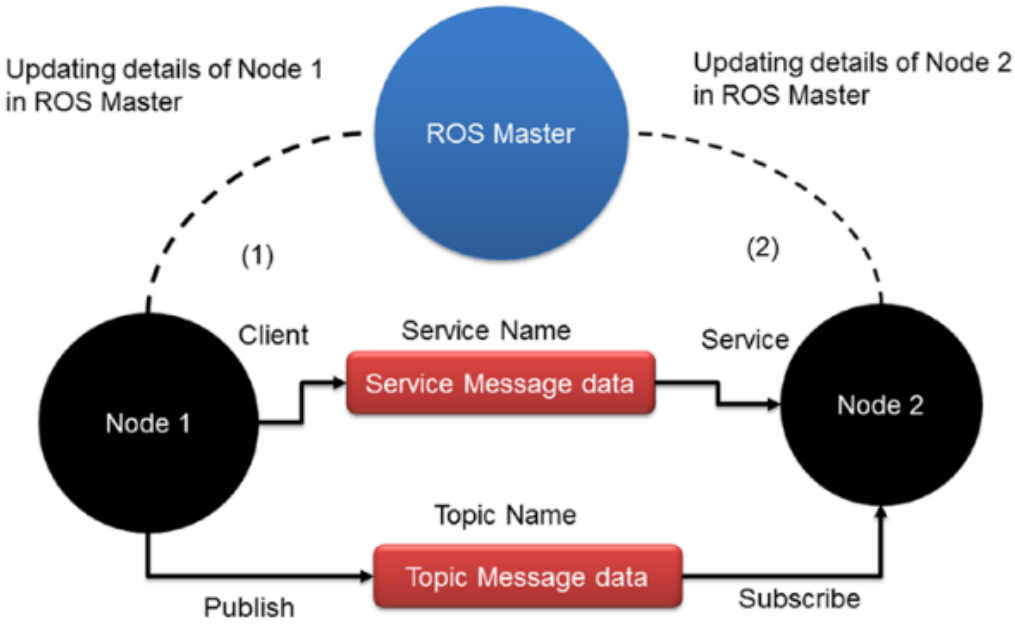


Figure 2.1: ROS Communication Block Diagram [3] [5]

figure 2.1 shows simply how the communication accrues. ROS manages the communication between the nodes.

MOBOT is an autonomous mobile robot based on ROS Robotic operating system as a software controller. The hardware controller is Raspberry Pi 3 Model B+. It uses a LiDAR laser scanner sensor for navigation. there are other sensors help the robot to perceive the environment around to and interacts intelligently. AI technology been added to th robot in the aspect related to speech recognition. Through this technique, a robot can interact with humans and resemble domestic service robots. For example, one can tell the robot to move froward or move backward, the robot understand the command and takes the action. This subject has been discussed in chapter 5.

The robot relies on the exploration of the place through the laser sensor where it moves from one point to another avoiding obstacles in front of it or colliding with walls. It creates a map of the place at the same time of movement by the laser sensor or that the map of the place is

reserved in advance for the robot. One of the tasks it can do is to search for the goal and go to goal commands. One of the things that has been done in the project is the so-called visualization and simulation, whereby the robot can be monitored from a separate computer through the ROS Networking. A system is installed on a computer to perform simulation tasks

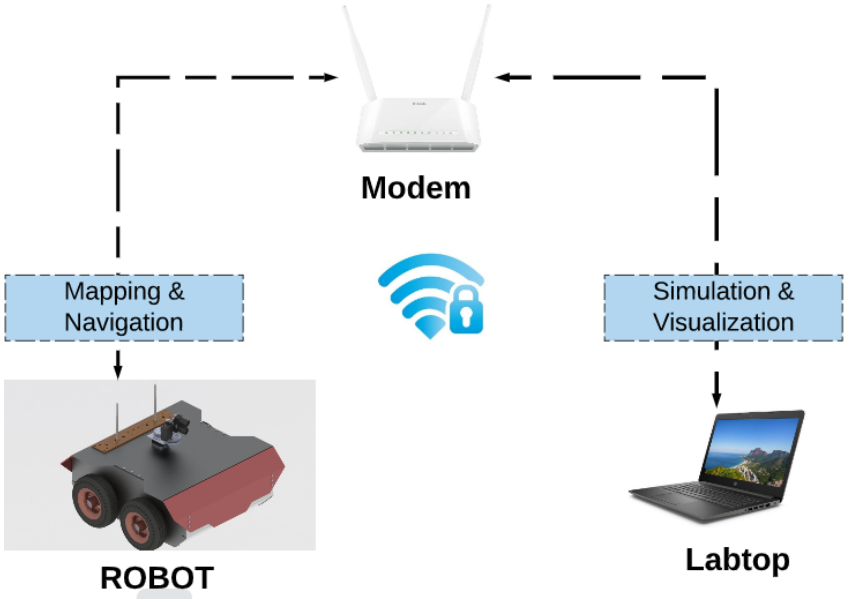


Figure 2.2: ROS Networking

2.5 The constrains in the approach of the problem area

As the mainstream projects we have dealt with so far for through the five years, we have faced many difficulties and constraints in many parts in the design and implementation stages. Of course, the robot is designed to perform tasks that are not addressed to the current robot due to the scarcity of possibilities and lack of pieces in the local market, in addition to the difficulty of importing some pieces due to some government policies. Therefore, we have been working on the available tools and the result of working on the project is a mobile robot based on the operating system of robots. The problems mentioned above were among the most significant obstacles that we had nothing in our hands to find solutions to them and continue to work as planned. However, there are many and many of these problems related to the implementation of what we have. These problems include what is related to the software part and what is related to the hardware part.

All problems encountered in the part of the software part are all in the system of the ROS

starting from the stage of the installation of the system and through the problems that appear during the implementation of some codes as a result of the lack of some packages or as a result of incompatibility with the version we are working on. In addition to the errors that occur in some Packages while executing some code or downloaded from the Internet. Some of these problems erode the time as the fire erodes firewood, such that to the extent that some problems take to solve from two to three days and as a result delay the completion of tasks in a discursive way. As for the problems related to the hardware part, they are no less than those related to the software part where we faced a lot of problems related to work on the mechanical part of the robot in the workshop section, including the installation of motors on the base of the robot and linking them with tires there are also problems we also encountered in the section on electrical work In Android wires and sensors mounting.

The problems stated so far are time-oriented costly problems, however, there are some other money-oriented costly problems related to the components that has gone out of service like electronic components such as controllers and drivers.

Altogether, the problems we have faced are associated with each field we have worked on in this document. For the problems we faced in the design and the problems we have faced in the implementation of the project.

In MOBOT there are modules for perception which handle LiDAR data and finding necessary objects from the scene, speech recognition/synthesis, artificial intelligence modules, robot controller modules for controlling the actuators, $decision - making$ node which combine all data from sensors and makes the final decision on what to do next

2.6 Previous Study

In this section, we are going to discuss the similar projects to us and how ours is different from them. There are many mobile robots that are ROS-based software controlled. Different methods and technologies are used. ROSBot is one of the mobile robots that uses ROS as a controller. It is an open source robot which has its own dedicated controller platform.

CHAPTER 3

ROBOTIC OPERATING SYSTEM ROS

Robotic Operating System ROS is a meta-operating system not a standalone operating system like Windows or Linux. Meta here means that it is an environment that provides services expected from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management.
ROS is a standard software framework that is widely used in robotics programming whose main goal is to make the multiple components of a robotics system easy to develop and share so they can work on other robots with minimal changes [5].

3.1 History of ROS

In 2007, Willow Garage, a nearby visionary robotics incubator, provided significant resources to extend these concepts much further and create well-tested implementations. The effort was boosted by countless researchers who contributed their time and expertise to both the core ROS ideas and to its fundamental software packages. Throughout, the software was developed in the open using the permissive BSD open-source license, and gradually has become a widely-used platform in the robotics research community. https://www.ros.org/history/
The ROS ecosystem now consists of tens of thousands of users worldwide, working in domains ranging from tabletop hobby projects to large industrial automation systems.

3.2 Why ROS was Used in MOBOT

Although there are many reasons that make people prefer not to use ROS for their robotic projects due to its difficulty in learning, difficulties in starting with simulation, difficulties in robot modeling [1]. We preferred to use it due to the capabilities it has such as operating

$system\breve{}like$ features, high-level programming language support and tools such as C++ or Python, availability of third-party libraries such as OpenCV integrated for robotic vision and Point Cloud Library **PCL! (PCL!)** integrated for 3D robot perception, off-the-shelf algorithms like Simultaneous Localization and Mapping (SLAM), extensive tools and simulators such as Rviz used for visualization with cameras, laser scanners and Gazipo used for robot simulation [3]. There are other reasons such that it supports high-end sensors and actuators packages and drivers such as LiDAR sensor, and its modularity [1].

3.3 Hardware ans Software Requirements for ROS

3.3.1 Software Requirements

There are some requirements for installing ROS

3.3.1.1 Installing Ubuntu on Disk Flash

As mentioned in the previous section that Ubuntu is the most suitable operating system for ROS to be installed on. So Ubuntu is installed on PC as we know, and the ROS is installed inside it. There is another choice for using Ubuntu operating system in the personal computer without installation. This can be done by burning the Ubuntu image into a flash disk. After the image has been mounted into the Flash Disk as persistence and booting from it, all the features of the Personal Computer (PC) are exploited by the flash disk including the memory and the Central Processing Unit (CPU). In this section, we show how to mount Ubuntu operating system on USB flash as a persistence user.

The following steps summarize the installation:

- **Download Ubuntu Desktop you need to mount it into USB from here.**
  If you already have it in your PC skip this step.
  The recommended system requirements are shown below [1]
  - 2 GHz dual core processor or better
  - GB system memory

[1] https://ubuntu.com/download/desktop

- 25 GB of free hard drive space
- Either a DVD drive or a USB port for the installer media
- Internet access is helpful

■ **Install Rufus Software** This software is used to create a bootable USB drive. It can be downloaded from here

- Windows Operating System
  If you are a windows user download it normally and use it directly.
- Linux Operating System
  If you are a windows user use the terminal to download it.

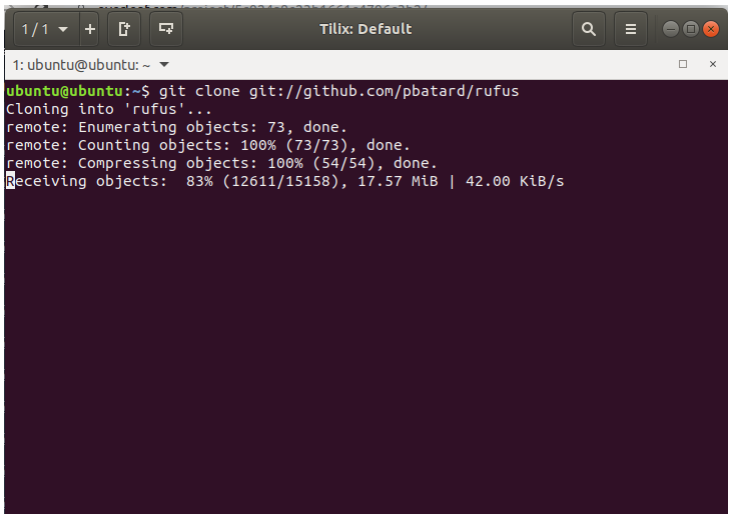


Figure 3.1: Rufus Install

■ **Open Rufus and plug in USB flash** Make the following configuration:

- Make sure that you have selected the right USB in the *Device field*.
- in the *Boot selection* field click on select to brows the **ISO!** (**ISO!**) image for Ubuntu you have in your computer.
- In the format options, you can change the *Volume label*.
- Keep the default configuration for the rest of the fields.
- Click on *Start* and wait a few minutes

■ **Restart the PC and boot from the BIOS. For HP Laptop, press on f9 when starts up**

- Use the arrow keys to move up and down and choose the USB flash you want to boot from and press *Enter*. Figure 3.3 shows this.

