A project report on

Indoor Navigation and Localization Mobile Robot

(Final year project)
Submitted in partial fulfillment of the requirement for the award of the Degree of Bachelor in
COMPUTER AND SYSTEMS ENGINEERING

Submitted by

Ahmed Abdelbadee Elsayed
Ahmed Abdelbasit Mohamed
Aya Ibrahim Elsayed
Nourhan Mansour Mohamed
Omar Raafat Abdullatif
Yasmin Ahmed Abdelbasit

Supervised by:

Dr.Ahmed Mohamed Helmi

July 20, 2017

Faculty of Engineering
Zagazig University

Abstract

This project introduces a small-size mobile robot to be used for indoor navigation. It can be operated either autonomously or controlled by man remotely over the network. Its size make it able to navigate in small places and narrow paths man can not go through. Man can discover the environment around it via a video stream transmitted by a camera free to rotate right, left, up and down. A user-friendly controller box is provided for manned control to guide robot's motion, set camera orientation and switch on/off flash light.

The autonomous navigation is based on graph theory and artificial intelligence search algorithms in a predefined map for the environment around robot. With the help of Landmarks detection by computer vision, the robot can identify its location periodically on his path from source to destination. This provides a great help to avoid accumulated errors caused by hardware or sensors in accuracy.

A computer graphical user interface (GUI) application is developed to easily reach the functionality of robot. By this application one can select either autonomous or manual mode and deal with each mode utilities.

The project is based on Robots Operating System (ROS) which makes its functionality reusable in other projects. Modularity and readable codes are considered in the design and implementation of software nodes. Also an optimized communication protocol is developed among project's parts.

About future work there is a wide field of updates like object detection, On-line Mapping of new Environments and installation of manipulator (i.e. robot arm) for a variety of tasks.

Acknowledgments

We want to thank our department, computer and systems dept. at Zagazig University, who provided valuable comments, ideas, and assistance, which were essential to this study. This work is developed for and fully funded by Zagazig University, Faculty of Engineering as a part of development for a robotic platform.

First of all, We want to thank our supervisor Dr.Eng. Ahmed Helmi for giving us the chance to work on this project and helped us to participate in a lot of conferences and competitions. Also we want to thank him for his valuable notes and advices for developing the proposed algorithm applied in our project.

We also want to thank Dr.Ing Mohamed Nour for his helpful guidance and opening the way to learn and deal with Robots Operating System platform which represents the main core of our project.

We would like to thank our friends for helping us in the manufacturing process of the robot frame. And many thanks to Eng. Mahmoud Ibrahim for providing us the material of the robot frame which makes our project more robust and leads to better performance.

We would like to thank our parents and family for giving us the encouragement to reach our goals and saved no effort to provide us all the needs.

Contents

Key abbreviations

ROS	Robots Operating System
GUI	Graphical User Interface
RPi	Raspberry Pi
NiMH	Nickel–Metal Hydride
BMP	Bitmap
GBFS	Greedy Best First Search
PPR	Pulse Per Revolution
IR	Infra Red
SSH	Secure Shell
API	Application Program Interface
IDE	Integrated Development Environment
JSCH	Java Secure Channel
mAh	milli Ampere Hour
PID	Proportion Integral Derivative
PWM	Pulse Width Modulation
I^2C	Inter-Integrated Circuit
IMU	Inertial Measurement Unit
LiPo	Lithium Polymer
RPM	Revolution Per Minute
ADC	Analog-Digital Converter
LPF	Low Pass Filter
FPS	Frame Per Seconds
GND	Ground

List of Figures

Chapter 1

Introduction

In daily life there are a lot of situations in which robots are needed to perform some tasks man can not deal with. Some of these situations may be risky, difficult or such impossible for man to do. Think about a risky place we want to discover like disasters area, places of extreme environmental conditions or military purposes. In these situations introducing a robot is important for saving human life.
Robots also can help people of special needs with what they can not do like carrying heavy things, holding and placing parts or even home cleaning.
Robots have many configurations, styles and mechanisms for motion. Some are legged, others are wheeled and the rest can fly, swim or dive. Each configuration has its functionality that others can not do and also has limitations. In our project we introduce a robot with good navigation and localization technique to solve a lot of problems mentioned above in this section.

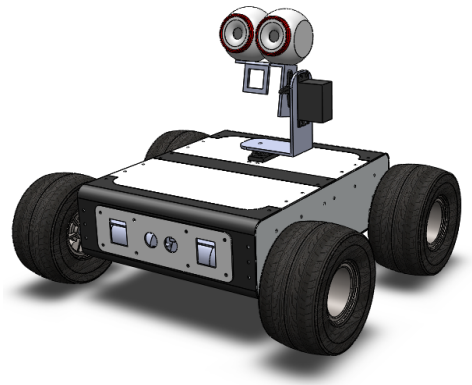

Figure 1.1: A 3-D model for robot frame

1.1 History of Mobile Robots

Mobile robots have the capability to move around in their environment and are not fixed to one physical location. Mobile robots can be "autonomous" (AMR - autonomous mobile robot) which means they are capable of navigating an uncontrolled environment without the need for physical or electro-mechanical guidance devices. Alternatively, mobile robots can rely on guidance devices that allow them to travel a pre-defined navigation route in relatively controlled space (AGV - autonomous guided vehicle). By contrast, industrial robots are usually more-or-less stationary, consisting of a jointed arm (multi-linked manipulator) and gripper assembly (or end effector), attached to a fixed surface.
Mobile robots have become more commonplace in commercial and industrial settings. Hospitals have been using autonomous mobile robots to move materials for many years. Warehouses have installed mobile robotic systems to efficiently move materials from stocking shelves to order fulfillment zones. Mobile robots are also a major focus of current research and almost every major university has one or more labs that focus on mobile robot research. Mobile robots are also found in industrial, military and security settings. Domestic robots are consumer products, including entertainment robots and those that perform certain household tasks such as vacuuming or gardening. [5]

Figure 1.2: Different applications for mobile robots; industrial, military and transport

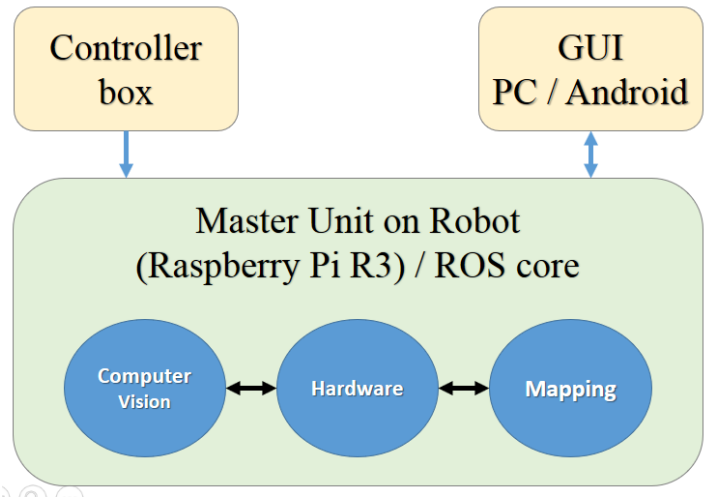

Figure 1.3: Clustered view of main system structure

1.2 Project Objectives

The main objective of this project is to demonstrate a robust mobile robot in a small scale to perform an autonomous navigation from point to another. Also it is required for the robot to localize itself when asked to do. Both navigation and localization depends on the ability of robot to detect and recognize texts on landmarks that uniquely identifies specific nodes in the map. For new environments whose map is not known, the robot can be guided remotely over the network to explore that location and a camera is provided for both steaming live video for the site around robot and to perform the computer vision task. In the hardware level a robust controller is required to perform motion instructions with acceptable precision.

1.3 Limitations

During our work we faced a lot of problems associated mostly with sensors. As we the main part in any control system is the feedback. This is because if we got a wrong indication for current state of system, we will perform a wrong reaction and the error increases more and more. In our project we need sensors to get information about the robot like position, velocity and orientation. Any missing part of them leads to both wrong estimation of state and wrong controller action. In next sections we will talk about our trials, results and algorithms implemented to get the advantage of each sensor and avoid its misleading data. But lets start from a high-level point of view and gradually take important topics with some details.

1.4 Overview on the project parts

The main system structure of our project as shown in figure 1.3 consists of a GUI and controller box at user side and the master unit (RPi) at robot side. The operation starts from GUI to select the function needed and then a flow of communication commands are passed to master unit over the network to perform the required task. Controller box is used in manned mode of operation to guide robot motion and camera orientation. In next chapter we talk about the GUI; how you can use, how it is implemented and a quick over view on ROS (Robots Operating System).

Chapter 2

User Manual and GUI

For the ease of controlling the robot by non-technical users, a user-friendly desktop application is created so that the user can operate on the robot with a few button clicks and also can follow the robot behavior with a video stream from a camera installed on it.

2.1 Main Scene of Application

For controlling the robot, the user can choose one of two modes from the first screen of the application:

2.2 Manned Mode

The user can control the robot manually by clicking on the Manned button and he will be forwarded for the manned mode screen:
At the click on *Start Camera* button, a video stream will be started on the black rectangle on the screen so the user can follow with the robot.
At the click on *Start Connection* button, the robot will start operating so the user can control it from a handy stick.

2.3 Autonomous mode

For the robot to operate independently, the user can go with the autonomous mode by clicking *Autonomous* button on the first screen to be forwarded for this window:
The Start Camera and Start Connection button functions are the same as the manned mode, for the robot to start operating autonomously, it needs the destination to go to so firstly, the user need to click on the Get Location button so the robot can recognize its

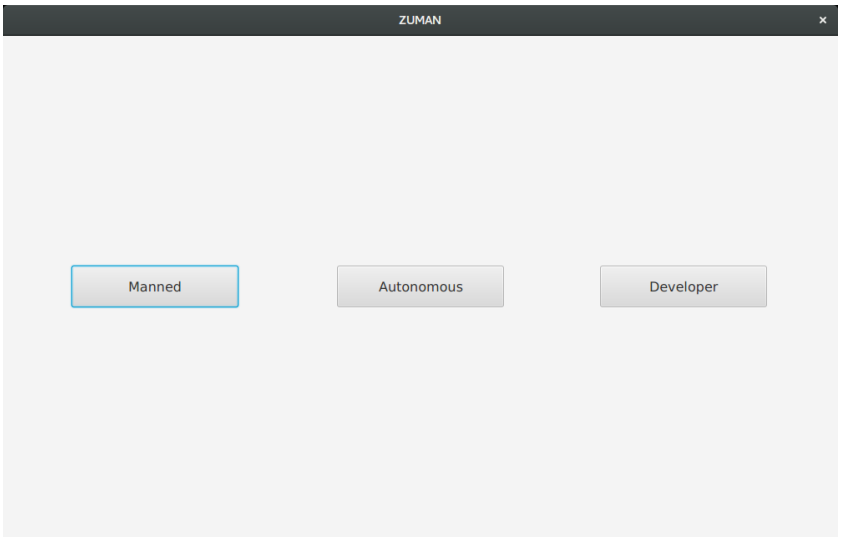

Figure 2.1: Main scene of the application

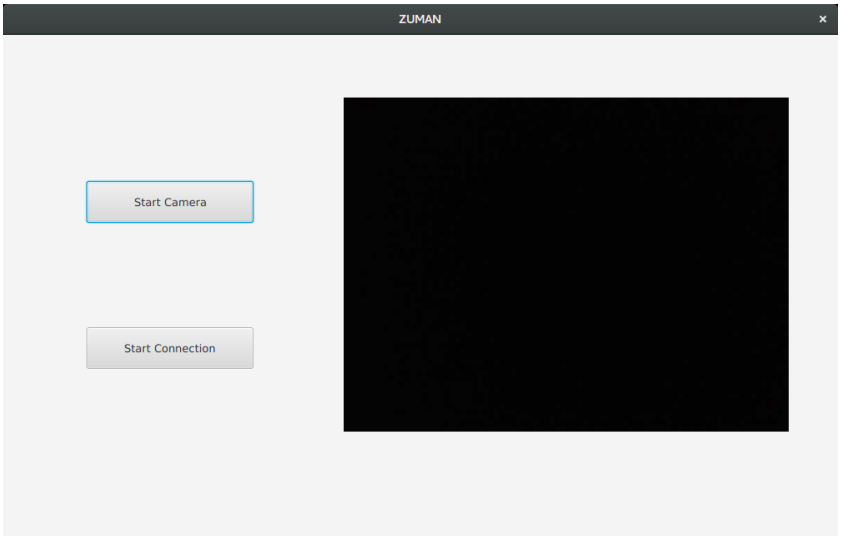

Figure 2.2: Manned mode scene in the application

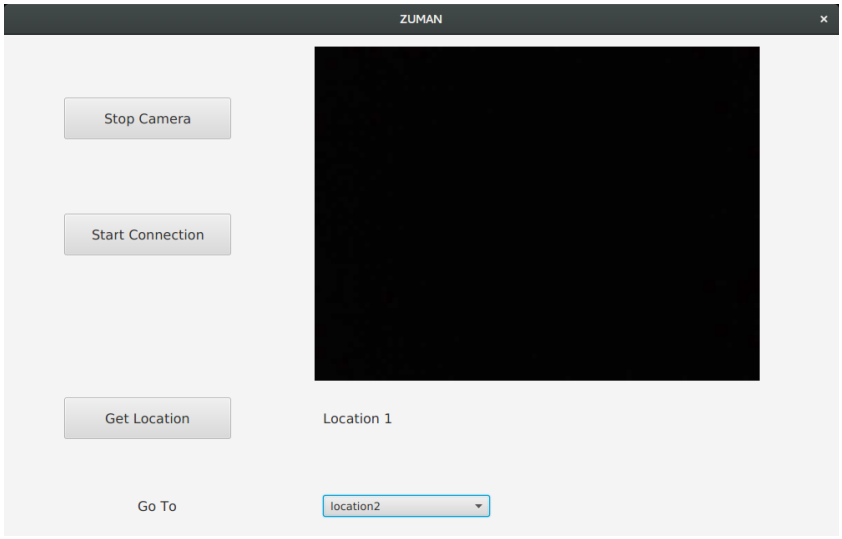

Figure 2.3: Autonomous mode scene in the application

current location and secondly, the user need to choose a destination from the drop-down menu and the robot will start operating.

