

# **Evi – THE RESCUE ASSISTING ROBOT**

## **DESIGN PROJECT- II REPORT**

*Submitted by*

<b>P SAI CHARAN</b>	<b>(17130029)</b>
<b>S GOWTHAM</b>	<b>(17130044)</b>
<b>M PARTHIBAN</b>	<b>(17130048)</b>
<b>R PRAVEEN</b>	<b>(17130071)</b>

**Under the guidance of**

**MS. N SEENU, M.E., (Ph. D.)**

*in partial fulfillment for the award of the degree of*

**Bachelor of Technology**

**in**

**MECHATRONICS ENGINEERING**



**DEPARTMENT OF MECHANICAL ENGINEERING SCHOOL  
OF MECHANICAL SCIENCES**

**HINDUSTAN INSTITUTE OF TECHNOLOGY AND SCIENCE PADUR 603  
103**

**APRIL 2020**



# HINDUSTAN

INSTITUTE OF TECHNOLOGY & SCIENCE  
(DEEMED TO BE UNIVERSITY)

## BONAFIDE CERTIFICATE

Certified that this Project Report titled “**Evi - THE RESCUE ASSISTING ROBOT**” is the bonafide work of **Mr. P SAI CHARAN (17130029), Mr. S GOWTHAM (17130044), Mr. M PARTHIBAN (17130048) , Mr. R PRAVEEN (17130071)** who carried out the project work under my supervision during the academic year 2019-2020.

### HEAD OF THE DEPARTMENT

Dr. D. Dinakaran, Ph.D.  
ANRO

### SUPERVISORS

Ms. N. Seenu , M.E.,(Ph.D.)  
ANRO

### INTERNAL EXAMINER

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

Designation:\_\_\_\_\_

### EXTERNAL EXAMINER

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

Designation:\_\_\_\_\_

Project Viva-Voce conducted on\_\_\_\_\_

## ACKNOWLEDGEMENT

We express our heartfelt thanks to the department faculties for their support, encouragement and cooperation for the completion of our project. We would like to thank **Dr D Dinakaran** (HOD) of Centre for Automation and Robotics for giving us an opportunity to work on our project. It's our pleasure to thank our mentors **Ms. N. Seenu** and **Dr. R.M. Kuppan Chetty** for spending their valuable time and sharing their knowledge for the development and completion of our project. We also thank our class teacher **Ms. Manju Mohan** for supporting us throughout the project.

P SAI CHARAN

S GOWTHAM

M PARTHIBAN

R PRAVEEN

## **ABSTRACT**

Ever since the past few decades, surveillance systems have played an important role in acquiring intel from widely spread as well as physically in-accessible areas for statistical analysis and trouble shooting to perform necessary modifications to the required system. Researchers and various technical enthusiasts have dedicated towards the development of surveillance systems. The usage of manually operated surveillance system has been transitioned to automatic surveillance, by equipping the robots with a set of capabilities to self-analyse an unknown environment, to generate a real time map of it, to undertake path planning across it and ultimately, to perform autonomous navigation within the generated map. Such autonomous systems, can act as a means for rescue-assisting operations that are carried out by the disaster management sector. This project focusses on the design and development of a rescue-assisting robot capable of path planning and autonomous navigation.

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	Pg.no
	ABSTRACT	
	LIST OF FIGURES	
	LIST OF TABLES	
1.	<b>OVERVIEW OF THE PROJECT</b>	
	1.1 INTRODUCTION	11
	1.2 PROBLEM STATEMENT	12
	1.3 SCOPE OF THE PROJECT	13
	1.4 OBJECTIVE OF THE PROJECT	13
2.	<b>LITERATURE SURVEY</b>	
	2.1 MAP GENERATION USING A SLAM-BASED ROBOT	14
	2.2 SENSOR COMPARISON FOR REAL-TIME SLAM APPLICATIONS	15
	2.3 APPLICATION OF ODOMETRY TO IMPROVE LOCALIZATION	15
	2.4 SUMMARY OF LITERATURE SURVEY	16
3.	<b>EVI : THE RESCUE ASSISTING ROBOT</b>	
	3.1 OVERVIEW OF THE ROBOT	20
	3.2 EVI'S ASSISTANCE TOWARDS RESCUE OPERATIONS	20
	3.3 BLOCK DIAGRAM OF EVI	21
	3.4 ACQUISITION OF IMU/MPU AND ENCODER VALUES	22
4.	<b>COMPONENT DESCRIPTION</b>	
	4.1 COMPONENT SELECTION	24
	4.2 HARDWARE COMPONENTS	25
	4.2.1 RP LIDAR A1M8	25
	4.2.2 L298N MOTOR DRIVER	27
	4.2.3 LITHIUM POLYMER BATTERY PACK	28
	4.2.4 ENCODER GEAR MOTOR	29
	4.2.5 TEENSY MICROCONTROLLER	30
	4.2.6 RASPBERRY PI 3	31
	4.2.7 MPU 9250 ACCELEROMETER	32

4.3	SOFTWARE USED	32
4.3.1	ROBOTIC OPERATING SYSTEM	32
4.3.2	LINUX OS	35
4.3.3	RQT SOFTWARE FRAMEWORK	35
4.3.4	TELEOP_TWIST_KEYBOARD	36
5.	<b>HARDWARE INTEGRATION AND ITS USAGE</b>	
5.1	BLOCK DIAGRAM OF HARDWARE SETUP	38
5.2	HARDWARE OUTLOOK OF EVI ROBOT	40
6.	<b>SOFTWARE INTEGRATION AND METHODOLOGY</b>	
6.1	CONTROL FLOW DURING SOFTWARE INTEGRATION	41
6.2	DECLARATION OF PORTS	42
6.3	MAKING LOCAL SERVER:	42
6.3.1	RASPBERRY ROS SERVER:	43
6.3.2	PC ROS SERVER:	
6.4	SELECTION OF DRIVER AND MOTOR SPECIFICATIONS TO TEENSY	43
6.5	CALIBRATING IMU/ACCELEROMETER	45
6.6	CONFIGURING OF PID	45
6.7	ODOMETRY CHECK FOR ROBOT	46
6.8	ACQUIRING LASER SIGNAL FROM LIDAR	46
7.	<b>IMPROVED SLAM (SIMULTANEOUS LOCALIZATION AND MAPPING)</b>	
7.1	MAP GENERATED WITH SLAM METHODOLOGY	48
7.2	MAP GENERATED WITH ENHANCED G-MAPPING METHOD	49
7.3	PSEUDO CODE	50
7.4	APPLICATION OF SLAM TECHNOLOGY AND LIDAR	52
7.5	DIFFERENCE BETWEEN CONVENTIONAL AND TRI-LAYERED GMAPPING	53

8.	<b>IMPROVED NAVIGATION BASED ON AMCL (ADAPTIVE MONTE CARLO LOCALIZATION)</b>	
	8.1 WORKING PRINCIPLE OF IMPROVED NAVIGATION SYSTEM	54
	8.2 FUNCTION AND WORKING OF IMPROVED NAVIGATION	55
9.	<b>EXPERIMENTAL RESULTS</b>	
	9.1 OVERVIEW	57
	9.2 CREATING A LOCAL SERVER USING LAUNCH FILE	59
	9.3 POSE AND TWIST OUTPUT	60
	9.4 ODOMETRY OF THE ROBOT	60
	9.5 IMPROVED G-MAPPING GENERATED MAP OUTPUT	61
10.	<b>CONCLUSION</b>	64
11.	<b>FUTURE WORK</b>	65
	<b>REFERENCES</b>	66

## List of Figures

<b>Serial No.</b>	<b>Title of the figure</b>	<b>Page No.</b>
1.	Block diagram depicting the control flow of Evi robot.	21
2.	Process of acquiring IMU/MPU and encoder values to determine motion of the robot.	23
3.	RP LIDAR A1M8	25
4.	L298N motor driver	27
5.	Lithium Polymer battery pack	28
6.	Encoder gear motor	30
7.	Teensy microcontroller	30
8.	Raspberry Pi 3	31
9.	MPU accelerometer	33
10.	The fundamental control flow in ROS	34
11.	List of fundamental ROS commands	35
12.	Steps to work with a teleop_twist_keyboard	37
13.	Control flow of hardware setup	38
14.	Microcontroller interface with MPU 9250 accelerometer and L298N motor drivers	39
15.	The Complete Hardware Assembly of Evi	40
16.	Software Integration Block Diagram	41
17.	Assigning Teensy port to the launch file	42
18.	Assigning LIDAR port to the launch file	42
19.	Selection of drivers, accelerometer and type of robot	43
20.	Assigning motor and robot specifications	43
21.	Three-axis calibration in C++ library	43
22.	Final accelerometer values after calibration	44
23.	RQT application is used to tune PID	45



24.	Ros launch code for acquiring LIDAR values	46
25.	Acquired laser signal	47
26.	General Gmapping	48
27.	Three-layered Gmapping	49
28.	Code involved in SLAM technology	51
29.	Electronic components used in autonomous car	52
30.	Working principle of improved navigation system	54
31.	Launch file of navigation stack	55
32.	Hardware output	56
33.	Output of MatPlot from rqt framework for tuning PID	58
34.	Output of encoder motor from launch file for tuning PID	59
35.	Result of launch file used to create a local server	59
36.	Result of Pose and Twist of the robot	60
37.	Result for odometry of the robot	61
38.	Output of RP LIDAR laser signal	62
39.	Output of the Improved SLAM with navigation stack	63

## List of Tables

<b>Serial No.</b>	<b>Title of the table</b>	<b>Page No.</b>
1.	Summary of literature review	16
2.	Components description	24
3.	Basic command instruction set used in ROS	35
3.	Difference between conventional and Tri-layered G-mapping	53

# **CHAPTER 1**

## **OVERVIEW OF THE PROJECT**

### **1.1 INTRODUCTION**

There have been various surveying methods incorporated in industry-based surveillance systems as well as for individual needs during the past two decades. Surveillance has played a key role in tracking an object's movement in space with the help of vector algebra. We have essentially grown in need to track people's movements in the recent years. In order to chart a path for people to journey across several places and to navigate people during the journey, people have devised scientific means that involved tracking equipment that employed various sensors to provide real-time data of a moving object. The data obtained in databases were stored in redundant files in the form of earth's co-ordinates such as latitude(s) and longitude(s). Since the past decades, satellite communication has been put in use for navigation across long distances that can range within or beyond countries. This technological aspect has eliminated the use of tracking devices which were needed to be carried by people. The equipment was then replaced with cellular devices like smart phones, which have GPS (Global Positioning System) embedded within them. In the recent years, surveillance systems demand for the use of automated surveying robots in order to access smaller/riskier locations that are inaccessible to mankind. Since effective surveillance depicts the most accurate exploration operations carried out, it is required that every area has to be checked/explored within the area of interest to acquire useful intel. This notion calls for the need of man-made surveillance bots for wider exploration and to be capable of path planning in order to automatically move in/around an area. It requires object detection to avoid collision while it moves across various obstacles. A navigation system must be embedded in the robot so that the robot can know/identify its position within the generated map, so that once obstructed by an obstacle it cannot pass through or various unavoidable factors like terrain modifications, battery power loss etc., the navigation algorithm created and employed in the robot must allow it to reset by itself and resume its surveillance operation if the stopping factors have been eliminated. A storage unit is allocated to store the acquired/plotted map data so it can be used for future use once the terrain has been explored completely, or if one wishes to redo the process to detect minor changes. It has been statistically analysed during the recent decade that, delays occur during search and rescue operations in the disaster management department. Search and rescue operations involve operations that are undertaken in various terrains known to mankind. It is organized by a group of specialized personnel.

These operations can be undertaken in rough terrains similar to hilly areas, battlefield operations , operations carried out by intelligence agencies. The disaster management sector is considered to be of interest in this project.

## **1.2 PROBLEM STATEMENT**

The problem statements listed below are the problems faced in real time scenarios as well as the issues faced while implementing a set of techniques to overcome these problems.

- Disaster management agencies face various problems in real time, one among them is the delay caused while performing search and rescue operations, leaving human lives in danger.
- Few places within the area might have been left unchecked in traditional search and rescue operations.
- Conventional search and rescue operations cannot be completely relied upon due to its randomized nature, rather than being in systematic manner such Ops will tend to render ineffective.
- In most cases, collapsed buildings and other strongholds blocks the passages so the rescue teams were unable to push-in. Situations arise, wherein which firefighters cannot physically access certain locations to identify if people are trapped within them. In such cases employing an unmanned robot proves useful.
- Navigation of the robot tends to be a cumbersome task, while considering the implementation of PID (Proportional Integral Derivative) controller and odometry.
- Static mapping is comparatively easier to be performed, but it has very less reliability due to the lack of data that can be collected in this method. In this proposed methodology, dynamic mapping technique will require additional coding and tuning of the PID controller that has been put into use.
- Data transmission from the robot is a highly complex phenomenon.

### **1.3 SCOPE OF THE PROJECT**

- The SLAM technology incorporated into this project seeks to develop autonomous navigation and path planning in various vehicle sectors.
- Robots that work on line-following feature can expand its functionality to explore wider areas with increased mobility using this technology.
- Rescue assisting robots can be manufactured as a separate segment of products to be sold to various government and private organizations to ensure global safety in disaster management sector.

### **1.4 OBJECTIVE OF THE PROJECT**

- The primary objective of this project is to develop the existing grid-mapping methodology in a SLAM-based robot to improve the localization and other generated map characteristics in conventional mapping methods
- To improve the existing AMCL (Adaptive Monte Carlo Localisation) technique using a fine-tuned PID controller setup to perform effective navigation.
- The project aims to develop autonomous navigation and environment mapping methodology for search and rescue operations that operate in the disaster management sector.

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 MAP GENERATION USING A SLAM-BASED ROBOT**

The use of various types of LIDAR and their operation pertaining to acquire map data of a given area has been mentioned. In this selected research paper, LIDAR Lite- V2 has been used as the key component in their project. The above component is a fairly inexpensive and affordable LIDAR type, that was built for various short scale projects. It uses laser light as the sensing element with no rotating parts. This equipment led us come to a conclusion that, lite V-2 type is an effective instrument that can sense objects in front of it for a 30m scan range in order to calculate distances between the starting point and destination as well as noting the clearances between the obstacles that come across its line of sight in a single direction i.e, a straight line. In this paper, the algorithms used are based on matrices involving RFS (Random Finite Sets). It assigns the marked/covered areas as “True” and unmarked areas are detected to be “False”. The above software setup along with these algorithms are implemented in an economical hardware setup which consists of a two-wheeled mobile robot with a set of motor drivers controlling a set of NEMA17 Stepper motors , a 32-bit micro-controller and an OLED display. The use of wireless communication has also been applied into the hardware with a working frequency of 2.4GHz .The paper included various SLAM algorithm methods for obtaining map data. The listed types are HectorSLAM, LagoSLAM , CoreSLAM and Grid Mapping techniques. In this paper ,grid mapping technique has been used. The results showcased were 2-D mapping trajectory where true and estimated trajectories were split into color-coded travel lines to distinguish and identify. A slippage error mapping result has also been added. The future work of this project aims to replace the existing 2-D Lite-V2 LIDAR sensor with an advanced 3-D sensor and improving the existing hardware setup to carry bigger payloads and to employ autonomous control system.

## **2.2 SENSOR COMPARISON FOR REAL-TIME SLAM APPLICATIONS**

We have come across the various types of sensors that can be used along with the SLAM methodology. The different types of sensors such as RP LIDARS , Microsoft Kinect, etc have been broadly classified into various categories including ; cost , as in : inexpensive, affordable, unaffordable , each of their performance attributes : frequency , scan range , change in accuracy with respect to change in real time environmental factors , built-architecture consisting of either stationary /rotating parts in it and other essential factors like power requirements and weight of each sensor used. The results shown in this paper, are the maps generated by two different LIDAR sensors , the RP LIDAR and Microsoft Kinect. It is chiefly from this paper, we have come to know the vast overlay of LIDAR sensors used in various large scale and mini-projects and has given us an insight to choose the necessary sensor for our project's application.