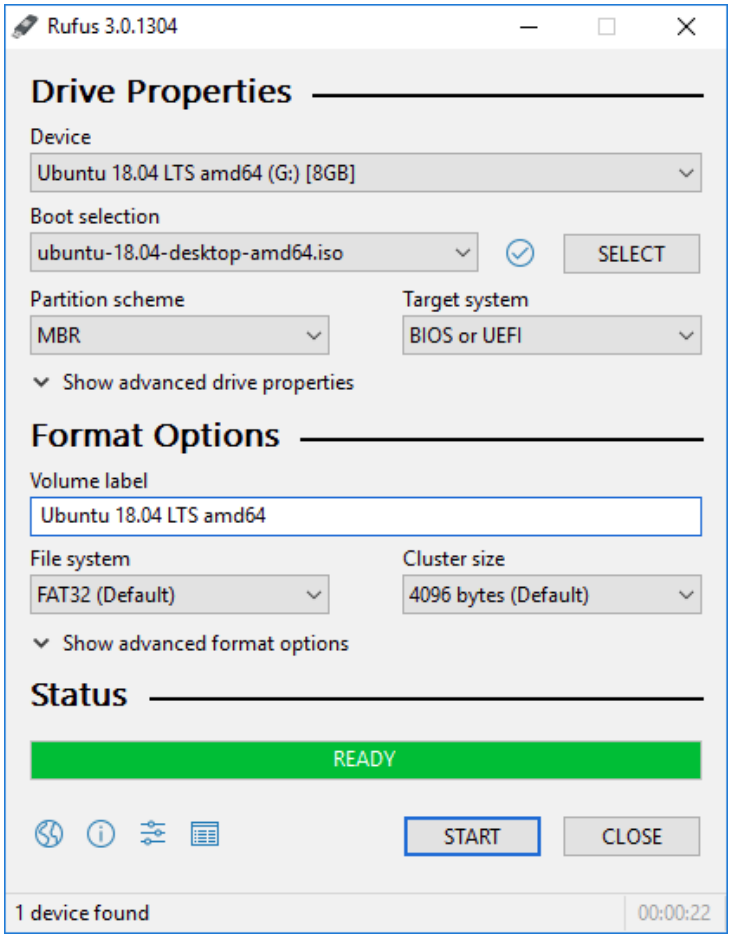
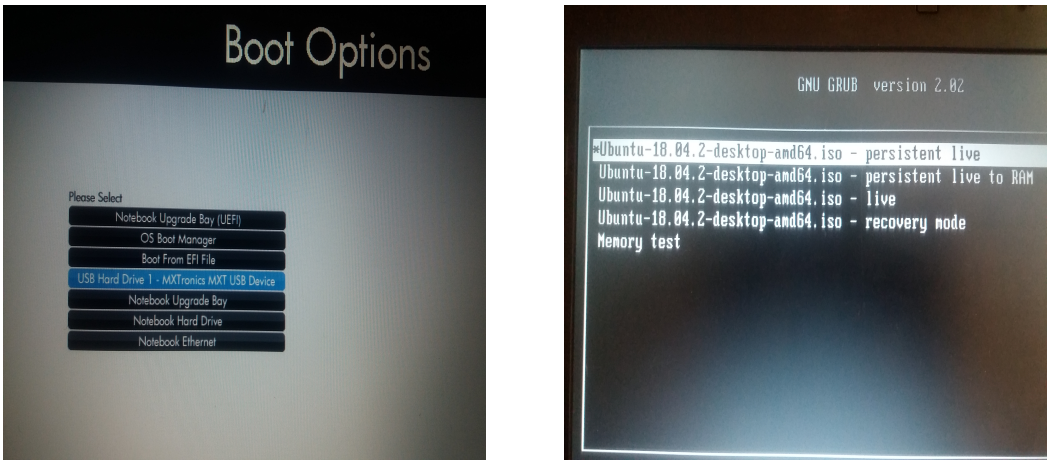


Figure 3.2: Rufus configuration for mounting image into USB flash



(a) (b)

Figure 3.3: Booting from the BIOS from USB

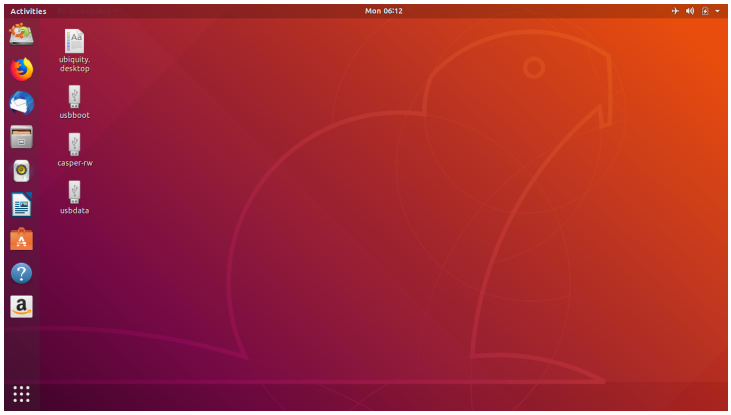


Figure 3.4: Ubuntu Desktop booted from the USB falsh

– Wait until the system boots up from the flash. The first thing you would see is the desktop of Ubuntu operating system shown in figure 3.4.

  Now this Ubuntu environment is not persistence, in other words, any changes in the system would not be saved after shutdown. So, we are going to make a persistence one using this Ubuntu on another USB flash. Up on finish the preparation of the system inside the flash, everything changed in the system such as creating folders and downloading files from the internet would be saved. It become just like the hard disk of the PC.

■ Open gparted built-in software in Ubuntu

– Select the flash disk you you want to install Ubuntu into.

– Click on the bar of the disk and writ click on it and choose *Unmount*

– Format the disk as ext4 as shown in the figure 3.5

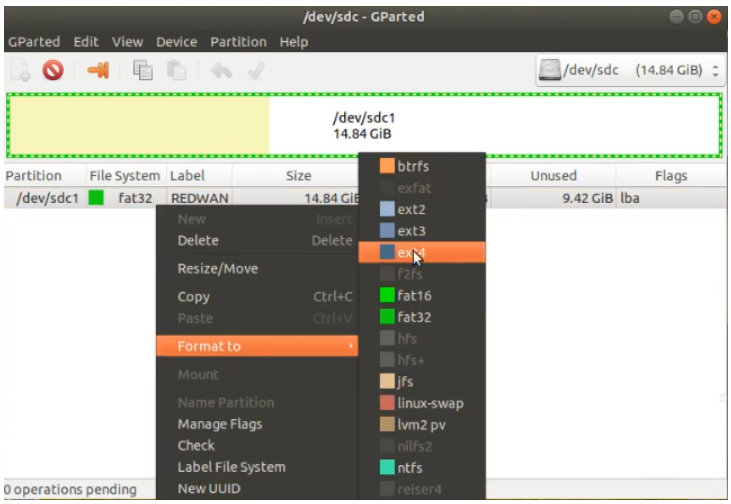


Figure 3.5: Preparing the USB falsh for Ubuntu installation

– Make sure that the flash disk does not have more than one partition, if so, these partitions must be deleted first using one of the icons in the tools bar.

■ **Open *Software Update application and chick the boxes as shown in figure 3.6***

Figure 3.6: Update the Repositories of the system

■ **Open Command Terminal and write down the following command**
$ sudo apt-add-repository ppa:mkusb/ppa. Click *Enter* two times
you should see as shown in figure 3.7.

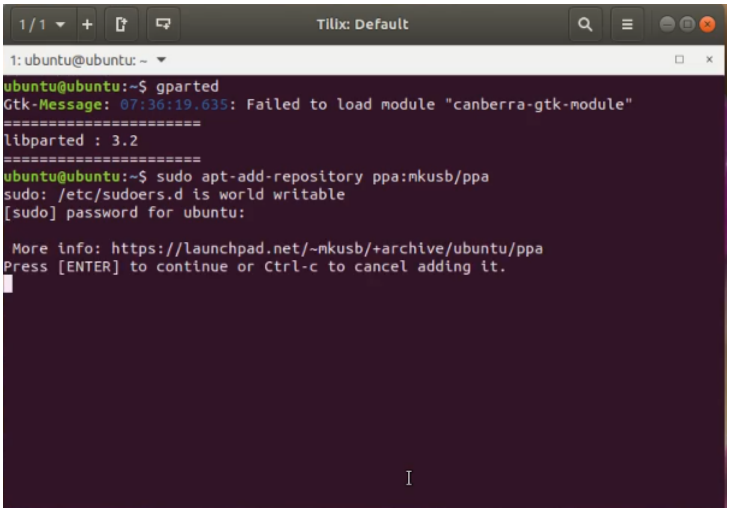


Figure 3.7: Prepare mkusb repository

3.3.1.2 Installing Ubuntu on Raspberry Pi3

3.4 Installing ROS on Raspberry Pi and Disk Flash

ROS has many distributions which they are versioned sets of its Packages. The purpose of the ROS distributions is to let developers work against a relatively stable code-base until they

are ready to roll everything forward. Depending on the versions of Ubuntu operating system, ROS is installed. Figure 3.8 shows some of these distributions

Distro	Release date	Poster	*Tuturtle*, turtle in tutorial	EOL date
ROS Melodic Morenia (**Recommended**)	May 23rd, 2018			May, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame	May 23rd, 2016			April, 2021 (Xenial EOL)
ROS Jade Turtle	May 23rd, 2015	Jade Turtle		May, 2017
ROS Indigo Igloo	July 22nd, 2014	I-turtle		April, 2019 (Trusty EOL)
ROS Hydro Medusa	September 4th, 2013	H-turtle		May, 2015

Figure 3.8: ROS Distributions [2]

The ROSversion used in MOBOT is Melodic Morenia dedicated for Ubuntu 18.04 Bionic **LTS!** (**LTS!**) for desktop and Mete for Raspberry pi. So the installation in Raspberry and in the desktop is the same.

3.4.1 Steps to install ROS

There are five steps to install ROS

- **Configure your Ubuntu repositories**
  It is the same as shown in figure 3.7. Repositories are where the Ubuntu softwares are organized, typically on servers in which users can access and install the applications [3]. These repositories are restricted, Universe and multiverse. In this configuration, the main server can be used or there is a best server depends on where you are in the world. This is considered form the most important steps in installing ROS.

- **Setup your sources.list**

3.5 ROS Architecture

The architecture of ROS has been divided into three level concepts [3] [1] [5]:

[3] http://wiki.ros.org/ROS/Concepts

1. **The File system level**
2. **The Computation Graph level, and**
3. **The Community level**

The block diagram in figure 3.9 shows ROS architecture.

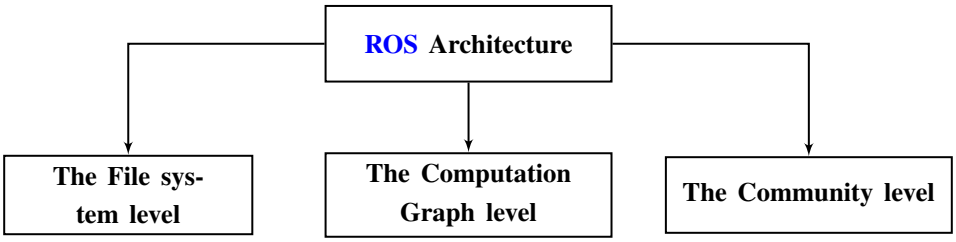


Figure 3.9: ROS architecture block diagram

3.5.1 The File system level

It concerns about the ROS resources on the disk such as the folder structure and the minimum files required to work. They are as follows:

1. **Packages**
2. **Metapackages**
3. **Package Manifests**
4. **Repositories**
5. **Message (msg) types**
6. **Service (srv) types**

CHAPTER 4

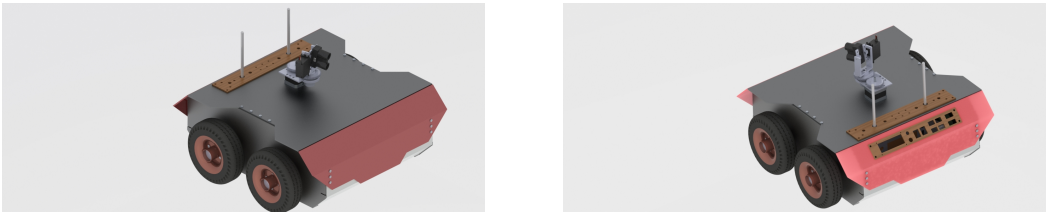
MOBOT DESIGN

In This chapter we discuss the design of MOBOT. The design of the related three sections of the mobile robot proceeded at one pace to achieve the concept disciplinary of Mechatronics Engineering. chapter 5 discusses everything related to the executive part of the project. The work proceeded as the design part was completed. While the mechanical design and implementation part is accomplished, the electrical part and the control part are taken into consideration as well.

The design of MOBOT is divided into three parts, the mechanical design discussed in section 4.1, the electrical design discussed in section 4.2, and the controller design discussed in section 4.3.

4.1 Mechanical Design

The final design of the MOBOT accomplished in Solidworks is shown in figure 4.1, 4.1(a) and 4.1(b)



(a) Front isometric view (b) Back isometric view

Figure 4.1: MOBOT Mechanical Design Isometric Views

The design of the robot is based on RosBot Open source mobile robot. Some of the parts have been changed to be fit to what resources that we have. The size of the robot has been changed to be bigger than RosBot.
The design has taken two weeks to complete, then we have started the implementation in the workshop. The implementation has been discussed in section 5.1

During the design, we consider the other two parts to be fit in order to get everything works properly. We have searched for the pieces at the local market to design in line with. For example, when we start to design the tires of the robot, we have considered the tires in the local market. On the other hand, the type of the motors in the local market have been taken into account and how they would be placed inside the robot and whether they would be able to move the robot or not. So we one motor have been bought to do the tests of its operating characteristics. There is also an important point which is how the connection between the tires and the motor to be accomplished. So before the design, these things have been studied thoroughly at the same time since everyone depends on the other.

4.1.1 External frame design

Figure 4.2 shows one of the parts of the robot designed in Sloidworks software. The rest of the drawings are provided in Appendix **??**.