2.4 Developer Mode

This mode is provided for technical users (i.e. maintainers) it they want to test the robot with different control commands before changing the source code. At the click on Developer button on the first screen, the user will be forwarded to this screen:
The command text box is used for sending messages to the robot and then click on send button. For subscribing to a specific topic, just write the topic name in the Topic name text box and click on echo button and the subscribed messages will be displayed on the Subscribed data text area.

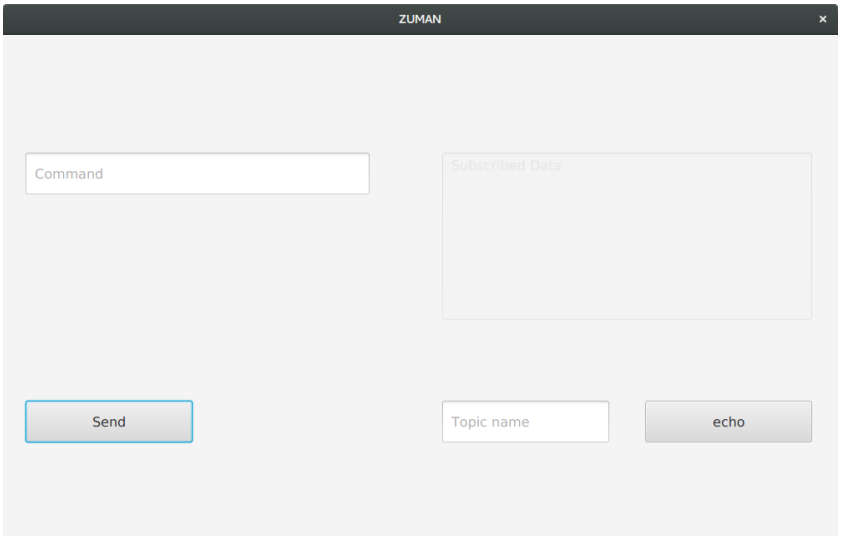

Figure 2.4: Developer mode scene in the application

Chapter 3

System Structure and Proposed Algorithm

In the design stage of project, three main parts arises to be implemented; mapping, computer vision and hardware. Each part represents an executable program that can interact with other parts in a manner to fulfill the required task. So, lets first talk briefly about ROS and how it provided a great help in connecting project parts and how communication between processes becomes easy.

3.1 Robots Operating System (ROS)

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. [3]
In ROS, Node is a common word that represents the executable file. So, in our project we have three main nodes; mapping, computer vision and hardware nodes. communication between nodes in ROS can be performed in many ways. The method we worked with is the message communication. Message represents the ROS data type. We developed a special type that can handle all communication needs between nodes. It is called Instruction and consists of a string variable that holds the command name and two float arguments.
Nodes can deal with messages in two manners; as a publisher, subscriber or both. Another word commonly used in ROS is the topic. It is considered as an intermediate program that holds any published message and forward it to all nodes that subscribed for it. This feature is very useful as a node can perform just one publishing command and any number of nodes can receive it. Now we know a bit about how ROS works and for more details and tutorials you can visit ROS tutorials site: http://wiki.ros.org/ROS/Tutorials.

Figure 3.1: ROS logo.

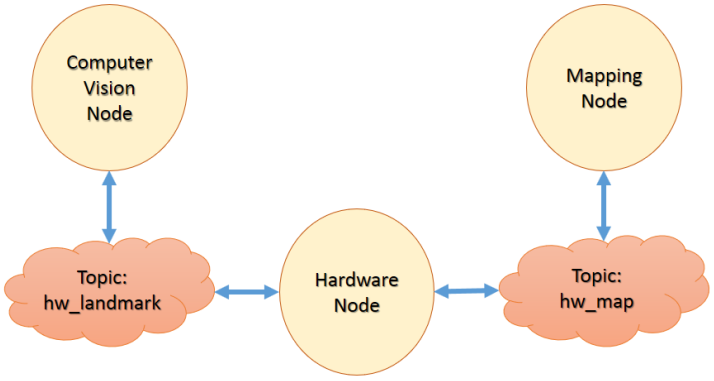

Figure 3.2: The implemented nodes and advertised topics between them.

In next section we are to talk about our project nodes and how communication performed among them.

3.2 Project Nodes and Communication Process

As mentioned before, we have three main nodes; mapping, computer vision and hardware. the communication between them were created as shown in figure 3.2.

There are two advertised topics; hw_landmark that connects hardware with computer vision and hw_map that connects hardware to mapping node. Over the hw_map topic the mapping node publishes the instructions that guide the robot from a landmark to another. When the robot reaches a new landmark, a tuning loop starts between hardware and computer vision node to put the robot exactly in front of the landmark. In this way, errors caused by inaccuracy of sensors or hardware motion performance are eliminated periodically resulting in a good autonomous navigation of the robot.
In next section we are to talk about the navigation algorithm and the sequence of instructions with some details.

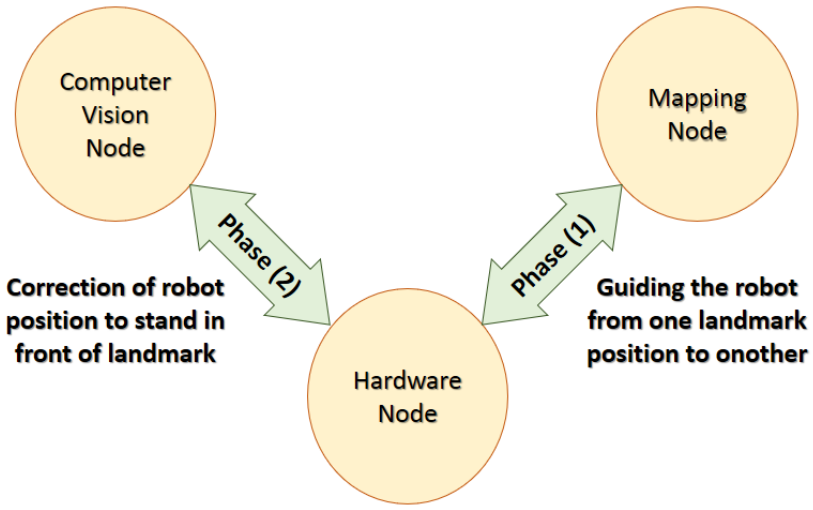

Figure 3.3: Proposed algorithm's phases

3.3 Proposed Algorithm for Autonomous Navigation

In the autonomous navigation mission, each of the three nodes has its own task to do. We can categorize these tasks in two phases. First phase is executed between Mapping and hardware node. Its goal is to guide the robot from one landmark to the next one on path to destination. Once the robot reaches that landmark or someplace near it, the second phase starts between computer vision node and hardware telling the robot how to move to stand exactly in front of the landmark. Figure 3.3 visualize these phases with brief description.
In the first phase, a sequence on instructions are passed from mapping to hardware node. These instructions can be 'move', 'rotate', 'rotate-camera' or a query for information of sensors. After each instruction the hardware responds by 'next_step' command as an acknowledgment to mapping node that the last instruction is done. When the mapping node receives acknowledgment of last instruction, it sends a 'tune' command telling the robot to start the second phase.
Second phase is a tuning process to eliminate any accumulated errors caused by hardware while performing mapping instructions. The computer vision node tries to guide the hardware to move in a way such that the land mark is detected at the center of picture frame. In this case, by knowing the distance between robot and wall we fully identify the robot position. Figure 3.4 shows how localization is done by computer vision and range finder sensor.
The flowchart of autonomous navigation process is provided ni figure 3.5 showing the whole communication steps and conditions involved in such process. You can identify in this flowchart the role of each node and how it deals with others.

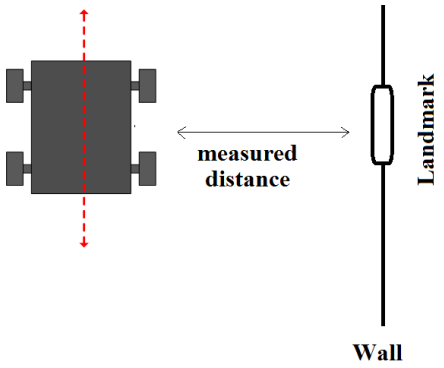

Figure 3.4: Localization using computer vision

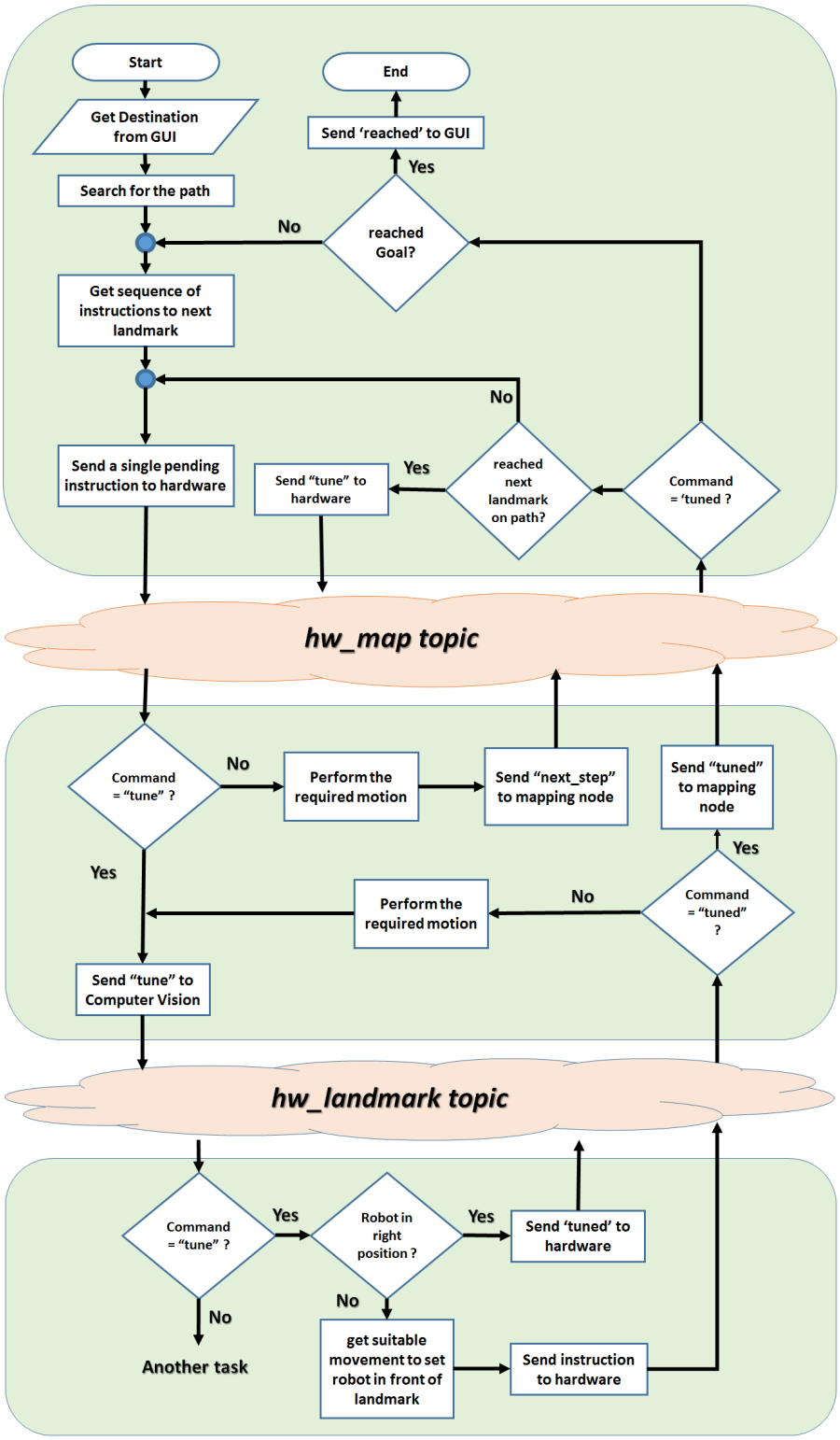

Figure 3.5: Flowchart for whole algorithm applied for autonomous navigation.

Chapter 4

GUI Implementation

The main tasks of the user interface are two:
1. Instantiate a connection with the main processing unit (Raspberry Pi) of the robot so it can control its functionality.
2. Connect to the IP camera installed on the robot to retrieve a video stream and show it to the user.