## **2.3 APPLICATION OF ODOMETRY TO IMPROVE LOCALIZATION**

The use of RP LIDAR using SLAM methodology has been elaborately mentioned. An additional odometer has been used along with the LIDAR sensor to incorporate odometry in the project. It is done to fix the wobbling in motors while tracing a path through the obstacles in an area throughout the robot's course, effectively it is seen to trace a smoother curve than the former models. An RPF (Regular Particle Filter) has been used as the SLAM method's localization algorithm to improve the accuracy of regular SLAM-based results. Furthermore, traditional SLAM-based algorithms are compared with RPF-based algorithms to showcase the improved geometric characteristic localization in the latter approach.

Over fourteen papers in Google Scholar have been reviewed by the team, out of which we have found eight papers to be closely related and helpful for our project. These papers have given us an insight into the selection of components and methodology to incorporate into our project. The following tabular column summarizes our literature survey :

## 2.4 SUMMARY OF LITERATURE SURVEY

Serial No:	Title of the paper	Journal details	Description	Inference
1	RFS-SLAM Robot: An Experimental Platform for RFS Based Occupancy- Grid SLAM.	20th International Conference on Information Fusion Xi'an, China - July 10-13, 2017.	It's an experimental paper. Tfmini LIDAR has been used to create grid mapping .They used MATLAB for mapping. This does not include navigation and obstacle avoidance.	Cost-effective robotic platform implementing a robust RFS-based algorithm using occupancy-grid SLAM.
2	Sensor Comparison for a Real- Time SLAM Application.	International Journal of Information and Electronics Engineering, Vol. 8, No. 1, March 2018.	It's a simulation paper. They have created a map using RPLIDAR and Microsoft Kinect and there is not inclusive of navigation and obstacle avoidance.	Within this scope, mapping of an area of 16 square meters has been done. The information given by the Kinect sensor is used as laser data. Implementing navigation and path planning.



3	Research and Implementation of SLAM Based on LIDAR for Four-Wheeled Mobile robot	2018 the International Conference of Intelligent Robotic and Control Engineering	Its an experimental paper. Usage of hardware is mentioned but its design and working methodology isn't.	To make a robot capable of self-navigation using SLAM algorithms.
4.	An Approach to Restaurant Service Robot SLAM	Proceedings of the 2016 IEEE International Conference on Robotics and Biomimetics Qingdao, China, December 3-7, 2016	It is a simulation paper. A concept robot for restaurant service is presented and the paper is based on mathematical modelling. There is no mapping or navigation implemented.	Localization and navigation for restaurant service robot. This method uses depth camera and lowcost laser to instead of heavy and expensive laser sensors. The experiments taken in different environments indicate that this approach has good performance in complex indoor environments.

5	Autonomous Wheelchair Navigation in Unmapped Indoor Environments .	2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC).	It is an experimental model. They have used a robotic arm in an autonomous wheelchair. It uses RPLIDAR and ROS for mapping , but navigation is not applied.	The strength and novelty of the presented approach is the ability to navigate in an indoor environment without prior mapping or the need for external hardware for localization. Additionally, the proposed design approximately costs \$1,000 and consumes 100 watts of power (including the laptop), making it a low-cost, low-power solution.
6	Research on Indoor Robot SLAM of RBPF Improved with Geometrical Characteristic Localization	2017 29th Chinese Control And Decision Conference (CCDC)	It is an simulation. They used only algorithm no hardware is involved to create mapping. So there is no mapping and navigation.	Yet to create mapping and navigation in the simulation.

7	An Improved Serial Method for Mobile Robot SLAM.	2017 IEEE International Conference on Robotics and Biomimetics (ROBIO).	It is a simulation. Mathematical modelling is used instead of Rplidar they used Kinect is used in this research.	Implementing in the hardware and testing the algorithm.
8	A Fire Protection Robot System Based on SLAM Localization and Fire Source Identification.	2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC).	It is a simulation. Mathematical modelling is employed for SLAM and it uses image processing to identify fire.	It is effective to construct a picture on an indoor or relatively flat road surface, and the effect of building a picture on a non-flat road surface is general. In the future, the accuracy of the odometer can be improved.

Table 2.1 : Summary of literature review

## **CHAPTER 3**

### **EVI - THE RESCUE ASSISTING ROBOT**

#### **3.1 OVERVIEW OF THE ROBOT:**

Evi can be thought of an autonomous electric vehicle with minimalistic payload. This robot is basically a mobile robot whose actions are similar to the autonomous cars. Evi robot is fully a automated robot that functions environmental mapping and self-navigation. These kind of robots are majorly used in house cleaning purposes. An advanced level of this robot is SpotMini , was designed and produced by Boston Dynamics. Evi robot is capable of reaching its destination within 0.1 metre accuracy , that of a LIDAR (Light Detection and Ranging) .In order to reduce wobbling, PID (Proportional Integral Derivative) controller has been used. Evi is equipped with one of the best microcontrollers available in the market , Teensy 3.2 is comparatively better than the Arduino board for processing the laser and TC (Transfer Configuration) signal .We have used Raspberry Pi mini-computer version 3B+. Evi runs on ROS(Robotic Operating System) version: Kinetic in Ubuntu version: 16.04 Operating System , that runs with the help of Linux kernel in both Raspberry Pi as well as the developer computer. The developed robot in this project performs better environmental mapping and self navigation compared with the former published robots working on SLAM methodology due to the application of enhanced G-Mapping algorithims and the use of PID controller to trace smoother paths.The algorithms used , allows the LIDAR to perform three-layered mapping and improved navigation of AMCL (Adaptive Monte Carlo localization).

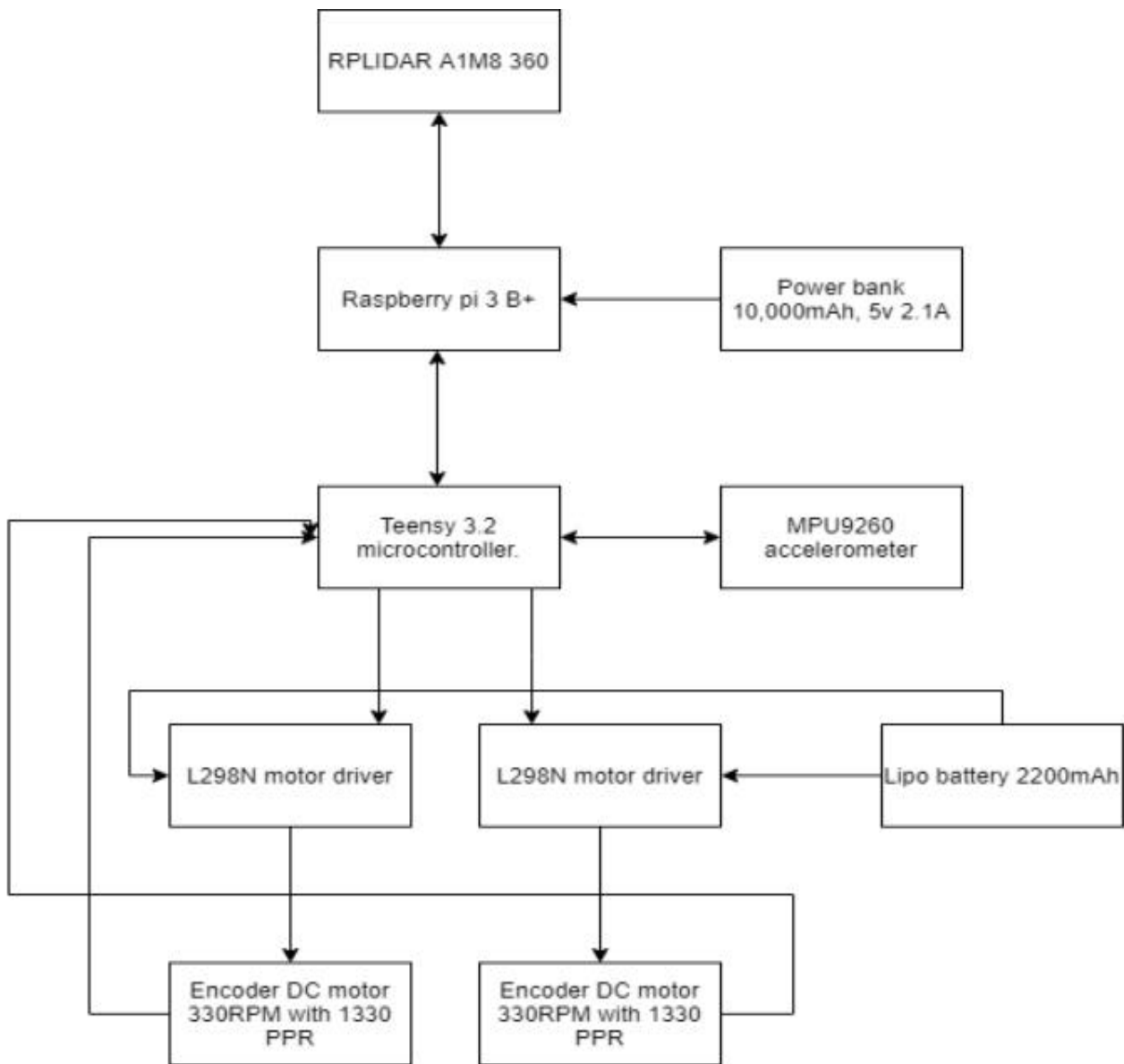
#### **3.2 EVI'S ASSISTANCE TOWARDS RESCUE OPERATIONS**

Conventional rescue operations carried out by searching for trapped humans in collapsed buildings.The search and rescue teams use infrared cameras to detect presence of trapped people in collapsed households , but deteriorating weather conditions such as heavy rains or thunderstorms would tamper with these instruments and live human exposure to these weather conditions puts the recue ops at risk . In such situations Evi can be used , it can be left on its own to move from a secured control room that holds visual systems to observe the plotted map and spots to detect trapped humans. Now-a-days rescue assisting operation is employed with mobile robots which are remotely controlled , these manually operated robots may collide or get stuck in the edges of obstacles in its path. To reduce the rescue time and to know the human location, this type of robot is used for rescue operation. Evi is capable of decision making. The navigation stack will know

robot configuration and will allow Evi to make decisions. More than two robots can be made to search at different locations simultaneously to reduce time. Furthermore, Evi is capable to pass through small gaps present within the collapsed household which humans are physically unable to access. In future, this robot will be equipped with Machine Learning to identify humans and to use AI chatbot.

### 3.3 BLOCK DIAGRAM OF EVI

The key component used in Evi is the RP LIDAR. All circuit components are interfaced to Raspberry Pi 3, it acts as a mini computer to store the generated map data and provide wireless communication in order to obtain remote desktop access to visualize the generated map.

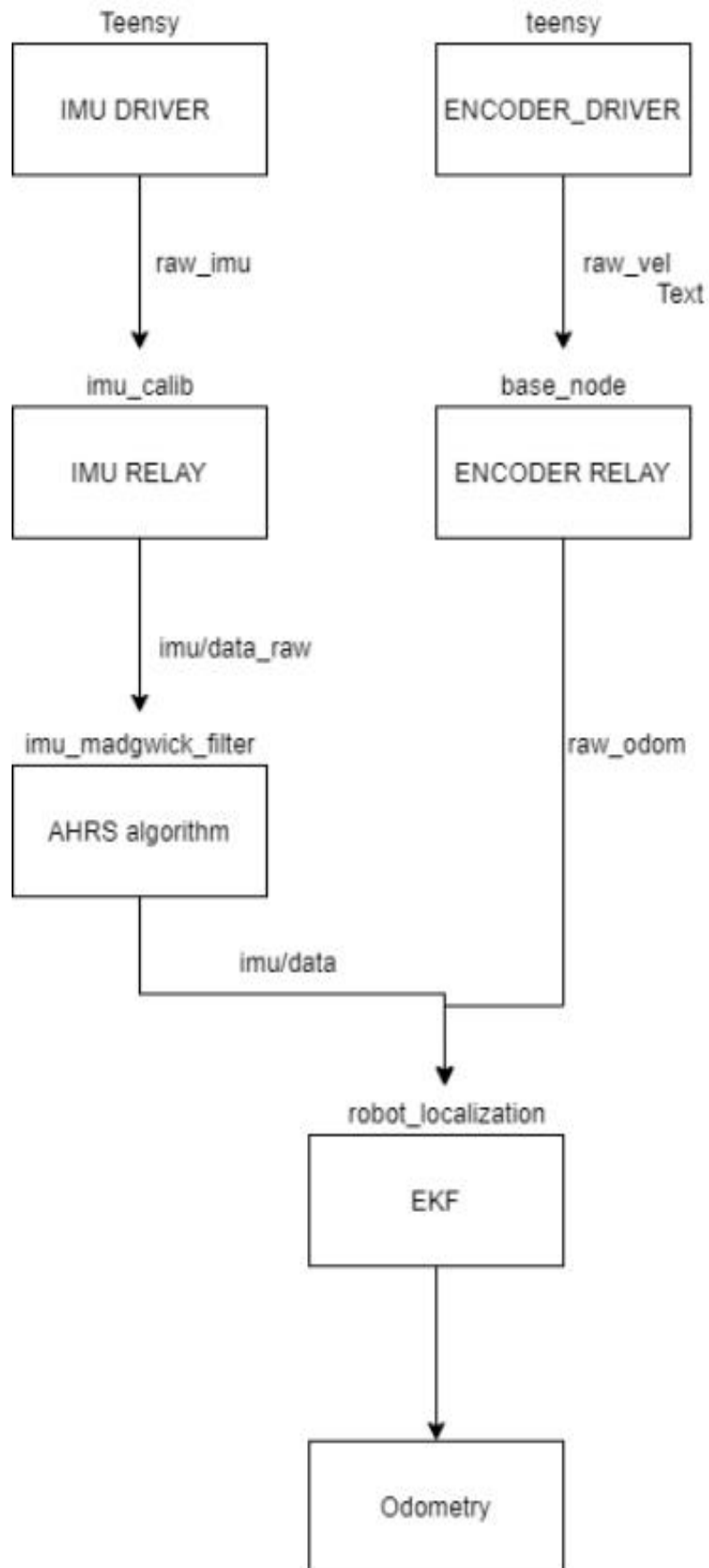


3.1 Block diagram shows the control flow of Evi robot.

### 3.4 ACQUISITION OF IMU/MPU AND ENCODER VALUES

The use of filters in any project depends on the nature of the system where it is being dealt in. If the system uses linearity principles based on time, the output expected from the system is not much of a complication as the user is aware that unknown factors are not present to influence such systems. One such example of a filter is a regular Kalman filter. This filter holds good with linear-systems that does not deal with much complication. In this project, the extended version of the existing Kalman filter is used, it is also preferred to be referred as EKF (Extended Kalman Filter). Since the SLAM-concept which includes autonomous path planning, localization and navigation involves various unknown variables that can change constantly with time, it becomes a system that should be able to cope up with such uncertainties. For such non-linear systems, EKF is used. To summarize this concept, as the name suggests, this concept is an extension of the existing Kalman filter to non-linear systems.

The IMU (Inertial Measurement Unit), is almost similar to an IMU sensor. Its sole purpose in this project is to report the feedback signals back into the virtual environment through the Raspberry Pi 3 about the force exerted on the object, in this case the LIDAR-setup and also the rate at which the object is being tilted, which can be obtained from the MPU 9250 accelerometer used. It is also used to notice how the setup is oriented overall/throughout, while it is in motion when it is being subjected to a flat/rough terrain that could be steep or might as well go downhill. The feedback signals from the accelerometer are given to the Raspberry Pi 3 that further enables the robot to adjust itself during its course. The figure below depicts the acquisition of IMU and encoder values to determine the motion of the robot (Fig 3.2).



3.2 Process of acquiring IMU/MPU and encoder values to determine motion of the robot.

## CHAPTER 4

### COMPONENTS DESCRIPTION

#### 4.1 COMPONENT SELECTION

The following components have been carefully selected based on the needs of the project after undertaking the literature survey. The components are distinguished and listed as hardware and software setups.