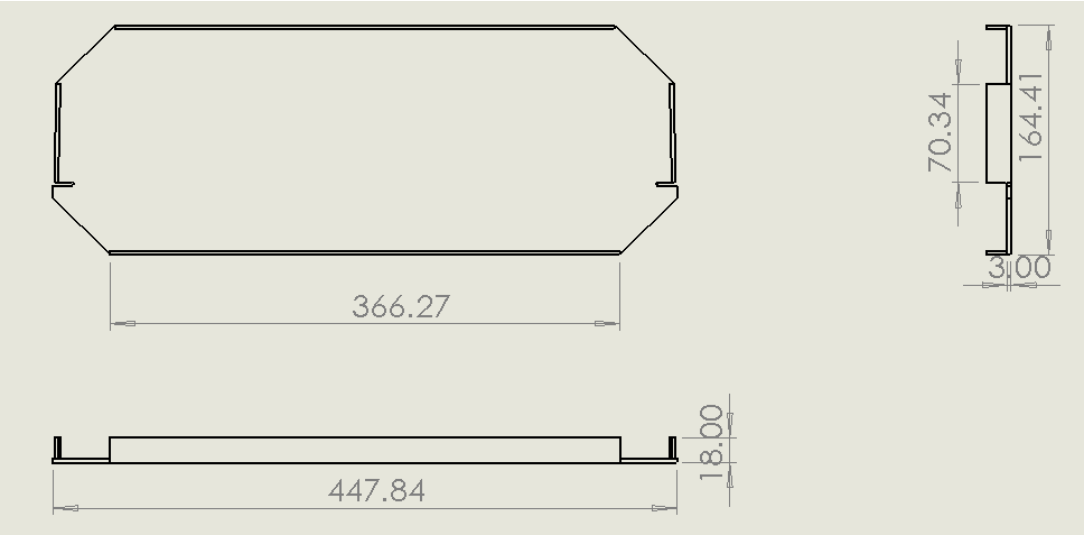


Figure 4.2: Dimentions of the side part

As the components and the tools available for us, some of the drawings of the design have been changed in accordance to the implementation. In other words, there are some cases in which parts cannot be implemented as per the design due to shortage of tools or components, so this enforced us to change the design to be in line with the real world assembly. Some of theses cases are discussed in section 5.1

4.1.2 Design of the LiDAR Laser scanner mechanism

4.1.3 Design of the motors of MOBOT

As mentioned above, the motor has no data-sheet anailable in the internet, so we have taken measurements by ourselves . Figure 4.3 shows the motor dimensions. The real part is shown in figure 4.5

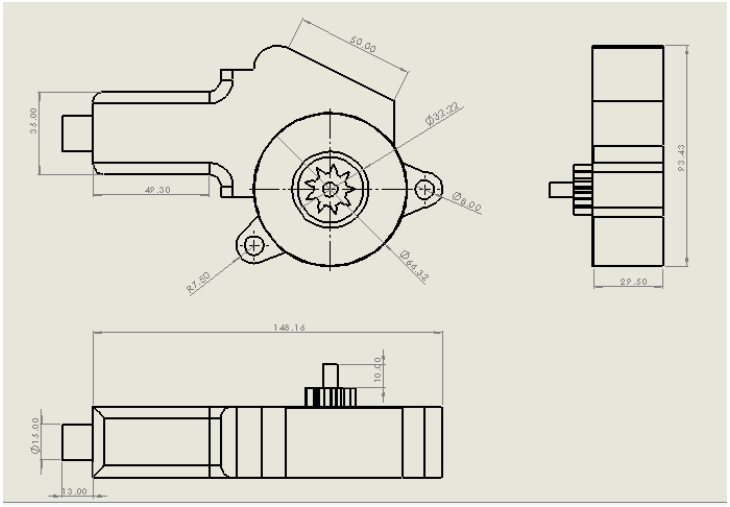


Figure 4.3: DC Motor dimensions

4.1.4 Design of the Coupling of the motors

4.1.5 Design of the Encoder slots

4.1.6 Design of the Bearings

4.1.7 Design of the panel of interfaces

4.1.8

4.1.9 Problems Faced in the Design

4.2 Electrical Design

The electrical design includes everything related to the power system including the control wiring of the sensors. After we had determined the functionality of MOBOT, we analyzed the electrical circuits we would have including the sensors and the actuators. The following points have been considered:

- **The battery that provides the robot with the required power**
- **Actuators that move the robot**
- **Sensors to be used in MOBOT to do its intended functions**
- **Preparing datas-heets of sensors and searching for them in the local market**
- **Electric circuit design**
- **Wiring criteria including the quality of wires and interference minimizing**
- **Safety System Design**

4.2.1 MOBOT Power design

The Battery we have used is 12V, 12AH is shown in figure 4.4 which is suitable for providing MOBOT with the required power for all its components. Tabel 4.1 shows the characteristics of the battery.

Figure 4.4: Battery of MOBOT

char	info
Voltage	12V
Capacity	12AH
Type	Gell Lead Acid

Table 4.1: Battery Characteristics of MOBOT

4.2.2 Actuators of MOBOT

The actuators we have planned to use in MOBOT should have a relatively high torque with 12V input. The most suitable for us was Hyundai Elantra Power Window Motor. This type of motors is available at the local market and has required characteristics, it is cheap as well. The actuator is shown in figure 4.5.

The characteristics of this motor have been taken experimentally since we did not find its data-sheet in the internet, so we have made tests on the current-voltage characteristics. Table 4.2 shows the characteristics of the actuator. [1], [2]

4.2.3 MOBOT Sensors

Perception the environment around MOBOT is one of the necessary aspects in navigation in which the data from sensors is being interpreted to extract meaningful data. The success of

[1] https://www.electricalengineering123.com/calculate-torque-dc-motor
[2] http://lancet.mit.edu/motors/motors3.html

Figure 4.5: MOBOT Actuator

char	info
Max. input voltage	12 V
free load current	3 A
Holding torque current	7 A
Speed	
Torque	
Power	

Table 4.2: MOBOT Experimental Current-Voltage Characteristics

the MOBOT to move from one point to another depends on the accuracy and reliability of the sensors dedicated to acquire environmental data. The sensors we have utilized in MOBOT for mapping, localization, obstacle avoidance and temperature sensing. The are as follows:

1. **Light Detection And Ranging LiDAR (LiDAR Light V3)**
2. **Motor Encoder Lite Blocking Sensor Module (RKI-3142)**
3. **Ulterasonic HC-SR04**
4. **IMU**
5. **DS18B20 temperature sensor**

4.2.3.1 LIDAR Sensor

LiDAR is an active optical remote sensing technology that can measure the distance to, or other properties of, a target by illuminating the target using a swept laser beam. LiDAR works in a similar way to Radar, it sends a pulse of laser and it calculates the time taken by light to hit an object or surface and reflected back to the scanner. LiDAR is shwon in the figure 4.6

Figure 4.6: LiDAR Lite V3 Sensor

Light travels very fast - about 300,000 kilometres per second, 186,000 miles per second or 0.3 metres per nanosecond so turning a light on appears to be instantaneous. The actual calculation for measuring how far a returning light has traveled to and from an object or surface is given by:

Table 4.3 shows the specifications of the sensor [4]

For more information about the sensor, the whole LiDAR Data-sheet is found here

Motor Encoder Lite Blocking Sensor Module (RKI-3142)

This sensor is used to measure the speed of MOBOT.

4.2.3.2 Ultrasonic Sensor

datasheet and specification

4.2.3.3 IMU Sensor

datasheet and specification

Specification	Measurement
Range (70 % reflective target)	40m (131 ft)
Resolution	+/- 1cm (0.4 in)
Update rate (70 % reflective target)	270 Hz typical
	650 Hz fast Mode
	> 1000 Hz short range only
Repetition rate	50 Hz Default
	500 Hz max
Power	5 Vdc Nominal
	4.5 Vdc min, 5.5 Vdc max
Current consumption	105 mA idle
	135 mA continuous operation
Operating temperature	-20 to 60 °C(-4 to 140)
User interface	I2C
	PWM
	External Triger
I2C interface	Fast-mode (400 Kb/s)
	Default 7-bit address 0x62
Laser Wavelength	905 nm (nominal)
Total laser power (peak)	1.3 W
Pulse width	0.5 µs (50 % duty Cycle)
Pulse train repetition frequency	10-20 KHz nominal
Beam diameter at laser aperture	12 × 2 mm (0.47 × 0.08 in.)

Table 4.3: LiDAR lite V3 specification

4.2.3.4 DS18B20 temperature sensor

datasheet and specification

4.2.4 Electrical Circuits Design

in this section, we introduce the design of the circuits to be utilized in MOBOT. The implementation of this part is mentioned in section 5.2 in chapter 5.

4.2.4.1 Sensors Circuit diagrams

Sensors are connected to one dedicated arduino to manipulate the their data and to transform processed data to Raspberry Pi 3 to enhance its performance. For more information on this aspect refer to chapter 5. the circuit diagram of the Sensors with arduino is shown in figure **??**.

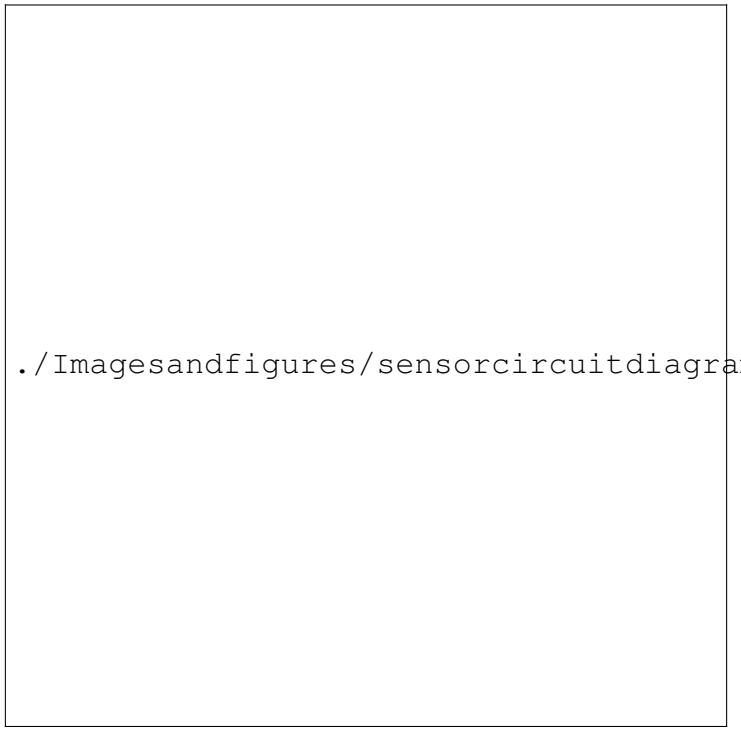


Figure 4.7: Sensors circuit diagram

4.2.4.2 MOBOT Motor Driver Design

In fact this part have been designed such that the drivers of the motors are going to be bought from the local market. After the required tests on the motors to be used in MOBOT had been done, the driver have been designed on this basis. During the test stage in this aspect, everything has gone properly without problems. However, problems have arisen in the implementation when the real test of the robot. The drivers burned out. We thought about another solution for this problem. It is to buy another powerful driver to meet the requirements of the motor. We decided to do this after we have analyzed the potential problem behind such problem. The problem in the driver itself. So ultimately we decided to design our custom driver from components available in the local market. It is more powerful than the those available in the local market.

Figure 4.8 shows the circuit diagram of MOBOT motors driver.

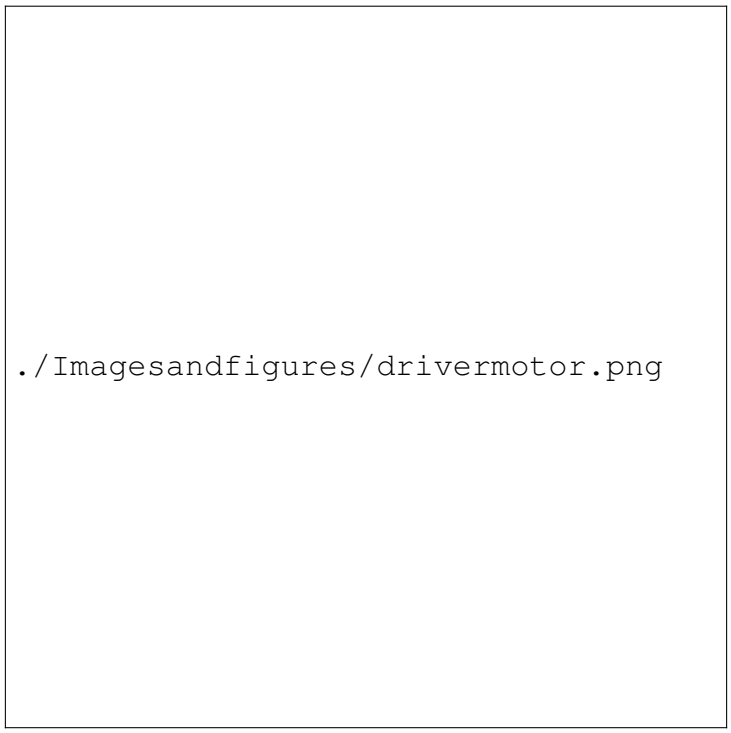


Figure 4.8: Driver motor circuit diagram

4.2.4.3 Safety Circuit Diagram

Due to its importance for any project, safety measures in MOBOT are considered to be one of the most significant aspect to be taken into account, since it protects the system from being

out of work. The safety system designed for MOBOT is to protect the system from overheat or short circuit problems as well as battery discharge.

There will be a temperature sensor to continuously read the temperature inside the robot where the controller are found then display this value on an LCD. If the temperature raised over a set point, an alarm arises with a buzzer producing an alarm.

There are going to be two fans one for Raspberry pi and the second for the driver of the motors to decreases their temperature. In the design as well, to protect the whole circuits from short circuit, a fuse is used to eliminate this problem. The safety system circuit is shown in figure 4.9

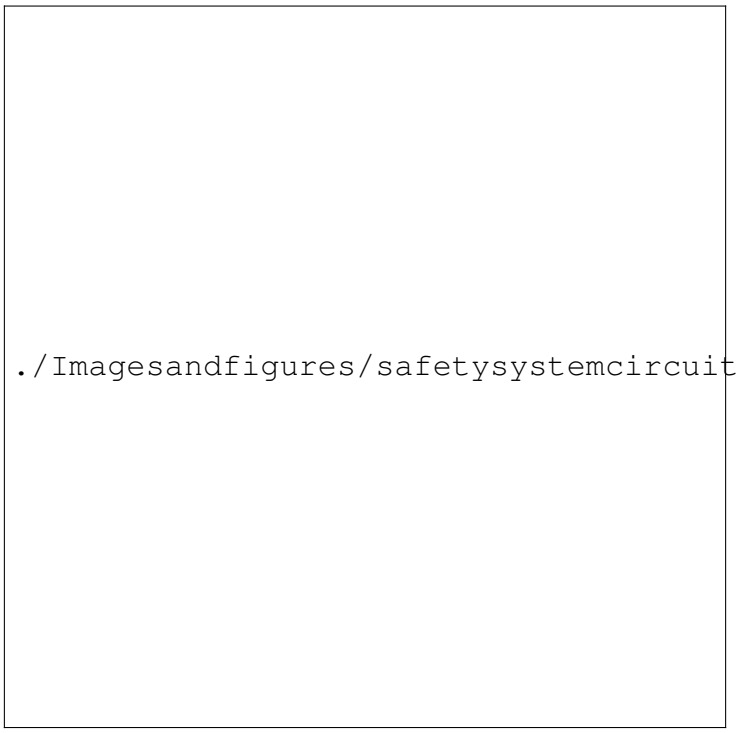


Figure 4.9: Safety System Circuit diagram

4.3 Controller Design

The controller of MOBOT is the most important part in the project, since it is base on ROS, the operating system dedicated for Robots control. From the working principle of MOBOT, the controller has been designed not only from the software point of view, but also from the hardware point of view. So the above components mentioned in the above sections are included in this design.

