TURTLEBOT3 WAFFLE PI: A FULLY AUTONOMOUS EXPLORATION ROBOT WITH OPEN SOURCE SOFTWARE

MINI PROJECT

by

BEERELLY SRINITHA (CS20B1004)
SIKANDER KATHAT (CS20B1020)

ಭಾರತೀಯ ಮಾಹಿತಿ ತಂತ್ರಜ್ಞಾನ ಸಂಸ್ಥೆ ರಾಯಚೂರು
भारतीय सूचना प्रौद्योगिकी संस्थान रायचूर
Indian Institute of Information Technology Raichur

to

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY
RAICHUR-584135, INDIA
May 2023

DECLARATION

We, **Beerelly Srinitha** (**Roll No: CS20B1004**) and **Sikander Kathat** (**Roll No: CS20B1020**), hereby declare that, this report entitled **'TURTLEBOT3 WAFFLE PI : A FULLY AUTONOMOUS EXPLORATION ROBOT WITH OPEN SOURCE SOFTWARE'** submitted to Indian Institute of Information Technology Raichur towards partial requirement of **Bachelor of Technology** in **Computer Science and Engineering** is an original work carried out by us under the supervision of **Dr. Suresh Chavhan** and has not formed the basis for the award of any degree or diploma, in this or any other institution or university. We have sincerely tried to uphold the academic ethics and honesty. Whenever an external information or statement or result is used then, that have been duly acknowledged and cited.

Raichur-584135 **Beerelly Srinitha**

May 2023 **Sikander Kathat**

CERTIFICATE

This is to certify that the work contained in this project report entitled **'TURTLEBOT3 WAFFLE PI : A FULLY AUTONOMOUS EXPLORATION ROBOT WITH OPEN SOURCE SOFTWARE'** submitted by **Beerelly Srinitha** (**Roll No: CS20B1004**) and **Sikander Kathat** (**Roll No: CS20B1020**) to the Indian Institute of Information Technology Raichur towards partial requirement of **Bachelor of Technology** in **Indian Institute of Information Technology Raichur** has been carried out by them under my supervision and that it has not been submitted elsewhere for the award of any degree.

Raichur-584135 Dr. Suresh Chavhan

May 2023 Project Supervisor

ABSTRACT

The proposed work investigates the feasibility of a mobile robot navigating and discovering its location in unknown environments, followed by the creation of maps of these navigated environments for future use.First, a real mobile robot named TurtleBot3 Wafflepi was used to achieve the simultaneous localization and mapping (SLAM) technique for a complex environment with obstacles of different sizes based on the Rviz library, which is built on the robot operating system (ROS) booted in Linux. It is possible to control the robot and perform this process remotely by using a Remote PC. Then, the map of the environment is drawn and can be further saved. This provides a database to display maps and use them at any time for navigation without the need to redraw the map. This map can be accessed by using an authentication process (username and password) supervised in Remote PC. Experiment results demonstrated the ability to build a map of an unknown location in a complex environment and use it for navigation tasks on a real mobile robot via remote control. It also showed the success of the process of storing the map for future use and the process of modifying the map. Before actual implementation on hardware, all the processes were first tested in simulation. The time-synchronized data feed is collected from the sensors of the robot. Then the collected data is further processed to perform controlling actions using actuators. The robot's performance is tested and optimized for the same algorithms by varying different parameters in the simulation and on a real robot.

Contents

List of Figures

Chapter 1

Introduction

TurtleBot3 is an open-source robot platform designed for education, research, and hobbyist purposes. It is developed by Robotis, a company that specializes in robotics and automation. The TurtleBot3 is the third generation of TurtleBot, and it is a highly customizable, modular, and affordable robot.TurtleBot3 comes in two different versions: Burger and Waffle. The Burger version is smaller and lighter than the Waffle version, making it more suitable for indoor environments. The Waffle version, on the other hand, is larger and heavier, making it better suited for outdoor environments. Both versions are equipped with a variety of sensors, including a 360-degree laser range finder, a camera, and an inertial measurement unit (IMU).One of the most notable features of the TurtleBot3 is its compatibility with the ROS (Robot Operating System) software. ROS is a popular open-source framework for building and programming robots. With ROS, users can easily program the TurtleBot3 to perform various tasks, such as navigation, mapping, and object detection.

The TurtleBot3 also has a strong community of developers and users who contribute to its development and provide support for others. There are many online resources available, including documentation, tutorials, and forums, making it easy for anyone to get started with the platform.Overall, the TurtleBot3 is a versatile and affordable robot platform that is ideal for students, researchers, and hobbyists who want to learn about robotics and explore new applications. Its compatibility with ROS and its modular design make it a popular choice for a wide range of projects and applications.

1.1 What Is TurtleBot?

TurtleBot is a robot especially designed for military purpose. There are many different kinds of military robots. They are Intelligence Surveillance and Reconnaissance (ISR), search and rescue robots, combat support, mine clearance, explosive ordnance disposal (EOD) and firefighting robots.TurtleBot is an Intelligence Surveillance andReconnaissance (ISR) robot which is used to monitor and help to gather data on the field.There are totally 3 versions based on the size of the turtlebots. They are Burger, Waffle and Waffle Pi.

1.2 History of TurtleBots

TurtleBot1 consists of a base, a battery, a power board, a Kinect sensor, a laptop with a dual core processor, and a hardware kit attaching everything together and adding future sensors. The first and the oldest TurtleBot was created at Willow Garage by Melonee Wise and Tully Foote in November

2010.

TurtleBot2 consists of a Kobuki base, a battery pack, a Kinect sensor, a laptop with a dual core processor, fast charger, charging dock, and a hardware mounting kit attaching everything together. Turtlebot2 was released on Oct 2012.

TurtleBot3 is made up of sectional plates that users can modify the shape according to their preference. Its available in three sizes: small size Burger and medium size Waffle, Waffle Pi. TurtleBot3 consists of a base, two Dynamixel motors, a 1800mAh battery pack, a 360 degree LIDAR,a camera), an SBC(single board computer: Raspberry PI 3 and Intel Joule 570x) and a hardware mounting kit attaching everything together and adding forthcoming sensors. Turtlebot3 was released on May 2017.

1.3 Limitations of the Existing System

Traditional method of identifying any malicious activity or recognizing arsonist is prone to disasters irrespective of preventive measures from being caught is taken into consideration. Even if technology is used for such things with improper security mechanisms it can lead to massive destruction, it can cause loss of lives, loss of sensitive data etc.

Traditional methods require lot of time and effort to master the skills and gather information and spies can be identified by the behaviour of the person and their belongings. For example, if a spy has to send evidence to his respective superior, he/she needs to acquire the evidence, that is, the hard copy or a photo of the evidence. Capturing pictures can be of high risk as it is easy to identify that a third person is capturing picture and can cause great threat to the life of the spy and to his/her organization. Storage is another drawback when it comes to preservation of evidence the traditional way/method. Considering the previous example given above, the pictures captured should be stored and secured in the right place which cannot be accessed by the opposite party and must be preserved in such way that it shouldn't get damaged until it is sent to its respective destination.