Hardware used	Software used
1. RP LIDAR A1M8	1. ROS (Robotic Operating System)
2. Lithium Polymer (Li-Po) Battery 2,200mAh	2. Ubuntu OS
3. Power bank 10,000mAh , 5V , 2.1A	3. Rqt software framework
4. L298N Motor driver (x 2)	4. Teleop_twist_keyboard
5. Encoder motor 330 rpm (x 2)	
6. Teensy Microcontroller	
7. Raspberry Pi 3	
8. MPU 9250 Accelerometer	



Table 4.1 Components used

## 4.2 HARDWARE COMPONENTS

### 4.2.1 RP LIDAR A1M8

RPLIDAR A1M8, is a 2D laser scanner which can rotate 360 degrees and can scan omnidirectionally. It is fairly inexpensive and is a much smaller version of the ones used in autonomous vehicles . The scanning range of this LIDAR is up to 6 metres. 2D points are used to point the co-ordinates which is used in mapping, localization and environmental modelling.

The frequency of scanning of this LIDAR is 5.5 Hz and goes up to 10 Hz. Its sampling points is about 8000 points per round .Its angular resolution is less than 0.5mm.Sampling duration, which is 0.125 milliseconds.This LIDAR uses laser triangulation and works excellently in indoor environments.



Fig 4.1 RP LIDAR A1M8

### 4.2.2 L298N MOTOR DRIVER

This motor driver works with H-bridge principle. In this manner stepper motors interfaced with this driver can change its polarities easily and effectively without delays.It consists of two ENABLE pins labelled as “A” and “B” and four logical pins with three power pins inclusive of the GND pin.This module is capable of controlling two DC motors as two sets of polarity reversing pins can be seen in blue color-coded sockets. Furthermore, L298N is cost-effective and widely used for speed and direction control in various mini projects and even for commercial purposes. The l298n Module has a specific high voltage Dual H-Bridge . It is designed to accept standard TTL

(Transistors-transistor logic) voltage levels. H-bridge drivers are used to drive inductive loads that requires forward and reverse motion with speed control like dc and stepper motor and it is manufactured by ST company.

#### HOW DOES L298N WORK:

The l298n is a dual H-Bridge motor driver which allows speed and direction control of two DC motor at the same time. The module is capable of drive DC motors hat have voltages between 5 to 35Voltage with a highest current up to ampere. This depends on the voltage used at the motors VCC.

#### CONTROLLING THE DC MOTOR:

To have a complete control over DC motor we have to control its speed and rotation direction. This can be achieved by combining these two techniques.

PWM – For controlling speed

H- BRIDGE – For controlling rotation direction

PWM( pulse width modulation):

The speed of a DC motor can be controlled by varying its input voltage. A common technique for doing this is to use PWM (Pulse Width Modulation).PWM is a process where average value of input voltage is adjusted by triggering a series of ON-OFF pulses.the average voltage is proportional to the width of the pulses called as duty cycles

#### H-BRIDGE:

The DC motor's rotating direction can be controlled by changing its polarity of its input voltage. A common process for doing this is to use an H-Bridge. An H-Bridge circuit has four switches with the motor at the center forming an H-like arrangement.

Closing two particular switches at the same time reverses the polarity of the voltage applied to the motor. This causes change in rotating direction of the motor.The driver weighs 26 grams and maximum power requirement is of 25Watts. It has an operating input voltage of 5-12 volts and will draw 0-36mA current from the given source.



Fig 4.2 L298N motor driver

### **4.2.3 LITHIUM POLYMER BATTERY PACK:**

Lithium Polymer Battery has 2200mAh capacity of power for supply. The model of the battery pack is Orange Lithium Polymer Pack 3S, it delivers full rated capacity of power as output. It is mainly used in Engineering projects as mini-projects and DC power supplies.

#### **USES OF LITHIUM POLYMER BATTERY:**

This type of battery has heavy duty discharge leads, it minimizes resistance and high current loads. Mainly it's stand up to punishing extremes of aerobatic flights and RC vehicles. IR matched cells are used in Lithium Polymer Battery while assembling it.

#### **PRECAUTIONS OF USING BATTERY:**

- o Do not short circuit terminals of battery.
- o Do not expose high temperature conditions.
- o Battery should not have contact with water.
- o Avoid over charging the battery or over discharging the battery.
- o Keep battery away from the kids and pets.
- o Check once or twice before connecting the battery to prototype.
- o When charging the battery connect the correct polarities properly.
  - Red = Positive (+)
  - Black = Negative (-)



Fig 4.3 Lithium Polymer (Li-Po) battery

#### CHARGING/DISCHARGING:

Every new battery comes with 30-60% battery pack it's depends on shipping. After getting the battery charge it fully before using it at 5C or less on LiPo setting. Do not charge LiPo pack on NIMH (nickel-metal hydride battery)/NICD (nickel-cadmium battery) charger. When not using LiPo battery try to maintain 60% of battery pack, At least use it once per month.

- o Don't keep battery in car or warm areas where the temperature is above 140F.
- o Before you dispose the battery discharge fully and throw it in dustbin.
- o Do not place the battery on metal tables or other areas.
- o If the electrolyte present in the battery touches your skin wash it thoroughly, especially if it has had contact with the eye rinse thoroughly with cool water and seek medical attention.

#### 4.2.4 ENCODER GEAR MOTOR

Two encoder gear motors have been used within the set of rear wheels in Evi .It is a permanent magnet motor which works on 12V DC supply and rotates upto 330rpm without any payload . Its rated load is 4.8kg.cm , with stalling torque of greater than or equal to 12.8 kg.cm

The feedback signal from the encoder present in the gear motor is fed back to the teensy microcontroller.

An encoder motor is basically an electro mechanical device which is use to gives an electrical signal which is used to control the speed and position of the motor and encoder which has a rotating and stationary circuit the rotor consist of a metal glass or may be a plastic that are fitted on the top of the encoder shaft ,the disc has a sort of optical pattern, that is electronically decoded to provide position information. This is a system which is use to referred as closed loop system. Encoder is usually convert the mechanical motion into electrical signal and then it is monitored by the control

system and then it makes the specific adjustment to maintain the machine operating in a desired motion. There are two types of motor encoder technology:

#### ABSOLUTE ENCODER:

The Absolute encoder has a coded disc, the construction of absolute encoder is complicated than the incremental encoder as they which need the decoder and advantage of this is that each angular position of the shaft can be noted at any time and though after a power failure ,that shows that there is no counter value to loose neither 1. counting the number of revolution with the multi turn absolute encoder, for counting the number revolution and internal reduction gear to know that it is connected to the drive shaft This shows that the output of encoder motor indicates the motion and the position of the motor shaft.

#### INCREMENTAL ENCODER:

Incremental encoders is a linear or rotatory electromechanical device which has two types of output signals A&B and Z and this Z has two type it is known as gated and ungated .Gated always depends on A phase 1-cycle OFF and UNGATED depends on A phase 1-cycle ON.It triggers a pulses for each incremental step and in this it has one transmitter and multiple receivers so encoder can generate a sequence of pulses and the output is measures the pulses per revolution which can be useful to determine rotational speed ,to detect the rotation direction it needs track A& track B in clockwise direction signal from track A leads and in counter clockwise direction signal from track B leads .This shows that the output of the encoder indicates that its control the speed of the motor shaft.



Fig 4.4 Encoder gear motor

#### 4.2.5 TEENSY MICROCONTROLLER

The Teensy controller has been selected to replace the arduino board in Evi. Its CPU architecture consists of ARM Cortex 32-bit , running on 72MHz with 256KB flash memory and 64KB RAM, consisting of a 3.3V voltage regulator. This microcontroller consists of 34 pins wherein which , 21 analog input pins are present along with 12 PWM ( Pulse Width Modulation) pins for output. Communication buses like I2C , SPI , CAN protocols are embedded within it. It is further compatible with micro USB cable to access bootloader.



Fig 4.5 Teensy microcontroller

Teensy 3.2 Development Board is 100% compatible for direct usage. We can use teensy 3.2 for all motor shields and add-on boards made for the previous version teensy 3.1 Teensy 3.2 is improved 3.3v regulator, to allow direct power ESP8266 Wifi. Teensy 3.2 comes pre-flashed with a bootloader it helps for program by using USB micro-B.

#### TEENSY MICROCONTROLLER OPERATION

It works with lower 3.3v signals, by the lower signals it become power-efficient. It can handle 5v signal by that it become flexible enough and inserted into any digital circuit. It comes with built-in DAC (Digital-to-Analog converter). Teensy work well with Sensors with large data output and refresh rate. Mainly Accelerometer, GPS, and Proximity sensor works easily with Teensy 3.2.

#### TEENSY 3.2 USER CONTROL

Teensy has 21 I/O pins with that you can control devices or read signals. These I/O pins are denoted with letter and number. Pins grouped into 8-bit port as letter, numbers denote individual bit. In controlling the OR and AND logical operations used together.

#### MEMORY UNIT:

It has the large memory type for projects 256K Flash Memory with 64K RAM. Teensy has 2K EEPROM type memory future. 64K allows for more advanced function in Teensy 3.2.

### 4.2.6 RASPBERRY PI 3

Raspberry Pi is incorporated in Evi as a mini-computer to organize the acquired map data from the RP LIDAR into a simulated form. In this way the user can see the map of an environment getting generated in real-time. It essentially acts as a storage unit for every set of samples collected by the LIDAR.



Fig 4.6 Raspberry Pi 3

The Raspberry Pi Model B is the newest raspberry Pi computer made, and the Pi Foundation knows you can always make a good thing better. It is the latest product in the Raspberry pi range, boasting an updated 64-bit quad core processor running at 1.4GHz with built in metal heatsink, USB 3 ports, dual-band 2.4Hz and 5GHz wireless LAN, faster Ethernet.

#### RASPBERRY Pi 4 KEY FEATURES:

- It has a high performance 64-bit Quad-core Processor.
- It has got dual-display support 4K resolutions.
- Hardware videos can be decoded up to 64KB.
- It contains up to 4GB of RAM.
- Dual-bands 2.4GHz or 5.0 GHz wireless Local Area Network.
- It supports Bluetooth 5.0 feature.
- Gigabit Ethernet portability.
- 2 USB 3.0 ports, 2 USB 2.0 ports.
- PoE capability.
- Fast and Reliable as compared to Raspberry Pi 4.

#### RASPBERRY PI 3 BENEFITS:

With 3GB RAM, the pi 3 no longer struggles with heavy web pages and apps, and is able to switch between full online services and java-Script sites without lagging. Raspberry Pi official OS has the chromium browser that chrome is based on. Raspberry Pi 4 can be used as a media center. Raspberry Pi 3 can run in Windows 10, Ubuntu and even in classic console games.

## APPLICATIONS OF RASPBERRY PI3

- Build your own Raspberry Pi computer
- Film your own Stop Motion Film
- Build your own Pi web Server
- Raspberry Pi Home Security System
- Home Automation System with Pi
- Build a virtual jukebox
- Create a Social Media Bot

Its overall specification sheet includes that: It has 3 Giga bytes of RAM with a quad-core processor clocked at 1.4 GHz in a 64-bit ARM Cortex architecture. It is powered with 5V supply and an easy-to-access micro USB cable. This module is further used for wireless communication to provide remote desktop access for the user. The simulation is not fixed to a single computer, any developer computer with usage access and containing ROS with Rviz library files can view, generate maps and visualize navigation in real time.

### **4.2.7 MPU9250 ACCELEROMETER**

This module is a combination of acceleration , gyroscope and magnetometer. It has an operating voltage from 3.3-5V and uses DC power supply. Its dimensions are 22mmx17mm , hence it is easily compactible within the chassis. It has 9 D.O.F (Degrees of Freedom) modules for tri-axial gyroscope and tri-axial accelerometer to incorporate 3-axis magnetic fields. It supports I2C interface communication protocol and has much higher vibration sensitivity compared to its predecessors.





Fig 4.7 MPU accelerometer

## 4.3 SOFTWARE USED

### 4.3.1 ROBOTIC OPERATING SYSTEM

The Robotic Operating System , often referred to as ROS is not a typical Windows-based OS or Linux OS , but it is a virtual environment that does have a set of user-defined operating features that enables it to perform activities similar to that of any other existing operating system. The main features of ROS is inclusive of data acquisition , information passing capabilities , device control , etc. ROS is an open source software that has been commercially used across the globe in various core-related software products. ROS supports a plethora of plug-ins , nodes and other softwares that can be programmed in C++ , Python , etc . Activities like acquiring output from sensors or observing the real-time performance of a servo system can also be performed in ROS. ROS allows communication between various nodes. A node can hold specific information while another node can “subscribe” to the former to obtain that information. There are two existing version of ROS available namely, ROS Melodic Morenia and the ROS Kinetic Kame. The former was published as the recent release , whereas the latter is the 10<sup>th</sup> official version of ROS and is used excessively. As ROS is not a real time software framework ,though it can be integrated with programming codes of various languages.

Various features of ROS framework are listed below:

- ROS is highly compatible with programming codes that are used in any other robot software. Thus, software integration is made easier.
- ROS libraries , can be accessed to perform operation on various robotic models with easier functionality.
- Programming language feasibility , the set of plug-ins can support the mainly used in the current programming languages like C++ , Python , etc.

- ROS provides a robust testing environment which is a built-in feature of the ROS Kinetic Kame version used in the project. Due to this feature virtual testing can be carried out instead of hardware prototyping. This is a cost-effective method widely used for testing purposes.

In this project , ROS Kinetic Kame has been used to create a node to control the encoder gear motor which has been subscribed to a topic with `_absolute` file name. Another node is created to publish the first topic and to subscribe to a `/cmd_vel` topic that comes through the `keyboard_node` which has a separate explanation dedicated to it.

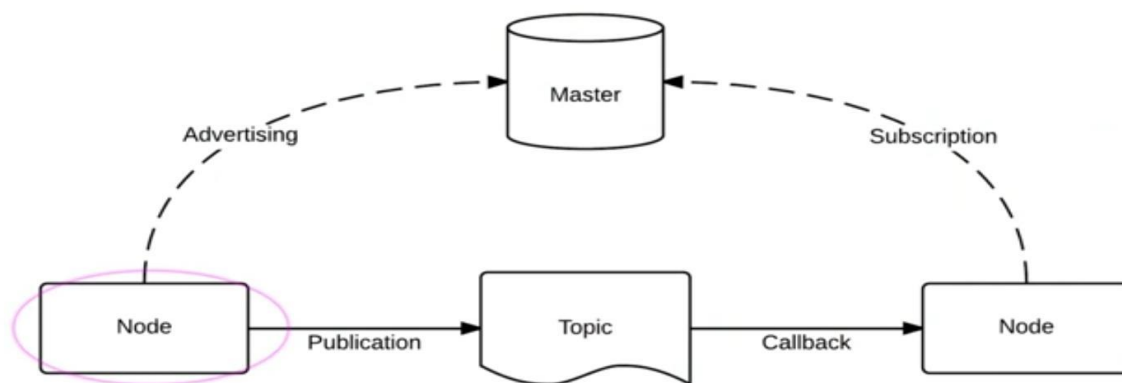


Fig 4.8 The fundamental control flow in ROS

ROS provides excessive command statements in its library. In order to publish a topic the syntax followed is : `rostopic pub / (Name of the topic)` and TAB key has to be pressed twice. ROS will start compiling and the direction of steering can be set manually.