4.3.1 Controller Block Diagram

Controller block diagram of MOBOT shown in figure 4.10 represented the road map for us to progress through achieving the goals of the project.

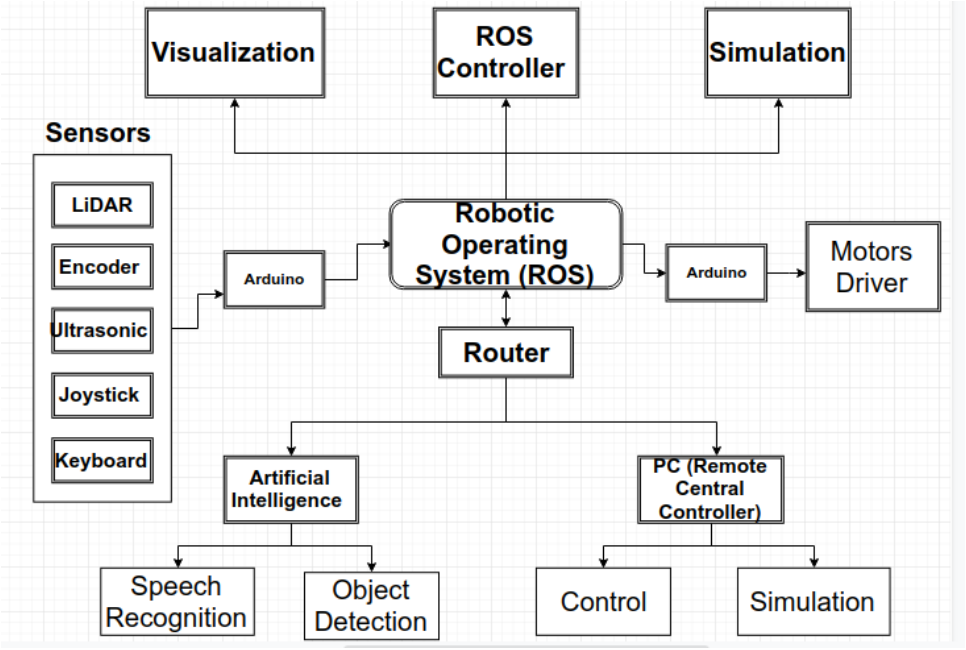


Figure 4.10: Controller block diagram

ROS is the master controller of the whole operation of our mobile robot. It takes the control over all the processes accomplished by the robot. ROS would send and receive data to and from both software packages and hardware devices. The software packages are the nodes that contains the codes specialized in reading information from hardware devices. Hardware devices include sensors and motors. ROS is to be installed in Raspberry Pi3 +B the controller platform we have. See section 3.4 for how to install ROS in Raspberry Pi. Two arduino controllers are going to be used in MOBOT, one would be for the sensors, and the other for the motor drivers.

The block diagram shown in figure 4.10 is divided into three levels, the upper level, the central level and the lower level. The uuper level includes the three processes *Visualization*, *Controller* and *Simulation*. The central Level include the operating system ROS and the *I/O end devices- Sensors and Actuators*. The lower level of the block diagram includes the *Remote Control* and the *Artificial Intelligence*.

As a prelude to what will be dealt with in the following sections of this report we will quickly explain what the format and the idea of the operations in which are in the block diagram.

The diagram shows the various processes that are performed by the robot to move from one point to another, including reading data from sensors, processing and analyzing them, and then the status of the robot is displayed accordingly. The three units at the top of the diagram work at the same pace. For example, the Visualization displays the robot and the details of the place around it completely based on the data coming from the sensors that have been analyzed in advance, and the analyzed data is further converted into a form to be understood by the controller according to which it sends commands to the motors to make the required movement through special drivers. All the movements and the commands are going to be sent from the simulation process such as search for goal or go to point commands. Search for goal, mapping, navigation, etc as well as the the flow control over the different branches are sub-processes being operated by the processes of the upper level and managed by ROS using the tools such as Riviz and Gazipo. All these processes are going to be discussed in details in the next sub-sections of this chapter

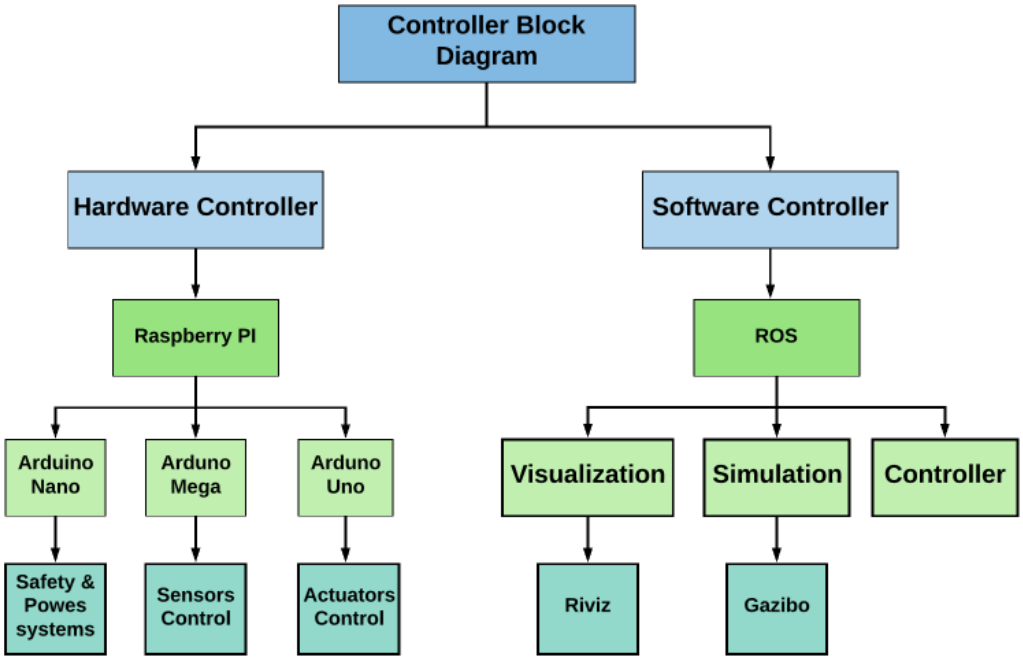


Figure 4.11: Hardware and Software Controller Block Diagram

In visualization, Rviz is used to show what the robot is doing, seeing, and thinking. The tools dedicated for this property is going to help us to see the robot in the a 3D environment similar to that of the real world, more details about this is going to be discussed in chapter 5.

4.3.2 Hardware Controller Platforms

Controller platforms we would use in MOBOT are Raspberry Pi 3 Model B+ and Arduinos. In the next subsections, we will show the exact types to be used in the project. There are two Raspberry Pi platforms, one is dedicated for ROS and the other is dedicated for the AI. With respect to Arduino platform, there are three different types Arduino Mega 2560 dedicated for sensors 4.2.3, Arduino Uno dedicated for the actuators 4.2.2 , and Arduino Nano dedicated for Safety and power systems.

4.3.2.1 Raspberry Pi 3 Model B+

Raspberry Pi 3 Model B+ is the product before the last, the latest product is Raspberry Pi 4. Raspberry Pi 3 Model B+ is shown in figure 4.12. It has a 1.4GHz 64-bit quad-core processor, a dual-band wireless LAN, a Bluetooth 4.2/BLE, faster Ethernet, and Power-over-Ethernet support (with separate PoE HAT) [6]

Figure 4.12: Raspberry Pi 3 Model B+ controller platform

The specifications of the Pi 3 are shown in the table 4.4 from its reference data-sheet

Property	Details
Processor	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz
Memory	1GB LPDDR2 SDRAM
Connectivity	2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE Gigabit Ethernet over USB 2.0 (maximum throughput 300Mbps) 4 × USB 2.0 ports
Access	Extended 40-pin GPIO header
Video & sound	1 × full size HDMI MIPI DSI display port MIPI CSI camera port 4 pole stereo output and composite video port
Multimedia	H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
SD card support	Micro SD format for loading operating system and data storage
Input power	5V/2.5A DC via micro USB connector 5V DC via GPIO header Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment	Operating temperature, 0–50 °C
Production lifetime	The Raspberry Pi 3 Model B+ will remain in production until at least January 2023.

Table 4.4: Raspberry PI 3 +B Specifications

4.3.2.2 Arduino Mega 2560 controller

This controller Arduino Mega 2560 The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila. It is the recommended board for 3D printers and robotics projects.

The whole datasheet of this arduino can be found here.

4.3.2.3 Arduino Uno Platform Controller

Arduino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz

Figure 4.13: Arduino Mega 2560 controller platform

quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worring too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

The datasheet of this Uno Arduino can be found here.

4.3.2.4 Arduno Nano Platform controller

The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328P (Arduino Nano 3.x). It has more or less the same functionality of the Arduino Duemilanove, but in a different package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one.

The datasheet of this arduin can be found here.

Figure 4.14: Arduino Uno controller platform

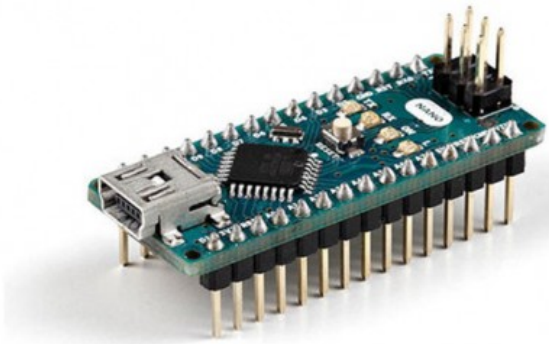


Figure 4.15: Arduino Nano controller platform

4.3.3 Software controller

Section 4.3.2 discusses the hardware controller platforms used in the control of MOBOT. In this section the software controller of the project is going to be discussed. This is one of the most important and essential parts of the project, which contains ROS through which all operations are controlled that occur when the robot moves from one point to another in a particular environment.

4.3.3.1 ROS Design for MOBOT

In this section, the design of the ROS controller is discussed.

The design of ROS for MOBOT includes the following points:

- **Installing and configuring ROS**
  After the preparation of Ubuntu on Raspberry Pi and installing ROS on it, the pre-requisite configurations are done on ROS. These aspects have been discussed in chapter 3.
- **Preparing ROS Architecture**
  As provided in chapter 3, section 3.5, ROS of MOBOT is going to be designed in line with that architecture. In fact there is no way to change the standards of this architecture, but instead, there are some important points to be addressed and taken into consideration in a way that would serve the final goal of our mobile robot. Some of these custom changes are to create the special workspace of MOBOT and to rename some of the files to help understand each file in the file system of ROS.
- **Preparing the 3D Model of MOBOT and URDF**
-

CHAPTER 5

MOBOT IMPLEMENTATION

The implementation of the project represented the most challenging step. Since it is educational oriented project, we have tried to do our best to achieve the goal that was initially set, and therefore benefits the students after us as well. Furthermore, for the development of MOBOT to continue after us, it s implementation is based on a clear and well-known basis in terms of design and documentation.

In this chapter, we are going to discuss the implementation of MOBOT in terms of its three parts, the mechanical, electrical, and control parts. There are many challenges we have been facing added to those of the design phase, but more difficult. In spite of the problems we faced in completing this part, we have tried with all our might to make the work very neat and tidy. There are three main sections in this chapter, section 5.1 talks about the mechanical hardware implementation of MOBOT accomplished in the workshop of the university based on the drawings curreid out in the design phase. Section 5.2 discusses the electrical implementation of MOBOT including power systems and wiring installation of sensors, actuators and controllers. Finally The Controller implementation was discussed in section 5.3. This does not necessarily means that our actual work has gone in this order, instead, the work on the three parts has gone in parallel. Because The work in Mechatronics projects requires integrated and parallel work between the main components of them.

5.1 Mechanical Hardware Implementation

Mechanical implementation has been accomplished at the workshop of the faculty. In this section, everything related to mechanical construction of MOBOT are discussed. In the designing stage, we have determined the material of the mobile robot, the types of actuators and how they would be mounted with the tires. The tasks in completing the robot structure were divided between iron cutting,drilling, milling, forming curves and preparing coupling bearings.

5.1.1 The material of the MOBOT construction

The material we chose for the construction of the robot is Hot Rolled Steel Sheet. It is shown in figure 5.1. The properties of this steel can be found here.

Figure 5.1: Hot Rolled Sheet

5.1.2 Mechanical Processes for MOBOT Structure Preparation

Many of the production processes in the workshop such as lathing, drilling and cutting were worked on in this part. The manual cutting machine was used to cut the steel, the bending machine was used to form the corners in the structure, the drilling machine was used to make the holes necessary to fix the pieces together via bolts, the lathe machine was used to equip the connecting shaft between the motor, bearing and tires, and the press machine was used to press Spindles on bearings. With a lot of difficulties and challenges, and one of the teem has been injured accidentally but it was not that dangerous. In the nect subsection, we introduce the steps of bulding the structure of the robot.

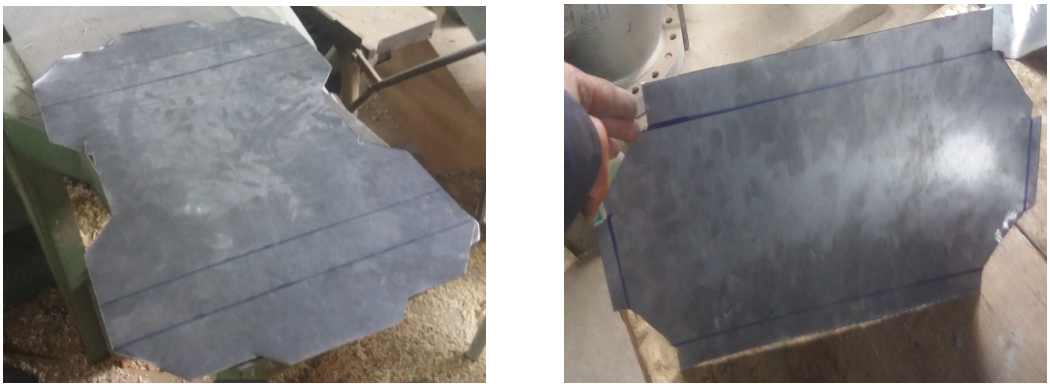
5.1.2.1 Steel Material Preparation

As mentioned above, the material used in the structure of the robot was Hot Steel, the material available in the workshop of the faculty. We have started preparing of the steel by taking the measurements of the drawings related to the design in Solidworks software on the steel as shown in figure 5.2.