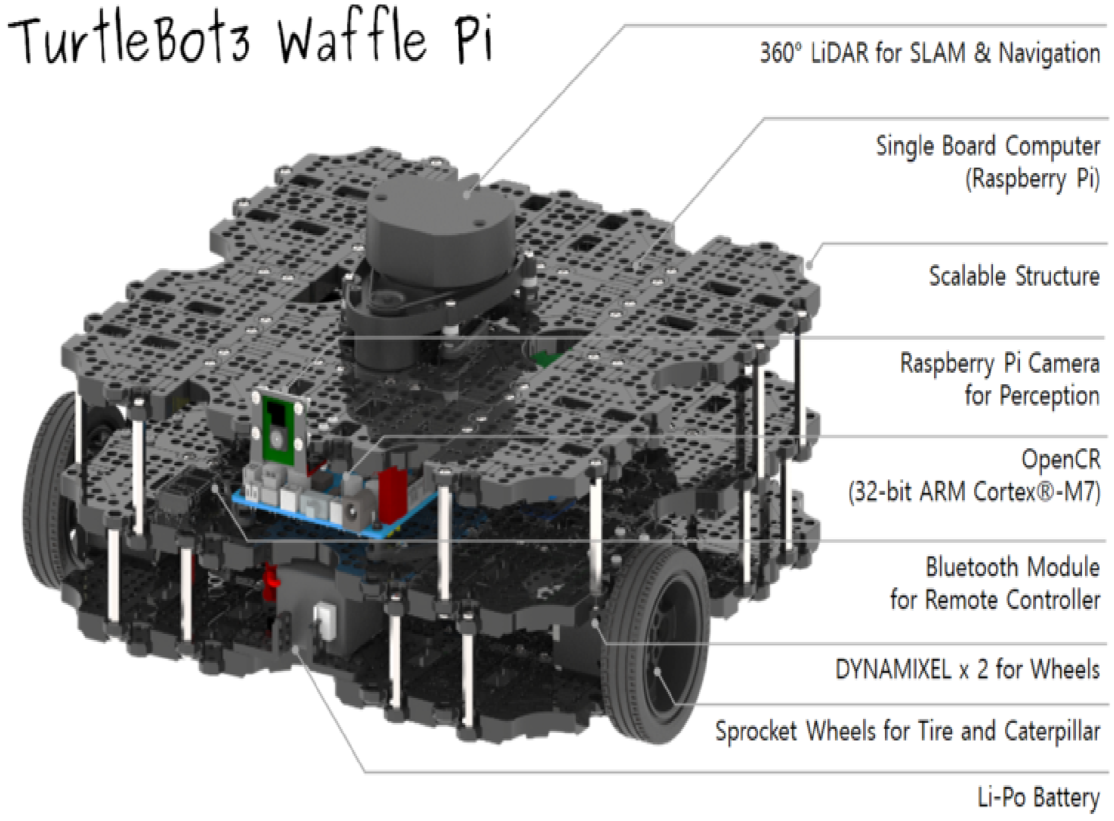

TurtleBot3 Waffle Pi
360° LiDAR for SLAM & Navigation
Single Board Computer (Raspberry Pi)
Scalable Structure
Raspberry Pi Camera for Perception
OpenCR (32-bit ARM Cortex®-M7)
Bluetooth Module for Remote Controller
DYNAMIXEL x 2 for Wheels
Sprocket Wheels for Tire and Caterpillar
Li-Po Battery

Figure 1.1: TurtleBot3 Waffle Pi

Chapter 2

PROJECT

2.1 PROJECT AIM

This project aims to develop a fully autonomous exploration robot with open-source software and LiDAR-based SLAM navigation.

2.2 OBJECTIVES

The objective is to successfully develop a robot that will be able to travel around the typical home environment without colliding with any possible human/pet/object moving nearby or dodging around. The robot must be able to also recognize the location once it has reached the desired destination. Therefore, the robot has to be able to perform the project's challenge of indoor navigation and indoor positioning as well as possible. To obtain the best possible results, an evaluation of different scenarios will be completed in order to observe the robot's functionality, and a wide research into and evaluation of different methods will be done in order to implement the most appropriate ones. The aspects that are going to be fulfilled for the project are:

-Select a suitable wheel-based robot for indoor use.

-Develop a SLAM-based software solution for exploration task

-Develop robot prototype with Turtlebot3 hardware and ROS Software

-Develop improvements for SLAM-based software to achieve fully autonomous and efficient exploration task performance

- Communication with the robot through an interface.

- Localization and navigation of the robot with no collisions (obstacle avoidance).

- Evaluate robot performance.

2.3 Development boards

A microcontroller (MCU) is an integrated circuit that can execute a set of instructions allocated to its memory. These instructions must be written using the appropriate programming language through an IDE (integrated development environment). Inside a microcontroller, the three main parts of a computer can be found: the central processing unit (CPU), the memory and the input/output peripherals. Different circuit boards include all the necessary elements to connect the peripherals to the inputs and outputs of the microcontroller [18]. These boards are named as microcontroller boards and they cannot be considered as a computer or as a simple board computer (SBC) due to several attributes that differentiate both. A basic desktop computer or a typical SBC is much more powerful than a simple MCU board. Generally, a computer can run multiple applications at the same time whilst most MCU boards can only run one program at a time. The fact of having an operating system (OS) and the I/O capabilities of the SBCs (HDMI, Ethernet or USB ports) are also differential factors when trying to draw a line separating single board computers and microcontroller boards. The MCUs run the code written on itself and nothing else, they are reliable, very economical and have low-power consumption. By attaching the external hardware to the I/O pins of the board, lots of different sensors and devices can be controlled, the reason why they are often used as a learning tool to take the first steps in electronics. However, plenty of robotics projects have been already developed using both SBC and MCU boards. Three of the main development boards available on the market will be be explained below, with information about their strengths and weaknesses.

2.3.1 Raspberry Pi

The Raspberry Pi is an SBC developed in the United Kingdom in order to stimulate informatics teaching in schools. It became more popular than expected, being used even in robotics applications. This small computer runs Linux and it is usually used to improve programming skills, nevertheless, the possibilities are almost unlimited. The Raspberry Pi community is huge and the vast quantity of theses and projects from different kinds of technology areas using this board is a significant help. The average price of the Raspberry Pi oscillates around 35€ depending on the model.

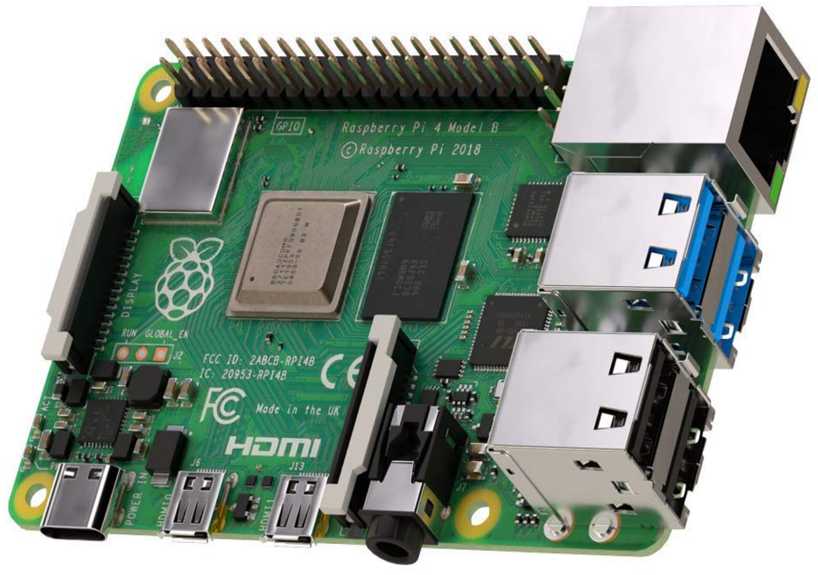


Figure 2.1: Raspberry-Pi

2.3.2 OPEN CR

OpenCR1.0 (Open-source Control module for ROS) is an open-source robot controller embedded with a powerful MCU from the ARM Cortex-M7 line-up. The hardware, software, schematics, PCB Gerber, BOM, and firmware source codes of the OpenCR1.0, the main controller used in the official ROS education platform TurtleBot3, are accessible and open to the public. Supports RS-485 and TTL to control the Dynamixels, and offers UART, CAN and a variety of other communication environment, development tools such as Arduino IDE are available as well. It has the advantage of being able to operate more powerfully when used with a host controller such as SBC (Single Board Computer). It provides various exclusive sources based on ROS, so that you can maximize the functions of OpenCR1.0 when using ROS.