4.1 Connection with the Raspberry Pi

As our main robot system is based on ROS that it can operate from the system terminal so it needs to connect through the SSH (Secure Shell Protocol) and start to launch the required nodes. For the SSH connection, we have used JSCH (Java Secure Channel) API, it is open source with a BSD style license, it is not the best but it meets our requirements.
The ROS nodes that need to be operated are packed in a roslaunch file and it is launched through the SSH connection inside the application.
For the SSH connections to occur, the code follows this flowchart:

4.2 Video Stream from IP Camera

For displaying a video, we need to process its frames first so we used OpenCV 3.2 for Java. The process of the camera connection is shown in the following flowchart:

4.3 JavaFx software platform

The application is implemented using JavaFx software platform as is is the most suitable for creating desktop applications that can run across a wide variety of devices. Our application is consisting of one stage and four scenes, the first and main scene is main

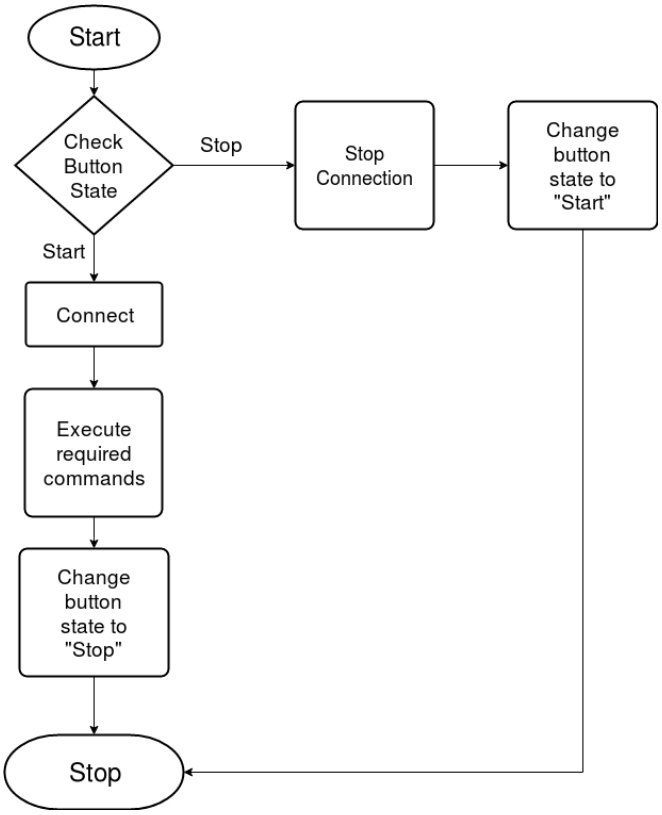

Figure 4.1: Flowchart for SSH connection with Raspberry Pi

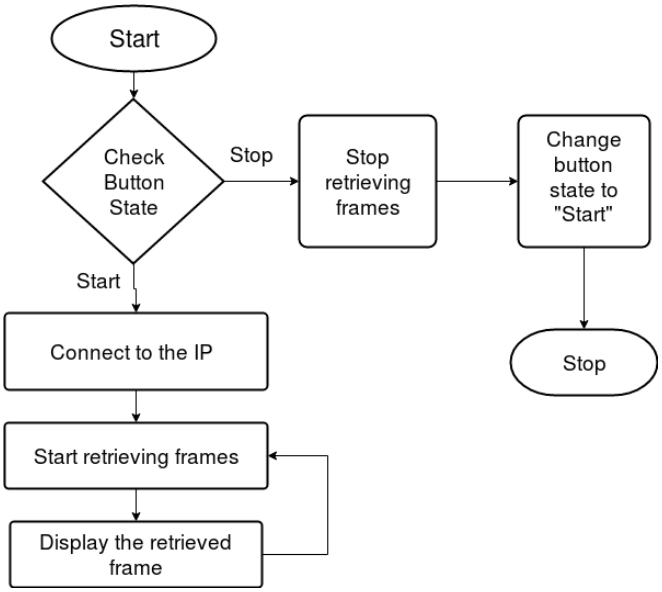

Figure 4.2: Flowchart for receiving video stream from ip camera

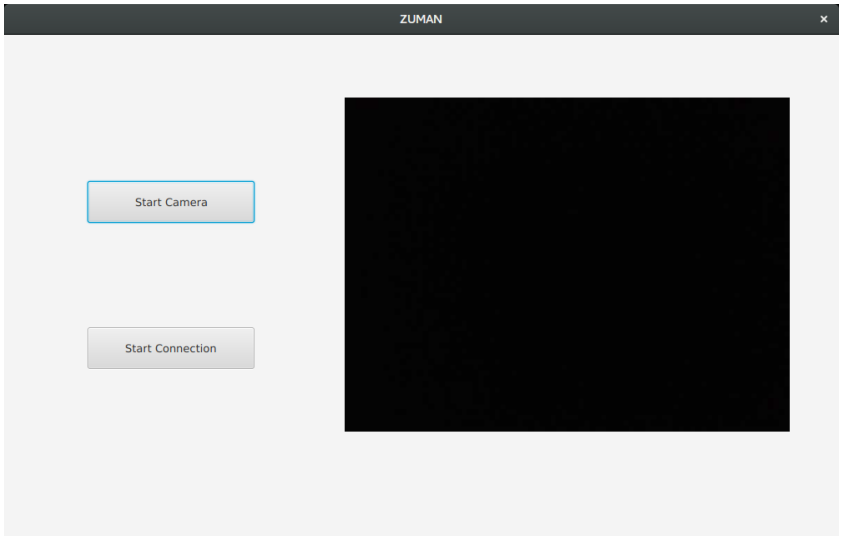

Figure 4.3: Video stream window in the application

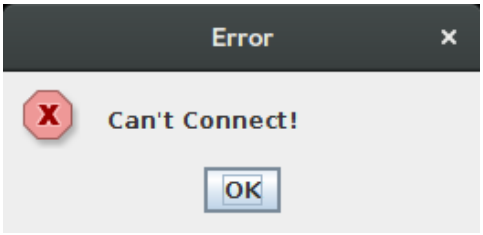

Figure 4.4: Error message for failed SSH connection

scene and it allows the user to choose a mode to operate the robot on as shown in figure 2.1.

Each of these three buttons is forwarding the user to a different mode with a different scene of controls at a click event. The second scene is for the manned mode which is used by the user to control the robot manually through a stick. At the click on button Manned in the first scene, the user is forwarded to the scene shown in figure 4.3:
At the click event of the Start Camera button, a video stream is retrieved from a specified IP and is displayed on this black rectangle of the scene.

At the click event of the Start Connection button, the application starts to connect to the Raspberry Pi.
On a successful connection, a specified roslaunch file is launched through a shell command and then the user can operate on the robot manually and the button state is changing to Stop connection so the user can stop the connection and exit at any time.
On a failed connection, an error dialog is displayed to the user as shown in figure 4.4.
At the click event of the second button, the user will be forwarded to the third scene of

the app for the Autonomous mode of the robot, as shown in figure 2.3.

This scene has the same elements and functions as the previous one with some additional functions.

At the click event of the Get Location button, a message is sent to the CV node to start recognizing its current location and then it sends it back.

For determining the destination for the robot, a drop-down menu has all the places of the robot environment and at the selection of a place, a message is sent to the MAP node to start determining the route between the current and the destination location.

Chapter 5

Mapping of Environment

Our robot is an off-line mapping system , meaning it only moves in a region when it has a map for it stored in its memory , we store the map in BMP format and the location of landmarks in a TXT file , and the mapping part gives the hardware part the instructions necessary to navigate from a start landmark to a destination landmark

5.1 Introduction

In this work, we present an algorithm for path planning to a target for mobile robot in known environment. The proposed algorithm allows a mobile robot to navigate through static obstacles, and finding the path in order to reach the target without collision. This algorithm provides robot the ability to move from the initial position to the final position (target). The path finding strategy is designed using a greedy and A* algorithms. The robot moves within the environment by sensing and avoiding the obstacles coming across its way towards the target. When the mission is executed, it is necessary to plan an optimal or feasible path for itself avoiding obstructions in its way and minimizing a cost such as time, energy, and distance. The proposed path planning must make the robot able to achieve these tasks: to avoid obstacles, and to make ones way toward its target.

5.2 Mapping Algorithms considered

Navigating a terrain and finding the shortest path to a Goal location is one of the fundamental problems in path planning. While there are many approaches to this problem, Our robot uses an off-line map to navigate through its environment , so it needs a search algorithm to find the best way to reach the destination node from a start one. We had two algorithms to consider , GBFS and A*.

5.2.1 Greedy Best First Search(GBFS) Algorithm :

This algorithm depends on a heuristic function which is the direct distance from the node to the goal node , and traverses its graph by selecting the node with the lowest heuristic in its frontier.

- **Advantages:**
  - Fast(less than a second) , which allows for remapping and obstacle avoidance
  - Consumes the least possible memory space
- **Disadvantages:**
  - It does not give the optimal route from source to destination
  - Not guaranteed to find the goal

5.2.2 A* Algorithm:

This algorithm depends on both a heuristic function (which is still the direct distance from the node to the goal node) and the cost which is the number of nodes that led to this node , and traverses its graph by selecting the node with the lowest sum of them in its .

- **Advantages:**
  - Balance between space-time consumption and accuracy
  - Guaranteed to find the goal
  - Gives the best possible path from start point to goal
- **Disadvantages:**
  - Slower than acceptable for real time (A few seconds)

5.2.3 Algorithm used in our project

After some tries and results demonstration We used a merge between them , good , but how?
We used a map with the path we want the robot to follow when possible is white and all others are grey (127 of 255 brightness) , the algorithm uses heuristic only in white areas (GBFS) and the sum of heuristic and cost in gray areas (A*) , this way , we ensure the algorithm moves in the wanted path unless necessary , we also used lines with (195 of 255) gray to indicate whether the robot is next to a landmark or not . We also noticed that the GBFS algorithm doesn't give the same results when going from source to destination and vice-versa , so we made the code run the algorithm both ways and choose the best to be operated. Flowchart of whole mapping role is shown in figure 5.1.

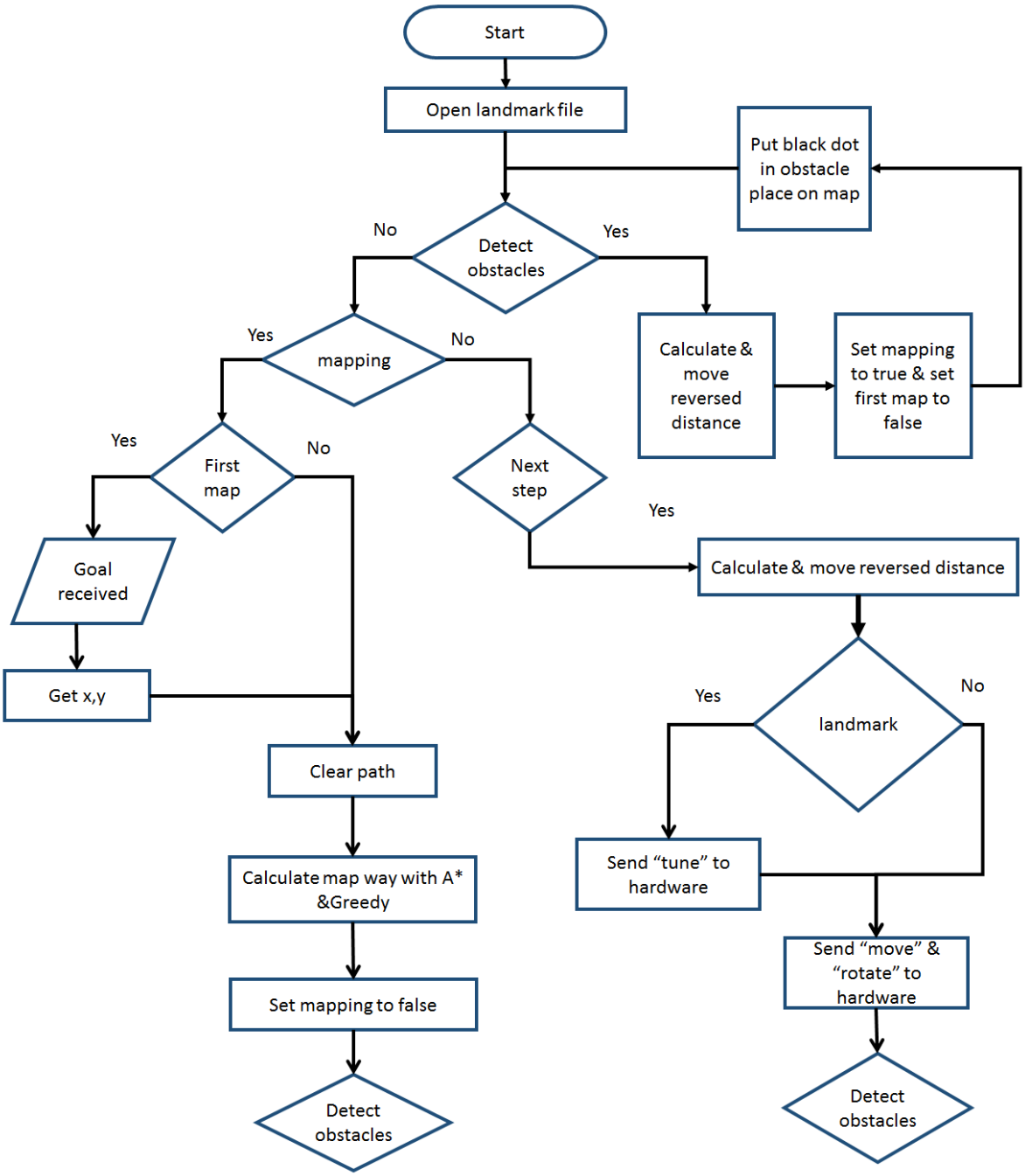

Figure 5.1: Flowchart of mapping process.