Figure 5.2: Taking Measurements on the steel before cutting

5.1.2.2 Cutting the Steel

This step was accomplished using the manual cutting tool. We have cut the pieces according to the drawings we had. Figure 5.3 shows some of these pieces. There are four similar pieces, one for the base, two for the right and left, and the fourth is for the top part.



(a) First piece after cutting (b) Second piece after cutting

Figure 5.3: Some Pieces after cutting

5.1.2.3 drilling the Wholes

In this step, we have used the drilling machine (Spindle) to make the wholes required for joining the pieces together using bolts. There are other wholes were drilled for mounting the bearings and the motors. Before drilling however, we had referred to the drawing to take measurements where to make wholes and at what diameter.

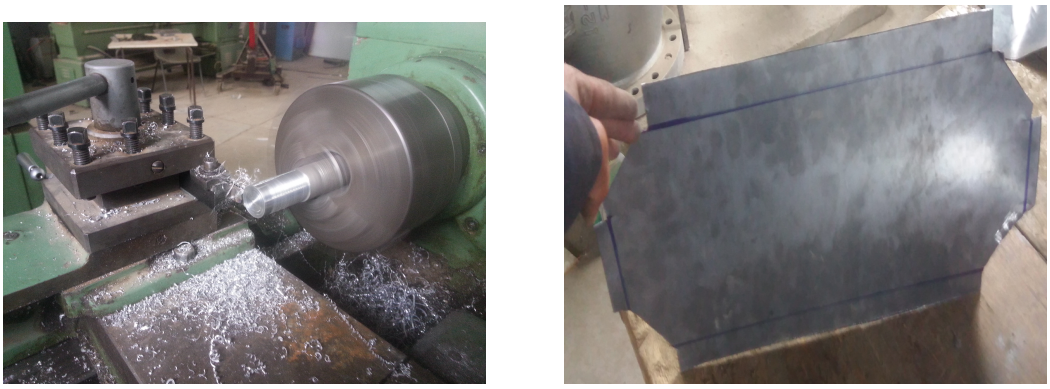
5.1.2.4 Coupling Preparation

This step was one of the most challenges for us to complete because there are number of things to be taken into consideration on the same time. We should consider the diameter of the tire, the diameter of the motor where the coupling shaft is compresses inside, the position of the motor, the height of the tire from the ground and so on. With the guidance of the software design of this step, we have faced these challenges though. In fact, some modifications on the software design were done in accordance to that of the implementation due to shortage of tools and components based on the design have been accomplished.

■ **Preparing Coupling Piece**

After we had taking the measurements of the real world components, the tires, bearings and the motor, we start designing the software part on this basis, and accordingly the production was being accomplished with the modifications required as a result of tolls shortage.

We have used the Lathe machine to prepare the coupling pieces between the three components, motor, bearing and the tire. 5.4 shows part of the work on the Lathe machine and the final fished part.



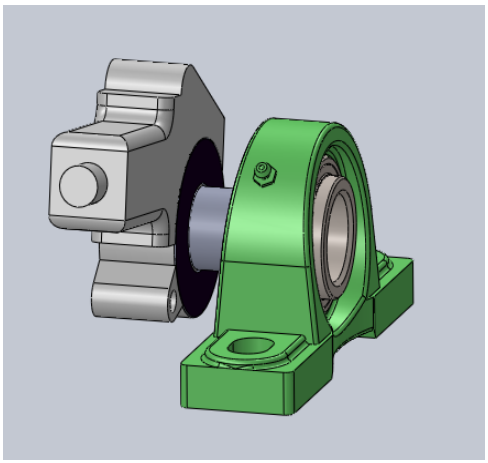
(a) Part of Working on the Lath Machine (b) Finished Coupling Piece

Figure 5.4: Coupling Piece Preparation

■ **Coupling Motor with Bearings**

Figure 5.5 shows the coupling of motors with bearings.

(a) Motor coupling with bearings



(b) Motor couple in the Design

Figure 5.5: coupling of motors with bearings

- **Coupling bearings and motors with tires**

It is Shown in figure 5.6

- **Mounting the side parts**

- **Completing the Final Structure**

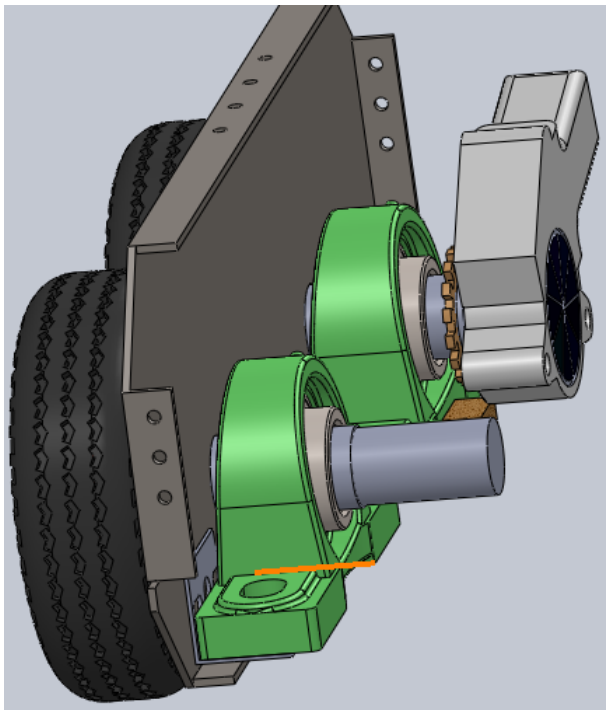
- **Mounting Sensors on MOBOT**

5.2 Electrical Implementation

This section includes everything related to the electrical components used in MOBOT. It shows how the sensors were mounted and where which is discussed in section 5.2.1. Section 5.2.2 discusses the electrical circuits designed for the purpose of motors driving as well as the circuit diagrams of the related circuits. The power system of MOBOT was discussed in section 5.2.3. In section 5.2.4, we discussed the wiring system of MOBOT and the methodology we have followed to amke everything well organized and neat. Section 5.2.5 was dedicated for discussing the safety part of MOBOT.

(a)

(b)



(c)

Figure 5.6: Coupling bearings and motors with tires

5.2.1 Placing sensors on MOBOT

Sensors plays a significant role in MOBOT since they are the devices on which the controller depends when taking an action. The perception process in navigation mainly depends on sensors. Sensors informs the controller where the robot is and what there is around it in the environment exists at. So it is very important to be placed in the right position on the robot where it is suitable to get valuable information form.

As listed in 4.2.3 in chapter 4 there are seven sensors mounted on MOBOT. They are Laser scanner LiDAR, two encoder sensors, IMU sensor, two ultrasonic sensors and finally temperature sensor. They are going to be discussed in the next subsections.

5.2.1.1 Mounting LiDAR on MOBOT

LiDAR Liser scanner sensor, its mounting place is very important for the controller, because there should be a static transformation matrix between it and MOBOT as well as a dynamic transformation matrix with respect to the objects around. So it is placed on the top middle of the robot. this is going to be discussed deeply in section**??**

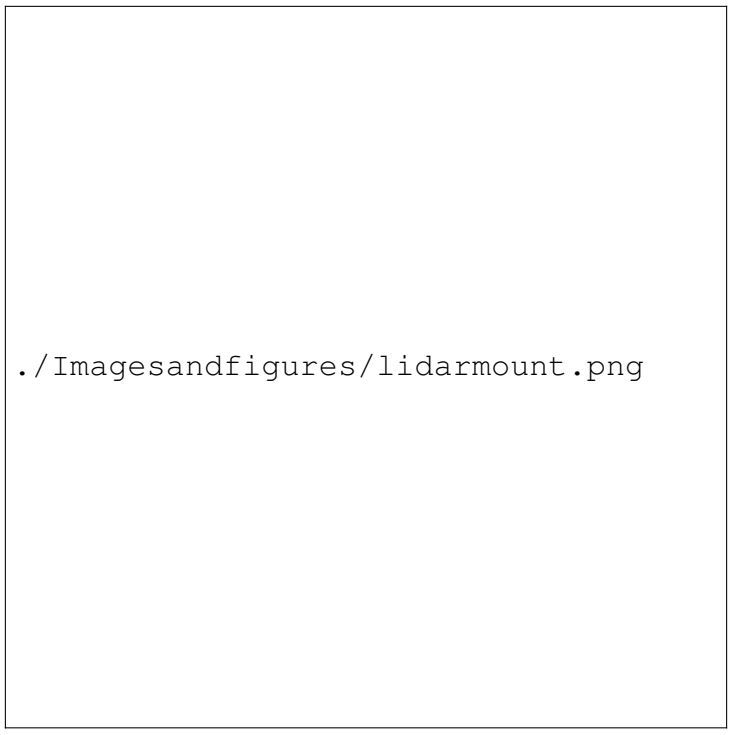


Figure 5.7: Mounting Place of LiDAR

5.2.1.2 Mounting Speed Encoders on MOBOT

There are two encoders dedicated for every motor on MOBOT. They are shown in figure **??**. The function of these encoders is to continuously read the speed of the motors to know at what speed the robot is moving and to use this data in the navigation and the search of goal and so on. This is going to be discussed in details in section **??**. Working on this sensor is interesting. In the design stage of the project, we have taken into consideration the fact its of working principle. Its working principle depends on the light cut when an object passes between its two terminals to produce a result each time this occurs. So we have designed a rotary slot as shown in figure **??** to be mounted on the shaft which couple the motor and the bearing. The Slots Ring where produced by a laser machine.

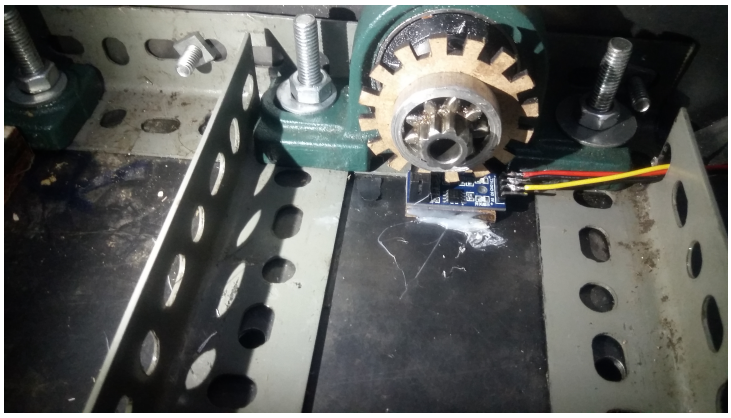


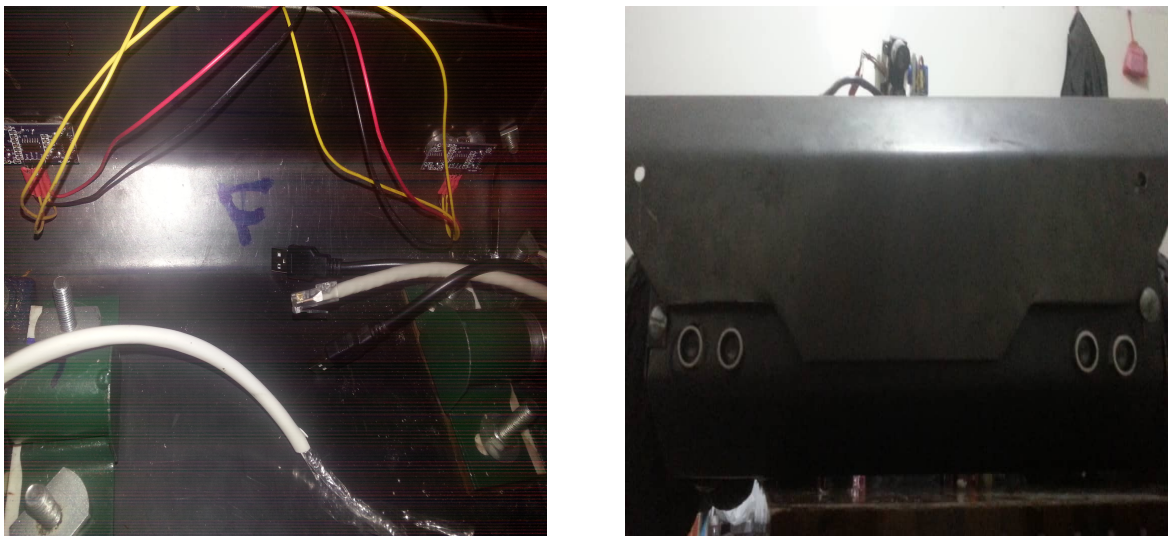
Figure 5.8: Mounting Place of encoders

5.2.1.3 Mounting Place Ultrasonic Sensors

Ultrasonic Sensors were used in the front head of MOBOT. They were placed at each side to make the robot avoid the obstacles in front of it that are located lower than the range of the laser scanner LiDAR. Figure 5.9 shows the place where they were mounted.

5.2.1.4 Mounting Place IMU Sensor

Have not been Used Yet.



(a) (b)

Figure 5.9: Mounting Place of Ultrasonic Sensors

5.2.1.5 Mounting Place Temperature Sensor

This sensor is placed inside the robot itself, it measures the internal temperature where the controllers are placed.

5.2.2 Implemented Electric Circuits

In this section we would provide the electrical circuits implemented for the project. The motor driver and the safety circuit are two examples. The different components of the circuits were to be highlighted with the circuits diagrams included as well.