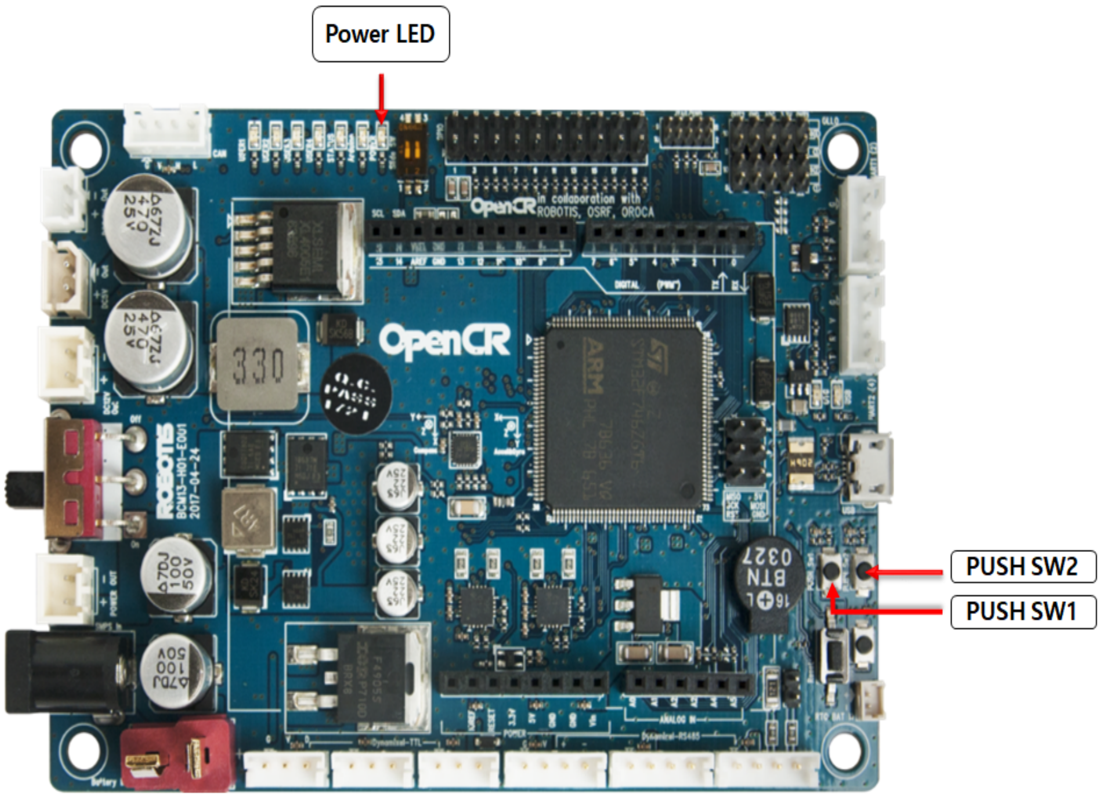


Figure 2.2: Open-CR Model

2.3.3 Lidar

The LiDAR sensor is used on most self-driving cars due to its good accuracy and longer range compared to conventional ultrasonic sensors. However, Li-

DAR's have some limitations that could cause trouble when moving through a map obtained with this sensor and so navigation should be done along with the information of other "obstacle avoidance" sensors. Different types of objects could not be detected. Transparent materials will not reflect the laser beam, letting it through. Some black surfaces may absorb part of the light received and create problems in the measurement. Also, if a mirror is not placed perpendicular to the laser beam, the laser will be deflected away from the LiDAR

Figure 2.3: LIDAR

Chapter 3

Features And Specifications

3.1 Features

World's Most Popular ROS Platform

TurtleBot is the most popular open-source robot for education and research. The new generation TurtleBot3 is a small, low-cost, fully programmable, ROS-based mobile robot. It is intended to be used for education, research, hobby, and product prototyping.

Affordable Cost

TurtleBot was developed to meet the cost-conscious needs of schools, laboratories, and companies. TurtleBot3 is the most affordable robot among the SLAM-able mobile robots equipped with a 360° Laser Distance Sensor LDS-01.

Small Size

The dimension of TurtleBot3 Burger is only 138mm x 178mm x 192mm (L x W x H). Its size is about 1/4 of the size of its predecessor. Imagine keeping TurtleBot3 in your backpack and develop your program and test it anywhere you go.

ROS Standard

The TurtleBot brand is managed by Open Robotics, which develops and maintains ROS. Nowadays, ROS has become the go-to platform for all the roboticists around the world. TurtleBot can be integrated with existing ROS-based robot components, but TurtleBot3 can be an affordable platform for whom want to get started learning ROS.

Extensibility

TurtleBot3 encourages users to customize its mechanical structure with some alternative options: open source embedded board (as a control board), computer and sensors. TurtleBot3 Burger is a two-wheeled differential drive type platform but it is able to be structurally and mechanically customized in many ways: Cars, Bikes, Trailers and so on. Extend your ideas beyond imagination with various SBC, sensors and motors on a scalable structure.

Modular Actuator for Mobile Robot

TurtleBot3 is able to get a precise spatial data by using 2 DYNAMIXEL's in the wheel joints. DYNAMIXEL XM series can be operated by one of 6 operating modes(XL series: 4 operating modes): Velocity control mode for wheels, Torque control mode or Position control mode for joint, etc. DYNAMIXEL can be used even to make a mobile manipulator which is light but can be precisely controlled with velocity, torque and position control. DYNAMIXEL is a core component that makes TurtleBot3 perfect. It is easy to assemble, maintain, replace and reconfigure.

Open Control Board for ROS

The control board is open-sourced in hardware wise and in software wise for ROS communication. The open source control board OpenCR1.0 is powerful enough to control not only DYNAMIXEL's but also ROBOTIS sensors that are frequently being used for basic recognition tasks in cost-effective way. Various sensors such as Touch sensor, Infrared sensors, Color sensor and a handful more are available. The OpenCR1.0 has an IMU sensor inside the board so that it can enhance precise control for countless applications. The board has 3.3V, 5V, 12V power supplies to reinforce the available computer device lineups.

Strong Sensor Lineups

TurtleBot3 Burger uses enhanced 360° LiDAR, 9-Axis Inertial Measurement Unit, and a precise encoder for your research and development. TurtleBot3 Waffle is equipped with an identical 360° LiDAR as well but additionally proposes a powerful Intel® RealSense™ with the recognition SDK. TurtleBot3 Waffle Pi uses high utilized Raspberry Pi Camera. This will be the best hardware solution for making a mobile robot.

Open Source

The hardware, firmware, and software of TurtleBot3 are open source which means that users are welcome to download, modify and share source codes. All components of TurtleBot3 are manufactured with injection molded plastic to achieve low cost, however, the 3D CAD data is also available for 3D printing. The 3D CAD data is released via Onshape which is a full-cloud 3D CAD editor. Users can get access with a web browser on desktop PC, laptops, and even portable devices. Onshape allows you to draw 3D models and assemble them with colleagues. Besides, for the users who want to make the OpenCR1.0 board by themselves, all details of the OpenCR1.0 board such as schematics, PCB Gerber files, BOM, and firmware source code are fully opened under the open-source licenses for users and the ROS community. You can modify downloaded source code and hardware to share it with your friends.