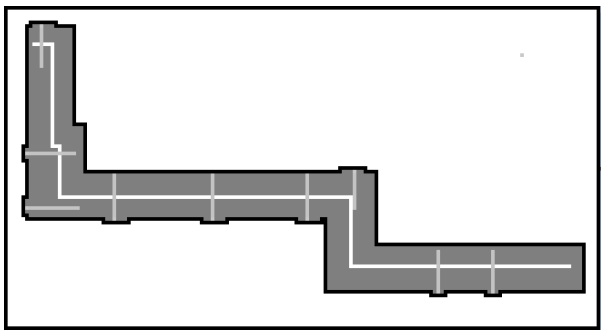

Figure 5.2: Map example represented in bitmap file.

5.3 Mapping Example of indoor environment

As we said first we use two kinds of search algorithm A* and GBFS. So, when the robot use them , as we show in this figure .. the map is build by colors from 0 to 255. We use three colors , if a robot in 127 color so it use A* search and the 195 color is detection rejoin if the robot in it should be ready to detect a land mark and the land mark is represented by 200 , otherwise the robot use greedy search , as this way we granted the robot always try to use greedy search to reduce the cost .

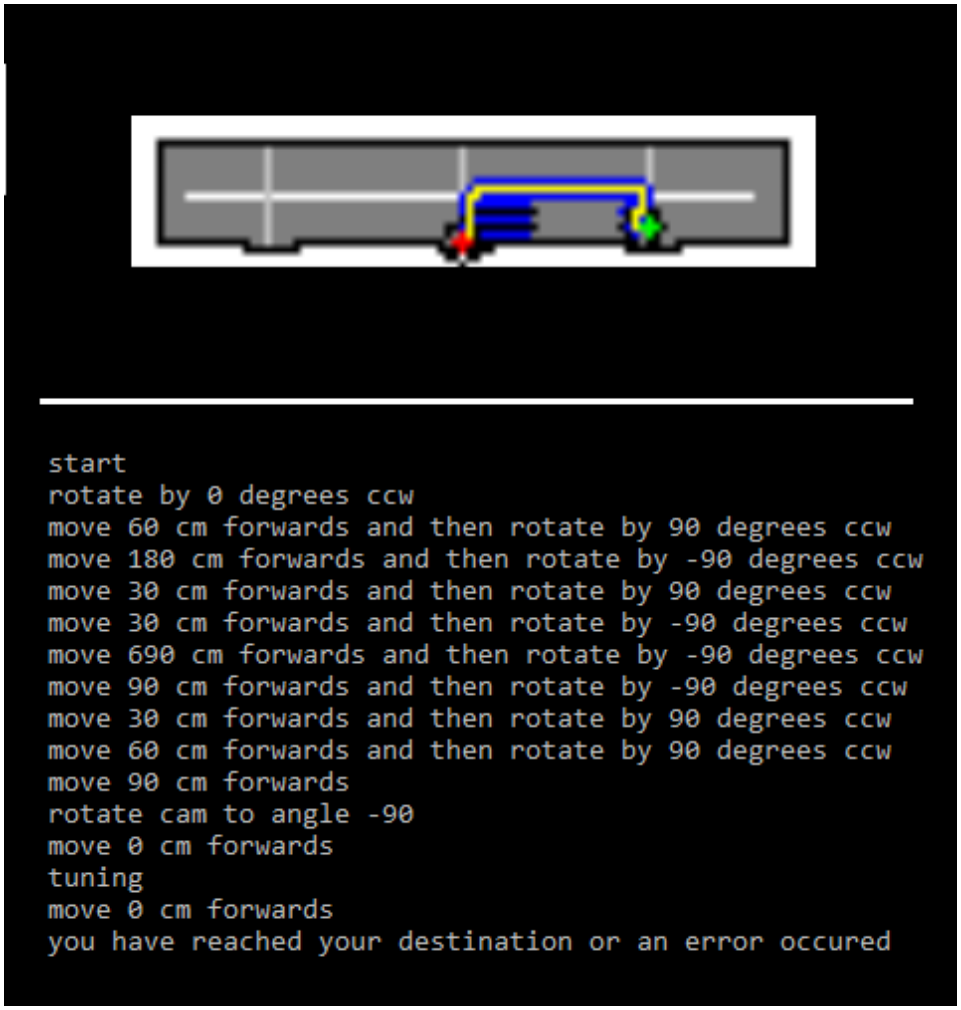

Figure 5.3: Path example and commands created from some source to destination in a trial map.

Chapter 6

Computer Vision

If robot is designed to blindly access the environment, its error rate will increase and its uses will be limited, So we used a Couple of Cameras with our robot for it to properly "see" the environment and better interact with it , we also used a library Called OpenCV which is – as the name implies – an Open source Computer Vision library , with methods used to process photos and extract info from them , we will use this library to help us locate our robots via landmarks , and detect and avoid obstacles (as a helping factor in addition to ultrasonic) , among other things.

6.1 Landmark detection for Localization

Our robot uses an offline map to navigate its environment , so it needs a landmark to use as a starting point , it may also use them as proof it has reached a specific point in the map and to compensate for errors , good , but how?

OpenCV is good with detecting circles , so we first chose our landmarks to be a thick circle with a number in it , when the robot is placed in a certain place , it spins around itself until it sees a landmark , then it goes and stands next to it , also when it knows (from the offline map) , that it is standing next to a landmark , it takes a photograph to confirm its location , the algorithm detects if there are two circle inside one another (the thick circle inner and outer perimeters) , then it confirms the results by cropping the outer circle and detecting the inner circle , the function returns the biggest thick circle in the scene and then recognizes it via OCR to determine which landmark it is , compare OCR results with landmark text it expects to find the best match.

This approach was not very good as circles get mis-recognized by the Tesseract OCR engine we used to recognize which landmark it is, so we decided to detect rectangles instead of circles, and we made the landmarks to be white small rectangular cards (which will work fine with better results than circles as long as the door it's sticked to is not

white too) , and to solve the OCR problem further we used two OCR engines : Tesseract and OCRad , to get better results (these were the only two free OCR engines we could find).

Experimental result for landmark detection and recognition is shown in figure 6.1.

6.2 Using Computer Vision for obstacle avoiding

Of Course an obvious question arises here , if we are already using Ultrasonic Sensors to detect obstacle , wouldn't it be redundant to use Computer Vision as well ? Of course extra precautions never hurts , but there is another reason , Ultrasonic sensor only detects the obstacle if its normal to the sensor , otherwise the ultrasonic signal will reflect away from the surface and won't be detected.

There are many ways to detect obstacles using computer vision , but each one has its pros and cons , we will discuss the methods we tried with our robot here.

6.2.1 Canny Technique

The technique depends on detecting the edges of the objects in the image , by blurring the image and using the canny transform , we get the edges of the objects in the image ,if the floor is not very rough it will have no edges and then it will be very simple to detect obstacles and walls!

- Advantages:
  - Easy to understand and implement
  - Super-Fast
  - Gives decent results in simple situations
- Disadvantages:
  - Does not work when floor is rough or has different colors

6.2.2 Floor Subtraction Technique

The technique depends on subtracting the most significant blob near the bottom of the image , which is bound to be the floor , keeping only the obstacles and the walls , the robot finds the longest path it can walk in without hitting anything and turns left or right accordingly , yes , it's as simple as that!

Figure 6.1: Detection and recognition of landmark.

- Advantages:
  - Easy to understand and implement
  - Very Fast
  - Gives decent results
- Disadvantages:
  - gets complicated (but solvable) when floor has different colors
  - The robots design obscures its view of the ground (not solvable)

6.2.3 Stereo BM/SGBM Techniques

That's how professionals do it ! BM/SGBM Stand for Block Matching / Semi Global Block Matching techniques , Both techniques are very similar , they both use two cameras as left eye and right eye , they both use a transform to find out which parts in the images are similar, they both return a grayscale map where near objects are brighter than far ones , this way we can detect obstacles as the near objects that are higher than a certain threshold from the floor , and find our way round them by finding the darkest part of the image and move towards it! • SGBM is more accurate than BM but it consumes much more time , so it's a tradeoff between time and accuracy

- Advantages:
  - Most accurate and professional way to find obstacles
  - Works on all kinds of floors
- Disadvantages:
  - Consumes large portions of processing power and memory
  - Slower than other methods

6.2.4 Used Method

We used SGBM while reducing the image size by 1/(8x8) of its original size to increase speed , the method uses threshold to determine whether there is something closer than a given distance and sends a remap command to the Mapping part if there is , it takes about half a second to process every frame so it should detect the obstacle when it's half a second away.

Chapter 7

Hardware Node and Modules Tree Structure

In prior sections we always look at hardware as a black box. This view is sufficient when we want to describe the overall operation of the robot. But lets now take this part in some details to know how instructions are processed after being received from either mapping or computer vision node.
In the design stage of hard ware, One of the most important aspects taken into consideration is modularity of design. Each Part in the hardware structure has a unique role and a method by which this role can be triggered for execution. Another important style of design followed is assigning low level tasks to multiple separate modules rather than having a central unit responsible for the whole operation. This method of design saves a lot of resources in the master unit opening the way for other complicated operations to be performed faster. In the following sections we will talk first about the role of hardware ROS node and then go down at low level.

7.1 Hardware ROS node

As mentioned in figure 3.5 the hardware node stands midway between both mapping and computer vision nodes. its main role is receiving instructions from other nodes, refining them and forwarding them to low-level modules to be performed by robot. This leads to two important results. First, the hardware ROS node is free of low-level operations which keeps its main role of regarding and communicating with other nodes. Second, low level modules save their processing capability for optimized motion and performance rather than being concerned with communication of multiple parts.

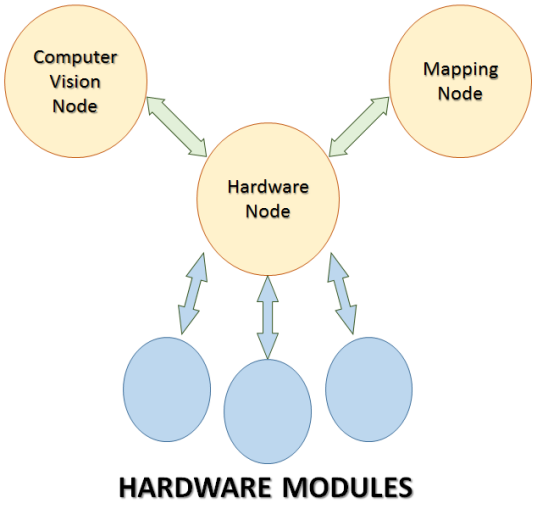

Figure 7.1: Hardware node standing midway between ROS nodes and hardware modules.

7.2 Low-level Hardware Kits and Modules

Here we reached to the stage of choosing the suitable kits and modules that can meet the needs of other nodes. In this case We must first encounter the whole instructions hardware may receive. These commands are as follow:

- move straight some distance
- rotate robot some degrees
- set camera orientation
- measure distance between robot and any object around
- switch on flash light
- set motors speed

Each of these instructions has some requirements to be done. The modules we brought to fulfill these requirements can be encountered as follow:

Component	Function	Number of units
Raspberry Pi R3	Master Unit	1
Arduino Nano Kits	processing units	3
NodeMCU	micro controller with embedded WiFi module	1
Motor driver	controlling motor speed and direction	1
DC Motors	manipulators for robot motion	4
Wheels	attached to motors	4
Motor shaft encoder	counting revolutions of motor	2
Compass Module	measuring heading angle	1
Ultrasonic module	range finder	4
Sharp IR sensor	range finder with longer distance span	1
Camera	capturing the environment around	2
Servo Motors	controlling camera orientation	2
Lithium Battery	Power source for motors and Light	1
Power bank 5V	Power source for Raspberry PI	1
NiMH Battery 5V	Power source for Servo motors	1

Components figures and data sheets are both included in appendix.

7.3 Hardware Tree Structure

In this section we get closer to know how exactly project parts are connected to communicate. Also we will define the exact role for each part. As shown in figure 7.2 there are three sub nodes underlying the hardware node. Each one has sub-modules to control or communicate with. All these nodes are implemented on Arduino kits and this is a magnificent advantage for ROS as it provides libraries that enables us to implement ROS nodes on such kits.[4] So, for each node it can publish and subscribe for message from the main hardware node on Raspberry Pi master. This enables us to have a single type of communication valid to use in all parts. We can identify the role of each node as follow:

- **Manipulator Node:**
  This node is responsible for executing motion commands. The controller is implemented in this node. As we can see in figure 7.2 there three underlying parts, the motor driver, flash light and ultrasonic. You may wonder why ultrasonic is here in the manipulator node. This simply because we needed to summarize all parts that participate in the motion control in a single node to minimize the flow of communication message as possible. The motor driver has two tasks; sending speed signals to motors and combine the encoder pulses to the Arduino. More details about each module will be figured out in next sections.