5.2.2.1 Sensors Circuit

Sensors are not connected directly to Raspberry Pi3, the main controller platform, instead, they are connected through separate arduino. Arduno would take the data from sensors and process this data, then it send it to the raspberry Pi to be ready for use in the codes of ROS. this enhances the performance of the Raspberry pi. Figure 5.12 shows the circuit diagram of the sensors connection to arduino. The arduino dedicated for sensor manipulation is Arduino Mega 2560, see section **??**

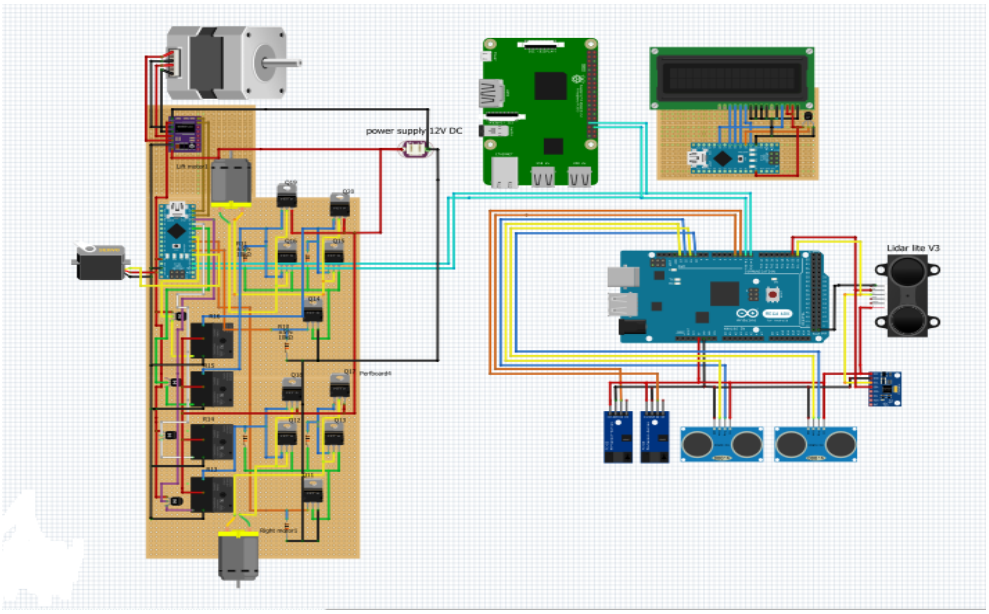


Figure 5.10: Sensors Circuit Diagram with Raspberry

5.2.2.2 Motor Driver circuit

The implemented motor driver circuit is shown in figure 5.11. This was a last resort after we have tried a ready built driver from the local market. Refer to section 4.8 and section **??** for more information about the problems we have faced in this aspect.

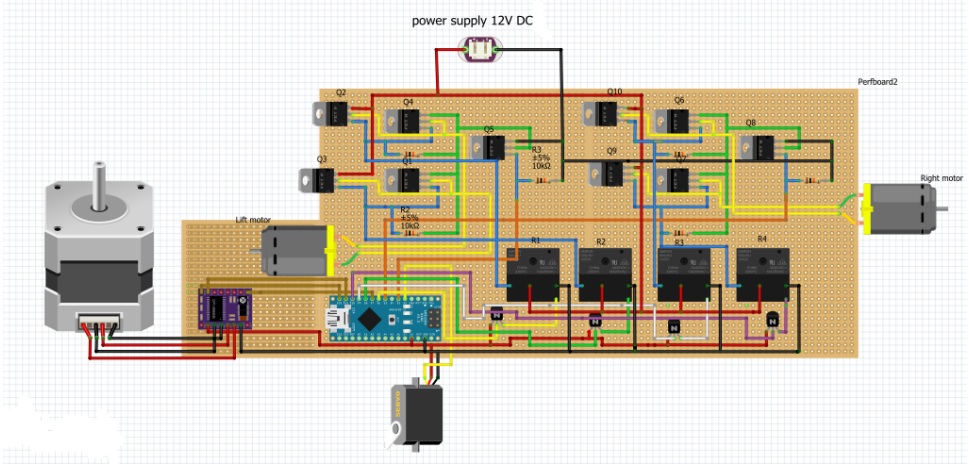


Figure 5.11: Motor Driver Circuit Diagram

5.2.3 Power System in MOBOT

5.2.4 Wiring System in MOBOT

Mobot Wiring was one of the most time-consuming tasks. It took us a lot of time to complete.

5.2.5 Safety System in MOBOT

The safety system has been implemented in MOBOT to protect it from short circuits and over temperature as well as low voltage The robot's protection system is based on protecting the robot from any short circuit in the power that feeds the robot in addition to the warnings associated with overheating within the robot. In addition, the protection system protects the battery from over discharging in order to conserve it. If the battery voltage drops below 12 volts, it triggers a warning that the battery voltage is low and a light on the control panel lights up.

For the overheat inside the robot, an alarm arises when the temperature exceeds a set point 35 °C. This make a led lights up at the control panel indicating over temperature. The sensor used for this purpose is DS18B20 discussed in section 4.2.3.4. The circuit is shown in figure 5.2.5. The circuit diagram is shown in figure 4.9. Refer to section 5.2.5 for more information about this section.

Here is a code fragment for the safety system in the MOBOT.

```
#include <SD.h>

File logfile;
byte logPin = 10;

void setup() {
  SD.begin(logPin);

  ///////// Create a new file //////////
  char filename[] = "LOGGER00.CSV";
  for (int i = 0; i < 100; i++) {
```

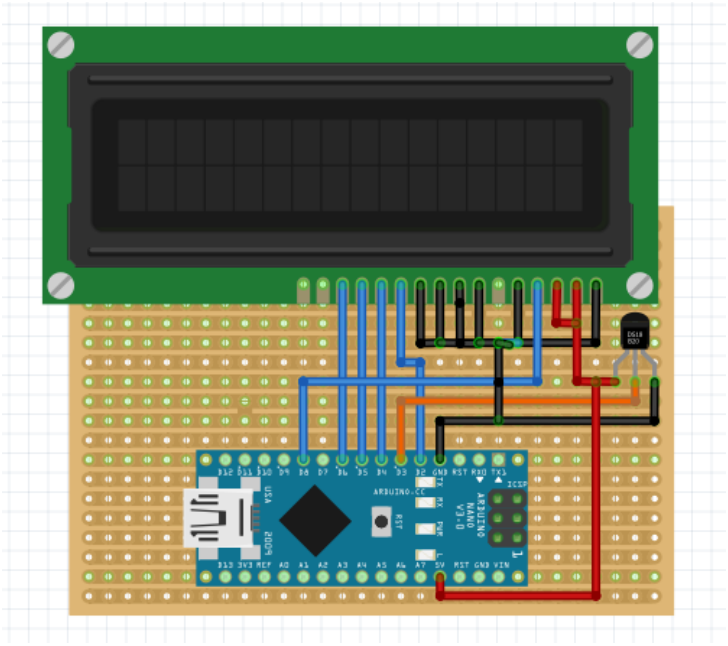


Figure 5.12: Safety System Diagram

```
    filename[6] = i/10 + '0';     // number the file
    filename[7] = i%10 + '0';     //
    if ( SD.exists(filename)==false ) { // only open a new file if it doesn't exist
      logfile = SD.open(filename, FILE_WRITE);
      break;  // leave the loop!
    }
  }
}

void loop() {
  // put your main code here, to run repeatedly:
  logfile.println("Hello");
  logfile.flush();
  delay(1000);
}
```

5.3 Controller Implementation

This part is the most important one in the project, but it depends on all the parts have been discussed so far. Navigation, simulation, visualization, and AI have been discussed here.

5.3.1 Preparing the workspace of MOBOT

Referring to section 3.5 in chapter 3, ROS file system is shown in figure 5.13.

```
$ sudo apt-get update
```

./Imagesandfigures/mobotfilesystem.jpg

Figure 5.13: MOBOT ROS file system

5.3.2 3D Model of MOBOT using URDF

Our MOBOT has been designed as a Computer Aided Design (CAD) model using Solidworks from scratch. The simulation of the robot in ROS requres the 3D model of the robot to be in a specific Format called URDF.. Some ROS packages, like *robot_localization*, are used to built a map and localize on it. In this section, we are going to explore the basics of MOBOT modeling using the. At the end of this video, we will have a model ready and running in Gazebo simulator. So Work on this part paves the work on other parts will come later such as simulation discussed in section 5.3.5 by inserting the model into Gazibo World and visualization in of MOBOT discussed in section 5.3.4

5.3.2.1 MOBOT 3D Description

The robot 3D description is in XML file which is called the URDF. this description contains of links and joints. The code part below shows part of the XML file of URDF.

```
<!-- Base Link -->
<link name="link1">
  <collision>
    <origin xyz="0 0 ${height1/2}" rpy="0 0 0"/>
    <geometry>
      <box size="${width} ${width} ${height1}"/>
    </geometry>
  </collision>
```

Listing 5.1: URDF example

URDF can only specify the kinematic and dynamic properties of a single robot in isolation. URDF can not specify the pose of the robot itself within a world. It is also not a universal description format since it cannot specify joint loops (parallel linkages), and it lacks friction and other properties [1].

5.3.2.2 URDF Package in ROS

fURDF is a package contains a C++ parser for the Unified Robot Description Format (URDF), which is an XML format for representing a robot model This package contains a number of XML specifications for robot models, sensors, scenes, etc. Each XML specification has a corresponding parser in one or more languages such as Python and C++

5.3.3 Navigation stack

Navigation can be defined as the combination of the three fundamental competencies:

- ☐ **Self-localization**
- ☐ **Path planning**
- ☐ **Map-building and/or map interpretation**

[1] http://gazebosim.org/tutorials/?tut=ros$_u rdf$

Robot navigation means the robot's ability to determine its own position by using its odometry and then to plan a path towards some goal location.

5.3.3.1 LiDAR Scanner

The sensor has been discussed in section 4.6 where we provided its specifications in table 4.3 and the working principle. LiDARs most frequently use the measurement time flight for of a laser pulse to be reflected off from the target to determine its target. In section **??** SLAM is discussed and LiDAR is considered to be the most important modalities on which SLAM depends. It is palned to make the LiDAR do 3D scanner by a rotary and up and down movement of the sensor. The sensor is shown in figure **??**, it was shown again in figure 5.14 for the sake of clarity.

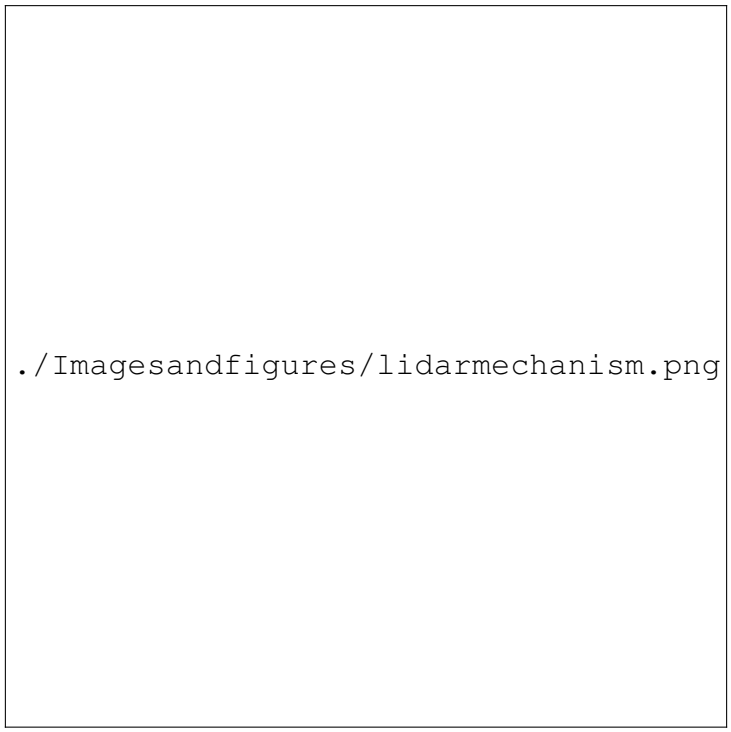


Figure 5.14: LiDAR Mechanism for 3D scanning!

5.3.4 Visualization

5.3.5 Simulation

5.3.6 ROS Networking

5.3.7 Artificial Intelligence in MOBOT

Speech recognition is a feature added to MOBOT as the part of the AI side. The idea behind it is that one can talk with the robot and to do a certain action. Some of the commands are move forward, move backward

5.3.7.1 Speech Recognition

In this section, we will implement the speech recognition with artificial intelligence which can communicate with people using text and speech. The reply from the MOBOT should be like a human's. We are going to implement a simple AI using Artificial Intelligence Markup Language (AIML) which can be integrated to a social robot. AIML is an **XML!** (**XML!**)-based language to store segments of knowledge inside XML tags. AIML files help us store knowledge in a structured way so that we can easily access it whenever required. [2]

The methodology to be used in controlling the robot is via speech commands such as move forward, move backward, turn right, turn left, stop and so on. The speech recognition engine *PocketSphinx* was chosen for this task because it works with little CPU and memory. The idea is that the user sends a speech command via the mic, inside the robot there is a ROS package called *pocketsphinx*. This package is responsible of converting the speech into text. See figure 5.15.

The speech of the user is converted into text using the speech recognition system in ROS. Then, it will input either to the AIML engine or send as a robot command. The robot commands are specific commands meant for robot control such those mentioned above for movements. If the text is not a robot command, it will send it to the AIML engine, which will give an intelligent reply from its database. The output of the AIML interpreter will be converted to speech using the text-to-speech module. The speech will be heard through speaker.