3.2 Hardware specifications

Hardware Specifications	
Items	Waffle Pi
Maximum translational velocity	0.26 m/s
Maximum rotational velocity	1.82 rad/s (104.27 deg/s)
Maximum payload	30kg
Size (L x W x H)	281mm x 306mm x 141mm
Weight (+ SBC + Battery + Sensors)	1.8kg
Threshold of climbing	10 mm or lower
Expected operating time	2h
Expected charging time	2h 30m
SBC (Single Board Computers)	Raspberry Pi
MCU	32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)

Hardware Specifications	
Items	Waffle Pi
Remote Controller	RC-100B + BT-410 Set (Bluetooth 4, BLE)
Actuator	XM430-W210
LDS(Laser Distance Sensor)	360 Laser Distance Sensor LDS-01 or LDS-02
Camera	Raspberry Pi Camera Module v2.1
IMU	Gyroscope 3 Axis,Accelerometer 3 Axis
Power connectors	3.3V / 800mA,5V / 4A,12V / 1A
Expansion pins	GPIO 18 pins,Arduino 32 pin
Peripheral	UART x3, CAN x1, SPI x1, I2C x1, ADC x5, 5pin OLLO x4
DYNAMIXEL ports	RS485 x 3, TTL x 3
Audio	Several programmable beep sequences
Programmable LEDs	User LED x 4
Status LEDs	Board status LED x 1,Arduino LED x 1,Power LED x 1
Buttons and Switches	Push buttons x 2, Reset button x 1, Dip switch x 2
Battery	Lithium polymer 11.1V 1800mAh / 19.98Wh 5C
PC connection	USB
Firmware upgrade	via USB / via JTAG
Power adapter (SMPS)	Input : 100-240V, AC 50/60Hz, 1.5A @max,Output : 12V DC, 5A

3.3 Technologies Included

The following technologies are included in this work, namely ROS, Gazebo, Lidar, TurtleBot3, and Raspberry Pi

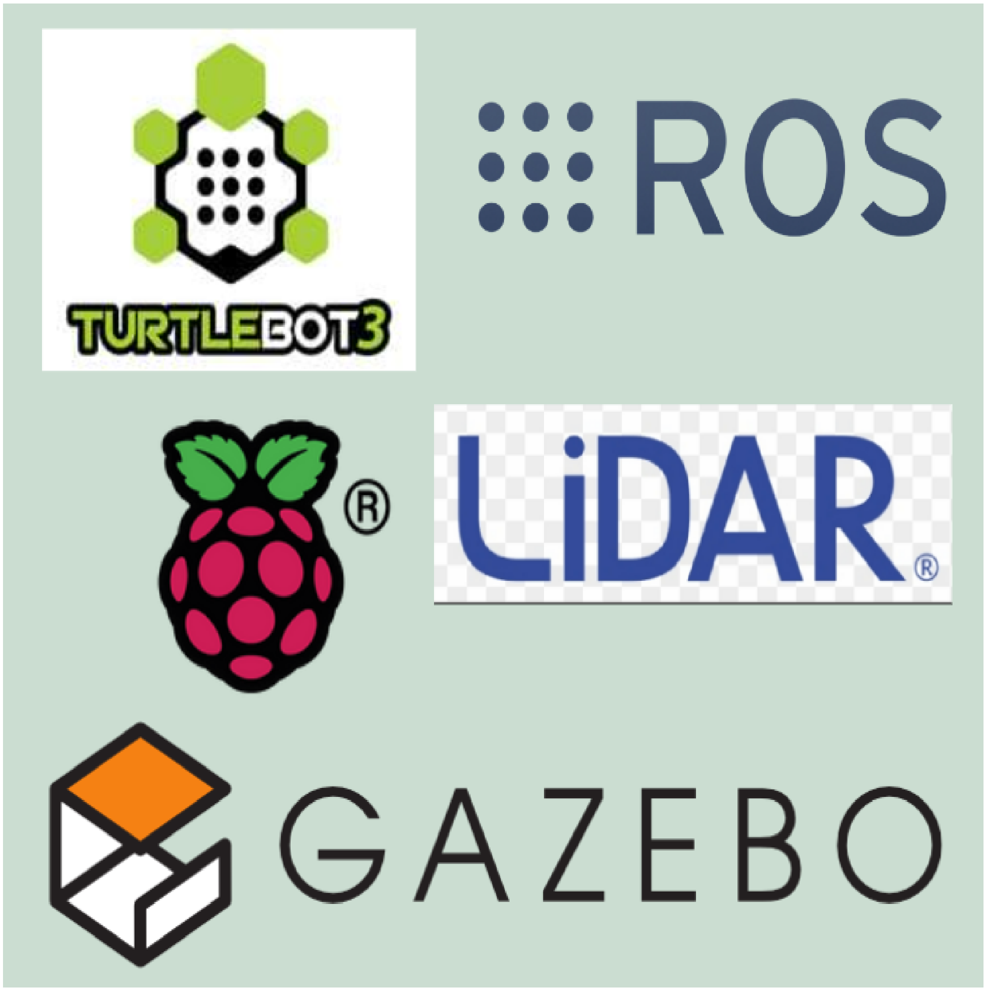


Figure 3.1: Technologies

3.4 Components / Parts List

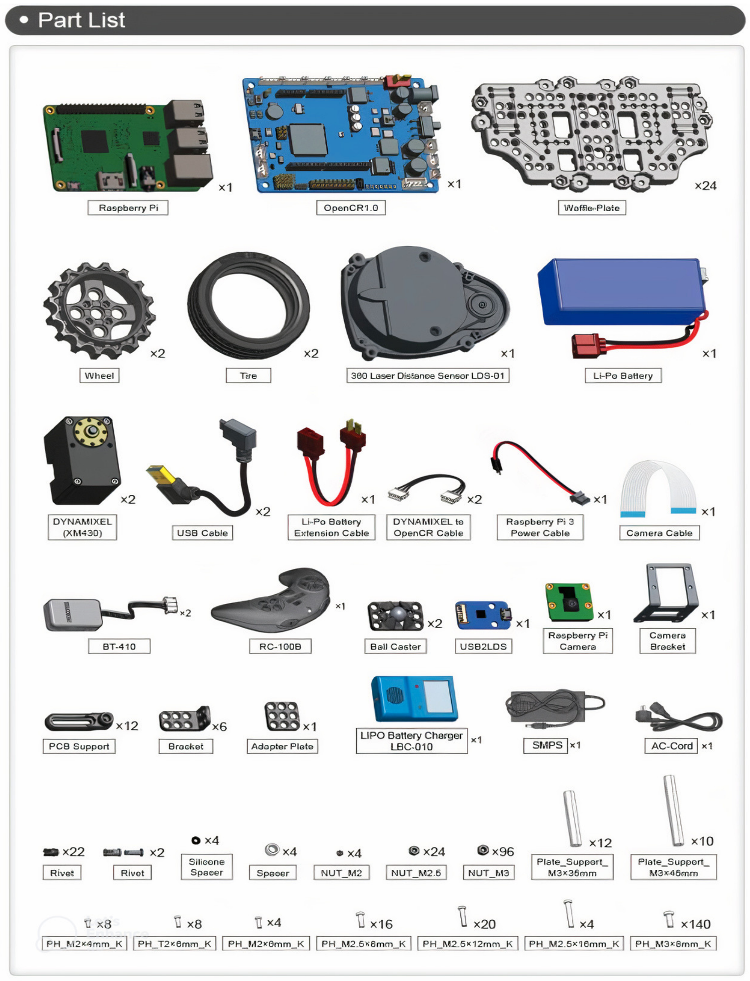


Figure 3.2: Parts List

Chapter 4

TECHNIQUES AND METHODS

4.1 SLAM

SLAM (simultaneous localization and mapping) is a method used for autonomous vehicles that lets you build a map and localize your vehicle in that map at the same time. SLAM algorithms allow the vehicle to map out unknown paths or environments. The map information is used to carry out tasks such as path planning and obstacle avoidance. SLAM has been the subject of technical research for many years. But with vast improvements in computer processing speed and the availability of low-cost sensors such as cameras and laser range finders, SLAM is now used for practical applications in a growing number of fields. There are two types of technology components used to achieve SLAM. The first type is sensor signal processing, including the front-end processing, which is largely dependent on the sensors used. The second type is pose-graph optimization, including the back-end processing.