The following list contains the basic ROS commands that have been used in this project:

Command	Action	Example usage and subcommand examples
<code>roscore</code>	This starts the Master	<code>\$ roscore</code>
<code>rostrum</code>	This runs an executable program and creates nodes	<code>\$ rostrum [package name] [executable name]</code>
<code>ronde</code>	This shows information about nodes and lists the active nodes	<code>\$ ronde info [node name]</code> <code>\$ ronde &lt;subcommand&gt;</code> Subcommand: <code>list</code>
<code>restock</code>	This shows information about ROS topics	<code>\$ restock &lt;subcommand&gt; &lt;topic name&gt;</code> Subcommands: <code>echo</code> , <code>info</code> , and <code>type</code>
<code>rosmg</code>	This shows information about the message types	<code>\$ rosmg &lt;subcommand&gt; [package name]/[message type]</code> Subcommands: <code>show</code> , <code>type</code> , and <code>list</code>
<code>rosservice</code>	This displays the runtime information about various services and allows the display of messages being sent to a topic	<code>\$ rosservice &lt;subcommand&gt; [service name]</code> Subcommands: <code>rags</code> , <code>call</code> , <code>find</code> , <code>info</code> , <code>list</code> , and <code>type</code>
<code>rosparm</code>	This is used to get and set parameters (data) used by nodes	<code>\$ rosparm &lt;subcommand&gt; [parameter]</code> Subcommands: <code>get</code> , <code>set</code> , <code>list</code> , and <code>delete</code>

Fig 4.9 List of fundamental ROS commands

The development environment can be sourced by the following syntax:

Source devel/setup.bash.

In order to access the details of a topic the following command can be used , its syntax is given by `:rostopic info / (Name of the topic)`. Using this basic command one can observe if a controller is subscribed to a topic. All the topics created can be seen by listing it using the command `rostopic list`.

### 4.3.2 LINUX OS

An OS(Operating system) , is basically a software that can run a computer. Linux is one such OS , similar to the well known WindowsOS and the Mac OS(Mackintosh). It consists of a file explorer to access pictures , videos and other media content and an internet browsing software like Mozilla Firefox which in compatible with all the OS platforms , in this way , it is similar to the other well known OS currently available. Linux is further officially known as the GNU/Linux. There are few key differences between Linux and MacOS.

- Freedom of usage: Linux is an open-source software, meaning that any user can customize the operating system or the user interface of the desktop and other applications however they prefer. This reason enables people to prefer Linux OS as the base platform to perform various simulations in virtual environments.
- Cost factor : The users tend to consider the cost of purchasing an OS along with its features. Though WindowsOS and MacOS can fulfil the needs of some people . There are others that require user-access to alter the OS in order to delete the features that annoy them most or the include features they prefer .Linux OS is cost-effective when compared with the above mentioned OS platforms , as most Linux distributions are completely free of cost. The additional softwares that are meant for Linux OS is also free to use eg ; offline games , photo or video editors, graphics design etc.

### **4.3.3 RQT FRAMEWORK**

Frameworks are required in ROS for GUI (Graphical User Interface) development. The rqt framework consists of various packages within it . It consists of a set of tools related to the graphical user interface in the form of plug-ins . The command syntax used to achieve access of an rqt is given by : `$ rqt` . The rqt is used to perform certain operations during the robot's runtime. One of which is used to control the runtime of the robot called `rqt_common_plugins`. Another package is used for interacting between various components within the robot's mainframe during program compilation during the bot's runtime , it is known as `rqt_robot_plugins`. It was able to be used in ROS Jade version upto the year 2013, after that the version was discontinued. Currently it can be used in ROS Kinetic version . This framework open various possibilities for users to change the GUI for better user-customization as multiple widgets can be projected in a single window like smartphones instead of opening them in separate windows. This rqt framework can be installed in Ubuntu OS and MacOS as well. Plug-ins for rqt framework can easily be created using C++ language.

### **4.3.4 TELEOP\_TWIST\_KEYBOARD**

Teleop\_twist\_keyboard is a generic term since it used as a virtual keyboard. The purpose of teleop\_twist\_keyboard in this project is to modify the linear and angular speeds of the rear end wheels , by integrating with the physical keyboard in a computer so that the commands can be given as key-inputs from the developer's computer. The below figure contains the list of commands that allow the movement of the robot with the specified keyboard inputs.

# 1. Installing

```
sudo apt-get install ros-melodic-teleop-twist-keyboard
```

# 2. Running

```
roslaunch teleop_twist_keyboard teleop_twist_keyboard.py
```

# 3. Controls

See the on-screen instructions:

```
Reading from the keyboard and Publishing to Twist!
-----
Moving around:
   u   i   o
   j   k   l
   m   ,   .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
anything else : stop

CTRL-C to quit
```

Fig 4.8 Steps to work with a teleop\_twist\_keyboard

## CHAPTER 5

### HARDWARE INTEGRATION

#### 5.1 BLOCK DIAGRAM OF HARDWARE SETUP

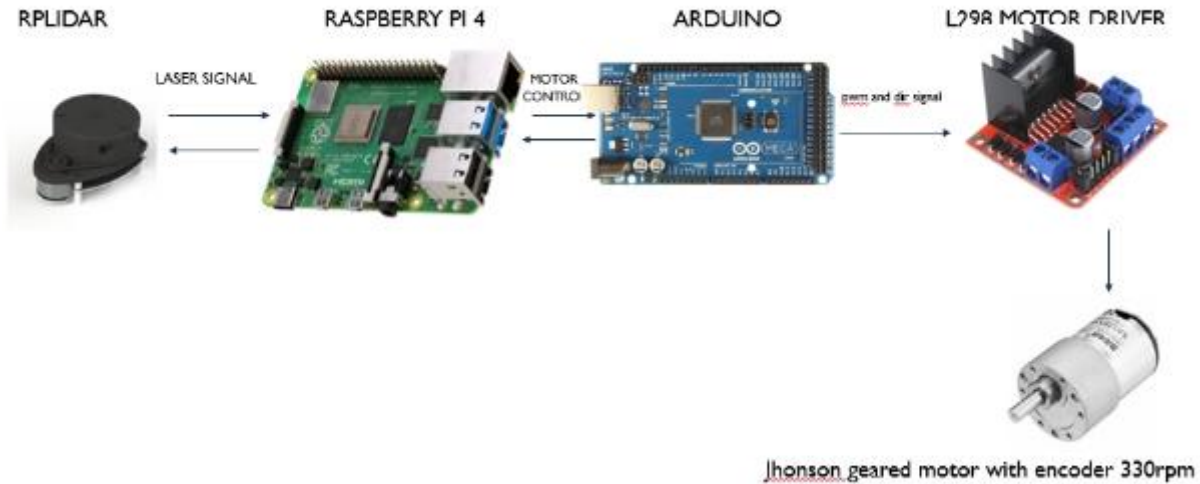


Fig 5.1 Control flow of hardware setup

Raspberry Pi 3b+ is used to process the LIDAR data and acquire the sensor values of the Teensy microcontroller. Raspberry Pi is equipped with ROS (Robotic Operating System) which runs on Ubuntu which requires the Linux kernel. Every I/O devices connected to the Raspberry Pi will have a port for Teensy and the LIDAR. The ports `/dev/ttyACM0` and `/dev/ttyUSB0` are normally present in the Raspberry Pi lead to issues in read/write operations. In order for I/O devices to overcome that, we have used special read write permission – `sudo chmod 666 /dev/ttyACM0` and `sudo chmod 666 /dev/ttyUSB0`. In this way, we will be able to transfer or receive data from I/O devices. Raspberry Pi is not so efficient to visualize the mapping so we have created a local server with `ROS_MASTER_URI` which is the accessible IP addresses and ports for the Raspberry and for visualizing the mapping PC with minimum spec of i3 processor, 2Gb RAM with ROS, used. For PC `ROS_MASTER_URI` raspberry's IP and port is given. For transmitting, PC should notify the IP to the Raspberry. For that purpose, `ROS_HOSTNAME` is given to both the computers. Teensy provides a much efficient approach compared with Arduino. The reason behind it is that, the MPU9250 accelerometer and encoder to the Raspberry because Raspberry will be consuming a lot of RAM while processing the LIDAR and the navigation stack. Teensy is used to control motors, configuring PID, receive accelerometer signal, and even to receive the encoder values.

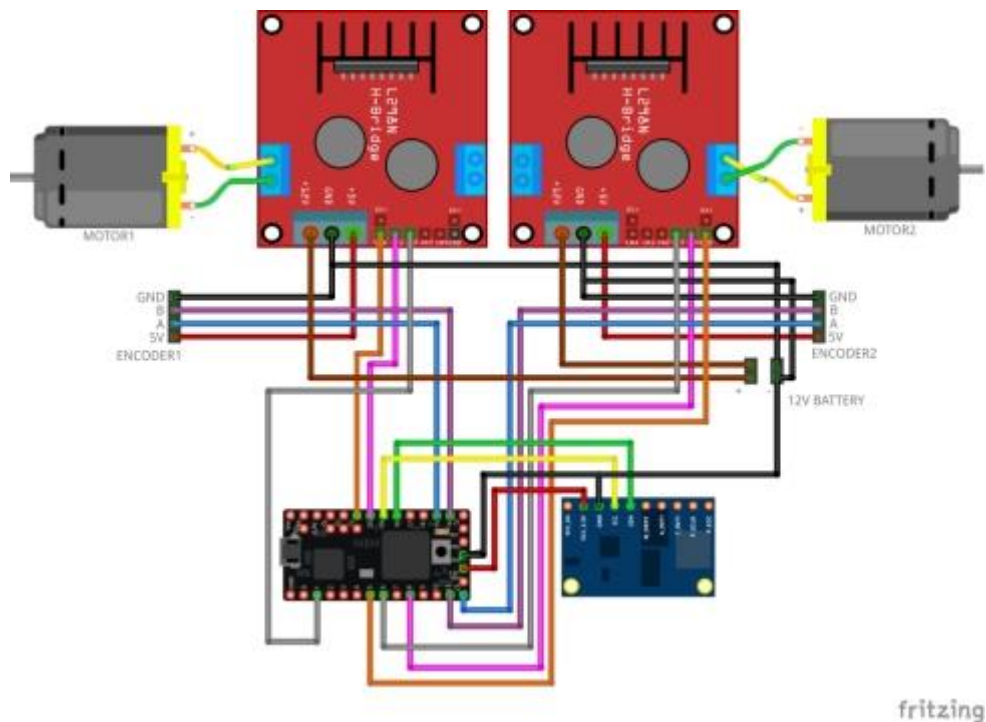


Fig 5.2 Microcontroller interface with MPU 9250 accelerometer and L298N motor drivers

All this information is passed to the Raspberry Pi by using separate controller space with this approach, RAM usage can be minimized. Interfacing to sensors becomes much easier with including library to the Teensy board. For Raspberry it needs a bridge to get in contact with teensy for that rosserial\_arduino is used to get serial values from teensy to Arduino by including ros.h library to teensy board. For uploading code from raspberry to teensy platformio is used to upload the code. RPLIDAR will give 8,000 sample signals per revolution of the turret that houses , a laser transmitter and receiver , but the signals produced by RPLIDAR cannot be processed with filtering the incoming signal for that ROS has a wonderful solution which is ros-kinetic-rplidar-ros node which will make the signal stable and filter the unwanted noise over the signal. Sometimes, error may occur due to wrong selection of LIDAR frequency, in the launch or param file or by low power supply to Raspberry/LIDAR.



## 5.2 HARDWARE OUTLOOK OF EVI ROBOT

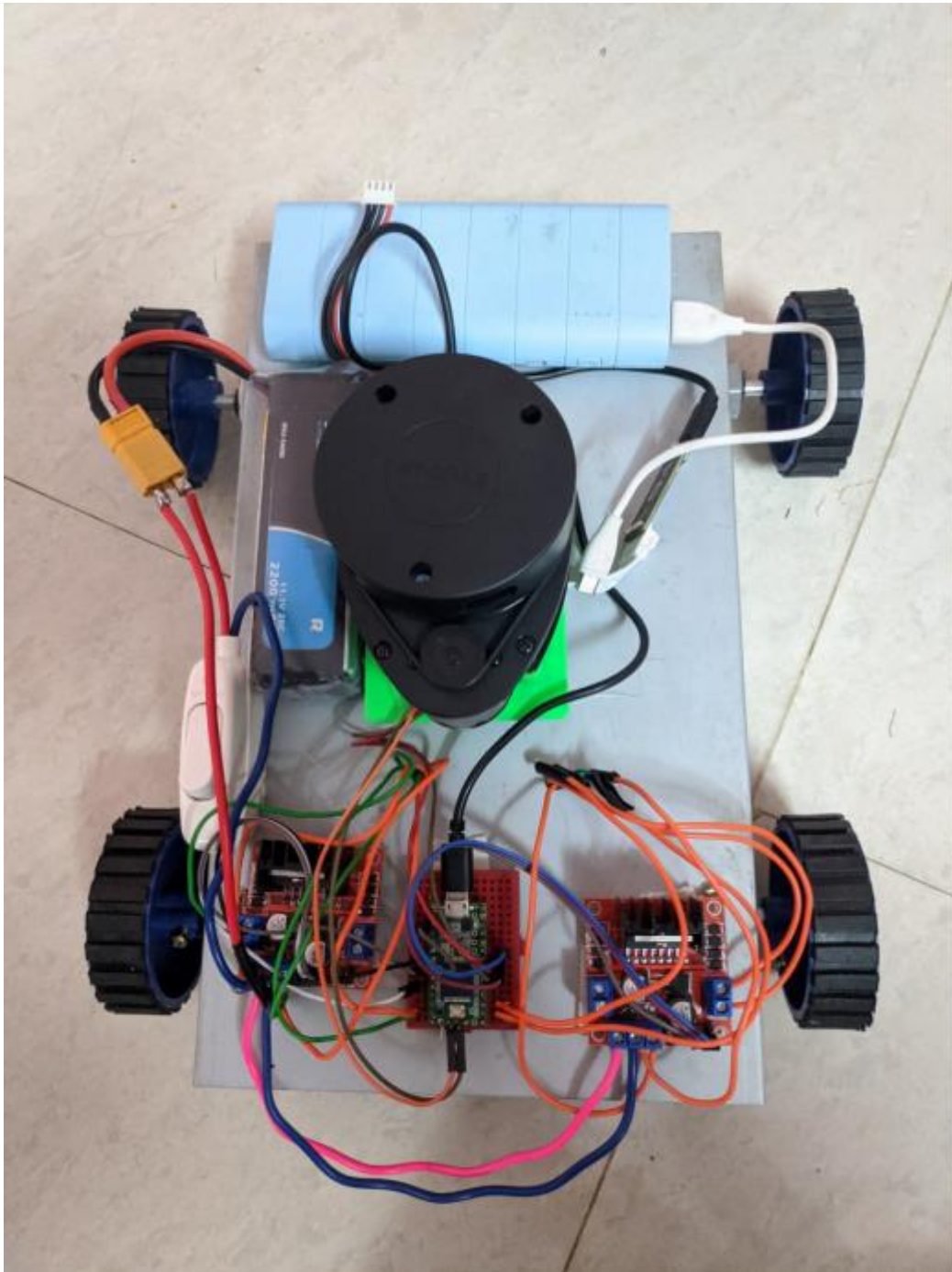


Fig 5.3 The Complete Hardware Assembly of Evi



## CHAPTER 6

### SOFTWARE INTEGRATION

#### 6.1 CONTROL FLOW DURING SOFTWARE INTEGRATION

The following operations are carried out in the ROS (Robotic Operating System) environment. As the Fritzing circuitry model shows Raspberry Pi 3 to be integrated with the RP LIDAR and Teensy microcontroller in Chapter 5 (Fig 5.3), the first fundamental step is to declare separate ports to both. The local server established is treated within the master computer. The LIDAR specifications (inputs) are given as separate labels into ROS. The collected frameworks are then used accordingly for establishing a bridge between the processor (within Raspberry Pi 3) and the Teensy microcontroller. The IMU (Inertial Measurement Unit) also referred to as accelerometer in this project is used to generate IMU values shown in Chapter 3 (Fig 3.2), which gives us information about the force acting on the robot while it is in motion, as well as providing the rate of tilt and the overall orientation of the robot while in motion. The calibration of IMU shown in Chapter 6 (Fig 6.6) unit is first completed by tilting the wheeled-robot in various directions and acquiring the values. Once calibration is complete, the calibrated firmware is uploaded once again with the recently acquired PID values. The data samples collected by the RP LIDAR and a sample static map is created for testing purposes, wherein which the sample points are joined together with imaginary lines generated by the LIDAR consecutively, to plot a sample map.