- **Sensors Node:**
  Like the previous node, this one is implemented on an Arduino Nano kit and attached to it other sensors included in our project; Compass, Ultrasonic, IR range finder and servo motors that control the orientation of camera. One of the critical tasks this node is responsible for is regarding the front path of robot to give alarms if there is some obstacle. Another task is participating in estimation of robot orientation when rotation motion is performed. More details about each sensor and how we get use of its readings will be discussed in lated sections.

- **Manual Controller Box:**
  This node is considered the main tool for adding a feature that the robot can be controlled remotely by man. Here in this node the main objective is giving user the ability to guide robot, change camera orientation and switching on/off flash light. This feature is so useful in exploration of new environments whose map is not available for autonomous motion. and it opens the way for adding on-line of such new environments.
  By determining the objective we can encounter the hardware units needed. So, for user-friendly controller we developed it using potentiometers and a switch connected to Arduino unit. And for publishing commands over the network a NodeMCU kit is used to implement a ROS publisher node on.

7.4 Electronics Involved in Project

In this part we are to talk about electronics modules and components used in out project. For each components we will discuss its function, how it is operated, advantages, limitations and how we used it as a port of the full structure. Electronic components are to classified into four categories; processing units, sensors, passive components and power sources

7.4.1 Processing Units

- **Raspberry PI R3**
  This part represents the master unit for the whole process. Its powerful processing capabilities opens the way complex processes that need much resources like CPU cycles or storage. One of these operations is the image processing and computer vision operated in out project. all the other parts are connected to Raspberry Pi by some way in a tree structure fashion. The operating system running on such unit is Ubuntu Mate 16.04 and we installed ROS platform base on it. We communicate and deal with this unit using SSH (Secure Shell) connection operated remotley from any other computer device.

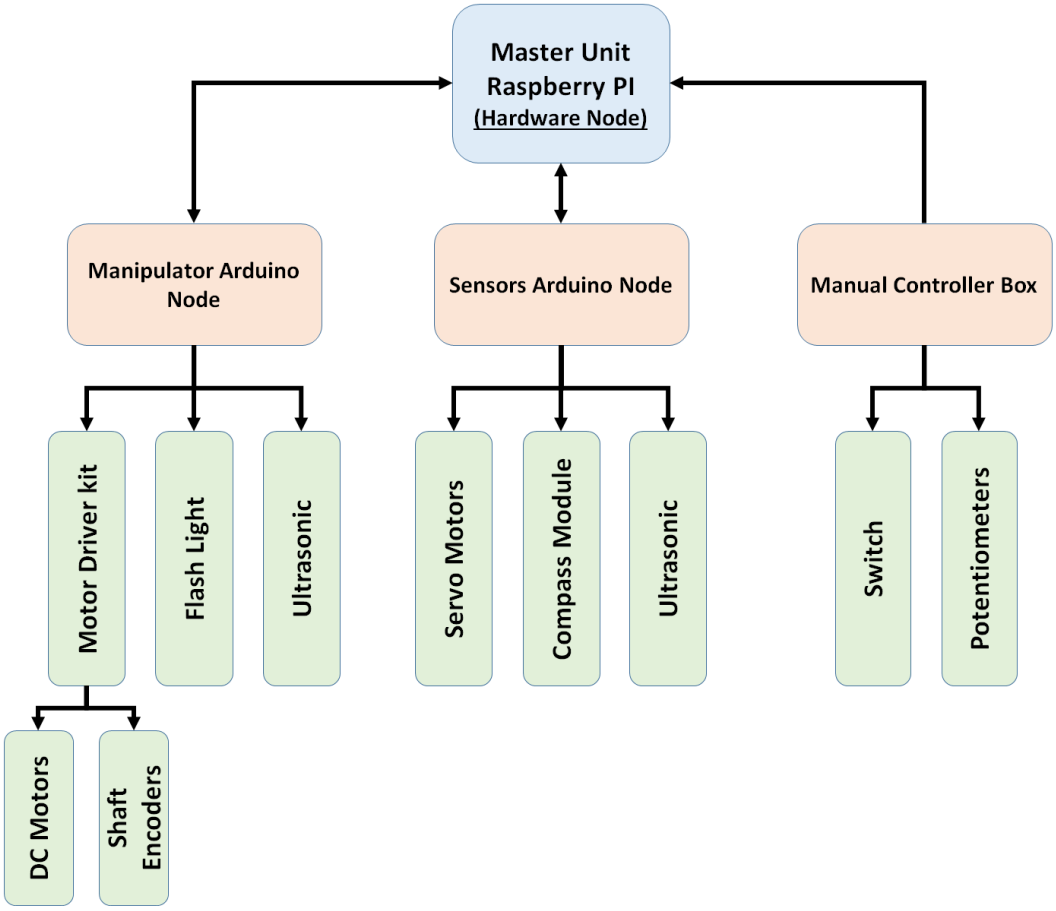

Figure 7.2: Hardware modules structure.

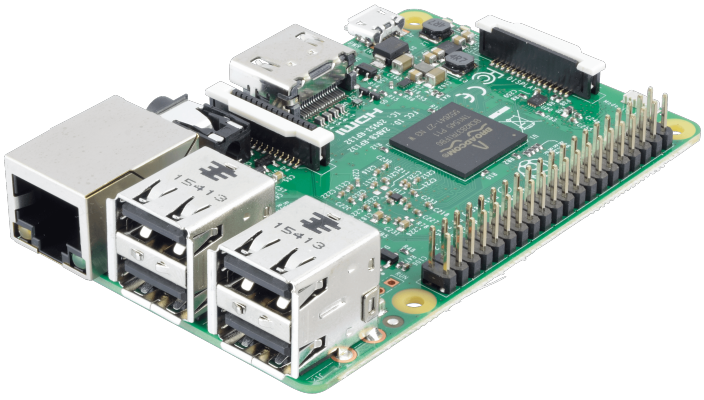

Figure 7.3: Raspberry Pi R3.

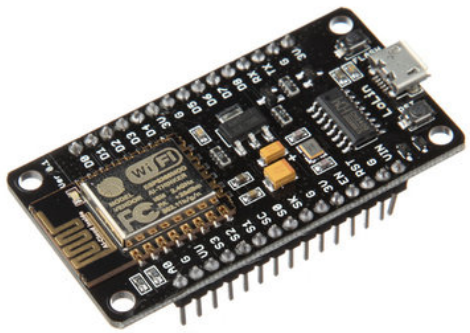

Figure 7.4: LoLin V3 NodeMcu

- **NodeMCU**
  This unit is a developed with integrated WiFi module. This helped us too much in the design of the manual controller box. We implemented a ROS node on it that publishes the instructions which can guide the robot and control camera orientation or flash light. This design and procedure will be explained in the software development section. We deal with this kit like an ordinary Arduino and program it using same Arduino IDE after installing some updates.

- **Arduino Nano**
  This kit is limited but powerful processing unit that can be used in too many tasks and applications. Its small size, low cost and ease of use make it most common in many projects. In our project we used three kits of such type. First in the manipulator node that gets the motion instructions, run the controller and send output signals to motors. Second is used in Sensors node that have all sensors underlying it. This kits helped us too much in maintaining the modularity and integration of parts of our project.

7.4.2 Kits and Sensor modules

- **Motor Driver**
  There are a lot of modules having the function of controlling DC motors. Each has it own specs like max current, operating voltage and temperature. We find this kit is most suitable for our robot as it contains 4 channels for 4 dc motors. Moreover it requires just two pins to control both speed and direction of motors. It also have important features; current sensor and impeded logic circuitry for handling Encoder signals come from shaft encoder sensor. Its Electrical specs are

Figure 7.5: Arduino nano kit

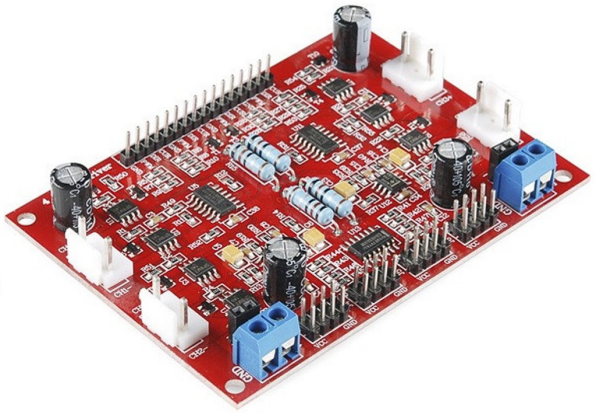

Figure 7.6: DC Motor Driver for Robot 4 channel

12V operating voltage and max current of 4.5A for each channel. These features and specs make it the best choice for our robot.

- **Shaft Encoder**
  One of the most popular method of tracking the robot position is calculating the number of revolution each motor made. This way is called Odometry feedback. Each encoder has a property of number of pulses generated per revolution. This is an indication of how much distance change the robot can feel and encounter. The type we used in out project is 3 PPR. But our motor is geared with a gear head of ratio 1:30. That makes the exact value 90 pulses generated for each wheel revolution Moreover, This Encoder has 2 channels each generates 3 pulses per revolution with phase shift 90 degrees. The logic circuitry in motor driver kit combines the two channels' pulses by an XOR gate resulting in an interrupt signal of 180 PPR. This gives a resolution of 0.21 cm per encoder interrupt pulse.
  But the most critical problem of this method of position estimation is that it may lead to fully wrong deviated measurements if the robot wheel slipped on the sur-

Figure 7.7: Motor shaft encoder 3PPR

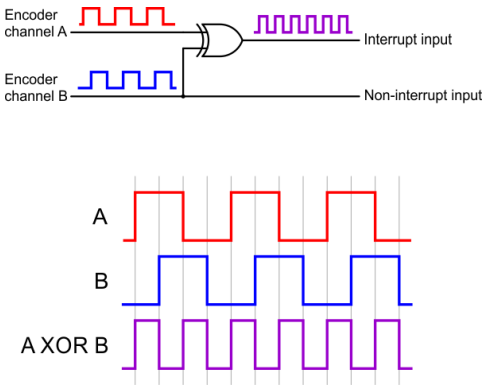

Figure 7.8: Quadrature encoder pulses explained

face. There is no guarantee that the robot has already moved that distance without slipping. So, This drawback opens the way to think about how to make sure that the robot is actually moving not slipping. Figure 7.8 shows the two channel pulses and the resultant interrupt signal used by combining them through XOR Gate.

- **Ultrasonic sensor**

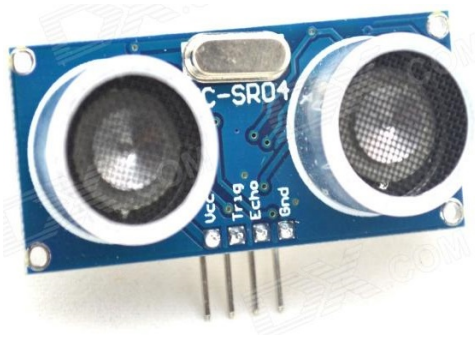

Figure 7.9: Ultrasonic sensor

This is a common sensor used for distance measurement between the sensor and the any object sitting in front of it. Its principle of operation depends upon sending an ultrasonic pulse and calculating the time passes until that signal echo come back and detected. The type we used has an operating range 1:300 cm. But there are many problems occur when using this sensor. These problems can be encountered as follow:

- accuracy of reading becomes bad while the measured distance increases.
- the procedure of measurement depending upon waiting until the signal comes back. This makes the processing unit idle and causes synchronization problems with other electronic modules running on the same micro-controller
- the sensor is not guaranteed to get a reading as the signal may be absorbed by some obstacle materials or reflected far away on inclined surface of the obstacle.

- **Compass Module**
  This sensor make use of measuring magnetic field of earth to get the heading angle(the angle between X axis of module and North directing), We paid this sensor much intention at first steps of work because it gives an absolute angle of robot heading which can be combined with encoder information about the displacement of robot to get a full estimation of robot location from the start point. But during our work we faced many problems that made us can not depend on its reading for estimation of robot position or heading.
  The worst problem associated with this module is that it is affected by any magnetic field around it. This is a big problem because indoor navigation makes the robot among too many electrical devices that generate different magnetic field patterns and so different effects on compass reading. Compass reading may encounter a scaling problem, dc shift or combination of them. To visualize this problem, we plot te compass readings while rotating it, the pattern we get is shown in figure 7.11.