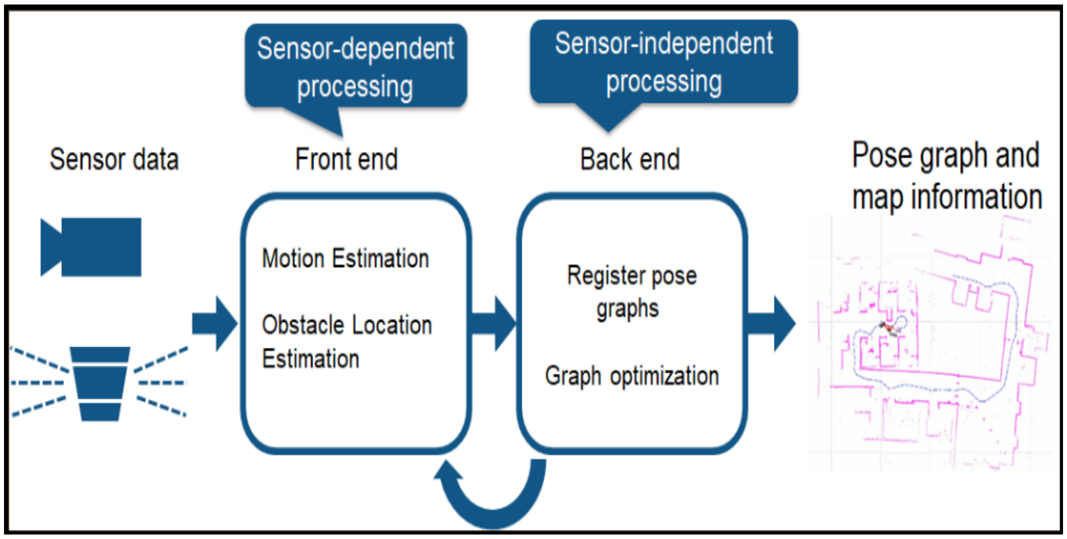

Sensor-dependent processing
Sensor-independent processing
Sensor data
Front end
Back end
Pose graph and map information
Motion Estimation
Obstacle Location Estimation
Register pose graphs
Graph optimization

Figure 4.1: SLAM WORKFLOW

4.1.1 Different methods of SLAM

VISUAL SLAM: Visual SLAM uses images acquired from cameras and other image sensors. Visual SLAM can be implemented at low cost with inexpensive cameras. In addition, since cameras provide a large volume of information, they can be used to detect landmarks.

LiDAR SLAM: Light detection and ranging is a method that primarily uses a laser sensor or distance sensor. Lasers are significantly more precise and are used for applications with high-speed moving vehicles such as self-driving cars and drones. The SLAM is a well-known feature of TurtleBot3 from its predecessors.TurtleBot3 will be exploring the unknown area of the map using teleoperation.

4.1.2 Packages

SLAM includes various useful packages like:

1) **gMapping**. It is used as a default SLAM method for turtlebot3. The gMapping package provides laser-based SLAM as a ROS node called slam-gmapping. Using slam-gmapping, we can create a 2-D grid map from laser and data collected by a mobile robot.

Few parameters which are a part of gmapping package and we are going to use them are:

maxUrange: This parameter is set the maximum usable range of the lidar sensor.

minimumScore: This can reduce errors in the expected position of the robot in a large area.

linearUpdate: When the robot translates longer distance than this value, it will run the scan process.

angularUpdate: When the robot rotates more than this value, it will run the scan process.

2) **Hector**. Hector SLAM uses only laser data collected to create the occupancy grid and it is able to work with laser mounted not planar to ground (as required by gMapping).

3) **Cartographer**. This graph-based system provides real-time SLAM in 2D and 3D across multiple platforms and sensor configurations.

4) **Frontier**. Frontier exploration involves dividing the whole environment into cells. The cells are used by three different classifiers (free, occupied, and unknown) and they are responsible for the representation of the boundary between the mapped and the unexplored areas.

4.2 MAPPING in TURTLEBOT3

The map is drawn based on the robot's scan information. These map data is drawn as the TurtleBot3 travels. After creating a complete map of desired area the map can be saved to the local drive for the later use.

The map uses two-dimensional Occupancy Grid Map (OGM). The saved map will have different coloured areas where white area is collision free area while black area is occupied and inaccessible area, and gray area represents the unknown area. This map is used for the Navigation.

4.3 NAVIGATION

Navigation is to move the robot from one location to the specified destination in a given environment. For this purpose, a map that contains geometry information of furniture, objects, and walls of the given environment is required. As described before, the map was created with the distance information obtained by the sensor and the pose information of the robot itself.(SLAM) The Navigation enables a robot to move from the current pose to the designated goal pose on the map by using the map, robot's encoder, IMU sensor, and distance sensor.

Chapter 5

DEVELOPMENT

5.1 Robot assembly

TurtleBot3 Waffle PI is developed by ROBOTIS, which is one of the leading manufacturers of robotic hardware. They specialize in the manufacture of robotic hardware and full platforms for use in all fields of study and industry, as well as educational robotics kits for all ages and skill levels. The TurtleBot3 Waffle PI kit includes a printed easy-to-follow assembly manual that explains step-by-step how to build the robot correctly. The e-Manual provides both a PDF to download and a set of videos explaining how to assemble it.

The shape of the Waffle plate leads to a highly versatile structure, allowing the user to customize the robot according to the project needs. OpenCR (Open-source Control module for ROS) is developed for ROS embedded systems to provide completely open-source hardware and software. OpenCR provides digital and analogue input/output pins and supports 12V, 5V, 3.3V power outputs for the different sensors and the Raspberry Pi SBC, where the Raspbian Stretch OS is located. The Raspberry Pi Model B communicates with the ROS master (computer) in order to run the required ROS packages remotely. The LiDAR component is positioned at the top of the robot to allow the robot to measure distance with a 360 range. The Li-Po battery has a capacity of 1800 mAh and is capable of providing a 2h 30m operating time to the robot with a charging time of 2h 30m. The hardware components are completely assembled by following the manual and the resultiong figure shows the final assembled robot.

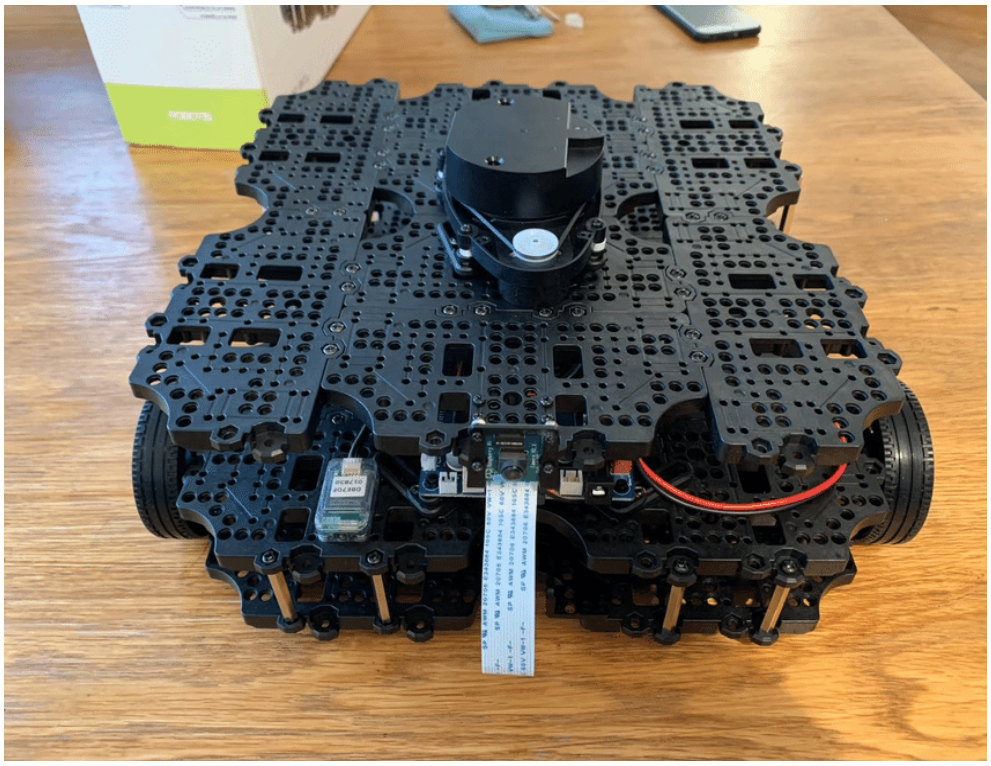


Figure 5.1: ASSEMBLED ROBOT

5.2 Software setup

TurtleBot3 is also developed by Open Robotics, it is in charge of the software and community activities. Open Robotics works with industry, academia, and government to create and support open software for use in robotics, from research and education to product development. Therefore, it is of great benefit for the project as a lot of documentation is available.