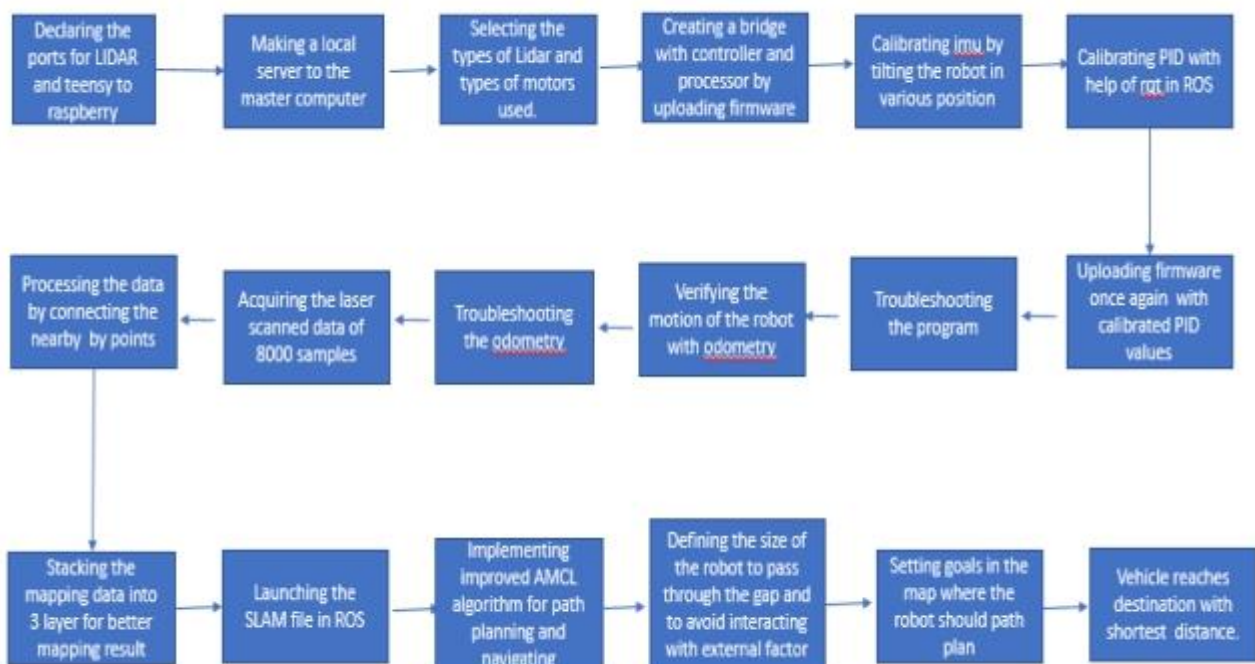


Fig 6.1 Software Integration Block Diagram

## 6.2 DECLARATION OF PORTS

```
1 <launch>
2   <node pkg="roscpp" name="roscpp" type="serial_node.py" output="screen">
3     <param name="port" value="/dev/ttyACM0" />
4     <param name="baud" value="57600" />
```

Fig 6.2 Assigning Teensy port to the launch file

There are two ways to input the port to the launch file one is declaring Udev rule (Fig 6.2) and other one is declaring ports straight away with giving read/write permission. Here, we have declared the ports directly by establishing nodes for permission because udev will not work on Raspberry Pi 3B+ without declaring port software cannot get access the data to the I/O devices.

```
1 <launch>
2   <node name="rplidar_node" pkg="rplidar_ros" type="rplidarNode" output="screen">
3     <param name="serial_port" type="string" value="/dev/ttyUSB0"/>
4     <param name="serial_baudrate" type="int" value="115200"/>
```

Fig 6.3 Assigning LIDAR port to the launch file

## 6.3 Establishing a local server:

### 6.3.1 ROS server:

```
export ROS_MASTER_URI=http://192.168.43.253:11311
export ROS_HOSTNAME=192.43.253
```

### 6.3.2 PC ROS server:

```
export ROS_MASTER_URI=http://192.168.43.253:11311export
ROS_HOSTNAME=192.168.43.82
```

It is mandatory to create a local ROS server between Raspberry and PC if we don't use server we should connect the raspberry and PC with wired connect. The reason behind server because Raspberry pi ROS doesn't have Rviz GUI because it does not have the GPU so Raspberry Pi can only filter the signals and convert and sends to PC ROS which will perform the SLAM operation and allow us to see the output visually.

## 6.4 SELECTION OF DRIVER AND MOTOR SPECIFICATIONS TO TEENSY

```
4  #define 2wd_BASE DIFFERENTIAL_DRIVE
5  #define USE_L298_DRIVER
6  #define USE_MPU9250_IMU
7
```

Fig 6.4 Selection of drivers, accelerometer and type of robot

```
15 #define MAX_RPM 330
16 #define COUNTS_PER_REV 1320
17 #define WHEEL_DIAMETER 0.10
18 #define PWM_BITS 8
19 #define LR_WHEELS_DISTANCE 0.235
```

Fig 6.5 Assigning motor and robot specifications

“2wd” is the library for 2 wheeled robot this is what we are developing. L298\_driver is the library for the motor driver. mpu9250 is the library for the accelerometer which is used to orient the map and used for motion of the robot. (fig 6.5)

In fig 6.5 The motor specification is included as the config library which will be used for navigation stack as well as PID of the robot .LR\_WHEELS\_DISTANCE is the distance between left and right wheel used to calculate turning or steering angle.

## 6.5 CALIBRATING IMU/ACCELEROMETER

The MPU9250 accelerometer is used as an IMU (Inertial Measurement Unit) or simply referred to as an IMU sensor in this project. It is used to perform calculations to provide information about the overall force acting on the robot, i.e to check if the component payload has exceeded a certain limit. It is also used to calculate the overall tilts and change in orientations caused while the robot is in motion (Fig 6.7).

```
58 orientations_.push(AccelCalib::XPOS);
59 orientations_.push(AccelCalib::XNEG);
60 orientations_.push(AccelCalib::YPOS);
61 orientations_.push(AccelCalib::YNEG);
62 orientations_.push(AccelCalib::ZPOS);
63 orientations_.push(AccelCalib::ZNEG);
64
65 orientation_labels_[AccelCalib::XPOS] = "X+ axis";
66 orientation_labels_[AccelCalib::XNEG] = "X- axis";
67 orientation_labels_[AccelCalib::YPOS] = "Y+ axis";
68 orientation_labels_[AccelCalib::YNEG] = "Y- axis";
69 orientation_labels_[AccelCalib::ZPOS] = "Z+ axis";
70 orientation_labels_[AccelCalib::ZNEG] = "Z- axis";
```

Fig 6.6 Three-axis calibration in c++ library

```

1  SN:
2  - 1.011195335294985
3  - -0.005478610695936285
4  - -0.005023688364125308
5  - 0.04996443396208794
6  - 0.999238462583232
7  - -0.03743896212053192
8  - -0.01279156783449262
9  - 0.08739016446413006
10 - 1.015073451343559
11 bias:
12 - -0.4328585554985973
13 - -0.3987367212348277
14 - -0.9287827723881869

```

Fig 6.7 Final accelerometer values after calibration

IMU/accelerometer should be calibrated before using the robot were IMU under RPLIDAR or middle of the robot. After that biringup.launch file should be initialed which consist of getting values of accelerometer and encoder from teensy. By giving command to ROS “ rosrn imu\_calib do\_calib” this will starts to calibrate the IMU. Total there are 3 major axis which splitted as +x,-x,+y,-y,+z,-z..

For +x robot front should be lifted up. For -x rear side of the robot should be tilted up. For +y right side of the robot is tilted up. For -y left side of the robot is tilted up. For +z topside of the robot should be tilted. For -z bottom side of the robot should tilted up. By doing this we can get the perfect motion of Evi while viewing the odometry of the robot in navigation.

## 6.6 CONFIGURATION OF PID CONTROLLER

The below figure shows tuning operations on PID controller using rqt framework. This process is done following a sequence of procedures, the first fundamental step being calibration of the PID (Proportional Integral Derivative) controller. The controller framework requires the use of three variables which are:  $K_p$ ,  $K_i$ ,  $K_d$ . All three values are set to '0' initially. Then ' $K_p$ ' variable is increased until the error noticed during tuning the PID controller is seen to lessen. The highest limit (Peak overshoot) is tested, at the threshold state of output values, i.e the output obtained for the lowest tilt.

This attribute also gives us the sensitivity of the PID controller used.  $K_i$  is then used as a final resort to compensate any minimal errors that are left behind.  $K_i$  can be varied starting from marginal factors like 0.0001 or even smaller.

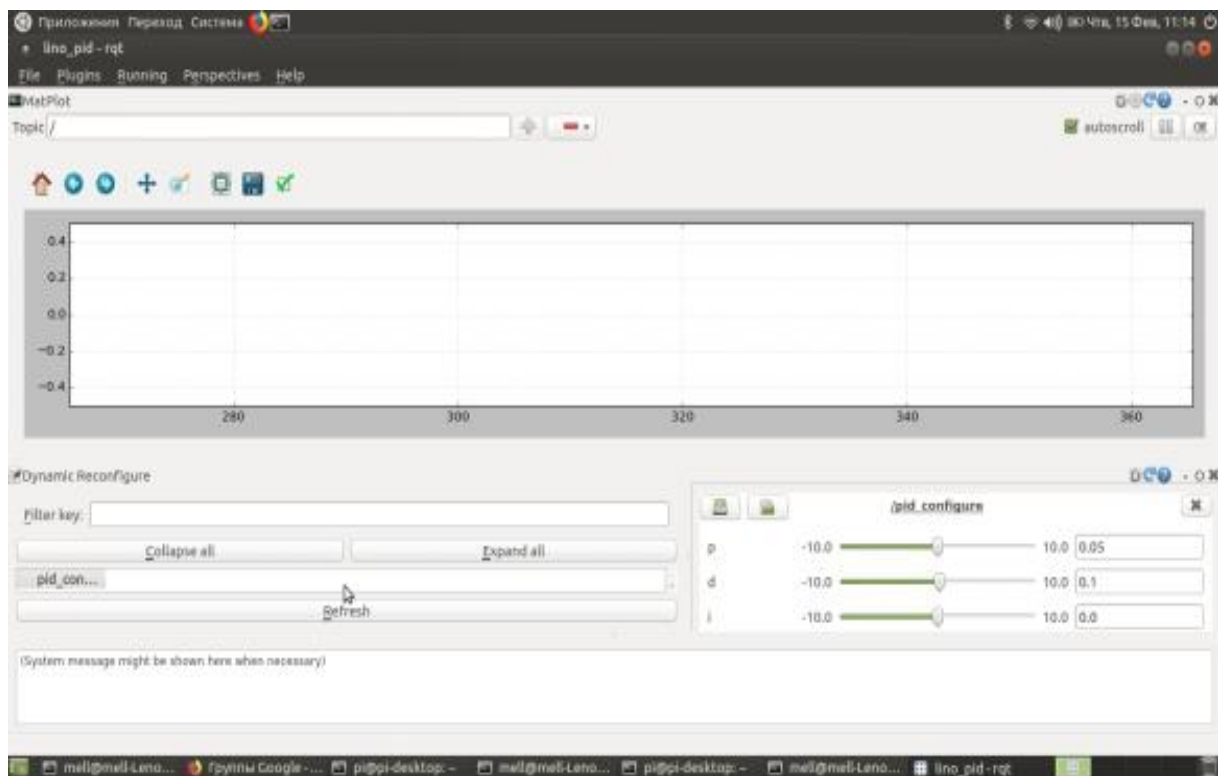


Fig 6.8 rqt application is used to tune PID

PID is necessary to reduce wobbling of the robot while tracing curves during navigation. rqt is the software in ROS which is used to configure PID with robot. PID value in rqt will be reflected in Arduino library PID. PID values will be sent to Teensy-Raspberry bridge. PID should be calibrated until the wobbling is neglected entirely during the navigation. teleop\_twist\_keyboard is used to drive the robot and correct the PID of the robot.

## 6.7 ODOMETRY CHECK FOR ROBOT

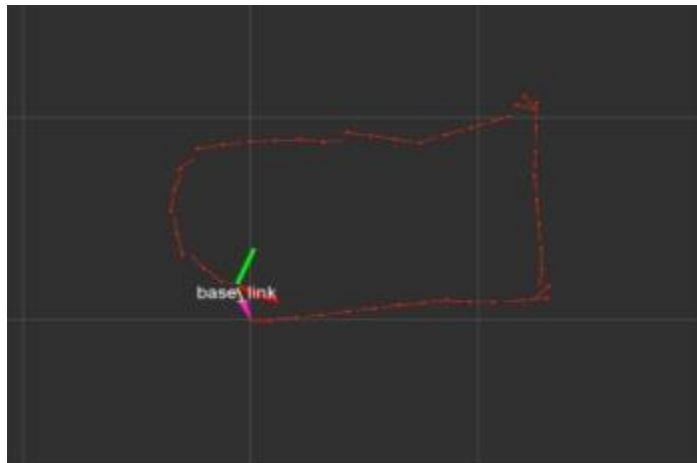


Fig 6.9 Acquired odometry

Odometry is the motion of the robot it should be in circle if PID, IMU and encoder works fine. This will collect the raw\_imu, raw\_enc data and creates a path. By using teleop\_twist\_keyboard press “J” to rotate the robot in anti-clockwise the path acquired in odometry should be circle. If the path formed is circle ,then all the SLAM and navigation stacks should work fine.

## 6.8 ACUIRING LASER SIGNAL FROM LIDAR

```
1 <launch>
2   <node name="rplidar_node" pkg="rplidar_ros" type="rplidarNode" output="screen">
3     <param name="serial_port" type="string" value="/dev/ttyUS80"/>
4     <param name="serial_baudrate" type="int" value="115200"/>
5     <param name="frame_id" type="string" value="laser"/>
6     <param name="inverted" type="bool" value="false"/>
7     <param name="angle_compensate" type="bool" value="true"/>
8   </node>
9 </launch>
```

Fig 6.9.1 roslaunch code for acquiring LIDAR values.

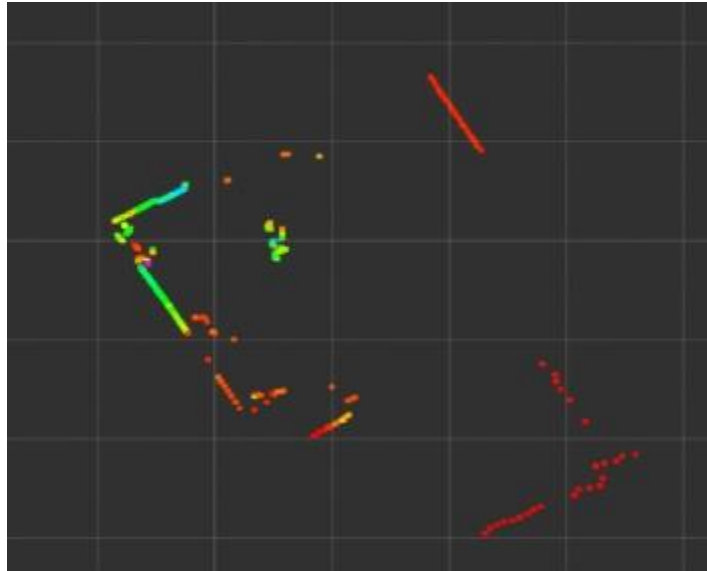


Fig 6.9.2 Acquired laser signal

rplidar.launch which consist of filters, port, baud rate of the LIDAR it will give co-ordinate points to the PC. To achieve autonomous navigation stack robot should have at least one laser signal to get to know about robot workspace and to avoid obstacle. Acquiring laser signal is the primary step which is mainly required before navigating the robot.