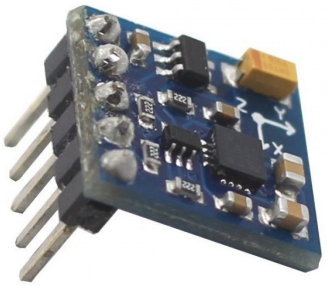

Figure 7.10: compass module HMC5883L

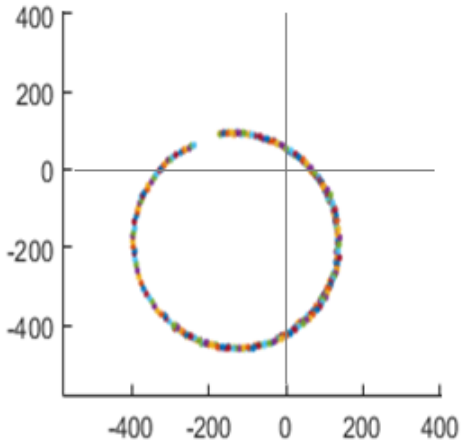

Figure 7.11: Showing the shifting problem in compass readings

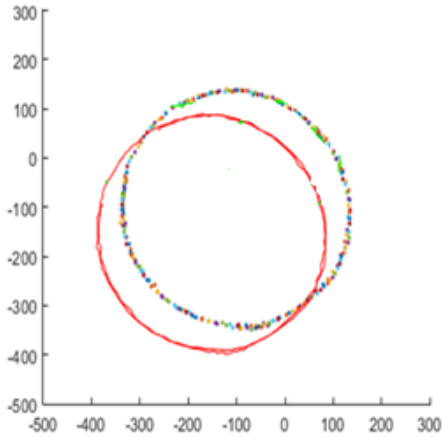

Figure 7.12: Showing the shifting problem in compass readings

Figure 7.13: MPU6050 Accelerometer and Gyroscope

About the sensitivity we took a reading pattern while rotating the compass and repeated the process at same place but a mobile phone was put at distance of 20 cm away from the module. the pattern was changed in both shift ans scale values. This result is shown in figure 7.12

- **Accelerometer and Gyroscope**
  This sensor provides measurements of both acceleration and angular velocity. So it can help in the estimation process as we know that displacement is the integral of integral of acceleration. And on the other hand, the heading angle of robot can be calculated from integrating the angular velocity measured by gyroscope. But the integration operation in such case leads to accumulated errors as we will show in next sections. So, it must be filtered to

- **IR Range Finder**

Figure 7.14: IR sharp sensor

Figure 7.15: Power Bank "Rechargeable Battery Pack" 7000mAh

This sensor is used for measurement of distances between the robot and objects in front of it. Its range is larger than the ultrasonic sensor but in short distances the returned readings goes fully wrong. Experiments figured out that the readings are reasonably acceptable in range from 80 cm to 500 cm.

7.4.3 Power Sources

- **Power Bank 5V**
  This power bank is 7000 mAh annd can support output current up to 2.4 A. This means about 2 hours and 48 minutes of continuous operation at full load. This is the power source that operates the Raspberry Pi unit.

- **Lithium-Polymer 12V Battery**
  This battery is 7000 Ah and is the power feeder to DC motors and flash light. It can support up to 7A making it lasts for one our of operation under full load condition.

- **NiMH 5V Battery**
  This battery powers up the servo motors controlling the camera orientation. Its capacity is 1500 mAh and has sufficient high discharge current

Figure 7.16: Lithium-ion Super Rechargeable Battery Pack (12V, 7000mAh)

Figure 7.17: NiMH Rechargeable Battery (5V-1500mAh)

Figure 7.18: Potentiometer 50 Kohm

7.4.4 Passive Components

- **Potentiometer**
  We used potentiometers in the manual contoller box to give a variable input in range we can use to map it into both speed and camera orientation angle. we used 4 potentiometers controlling left motor and right motors and camera yaw and pitch angle.

- **LEDs**
  We used LEDs so that the robot can navigate in dark environment without losing the ability of landmark detection. Moreover, when a man control the robot remotely he can switch on light to figure out the around environment via the video stream. The circuit's scheme we designed for this purpose is shown in figure 7.20 .

Figure 7.19: White LEDs 5 mm 3V

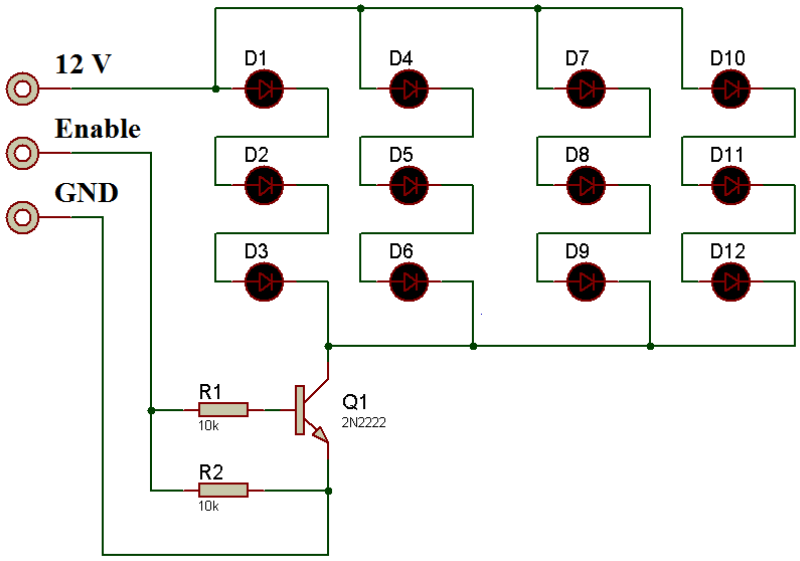

Figure 7.20: Circuit diagram for flash light

Chapter 8

Software Implementation of Micro-Controller nodes

Here we reached the final step in project, that is the implementation of low level control on Arduino boards to directly take actions on the robot. All programs, trials and results can be found as an open source provided here in this link:

https://github.com/AhmedAbdelbasit/ZUMAN-Robot/tree/master/Hardware

8.1 WiFi Remote Controller

The main task of this node as explained before is to enable the project user to guide the robot motion, camera orientation and flash light. Electronics used in this part are:

- NodeMCU
- Arduino Nano
- 5 Potentiometer
- Toggle switch

The reason for that choice of components it the the potentiometer need analog input pins to read. Arduino Nano has 8 analog-input pins so it is good choice. But the problem here is that we can not publish the commands on the network by it. We thought about attaching wifi module with the Nano kit but we found that Nano resources will be in unstable performance. So, we started to search for another solution and we find NodeMCU kit which opens the way for further upgrades and performance. This kit has many advantages compared with the Nano kit. It has higher processing speed, larger storage capacity and has an embedded wifi module. All these aspects make it a better

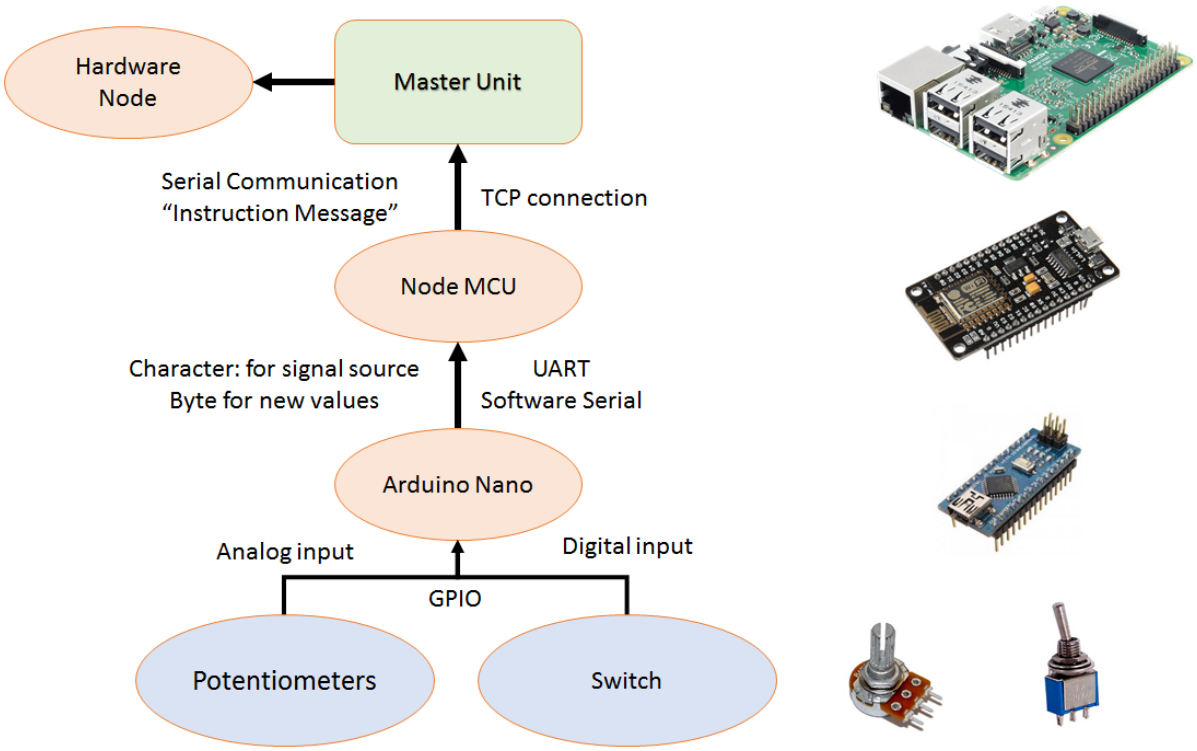

Figure 8.1: Hardware structure of Remote Controller Node

solution to replace the Nano kit. But there is one limitation that makes us use the Nano board with it. It has only one analog input pin. So, we established the following structure shown in figure 8.1 :

For network optimization and avoiding conflicts of unusable messages, both the Nano and NodeMCU are programed and tuned such that the message transmission takes places when there is a change in the potentiometer values which means that the user want to send one or more new values to the robot.
For communication between NodeMCU and Nano we uses the Software-Serial library. Messages consists of a character identify the command speed, camera orientation or flash light and a value mapped from potentiometer reading to the required range.
A video showing the operation of this node is here in this link:

https://www.youtube.com/watch?v=U6jN9-zoa5M

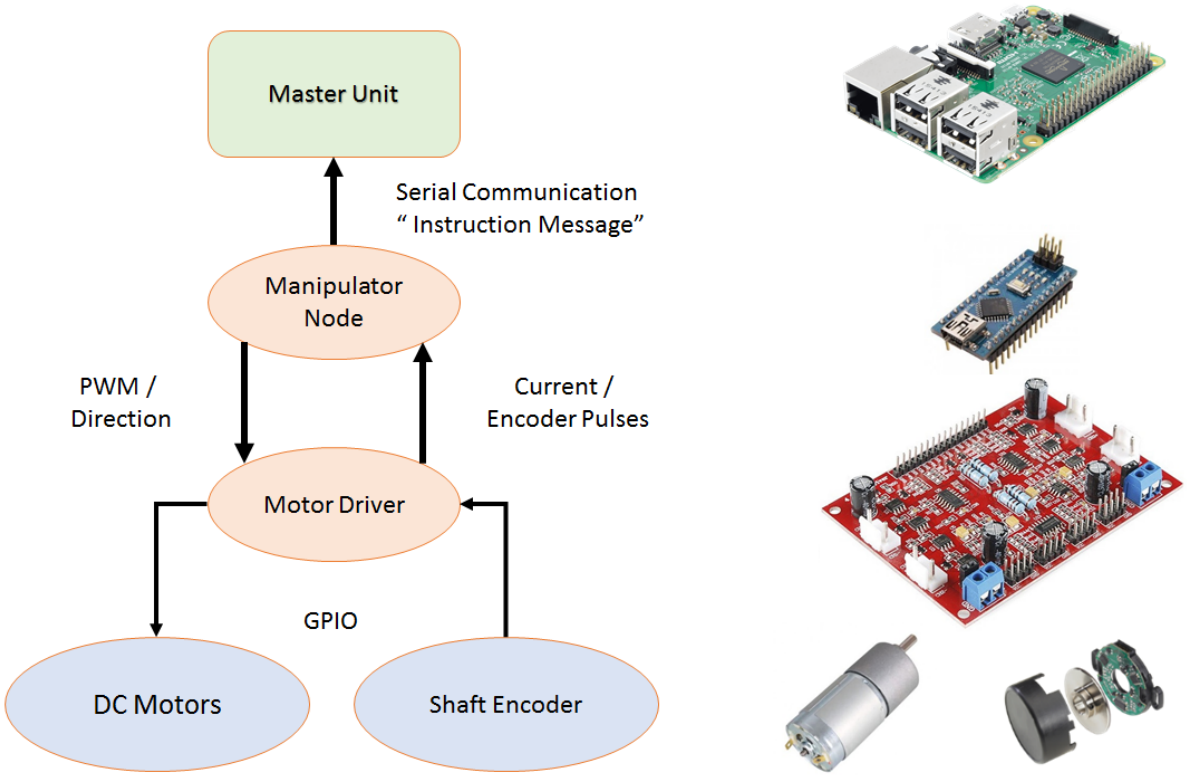

Figure 8.2: Hardware Structure of manipulator node

8.2 Manipulator Node

Here in this node, the control process takes place on the performance of robot motion. This node receives the low level instruction from the Hardware Node, initiates the controller and start the controlling loop and gives the suitable signals to the motor driver which in turn controls the DC motors. These signals are both PWM and Direction.
This node also has a feedback signal represented in the shaft encoder we talk about in a prior section. Another feedback available for this node is the current passed for each motor. This piece of information can be useful in the estimation of the load and actual speed that the motor rotates by.
The implementation of such controller and feedback system will be discussed in details next chapter.