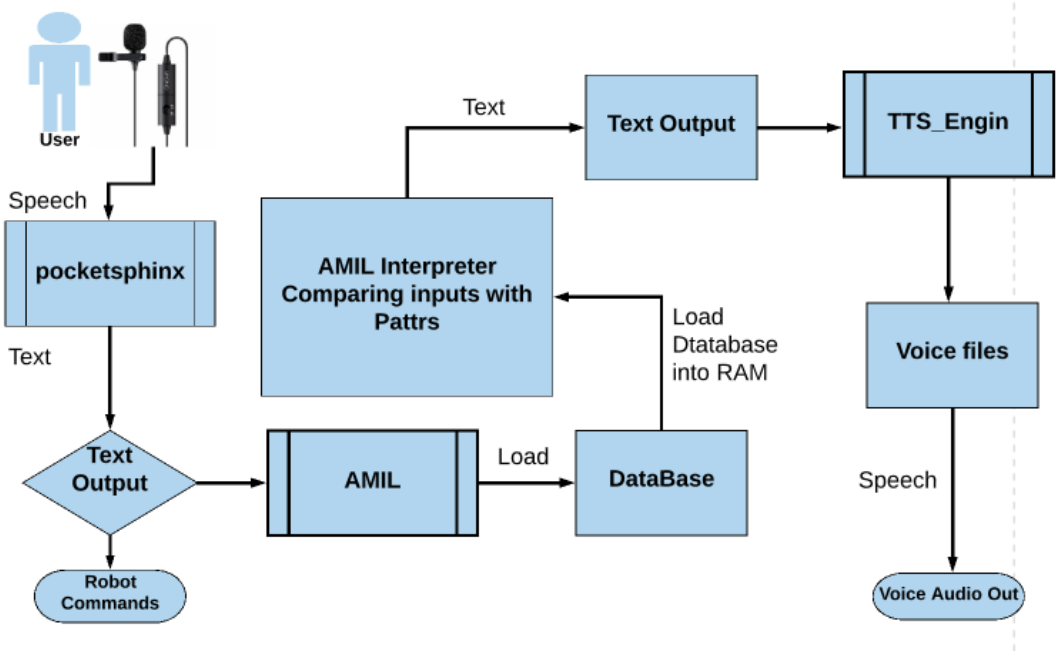


Figure 5.15: Speech recognition block diagram

Before dealing with the **AMIL!** (**AMIL!**) we have to prepare the ROS packages required.

■ **Installing the ROS *sound_play* package**

sound_play provides a ROS node that translates commands on a ROS topic (robotsound) into sounds.

☐ **Installing Dependencies**

First of all we have to update Ubuntu repositories using the command

```
$ sudo apt-get update
```

```
ubuntu@ubuntu:~$ sudo apt-get update
```

Figure 5.16: Update Ubuntu Repositories

Now we run the following command line to install the repositories. See figure 5.17

```
$ sudo apt-get install libgstreamer1.0-dev libgstreamer-plugins-
base1.0-dev gstreamer1.0 gstreamer1.0-plugins-base gstreamer1.0-
plugins- good gstreamer1.0-plugins-ugly python-gi festival
```

Install pip with cURL and Python.

```
$ curl "https://bootstrap.pypa.io/get-pip.py" -o "get-pip.py"
```

```
$ python get-pip.py
```

Install *pyaudio*

```
$ sudo pip install pyaudio
$ sudo apt-get install libasound-dev
$ sudo apt-get install python-pyaudio
```

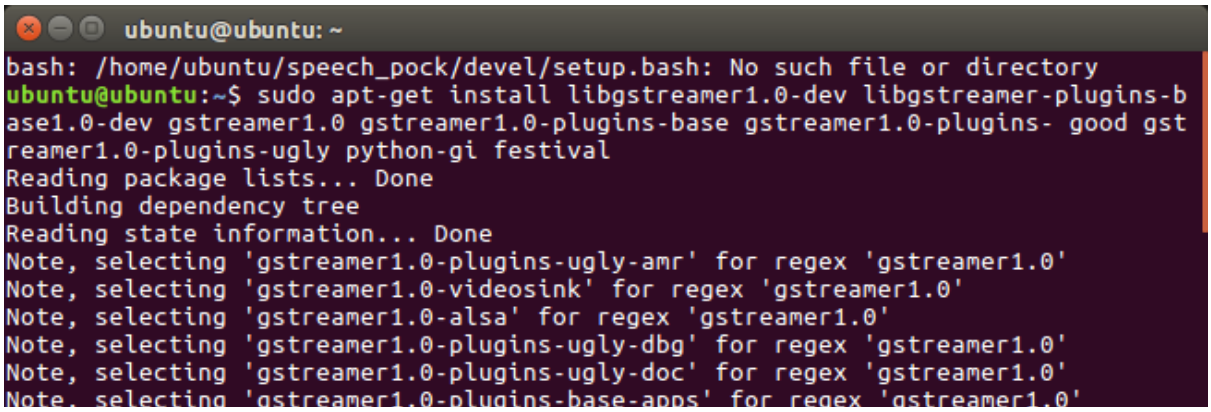


Figure 5.17: Install Repositories

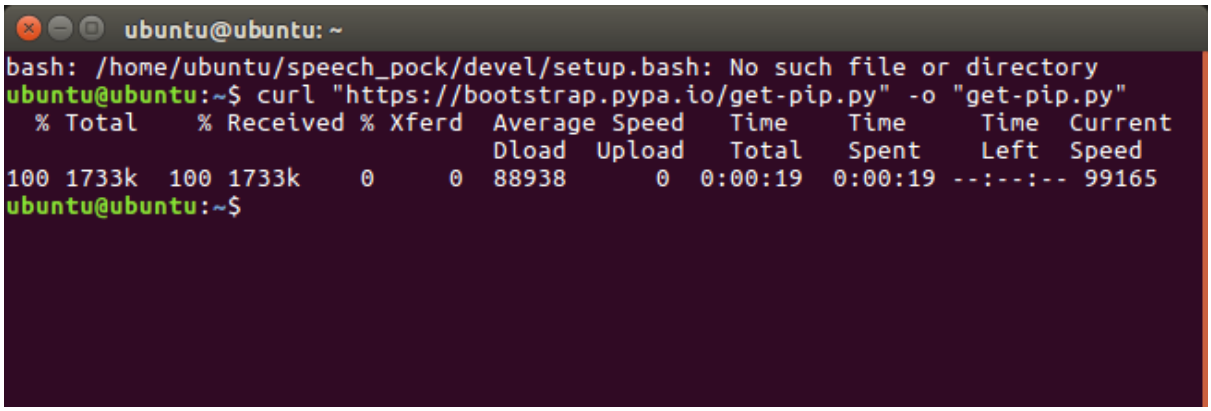


Figure 5.18: Install pip with cURL and Python

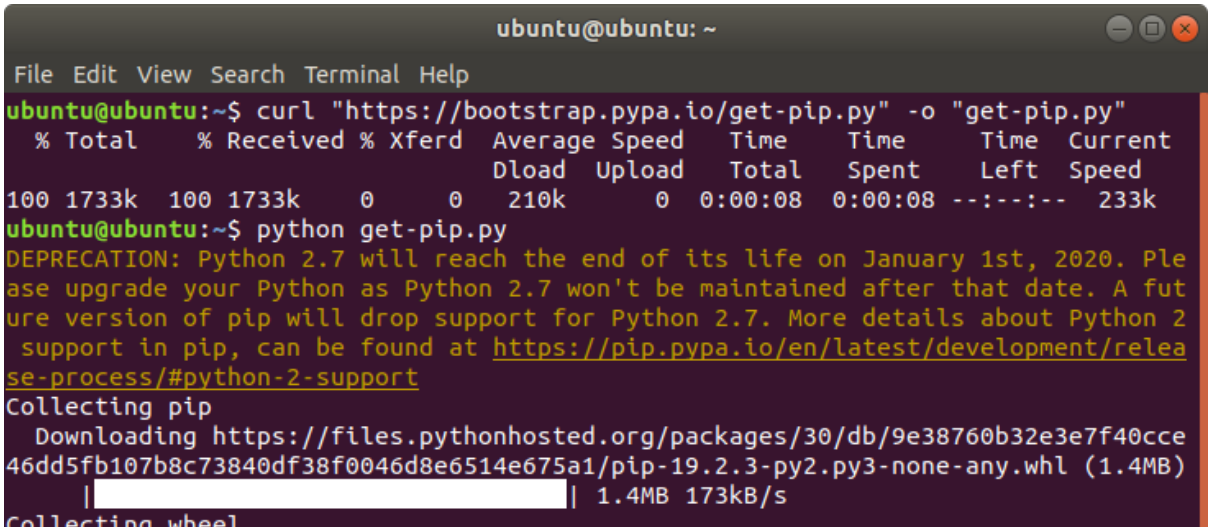


Figure 5.19: Install pip with cURL and Python second command

■ **Install pocketsphinx**

```
$ sudo pip install pocketsphinx
```

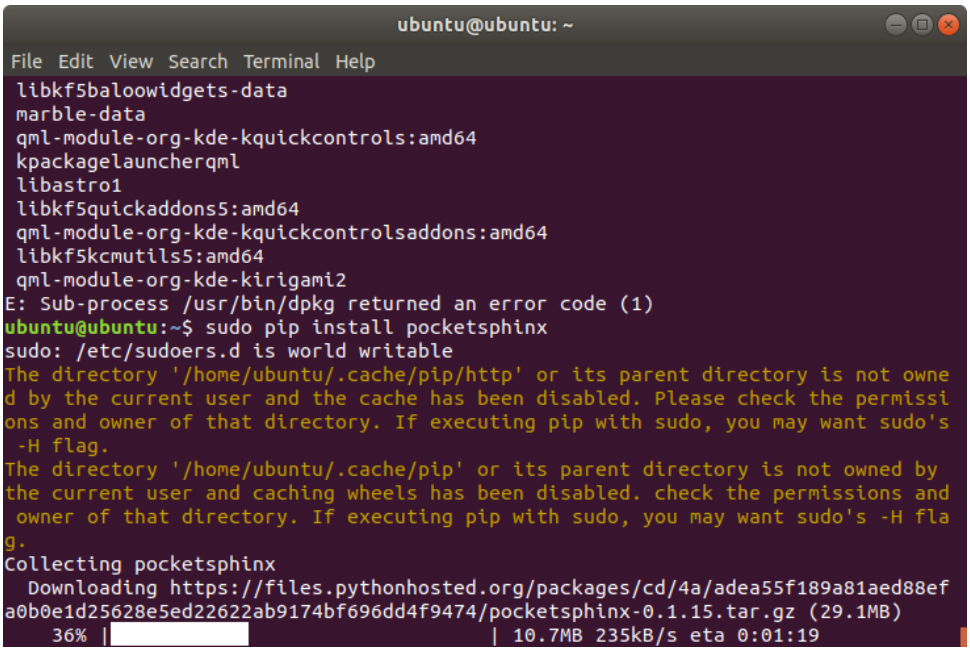


Figure 5.20: Install pocketsphinx Package

Now we will execute the process and try to send command to robot to move. However there is a problem in the mic of the Raspberry pi, we are going to send the command by keyboard inserted of sound. We have attempted to used ROS Networking discussed in section 5.3.6, unfortunately, this took us more than tow days. So Now we are going to explore the function of the three ROS packages and run the nodes to get actions.

■ ***PocketSphinx***

As mentioned above that this ROS Package that convets the speech into text. The first thing to do is to run the node of *PocketSphinx* which will start listening the to the sound comes from the user. The command line to start the node is:

```
$ roslaunch pocketsphinx robotcup.launch
```

This command should be run in the path directory of the node.

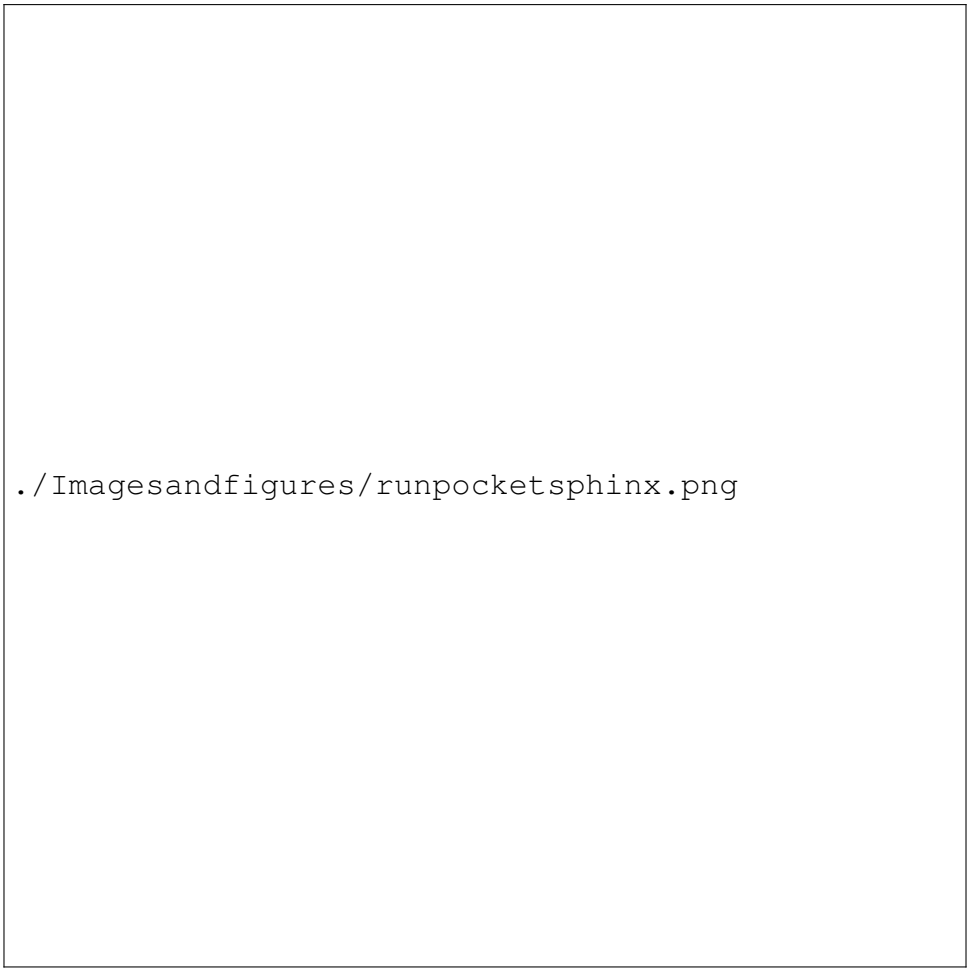


Figure 5.21: Output of the start speech chat launch file

The output topics in this package are shown in figure 5.22

Now after this package has been run and we know the type of the output, we see the possible actuating topics to control MOBOT by speech. The following command line shows a list of this topics to control the robot. See figure 5.24.

```
$ rostopic list
```

■ ***AIML Package***

This package plays the role of loading the database of the AI from the Storage memory into the **RAM!** (**RAM!**). One can create his own package and create folders called *data* , *scripts* , and *launch* to store the AIML files, Python scripts, and ROS launch files [2].