## CHAPTER 7

### TRI-LAYERED GMAPPING

#### 7.1 MAP GENERATED USING GENERAL GMAPPING

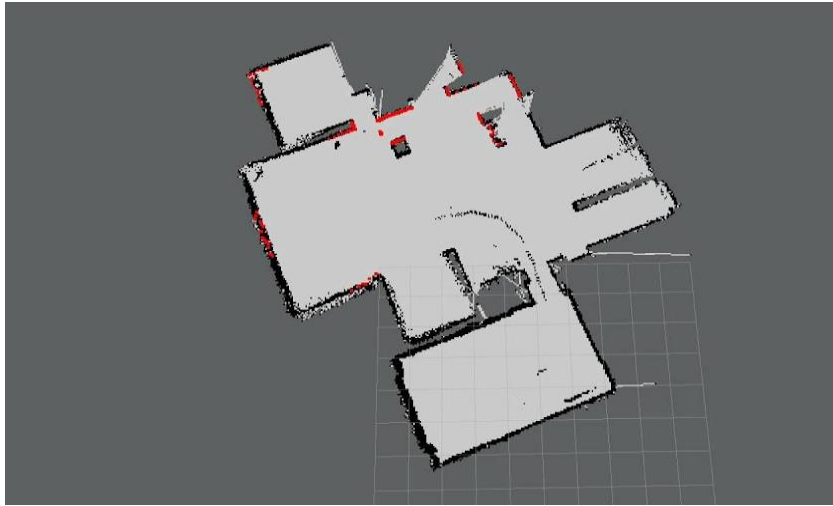


Fig 7.1 General Gmapping (single-layered)

The above map is created using the standard G-mapping SLAM technique. The map was generated in RVIZ, which is a 3D visualization tool inbuilt in the ROS application. It is quite similar to the TurtleBot simulator used in Gazebo which is also inbuilt in ROS, it is used to generate a real world within a virtual environment. In this world, the robot can be moved around the obstacles and simulation can be undertaken. Here rviz is used to acquire the sensor data from the RP LIDAR used in Evi and generate the captured data over a grid as shown below the map. The algorithms that have been used to come up with this generated map of a closed indoor environment is based on probabilities. The above map lacks clarity in depicting the object's external dimensions. The edges as seen in the map are not well-defined due to the fact that PID controller was not used. The controller was not compatible with the existing algorithms. The objects kept on the surface lacks further clarity, which causes the criss-crossing arcs around the centre of the generated map above the bottom rectangular map section.



## 7.2 MAP GENERATED WITH ENHANCED G-MAPPING METHOD

This method aims to improve the drawbacks seen in the generated map of the previous model by inculcating enhanced G mapping algorithms. The same 3D visualization tool rviz , from the above setup is used in this technique as well. The existing A\* algorithm has been improved , the difference incorporated in this technique is the improvements in the algorithm which is set to automatically calculate the displacement between the robot and the obstacle , the shortest route obtained , increases the robot's mobility and response in closely enclosed areas . We call this as the Three-layered mapping technique which involves enhanced g-mapping algorithms. The laser light transmitted from the rotating turret is used to acquire data in three layers. Those three layers are sandwiched within one another and are combined to form a map of a surface. The next layer generated will automatically discard the first layer that was scanned in the previous scan cycle. Leaving three layers to always be intact given any time interval. This technique was adopted to impart better stability to the map building process. As seen in the former case , the dimensions were mis-aligned due to weak laser signal transmission due to single-layered mapping. With this mapping technique the use of PID controller is appreciated by fine tuning the PID with the help of rqt framework .



Fig 7.2 Three-layered of G-mapping.

### 7.3 PSEUDO CODE:

Input : Laser scanner reading from LIDAR

Output : Map of the environment

1. RPLIDAR collects 8000 laser samples while rotating in 360° stitching the sample points and form a map.
2. Let  $S_1, S_2, S_3, \dots, S_n$  be the n-number of layers acquired by LIDAR.
3. First 3 revolutions( $S_1, S_2, S_3$ ) of LIDAR values are stacked layer by layer .
4. if robot move to new position;  
the third layer is modified with the new map layer
5. if the robot is static  
first layer is replaced with new map layer
6. repeat step 3-5 still the complete environment map is generated.

```

3   <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="std_out"
4       <param name="base_frame" value="/base_footprint" />
5       <param name="odom_frame" value="/odom" />
6       <param name="map_update_interval" value="15.0"/>
7       <param name="maxUrange" value="5.0"/>
8       <param name="minRange" value="-0.5"/>
9       <param name="sigma" value="0.05"/>
10      <param name="kernelSize" value="1"/>
11      <param name="lstep" value="0.05"/>
12      <param name="astep" value="0.05"/>
13      <param name="iterations" value="5"/>
14      <param name="lsigma" value="0.075"/>
15      <param name="ogain" value="3.0"/>
16      <param name="lskip" value="0"/>
17      <param name="minimumScore" value="100"/>
18      <param name="srr" value="0.01"/>
19      <param name="srt" value="0.02"/>
20      <param name="str" value="0.01"/>
21      <param name="stt" value="0.02"/>
22      <param name="linearUpdate" value="0.7"/>
23      <param name="angularUpdate" value="0.7"/>
24      <param name="temporalUpdate" value="-0.5"/>
25      <param name="resampleThreshold" value="0.5"/>
26      <param name="particles" value="50"/>
27      <param name="xmin" value="-50.0"/>
28      <param name="ymin" value="-50.0"/>
29      <param name="xmax" value="50.0"/>
30      <param name="ymax" value="50.0"/>
31      <param name="delta" value="0.05"/>
32      <param name="lssamplerange" value="0.05"/>
33      <param name="llsamplestep" value="0.05"/>
34      <param name="lasamplerange" value="0.005"/>
35      <param name="lasamplestep" value="0.005"/>
36      <param name="transform_publish_period" value="0.1"/>
37  </node>

```

Fig 7.3 Code involved in SLAM technology

## 7.4 Application of SLAM technology and LIDAR

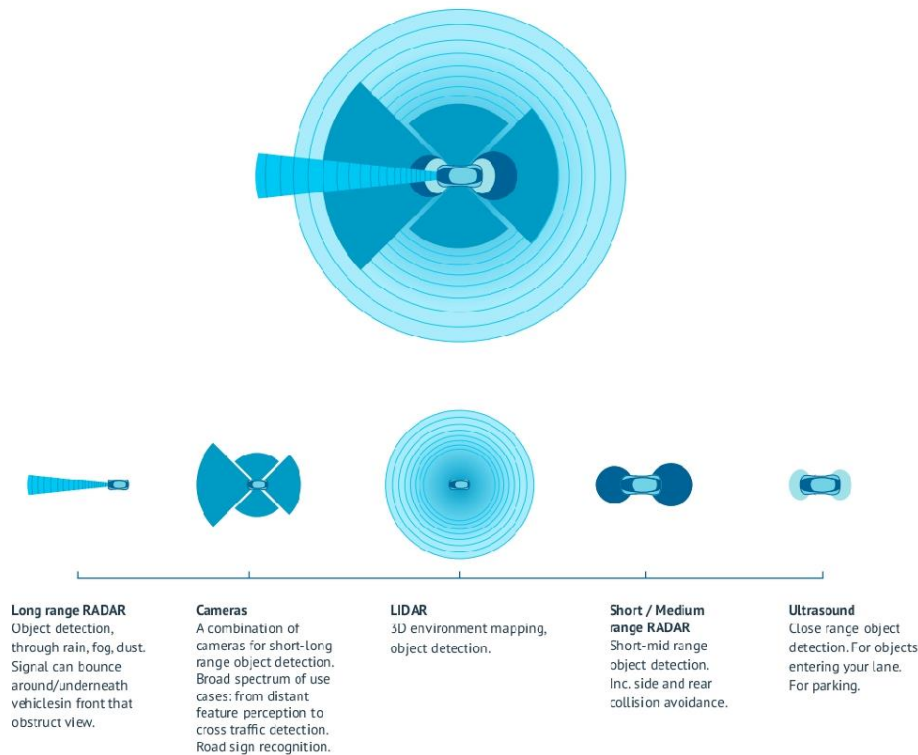


Fig 7.4 Electronic components used in autonomous car

SLAM (Simultaneous Localization and Mapping) is a complex technology that are used in autonomous cars SLAM technology is not just meant for LIDAR or laser scanner it is applicable to all time of image capturing or laser sensor. We have made some improvements in Gmapping which will give better viewing of the map generated by LIDAR. Single layer mapping the data won't sufficient as the environment is changing so we have decided to add 3 more layers to the map. At the static location A,B,C be the first, second and third layers of the map during one rotation cycle which at each rotation of the LIDAR 'A' will be discarded and the newest 4<sup>th</sup> layer will be added to it if the robot is starts to move with a fixed three layers of the map become, unchanged in a given point of time.

## 7.5 DIFFERENCE BETWEEN ACTUAL GMAPPING AND TRI-LAYERED GMAPPING:

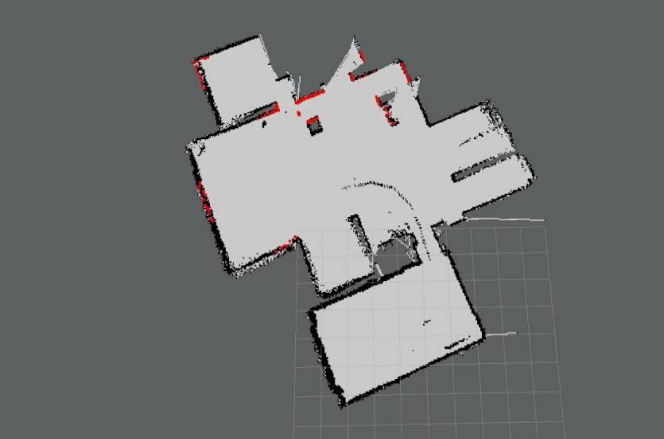
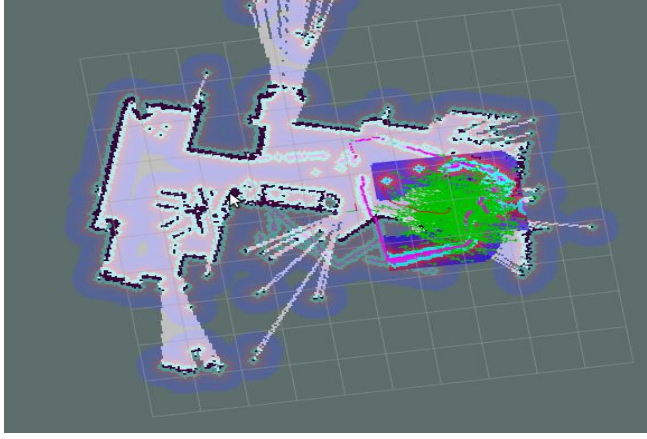
General G-mapping	Tri-layered G-mapping
	
<p>1. The general gmapping method consists of consists of a single layer involved in mapping.</p>	<p>1. The Tri-layered mapping shown above, consists of three layers stacked upon one another during a given interval of time, to provide better stability of the generated map.</p>
<p>2. It is not effective compared to the proposed methodology on the basis of navigation. Map can be easily scattered or shuffled and it cannot be reconstructed.</p>	<p>2. Due to better viewing angle and the use of PID controller for odometry it has great efficiency on navigation. If map starts to scatter it can be easily reconstructed with previous layer.</p>
<p>3. The external structure of the objects present in an environment is not seen to be represented accurately ,there can be various factors like offset in wheel alignments or undesired scattering of the map due to acquisition of inappropriate IMU values.</p>	<p>3. The sample points generated in this map is not scattered due to the layer-stacking counter measure incorporated into it. This generated map has the characteristic of repeatability, so any given point of time, the map generated in trial ‘1’ will almost resemble the same environment scanned in trial ‘2’ without the interference of any external factors can be retrieved easily taking into the account of the prebuilt layer.</p>
<p>4.Navigation can be created easily with help of various pre-existing navigation algorithms available. But it will rather deem ineffective, due to its lack of accuracy and reliability of the generated map.</p>	<p>4. Modified navigation is implemented for this tri-layered mapping</p>

Table 7.1: Difference between general and tri-layered gmapping.

## CHAPTER 8

### IMPROVED AMCL (ADAPTIVE MONTE CARLO LOCALIZATION)

#### 8.1 WORKING PRINCIPLE OF IMPROVED NAVIGATION SYSTEM

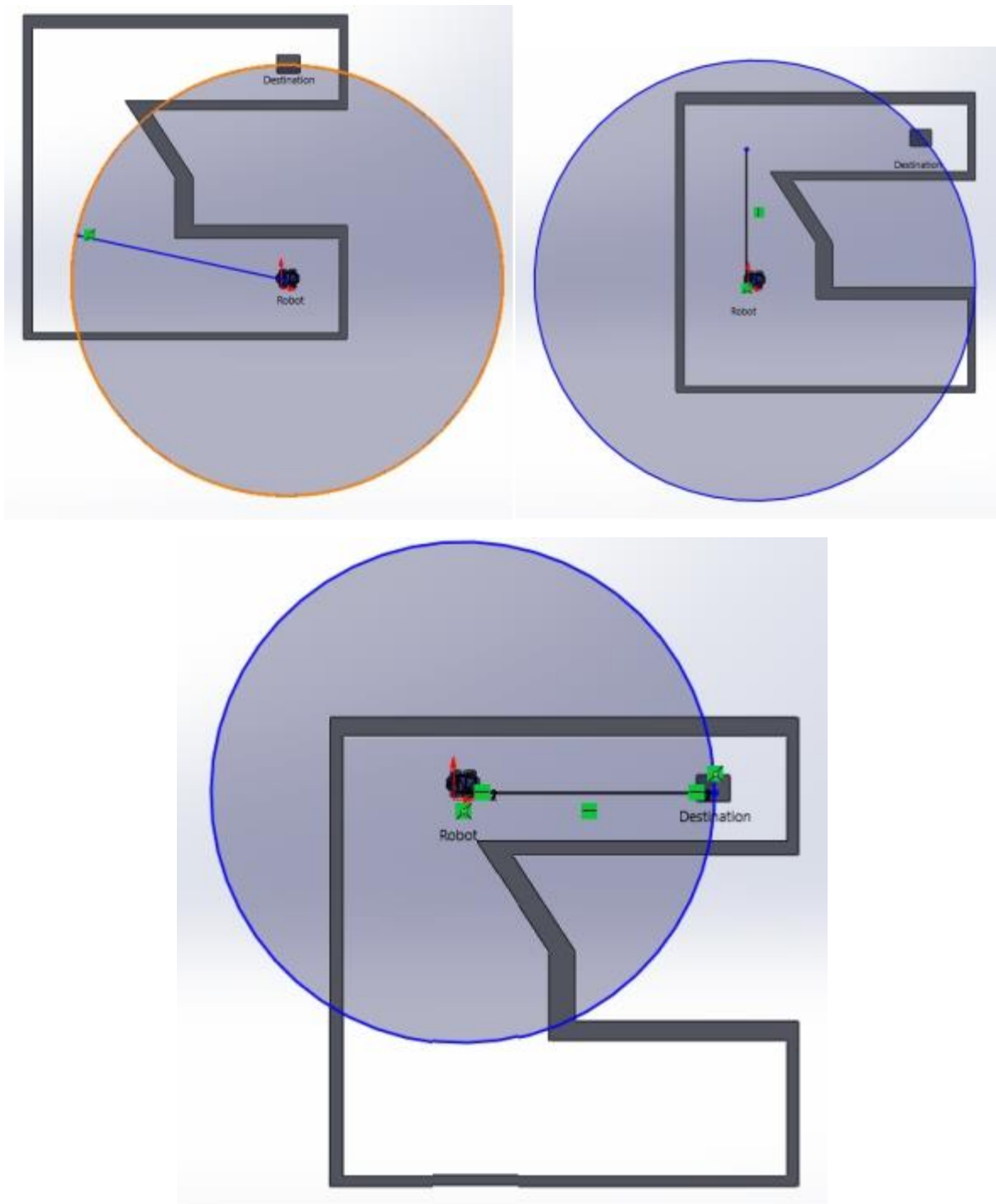


Fig 8.1 Working principle of improved navigation system

## 8.2 FUNCTIONS AND WORKING OF IMPROVED NAVIGATION

- Navigation is basically a logic by using ROS packages we can develop our own navigation stack
- Our navigation is to improve the existing AMCL algorithm which is majorly used to for navigation.
- If the robot has the generated map pertaining to the environment ,The SLAM-based wheeled robot can navigate to its destination with ease with the help of co-ordinates.
- The navigation robot uses a set of encoder motors to get co-ordinate values and LIDAR data to know the range of destination as well as to localize itself in an unknown environment.
- There are some aspects when it comes to navigation trajectory, robot footprint, goal tolerance, obstacles, optimization and homotopy class planner.

```
3 <param name="base_frame_id" value="base_footprint"/>
4 <param name="gui_publish_rate" value="10.0"/>
5 <param name="kld_err" value="0.05"/>
6 <param name="kld_z" value="0.99"/>
7 <param name="laser_lambda_short" value="0.1"/>
8 <param name="laser_likelihood_max_dist" value="2.0"/>
9 <param name="laser_max_beams" value="60"/>
10 <param name="laser_model_type" value="likelihood_field"/>
11 <param name="laser_sigma_hit" value="0.2"/>
12 <param name="laser_z_hit" value="0.5"/>
13 <param name="laser_z_short" value="0.05"/>
14 <param name="laser_z_max" value="0.05"/>
15 <param name="laser_z_rand" value="0.5"/>
16 <param name="max_particles" value="2000"/>
17 <param name="min_particles" value="500"/>
18 <param name="odom_alpha1" value="0.25"/>
19 <param name="odom_alpha2" value="0.25"/>
20 <param name="odom_alpha3" value="0.25"/>
21 <param name="odom_alpha4" value="0.25"/>
22 <param name="odom_alpha5" value="0.1"/>
23 <param name="odom_frame_id" value="odom"/>
24 <param name="odom_model_type" value="diff"/>
25 <param name="recovery_alpha_slow" value="0.001"/>
26 <param name="recovery_alpha_fast" value="0.1"/>
27 <param name="resample_interval" value="1"/>
28 <param name="transform_tolerance" value="1.25"/>
```