8.3 Sensors Node

This node was designed to keep track of all sensors readings and make suitable filtering on them. So, compass, ultrasonic, MPU are attached to it. However, due to lack of experience about system modeling and the bad accuracy of sensors we could not fulfill

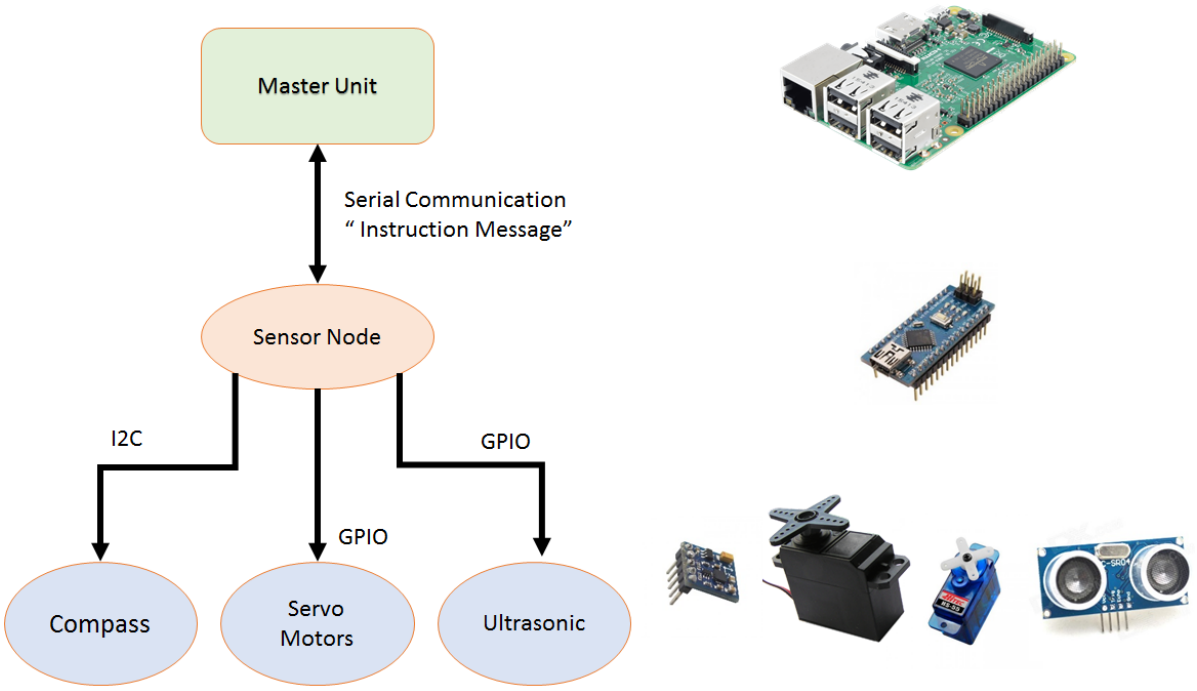

Figure 8.3: Hardware structure of sensor node

this step of design. but this node is still used for controlling the camera orientation by two servos attached to the robot frame in a manner to control both yaw and pitch rotation angles of camera. This node is visualized in figure 8.3.

Chapter 9

Design of PID Controller and feedback

9.1 About The Controlled Process

In our project the process here is the dc motor. The type we brought has a fast response time which makes the controlling process oscillatory. We did not spend much time trying to reach the exact model of such motor as we found out that the motor changes its behavior over short periods. Instead, we put a low pass filter midway between the motor and controller such that the over all response time for the open loop system become slightly slow and controllable by our limited processing resources. The result for this action was significantly good and enhanced the motion performance of robot.

9.2 PID Controller

9.2.1 Theoretical Background about PID

PID stands for Proportional-Integral-Derivative controller. It is a simple but effective digital controller and is commonly used in industrial processes. Each term contributes to rapidly correct the error in system.

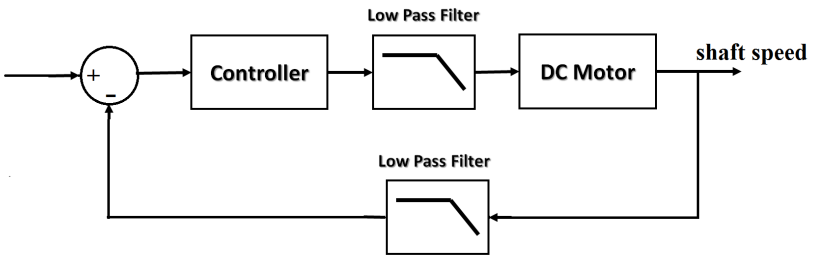

Figure 9.1: Low pass filter introduced before the DC motor

Proportional term works by multiplying the error by a factor causing the manipulated signal to be larger and so the plant process is given a higher signal to recover the error done in the output of system. Put the main side effect of increasing the P term is that it increases the overshoot of the system response. Moreover, in common digital microcontrollers there is a limit in output voltage which means that if the manipulated output exceeded this limit, the signal is clipped causing nonlinearity in the system.
The integral term works on eliminating the steady state error. It accumulates the whole past errors and multiply the accumulation by a factor to get the manipulated signal. But the bad side of the integral term in digital controllers is that it may cause oscillations around the steady state value because of the quantization error of digital signals. So, the accumulation reaches zero after many oscillations.
The derivative term can be used to overcome the problem of both Proportional and Integral terms. It works on the error change rate. So, if the error is recognized and the system going toward the steady state value, the derivative term decreases the manipulated signal so that it can fix the output just when it reaches the steady state for the first time without oscillations or overshooting. The case is called 'critical response'. Now, let's talk about the implementation of this type of controllers.

9.2.2 Practical implementation

The PID controller design basically depends on the overall required response of the system. For example, the system we try to implement is a controller for robot speed. So, our plant process here is the DC motor attached to each of robot sides. This DC motor has a transfer function which forms a relation between input signal 'voltage' and output 'speed of rotation'. So, want to implement a controller before the DC motor so that the overall transfer function of the system can meet out requirements like stability, no steady state error and low overshoot. But in our practical work we faced some problems in identifying the DC motor parameters and transfer function. So, we followed a practical algorithm to determine the values of P, I & D constants. This manual method for PID tuning is described as follow:

- Set Ki and Kd to zero.
- Increase Kp till the output oscillate.
- Then set Kp to approximately half of that value.
- Increase Ki until any offset is corrected in sufficient time.
- If required, increase Kd until the loop is acceptably quick to reaches its reference value.

9.2.3 Software Implementation of PID

Here is a flowchart describing the PID commands needed to be appended within the loop procedure of the micro-controller program.

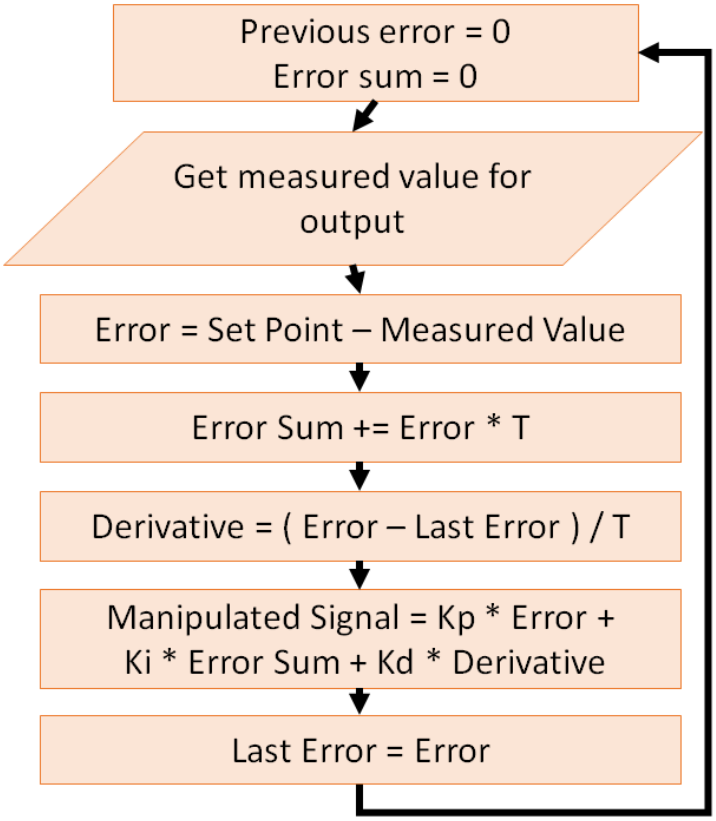

Figure 9.2: flowchart for the PID loop

9.3 Design of Feedback

As we know, the feedback is so critical in the closed loop system. This simply because having a wrong indication for system state leads to generating wrong manipulated signal which in turn makes the error increases more and more causing instability. So, we paid much attention about this part and tried many ways to fulfill an accurate feedback.
For our project the system state is represented as x, y positions and the heading angle of the robot and the final goal is to perform instructions come from other nodes with minimum error as possible. So we can encounter these tasks as follow:

- Robot must be able to move in a straight line
- Robot must know how far he moved from the start point.

- Robot must be able to rotate with a given angle

To accomplish these tasks we started to use the available sensors one after another until we got the reasonably acceptable performance that meets the needs of other nodes' instructions. Lets summarize our results now.
For measuring the distance that the robot moved by we used the shaft encoder. We managed to have both speed and position information from these pulses. Speed in rpm is measured as:

$$Speed(rpm) = \frac{60}{Time\ between\ two\ consecutive\ pulses \times 180}$$

This is because the the encoder we used generates 180 pulses for one revolution of wheel shaft. For measuring displacement these pulses are accumulated to get total pulses. then we can find the distance using this equation:

$$Distance = \frac{Total\ Pulses \times Wheel\ Circumference}{180}$$

for the rotation motion we used the differential drive relations and reached a relation that maps the angle to the number of pulses each encoder shout count to rotate by the required angle.

$$Num\ Of\ Pulses = \frac{105 \times Angle}{90}$$

All our tries with other sensors were not acceptable because of inaccuracy, the high sensitivity to external noise and low repeatability which makes the same input, at same place, by same controller has totally different responses.

Chapter 10

Mechanical Design

In this section we are about to take a close look on the design procedure of robot hardware both mechanical and electrical. The design stage of any application depends on the tasks and requirements it is supposed to fulfill. So, we will talk about functional requirements of project and how we select components to best suit them. Design engineers are usually constrained by one or more of these 5 concerns; time, money, knowledge, power and weight. So, there is no absolute good choice but we try to select the best one that meets the requirements and follow constraints. [1] Then we will get a brief view on frame design, materials used and how manufacturing operation done. After that we will get closer to the low level control and talk about electronic modules and sensors used, function, advantages and disadvantages of each. Finally we reach the controller design; how it was implemented, tuned and tested.

10.1 Choosing Suitable DC Motor [1]

In this project we introduce a robot for indoor navigation. So, in this case low speed is acceptable but at the same time it must be powerful enough for carrying and transporting most of parts we deal with at home or office. The requirements of robot were selected to be as follow:

- **Functional Requirements:**
  DC Gear head motor capable of accelerating a 20 Kg, four-wheel drive robot with wheel diameter of 6.5 cm at a rate of 1 m/s^2. Top speed required will be around 0.75 m/s. This speed is suitable as it reasonably approachs human walking speed.
- **Design Parameters:**
  Supplied Voltage = 12 V, Motor size limited to an overall diameter of approximately 4 cm and an overall length of not more than 10 cm (Less than robot frame width).

Here is the calculation steps based on the lecture notes referenced in section title:

- **Step One: Calculating Required Torque and rpm:**
  - **Required Torque:**
    $Force = Mass \times Acceleration$
    $F_{total} = ma = 20kg \times 1m/s^2 = 20N$
    $F_w = F_{total} \div NumberOfWheels = 20 \div 4 = 5N$
    $\tau = Fd = F_w \times WheelRadius = 5N \times 0.065m = 0.325Nm$
  - **Required rpm:**
    $WheelCircumference = C_w = \pi D = 3.142 \times 0.13m = 0.408m$
    $Speed = RPS \times C_w$
    $RPS = Speed \div C_w = 0.75 \div 0.408m = 1.838rps = 110.294rpm$
- **Step Two: Motor Selection to Meet the Requirements**
  After searching available electronics stores we got the most suitable motor for our project. Its model name is SG-555123000-30K shown in figure . From the data sheet provided for it find the following specs:[2]
  - Rated Voltage : 12 V
  - No load Speed : 100 rpm
  - Load torque speed : 73 rpm
  - Torque : 0.34 Nm

  By these specs we can conclude that robot features became as follow:
  - $Fullloadspeed = rpm \times C_w \div 60 = 73 \times 0.408 \div 60 = 0.4964m/s$
  - $Payload = \frac{\tau_w \times 4}{R_w \times a} = \frac{0.34 \times 4}{0.065 * 1} = 20.923kg$

This is still acceptable speed because the robot is not supposed to work in full load all time. Moreover the speed of about 0.5 m/s is suitable also for indoor navigation purposes.

10.2 Robot Dimensions and Design Considerations

Within the introduction section we figured out many problems and situations where robots are needed instead of man whether for safety issues or just it is impossible for man to do. So, in our design of robot frame we consider many points that lead to the frame presented. These points can be encountered as follow:

- Frame must be reasonably small so that the robot can path through narrow places man can do.

Figure 10.1: DC motor SG-555123000-30K selected for driving the robot.

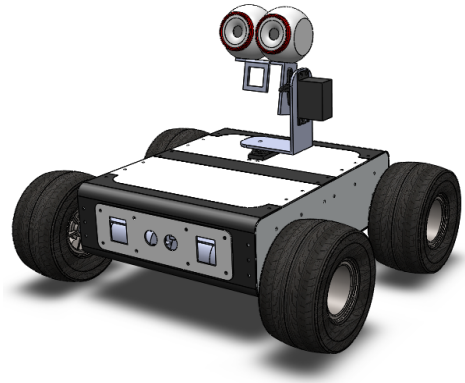

Figure 10.2: 3D Real view of robot frame

- Differential steering is to be used with four driven wheels.
- Distance between front and back wheels must be small enough to decrease friction forces the wheel exposed to during rotation motion in such differential steering.
- the overall frame style must be friendly so that people like to use
- The frame strength must be high enough to overcome any external shocks
- Contained parts must be attached well to frame and nothing may shake.