5.2.1 Bring up

Communication between devices was the first thing that was required for the robot to start. They consist of a Remote PC, which is the one used to communicate with the TurtleBot, and an SBC, which is used to communicate with the Remote PC. The first step was to set up the Remote PC by installing the Ubuntu 16.04 OS and ROS environment. The TurtleBot e-Manual [82] included the respective links for the OS download and the commands that were needed to run in the terminal window, as well as the required packages to be installed. ROS Kinetic Kame is the version that works best for the TurtleBot3, although it also supports Melodic Morenia. Secondly, ROS 1 requires network configuration to communicate between TurtleBot (Slave) and the Remote PC (Master). Once the communication had been configured, the SBC setup must be done. The Raspbian Stretch was the OS used on this project for the Raspberry Pi 3 Model B (TurtleBot3); it was flashed into a microSD card in order to be inserted in the robot. The dependency ROS and TurtleBot packages were needed to be installed and the network configuration on the TurtleBot was completed. After doing this, the IP address of the Remote PC was assigned as the ROS Master IP and the access to the Raspberry could be easily done from the Master using SSH (Secure Shell) protocol on the terminal window.

Finally, the OpenCR setup. The e-Manual provides two methods for uploading firmware although it is recommended to use shell script. Therefore, the Shell Script method was chosen, and the specified commands were used to connect OpenCR to Remote PC or connect OpenCR to TurtleBot PC.

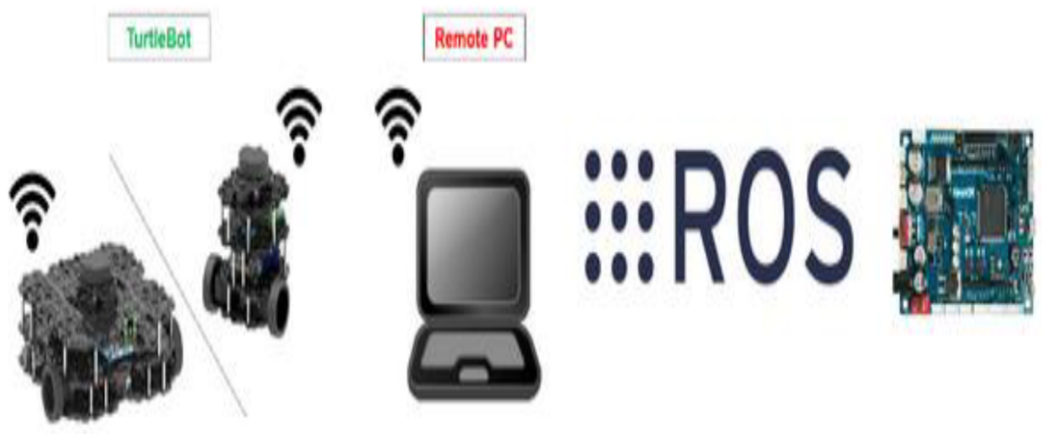


Figure 5.2: Communication between TurtleBot and Remote PC using ROS

5.2.2 Basic Operations

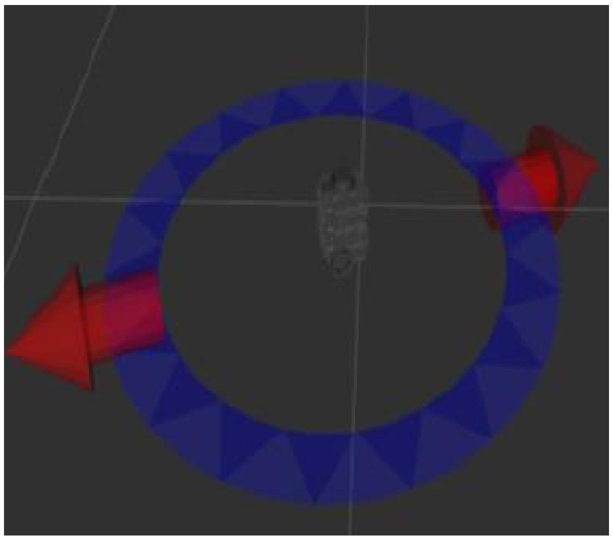


Figure 5.3: Interactive marker

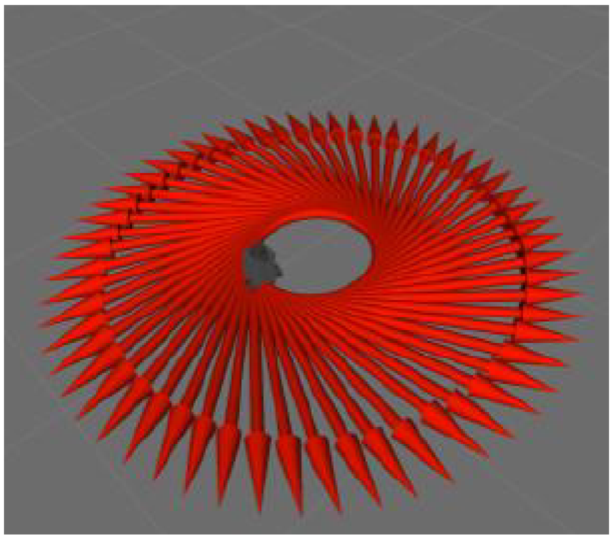
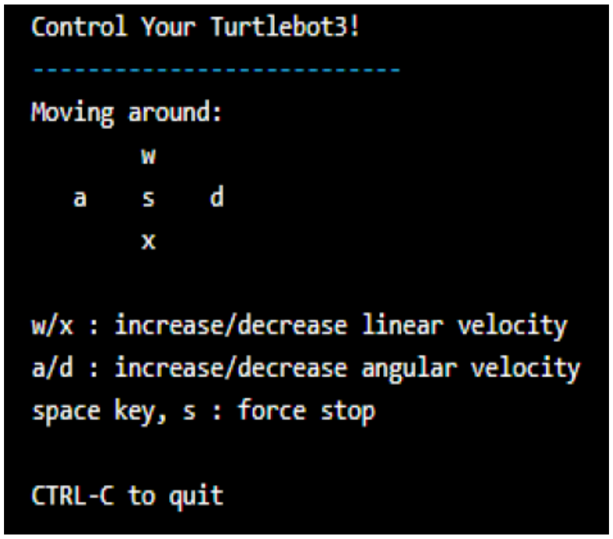


Figure 5.4: Circle pattern



```
Control Your Turtlebot3!
---------------------------
Moving around:
        w
   a    s    d
        x

w/x : increase/decrease linear velocity
a/d : increase/decrease angular velocity
space key, s : force stop

CTRL-C to quit
```

Figure 5.5: Teleportation command Window

Chapter 6

EVALUATION

6.1 SLAM methods

The four different SLAM methods were tested to evaluate which map works best for this project. Before launching the robot, the working area was defined by closing some corridors and adding two static obstacles to test the robot's obstacle avoidance when travelling from one point to another. To test each method, the teleoperation node was launched and driven through the same path until it reached every corner of the area.