Fig 8.2 Launch file of navigation stack

- For visualizing the navigation there should be rviz with mapping these parameters should be enabled base\_link, imu\_link, odom, map and laser.
- PC will visualize the rviz so it will send values to raspi by using ROS\_MASTER\_URI.
- In rviz PC will order raspi to bringup the planner and costmap which will be having vel\_theta, accel limit of x and y, minimum obstacle distance.
- By this method robot navigation stack creates virtual circle to the destination and it will calculate if there is any straight path if there is any obstacle it will path plan to the minimal angle so it will travel to the distance with co-ordinates produced by encoder and laser signal.
- At every rotation of the LIDAR so there will be virtual circle developed along with it so there will be new path will be planned at each virtual circle.
- Fig 8.1 , shows how the robot will reach the destination using improved AMCL (Adaptive Monte Carlo Localization) with using virtual circle.



## CHAPTER 9

### EXPERIMENTAL RESULTS

#### 9.1 OVERVIEW

Chapter 7 gave an insight into the two different G-mapping techniques. The former being conventional single-layered grid mapping technique and the latter being the Three-layered grid-mapping technique. Two different maps have been generated with both of these techniques and their difference in stability and reliability was seen to improve in the latter. This chapter is set to analyze the enhanced g-mapping method by applying it in a vehicle moving around in a physical environment. The figure shown below is the overall final integration of the robot.

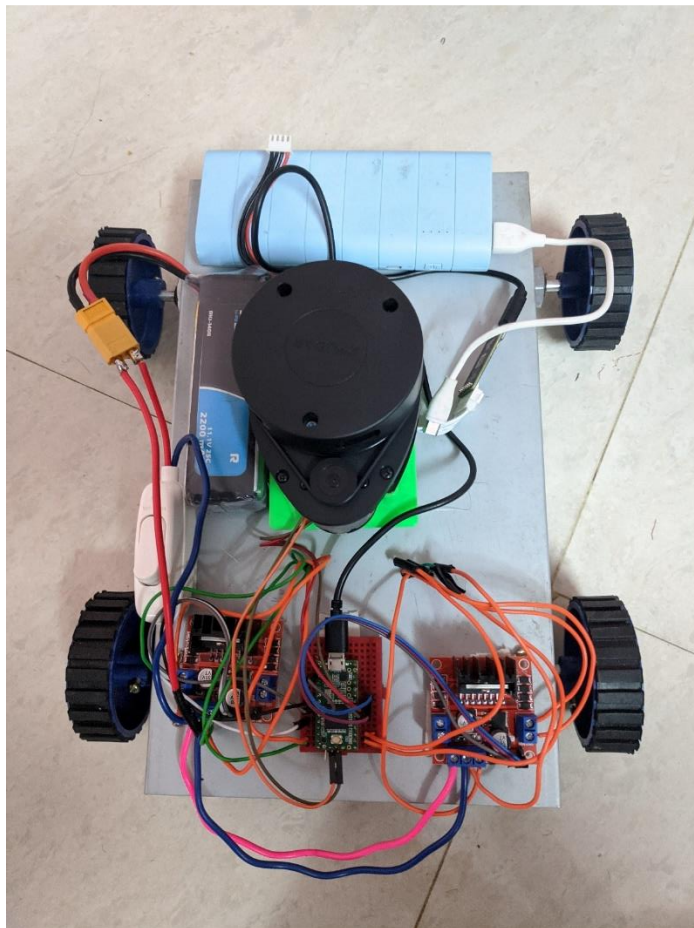


Fig 9.1 Hardware output

The overall setup used in the above figure consists of the components that were listed in Chapter 3. Single core wires color coded in orange is used along with few jumper wires soldered together are used to provide connections to interface both L298N motor drivers with the Teensy microcontroller as well as the MPU accelerometer.

The fluorescent green 3-D printed component is inculcated within the robot to stabilize the RP LIDAR that is set on top of the board beneath which, lies the Raspberry Pi 3. The dual encoder motors are drill-fitted into the metal chassis in order to impart accurate PID tuning and to trace a smoother curve while the robot undertakes the path planning operation. There are two separate power supply units used in the robot. The 10,000mAh Power bank unit is used to supply DC power to the Raspberry Pi. The other power supply unit is a Li-Po battery 2,200 mAh is used to supply power to the dual encoder gear motors.

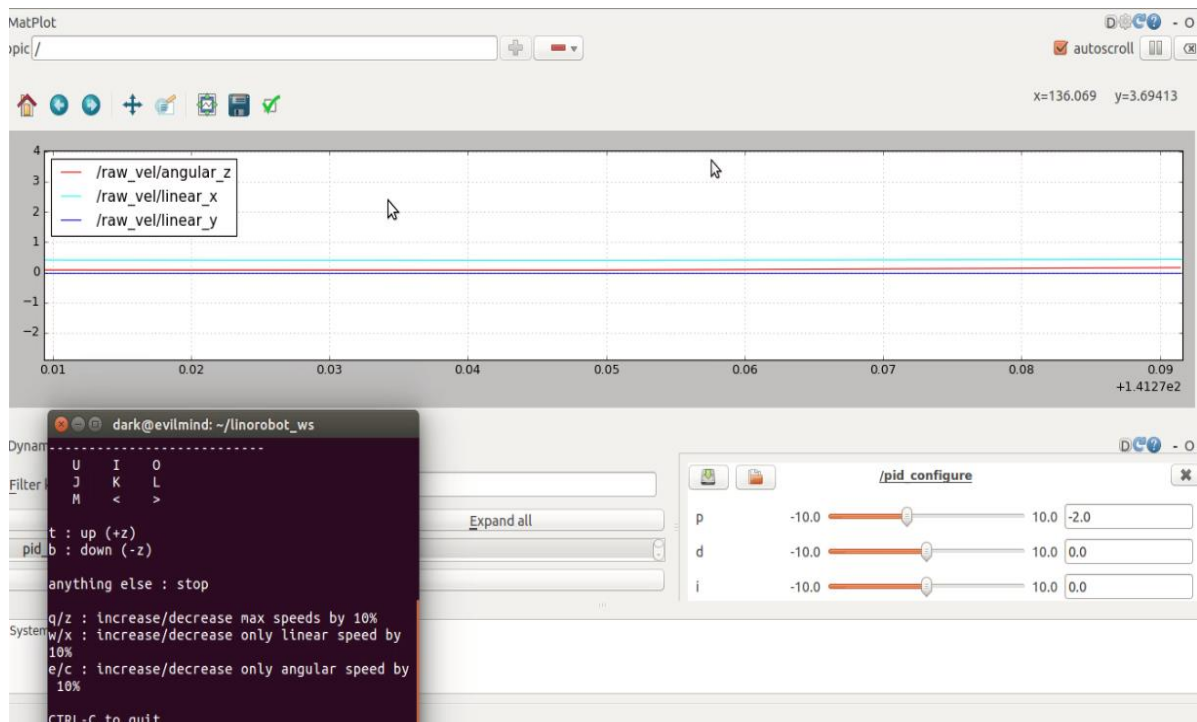


Fig 9.2: Output of MatPlot from rqt framework for tuning PID

- Rqt framework is used along with other software frameworks to configure the PID for the robot.
- Perspective is been created over the rqt for future use.
- Raw\_vel has been plotted with MatPlot framework as shown in the above figure.
- Where we need to find the correct PID for the robot by checking the error while running the robot with teleop\_twist\_keyboard in ROS ,whose complete description can be referred in

Chapter 4. The values are changing real-time when the feedback signal influences the direction of wheel turns. We have used EKF to filter the IMU data as well as encoder value.

```
[INFO] [1586578618.082922]: Encoder FrontRight : -20428
[INFO] [1586578618.087660]: Encoder RearLeft : 0
[INFO] [1586578618.092742]: Encoder RearRight : 0
[INFO] [1586578618.296302]: Encoder FrontLeft : -22908
[INFO] [1586578618.323021]: Encoder FrontRight : -20428
[INFO] [1586578618.328475]: Encoder RearLeft : 0
[INFO] [1586578618.334103]: Encoder RearRight : 0
[INFO] [1586578618.515859]: Encoder FrontLeft : -22908
[INFO] [1586578618.520849]: Encoder FrontRight : -20428
[INFO] [1586578618.525369]: Encoder RearLeft : 0
[INFO] [1586578618.529962]: Encoder RearRight : 0
[INFO] [1586578618.735603]: Encoder FrontLeft : -22908
[INFO] [1586578618.741023]: Encoder FrontRight : -20428
[INFO] [1586578618.746174]: Encoder RearLeft : 0
[INFO] [1586578618.760261]: Encoder RearRight : 0
[INFO] [1586578618.953621]: Encoder FrontLeft : -22908
[INFO] [1586578618.959160]: Encoder FrontRight : -20428
[INFO] [1586578618.964645]: Encoder RearLeft : 0
[INFO] [1586578618.970146]: Encoder RearRight : 0
[INFO] [1586578619.173859]: Encoder FrontLeft : -22908
[INFO] [1586578619.179720]: Encoder FrontRight : -20428
[INFO] [1586578619.184884]: Encoder RearLeft : 0
[INFO] [1586578619.190415]: Encoder RearRight : 0
```

Fig 9.3: Output of Encoder motor from launch file for tuning PID

In the above figure , each of the four wheels attached to the bot via drilling is labelled. The labels shown above are : FrontRight, FrontLeft, RearRight, RearLeft ; to denote all four wheels.

## 9.2 CREATING A LOCAL SERVER USING LAUNCH FILE

```
File Edit View Search Terminal Help
ekf_localization (robot_localization/ekf_localization_node)
imu_filter_madgwick (imu_filter_madgwick/imu_filter_node)
lino_base_node (linorobot/lino_base_node)
rosterial_lino (rosterial_python/serial_node.py)

auto-starting new master
process[master]: started with pid [3933]
ROS_MASTER_URI=http://192.168.43.253:46765

setting /run_id to 5b8c658e-7bad-11ea-a6cb-b827eba078c4
process[rosout-1]: started with pid [3946]
started core service [/rosout]
process[rosterial_lino-2]: started with pid [3963]
process[apply_calib-3]: started with pid [3964]
process[imu_filter_madgwick-4]: started with pid [3965]
process[base_footprint_to_imu_link-5]: started with pid [3976]
process[lino_base_node-6]: started with pid [4003]
[ INFO] [1586579514.775963483]: Starting ImuFilter
process[base_footprint_to_base_link-7]: started with pid [4016]
process[ekf_localization-8]: started with pid [4028]
[ INFO] [1586579514.930530769]: Using dt computed from message headers
[ INFO] [1586579515.222892053]: Imu filter gain set to 0.100000
[ INFO] [1586579515.223131546]: Gyro drift bias set to 0.000000
[ INFO] [1586579515.223282648]: Magnetometer bias values: 0.000000 0.000000 0.0
```

Fig 9.4: Result of launch file used to create local server

The figure shown represents the output value after filtering the input signal. It can be seen that the varying values of the filter gain set and gyro drift bias as well as the magnetometer is being shown during the motion of the robot.

This launch file is used to create local server to transmit IMU and encoder data to the PC computer.

- Output majorly transmit EFK signal with both read/write permission.
- Basically this launch file convert filters into odometry values in PC.

### 9.3 POSE AND TWIST OUTPUT

```
pose:
pose:
position:
x: 0.0530822266891
y: 0.0652583920816
z: 0.0
orientation:
x: 0.0
y: 0.0
z: -0.945670513536
w: 0.325126559709
covariance: [11.65659474908849, -0.04274906595707535, 0.0, 0.0, 0.0, -0.971756094009087, -0.042749065957075186, 11.515199475203582, 0.0, 0.0, 0.0, 0.3369881071168502, 0.0, 0.0, 9.996646020662635e-07, 8.936513576375267e-44, 1.1305600912710906e-38, 0.0, 0.0, 0.0, 8.936513576375265e-44, 9.993295041347473e-07, 1.3894679550337065e-34, 0.0, 0.0, 0.0, 1.1305600912710906e-38, -1.3894679550337065e-34, 9.993295041347473e-07, 0.0, -0.9717560940090862, 0.33698810711685073, 0.0, 0.0, 0.0, 13.794936641846796]
twist:
twist:
linear:
x: -3.83721937433e-25
y: 0.0
z: 0.0
angular:
x: 0.0
y: 0.0
z: 1.45410613955e-05
covariance: [9.324316208289579e-05, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 9.324316208289579e-05, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 9.99497015626974e-07, 1.7733178858751823e-51, 2.8452989352762126e-46, 0.0, 0.0, 0.0, 1.7733178858751835e-51, 9.979921005681246e-07, -7.612040218405935e-42, 0.0, 0.0, 0.0, 2.845298935276213e-46, 7.616449146700727e-42, 9.979921005681246e-07, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 9.086245687552163e-05]
---
```

Fig 9.5: Result of Pose and Twist of the robot

## 9.4 ODOMETRY OF THE ROBOT

The received laser light into the rotating turret of the RP LIDAR acts as a typical sensor output which is used in odometry to predict the change in position of the robot over a period of time. Pose and twist are the two attributes used in this model for inculcating odometry in rviz. The odometry feature also requires initial parameters/dimensions of the robot such as : diameter of the wheel and its circumference , the distance between two wheels in both sections (length, breadth) , the endpoint of the circular wheel to the end point of the consecutive circular wheel as well as the shortest distance between each endpoints of the wheels.

- The output of raw\_odom is the value which is requested from the Raspberry Pi3
- Values are mostly of encoder and accelerometer with the ekf filter.