After more than one design plan we reached this frame shown in following figure and robot dimensions with different views are shown in figure 10.3.

10.3 Mechanical Parts

All solid work file for parts, assemble and dxf are available in this link: https://goo.gl/YEUib9

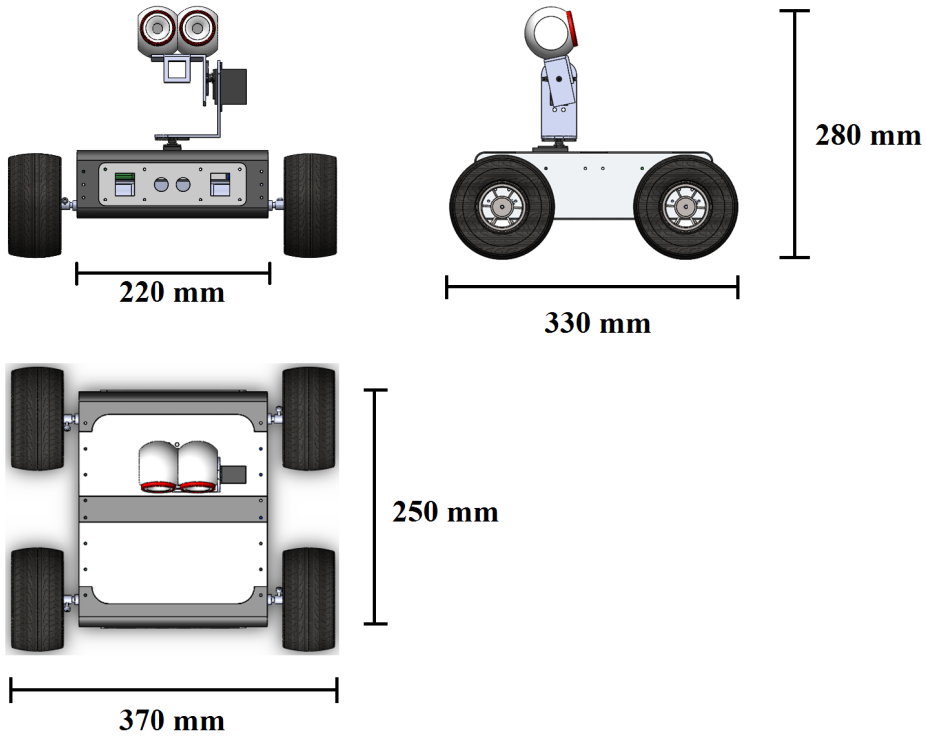

Figure 10.3: Front, side and top planes for assembly of robot frame

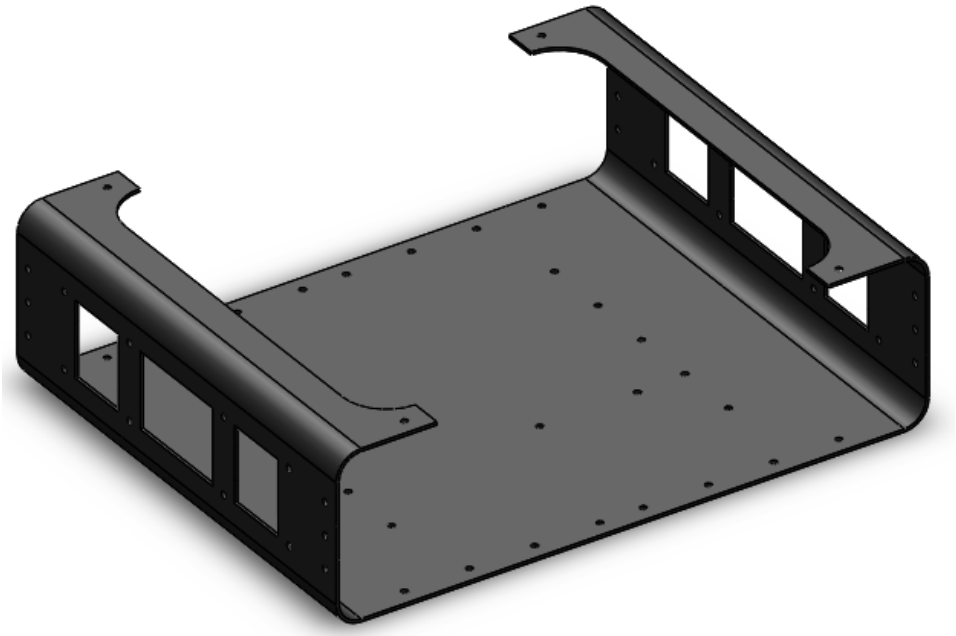

Figure 10.4: Aluminum Base frame

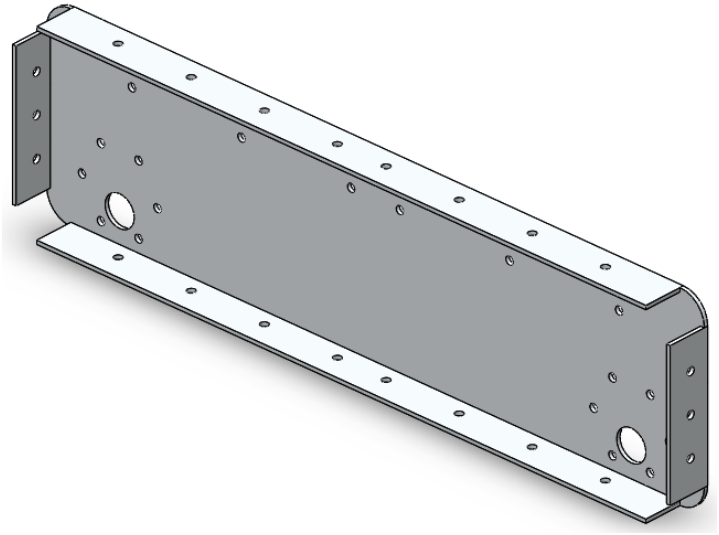

Figure 10.5: Side frame

Figure 10.6: Top front cover

Figure 10.7: Top mid link

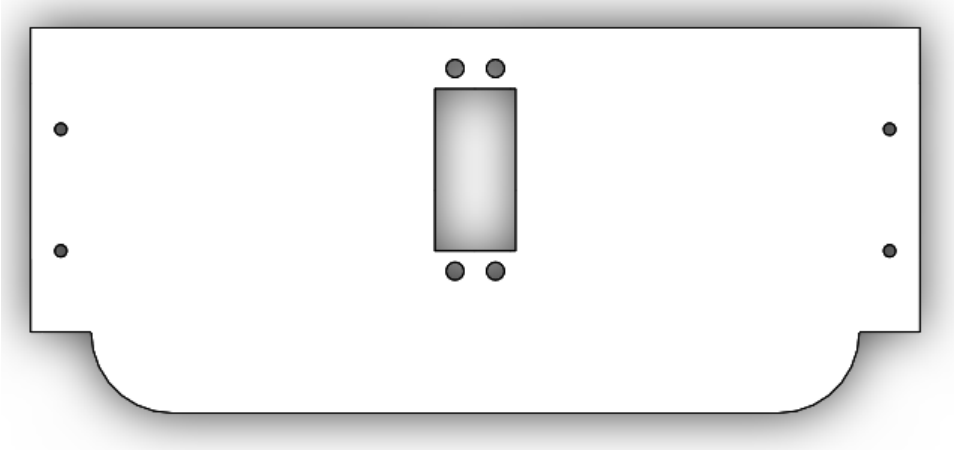

Figure 10.8: Top back cover

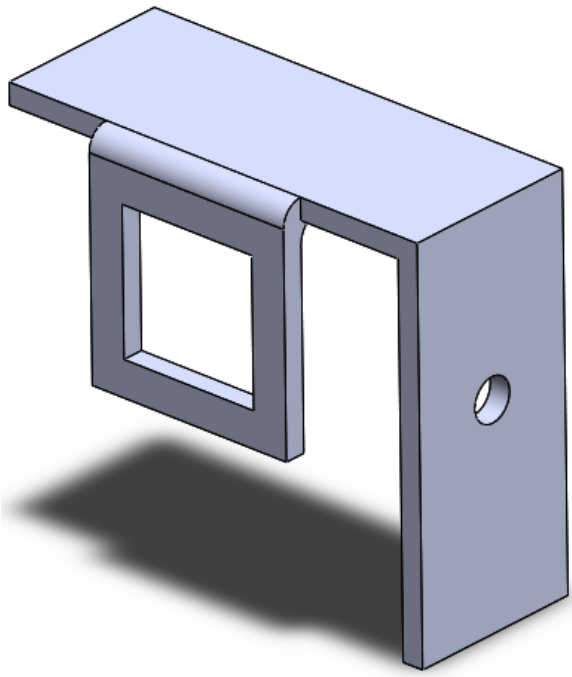

Figure 10.9: Upper Link of Camera Holder

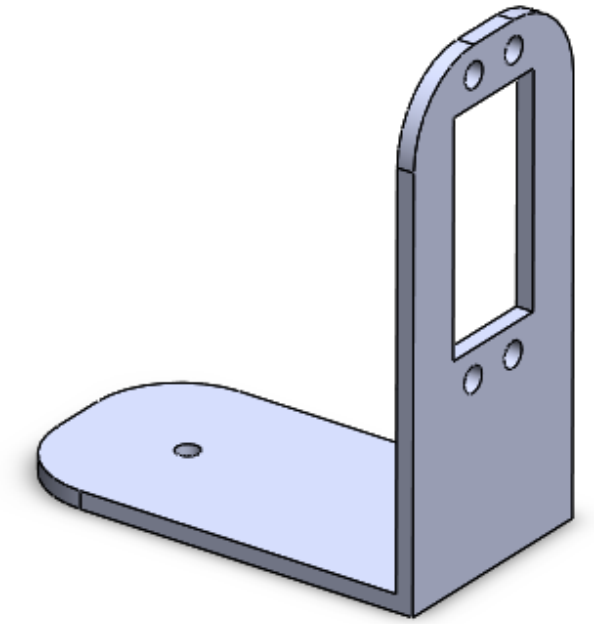

Figure 10.10: lower Link of Camera Holder

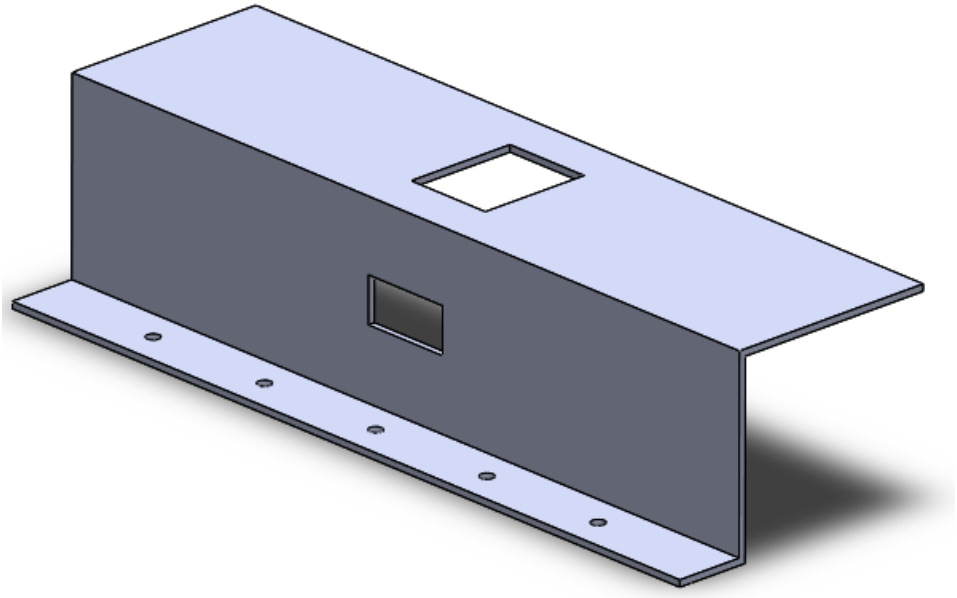

Figure 10.11: DC motors Cover

Chapter 11

Future Work

This project is considered as a long term one and another group will work on it as their graduation project. We plan to provide sessions to put them on the way and save their time from starting from the beginning.

First we will pay much attention to finish th over all algorithm and debug the problems appeared with us during our work. and then we will search for more accurate sensors to include in the hardware low level control.

Another field of upgrade is installing robot arms on it to start interacting with the environment. Moreover, Research papers will pe introduced in conferences and participate in competitions.

Bibliography

[1] M. Ahmed. Lecture notes understanding using dc motors specs. https://sites.google.com/site/mec304actuatorsanddrives/, Mar 2016.

[2] Ram shop, 32 El Falky St. Bab El Louk, El Tahreer, Cairo, Egypt. *DC Geared Motor "SG555123000-30K"*.

[3] ROS. About ros. http://www.ros.org/about-ros/, Mar 2017.

[4] R. wiki. rosserial arduino tutorials. http://wiki.ros.org/rosserial_arduino/Tutorials, Sep 2014.

[5] Wikipedia. Mobile robots. https://en.wikipedia.org/wiki/Mobile_robot, Jan 2015.