Based on the visual comparison, it was determined that the Hector and Frontier methods produced a clearly defined result, whereas gMapping and Cartographer showed errors in the output. When Cartographer was launched, some erros appeared when mapping the environment. Furthermore, gMapping showed some errors as the method updates the representation of the map based on the robot's position. Drift error in position increases as the robot travels causing the curvature. This issue is solved with the Hector method as it does not depend on the odometry data, although the map has to be created using a much lower vehicle speed when turning into a new corridor. The frontier method encompasses the same issue when changing the robot's direction, but it also updates the map when it recognizes that the robot has revisited the same place. This causes the map to shifts slightly from the real environment due to drifts in localization along the whole path.

The results of the testing of four methods were captured as below:

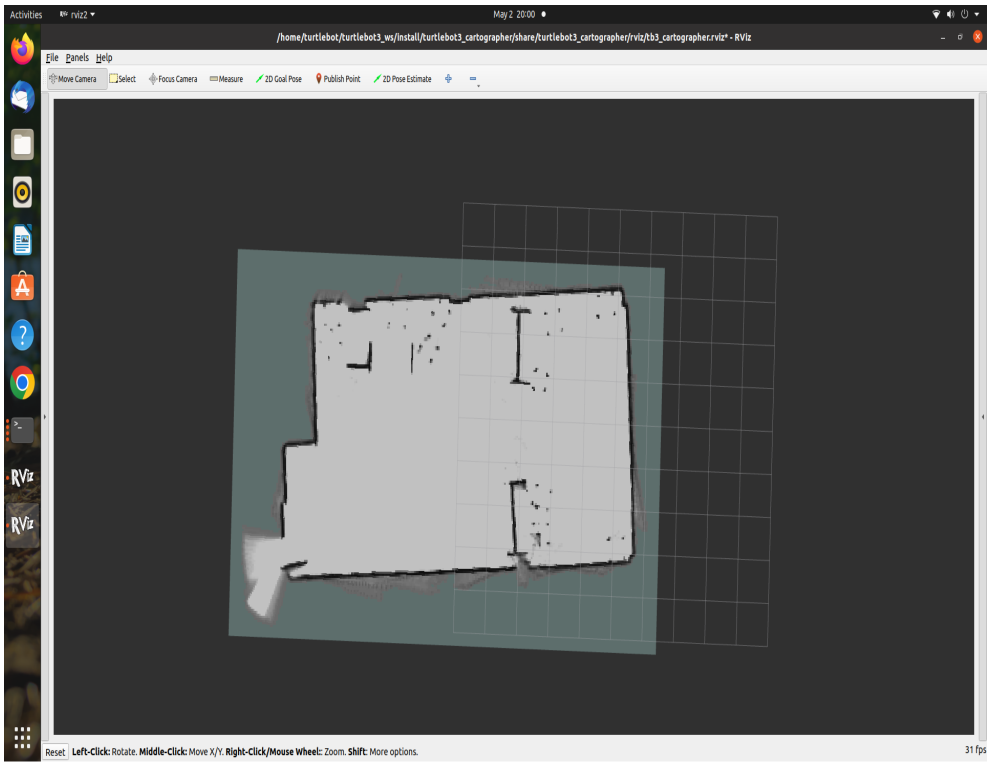

Activities
rviz2
May 2 20:00
/home/turtlebot/turtlebot3_ws/install/turtlebot3_cartographer/share/turtlebot3_cartographer/rviz/tb3_cartographer.rviz* - RViz
File Panels Help
Move Camera
Select
Focus Camera
Measure
2D Goal Pose
Publish Point
2D Pose Estimate
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
31 fps

Figure 6.1: gMapping

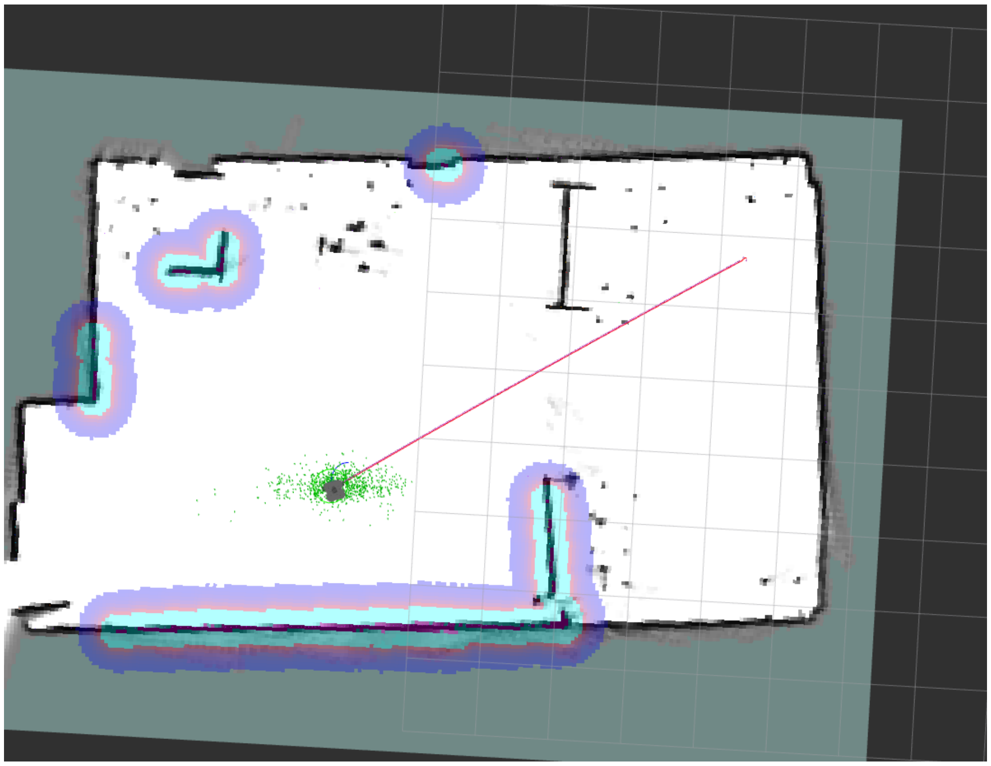


Figure 6.2: Cartographer

6.2 Navigation

The aim was to move the robot from one location to the specified destination inside the working area. To do so, both the map previously created, and the established initial point and orientation are used to test navigation. Ten different points on the map were set as goals to evaluate how the robot responded to it. The ability of the vehicle to reach its destination even with the presence of disturbance was also evaluated.

The robot succesfully reached the ten different established targets despite the obstacles presented along the vehicle's path.