The command line to launch this package is:

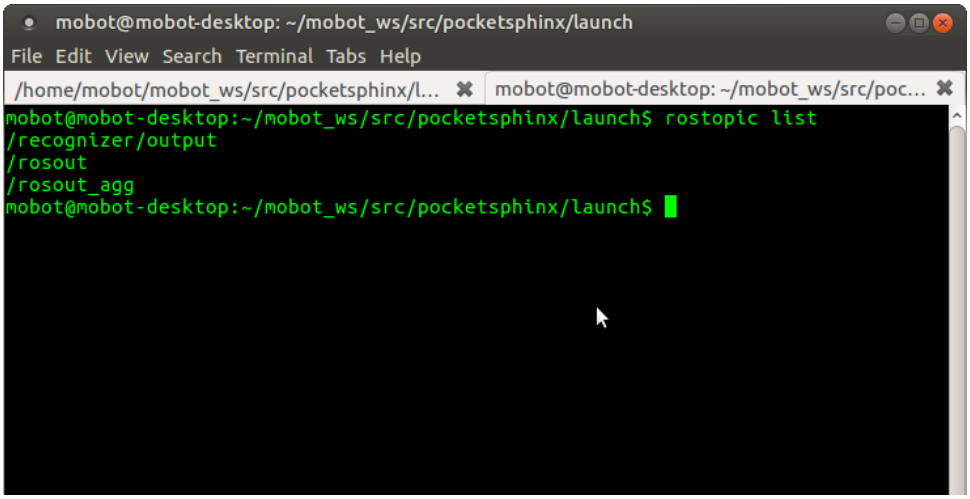


Figure 5.22: List of topics in this pocketsphinx

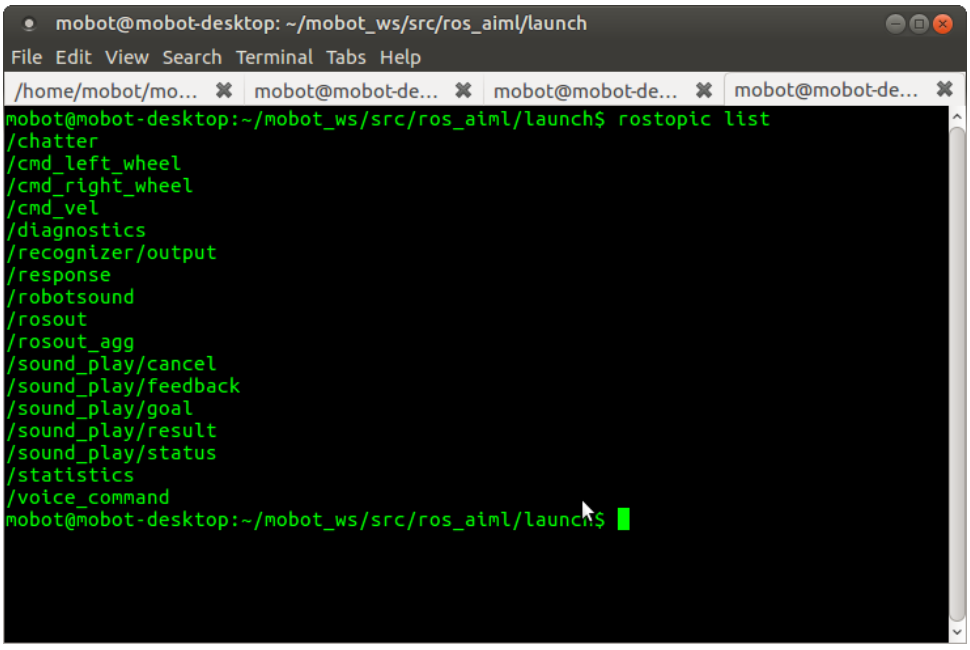


Figure 5.23: List of topics to move MOBOT by speech

```
$ roslaunch ros_aiml start_chat.launch
```

This will start the strat_speech_chat which starts the **AMIL!** server AIML TTS node, and speech recognition node.

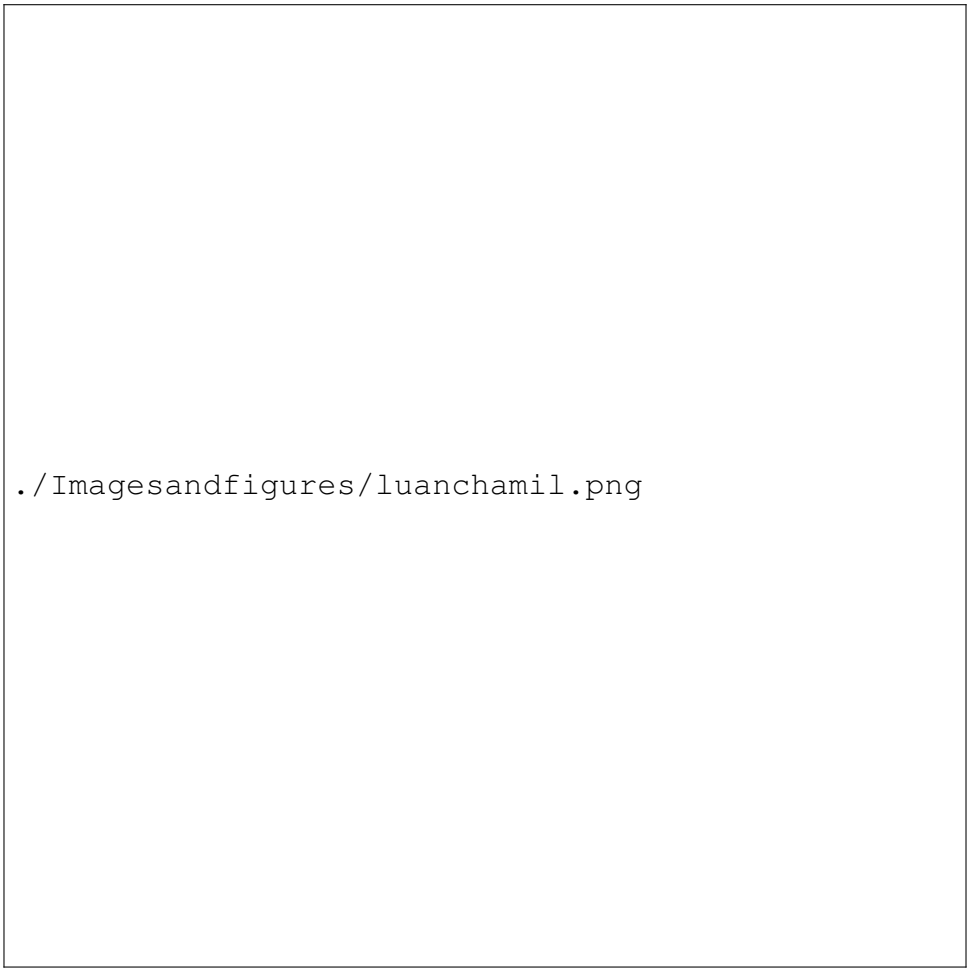


Figure 5.24: The output of strat_speech_chat

We can use the following command to start interacting with the AIML interpreter and send commands to the robot to control it. The response will be converted to speech as well.

```
$ roslaunch ros_aiml start_tts_chat.launch
```

To enable the speech recognition of the **AMIL!**, we use the folloeing command

```
$ roslaunch ros_aiml start_speech_chat.launch
```

Now we are ready to control the robot using the speech. before showing this we are going

to show the nodes created so far. they are shown in figure 5.25

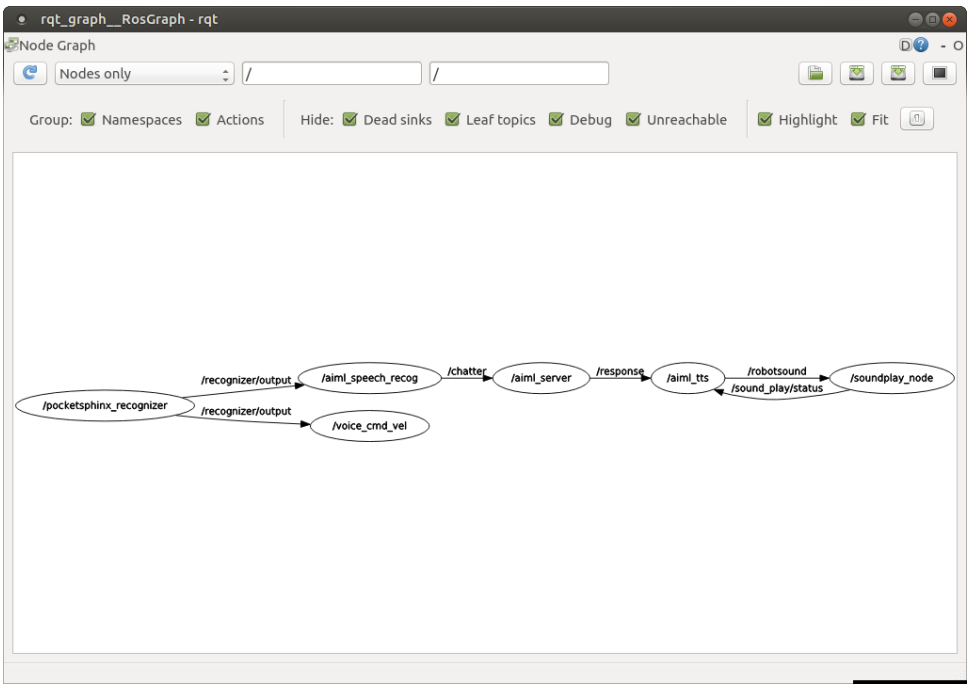


Figure 5.25: The nodes running with the topics

The following command is used to set up the speech recognition.

```
$ roslaunch pocketsphinx.launch
```

If we show the running nodes in figure 5.27 after running the actuating nodes

Figure 5.26: Controlling the robot using the speech

Figure 5.27: Running ROS nodes after adding the actuating nodes

CHAPTER 6

RESULTS AND EVALUATION

CHAPTER 7

FUTURE ENHANCEMENT

CHAPTER 8

CONCLUSION

APPENDIX A

CODES

This appendix contains the complete codes

A.1 C++ Codes

A.2 Python Codes

A.3 Arduino Codes

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(9, 8, 7, 6, 5, 4);
const int operation=A0;
int temperature = A1;
int overtemp= 13;
int lowBettary= 12;
int voltage= A2;
float voutV = 0.0;
float voutT = 0.0;
float temp;
float vin = 0.0;
float R1 = 100000.0; // resistance of R1 (100K) -see text!
float R2 = 10000.0; // resistance of R2 (10K) - see text!
int value = 0;
void setup(){
   pinMode(overtemp,OUTPUT);
    pinMode(lowBettary,OUTPUT);
  pinMode(operation,OUTPUT);
   pinMode(temperature,INPUT);
   pinMode(voltage, INPUT);
   lcd.begin(16, 2);
```

```
   lcd.print("Batery V");
   lcd.setCursor(0,1);
   lcd.setCursor(0,2);
   lcd.print("Temp");
}
void loop(){
   // read the value at analog input
digitalWrite(operation, HIGH);

   voutT=analogRead(temperature);
voutT=(voutT*500)/1023;
temp=voutT;
lcd.setCursor(9,1);
lcd.print(temp);
delay(1000);
  digitalWrite(operation, LOW);
if(temp>=25){
     digitalWrite(12, HIGH);   // turn the LED on (HIGH is the voltage
    level)
   digitalWrite(operation, HIGH);

   delay(1000);
  digitalWrite(operation, LOW);
   }             // wait for a second
    else
  {digitalWrite(12, LOW); // turn the LED off by making the voltage LOW

   delay(1000);
   digitalWrite(operation, HIGH);
  }
   value = analogRead(voltage);
   voutV = (value * 5.0) / 1024.0; // see text
   vin = voutV / (R2/(R1+R2));
   if (vin<10) {
   digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
  digitalWrite(operation, LOW);
   delay(1000);// wait for a second
   digitalWrite(operation, HIGH);
 }
```

```
60     else
61   {digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
62    delay(1000);
63 }
64 lcd.setCursor(10, 0);
65 lcd.print(vin);
66  digitalWrite(operation, HIGH);
67 delay(1000);
68  digitalWrite(operation, LOW);
69 }
```

APPENDIX B

ALGORITHMS

B.1 Algorithms

B.2 Speech recognition algorithm

B.3 Object detection algorithm

REFERENCES

[1] Joseph, L. (2015). *Mastering ROS for robotics programming*. Packt Publishing Ltd.

[2] Joseph, L. (2017). *ROS Robotics Projects*. Packt Publishing Ltd.

[3] Joseph, L. (2018). *Robot Operating System (ROS) for Absolute Beginners: Robotics Programming Made Easy*. Apress.

[4] Lite, L. (2016). "v3 operation manual and technical specifications.

[5] Mahtani, A., Sánchez, L., Fernández, E., and Martinez, A. (2016). *Effective robotics programming with ROS*. Packt Publishing Ltd.

[6] Pi, R. (3). "Model b+-raspberry pi", raspberry pi, 2018.