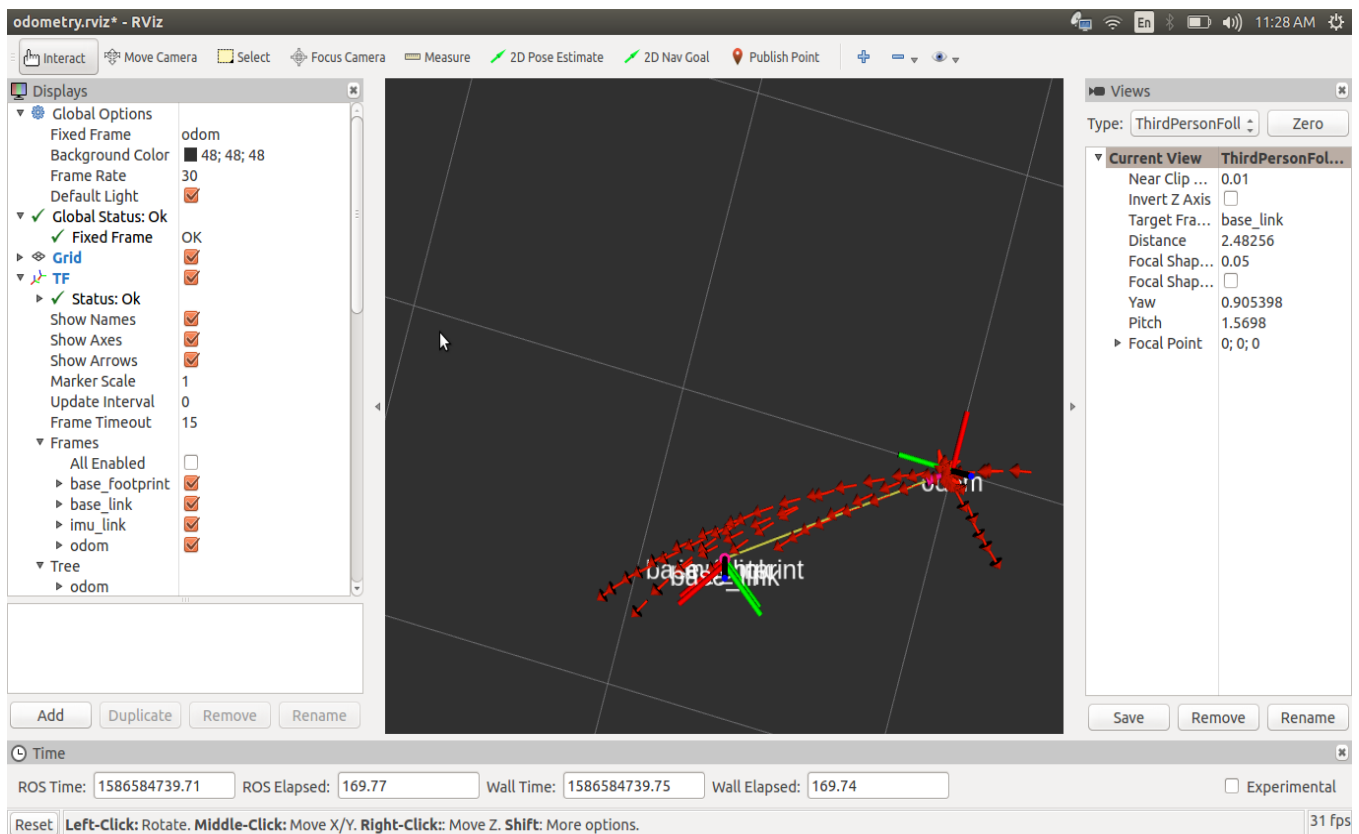


Fig 9.6 Results for odometry of the robot

- Odometry is used to predict the change in position of the robot over time. The motion of the robot is basically the trajectory of the robot as projected in the figure.
- It is built using ekf and odom\_value of the raspberry pi value.
- Using this application we can develop advanced motion principles into the robot and error detection is also possible as it compares the real-time values to the desired value. There four

major aspects to be considered while acquiring the trajectory output are– base\_footprint, base\_link, imu\_link, odom\_value.

- By checking the Odom\_value , one can find the misdirection of the robot and can be corrected the error by the application of a PID controller into the robot.

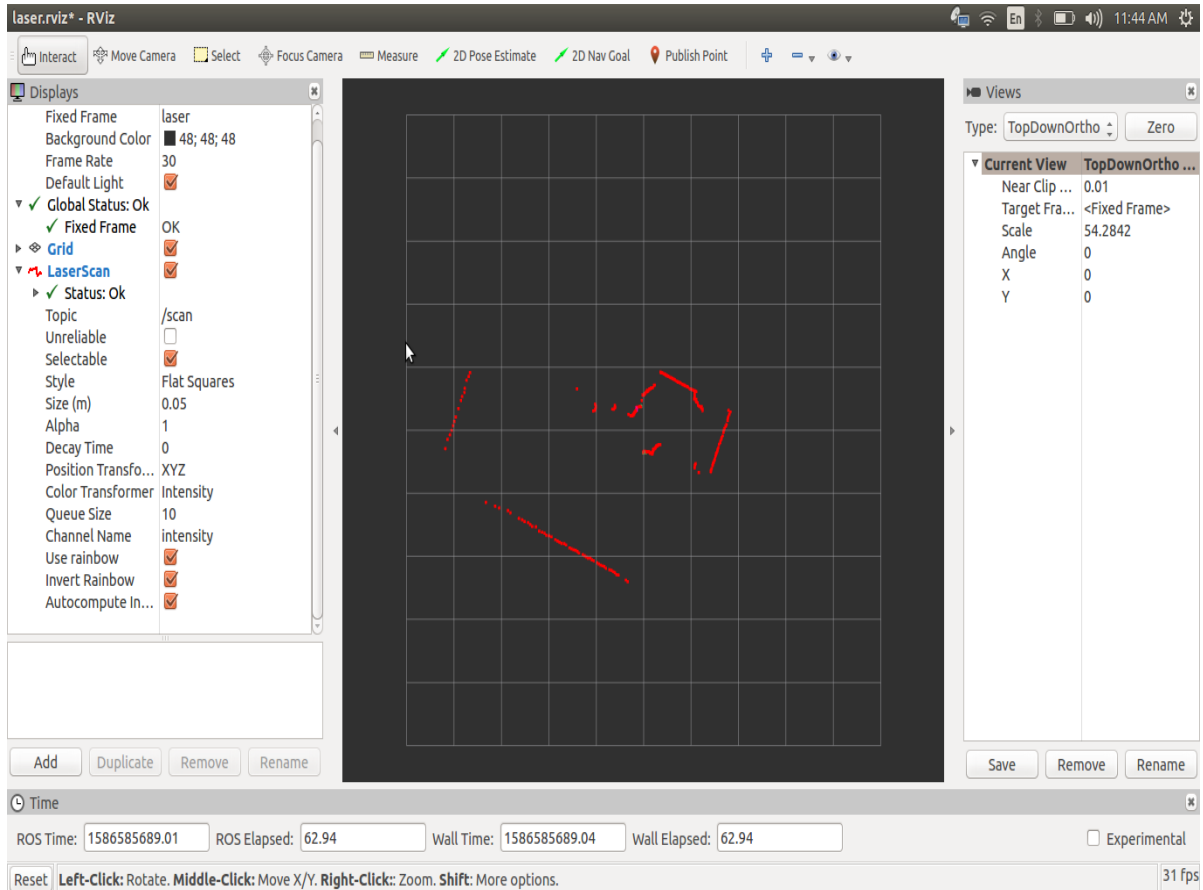


Fig 9.7: Output of RP LIDAR laser signal

- The above figure represents the output data of RP LIDAR which collects upto 8000 sample points of data.
- This output is transmitted to the local server from Raspberry Pi3 from a defined port /dev/ttyUSB0



## 9.5 TRI-LAYERED G-MAPPING GENERATED MAP OUTPUT

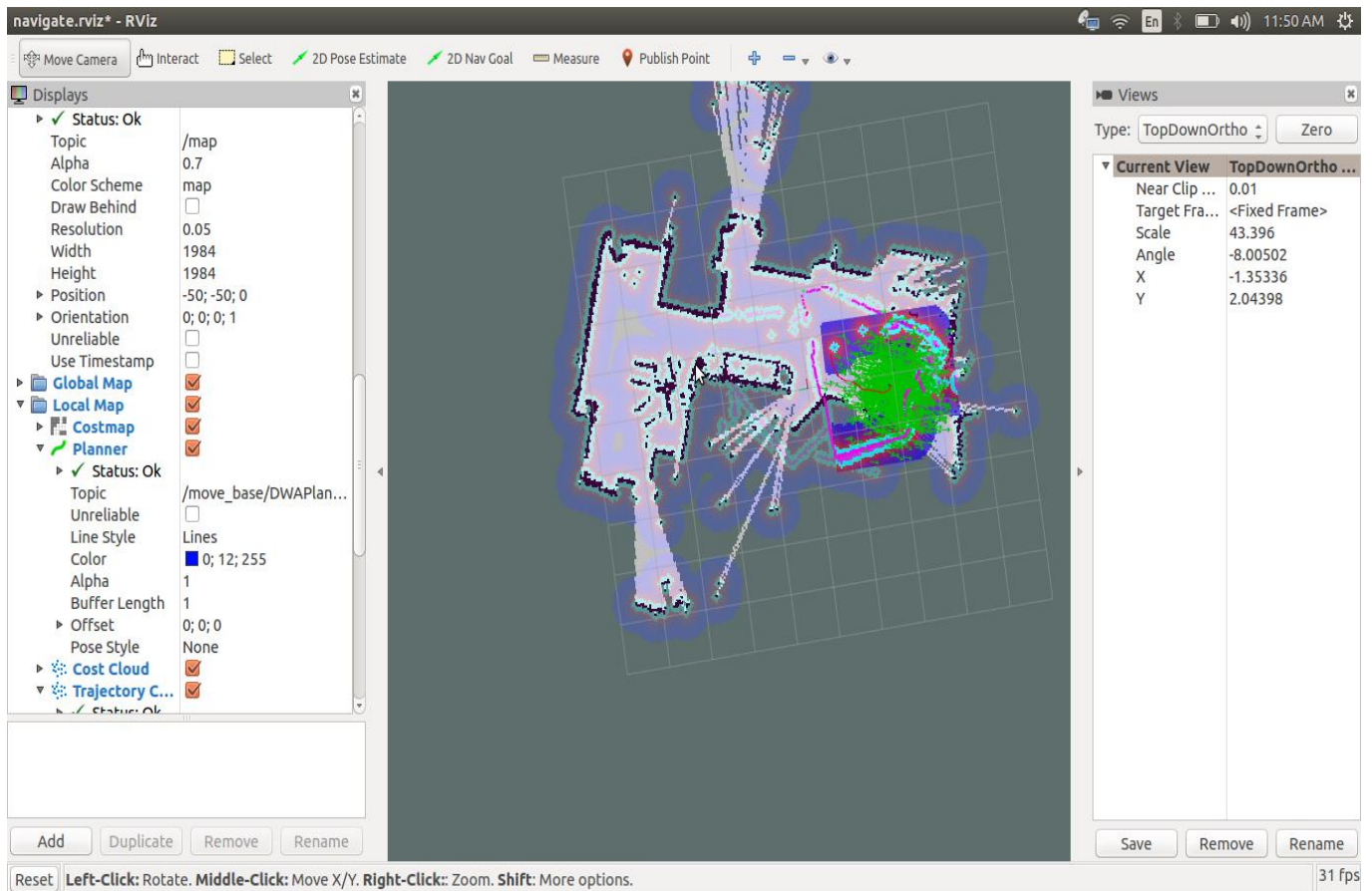


Fig 9.8 Output of the improved SLAM with navigation stack

- It requires EKF, improved Gmapping, odometry of the robot and PID to control the motors with ease.
- The above figure gives the final output of the robot i.e the generated map of a physical environment whilst it is in motion. The improved g-mapping as well the navigation stack is included in it.

## **CHAPTER 10**

### **CONCLUSION**

Surveillance systems play a key role for acquiring vital information to Intelligence agencies as well as providing valuable intel for commercial use. Human interference with a disaster-prone environments with deteriorating weather conditions is risky. In order to ensure personnel safety , this notion calls for the need of man-made surveillance bots for wider exploration and to be capable of path planning in order to automatically move in/around an area. It requires object detection to avoid collision while it moves across various obstacles . A navigation system must be embedded in the robot so that the robot can know/identify its position within the generated map. In most cases, collapsed buildings and other strongholds blocks the passages so the rescue teams were unable to push in. Situations arise , humans cannot physically access certain locations to identify if people are trapped within it. In such cases, employing an unmanned robot proves useful. Rescue assisting robots can be manufactured as products to be sold to various government and private organizations to ensure global safety in disaster management sector. These causes have led us to think of an efficient means to carry out rescue operations , leading to the Evi robot. With the application of SLAM technology and enhanced G-mapping algorithms , equipped with high rpm encoder motors and integrated with PID controller with effective tune-up to reduce wobbling while tracing its path as much as possible .Further interfaced wirelessly with a mid-range computer , it can let us visualize the map of an environment getting generated in real-time. Thus, such rescue assisting robots can be deployed to undertake search and rescue operations effectively.



## **CHAPTER 11**

### **FUTURE WORK**

Since this is the initial phase of robot development , we have discussed and decided to add various features than can enhance this existing robot. The features that will be added in the near future is listed below as follows:

- Machine learning will be adopted into this existing project in order to impart human detection. The currently decided software for human detection is TensorFlow whose input is taken using a necessary camera module.
- The two sets current wheels will be replaced with all terrain wheels similar to an aluminium tank track chassis. For better mobility and to combat against rough , wet terrains and to reduce the payload on the motor to reduce power consumption.
- To extend this scope of dual-encoder motor-operated wheels into a four-wheeled motor operated robot to improve mobility and path tracing.
- Based upon wider applications , it has been thought of implementing this technology in autonomous self-driving vehicles once the expected accuracy and robustness has been achieved.

## References:

1. RFS-SLAM Robot: An Experimental Platform for RFS Based Occupancy-Grid SLAM 20<sup>th</sup> International Conference on Information Fusion Xi'an, China - July 10-13, 2017.
2. Research and Implementation of SLAM Based on LIDAR for Four-Wheeled Mobile robot International Journal of Information and Electronics Engineering, Vol. 8, No. 1, March 2018.
3. An Approach to Restaurant Service Robot SLAM Proceedings of the 2016 IEEE International Conference on Robotics and Biomimetics Qingdao, China, December 3-7, 2016
4. Autonomous Wheelchair Navigation in Unmapped Indoor Environments 2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC).
5. Research on Indoor Robot SLAM of RBPF Improved with Geometrical Characteristic Localization 2017 29th Chinese Control And Decision Conference (CCDC)
6. An Improved Serial Method for Mobile Robot SLAM. 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO).
7. A Fire Protection Robot System Based on SLAM Localization and Fire Source Identification. 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC).
8. [http://wiki.ros.org/roserial\\_python](http://wiki.ros.org/roserial_python)
9. [http://smithcsrobot.weebly.com/uploads/6/0/9/5/60954939/pid\\_control\\_document.pdf](http://smithcsrobot.weebly.com/uploads/6/0/9/5/60954939/pid_control_document.pdf)
10. [https://en.wikipedia.org/wiki/Extended\\_Kalman\\_filter](https://en.wikipedia.org/wiki/Extended_Kalman_filter)
11. [http://wiki.ros.org/robot\\_localization](http://wiki.ros.org/robot_localization)
12. <http://wiki.ros.org/navigation/Tutorials/RobotSetup/Odom>
13. <https://robohub.org/robots-101-lasers/>
14. <http://wiki.ros.org/navigation/Tutorials/RobotSetup/TF>
15. <http://wiki.ros.org/gmapping>
16. <http://wiki.ros.org/navigation>
17. [http://wiki.ros.org/turtlebot\\_navigation/Tutorials/indigo/Setup%20the%20Navigation%20Stack%20for%20TurtleBot](http://wiki.ros.org/turtlebot_navigation/Tutorials/indigo/Setup%20the%20Navigation%20Stack%20for%20TurtleBot)
18. [http://wiki.ros.org/husky\\_navigation/Tutorials/Husky%20Frontier%20Exploration%20Demo](http://wiki.ros.org/husky_navigation/Tutorials/Husky%20Frontier%20Exploration%20Demo)