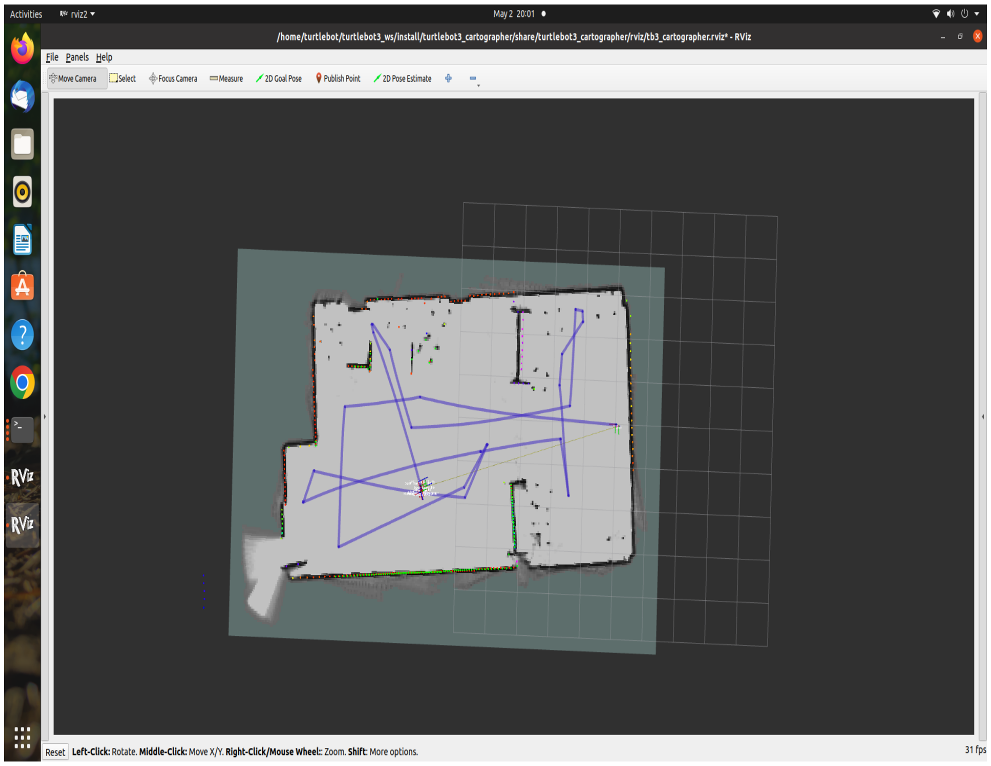

Activities
rviz2
May 2 20:01
/home/turtlebot/turtlebot3_ws/install/turtlebot3_cartographer/share/turtlebot3_cartographer/rviz/tb3_cartographer.rviz* - RViz
File
Panels
Help
Move Camera
Select
Focus Camera
Measure
2D Goal Pose
Publish Point
2D Pose Estimate
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
31 fps

Figure 6.3: Hector

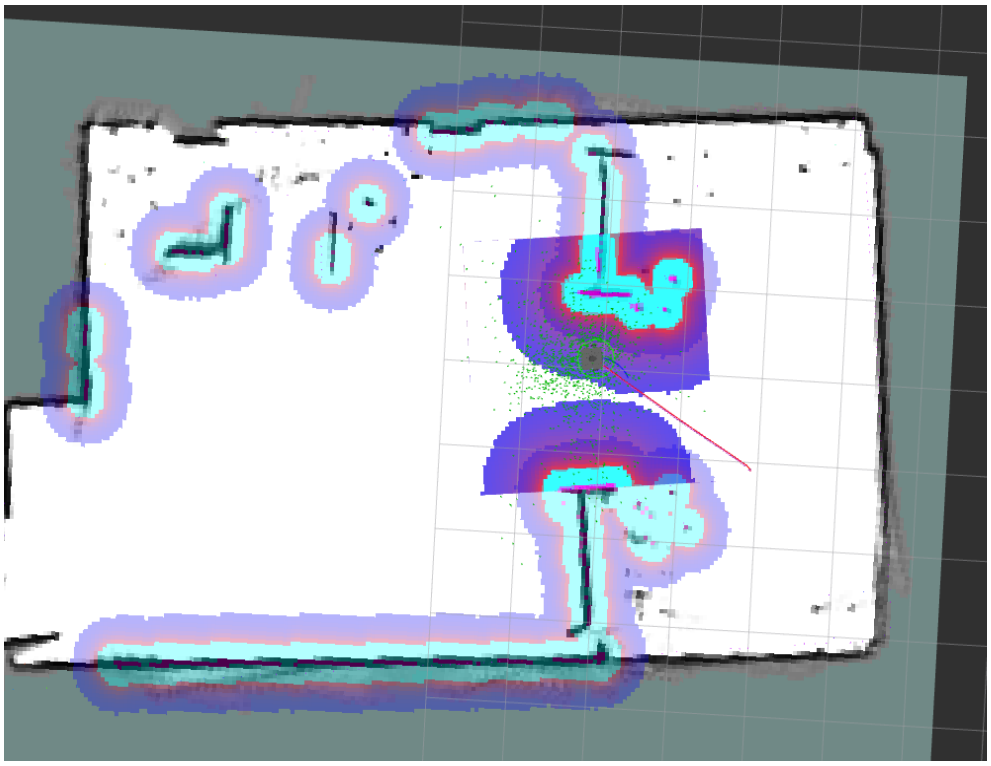
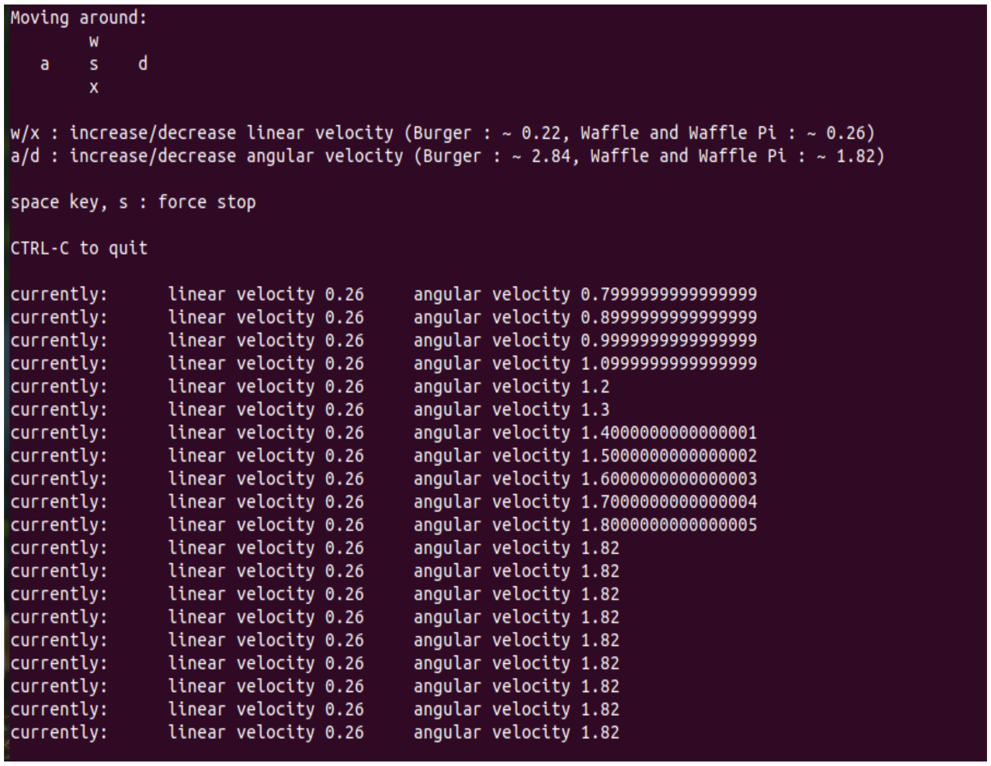


Figure 6.4: Frontier



```
Moving around:
        w
   a    s    d
        x

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

space key, s : force stop

CTRL-C to quit

currently:      linear velocity 0.26     angular velocity 0.7999999999999999
currently:      linear velocity 0.26     angular velocity 0.8999999999999999
currently:      linear velocity 0.26     angular velocity 0.9999999999999999
currently:      linear velocity 0.26     angular velocity 1.0999999999999999
currently:      linear velocity 0.26     angular velocity 1.2
currently:      linear velocity 0.26     angular velocity 1.3
currently:      linear velocity 0.26     angular velocity 1.4000000000000001
currently:      linear velocity 0.26     angular velocity 1.5000000000000002
currently:      linear velocity 0.26     angular velocity 1.6000000000000003
currently:      linear velocity 0.26     angular velocity 1.7000000000000004
currently:      linear velocity 0.26     angular velocity 1.8000000000000005
currently:      linear velocity 0.26     angular velocity 1.82
currently:      linear velocity 0.26     angular velocity 1.82
currently:      linear velocity 0.26     angular velocity 1.82
currently:      linear velocity 0.26     angular velocity 1.82
currently:      linear velocity 0.26     angular velocity 1.82
currently:      linear velocity 0.26     angular velocity 1.82
currently:      linear velocity 0.26     angular velocity 1.82
currently:      linear velocity 0.26     angular velocity 1.82
currently:      linear velocity 0.26     angular velocity 1.82
```

Figure 6.5: Navigation through terminal

Chapter 7

CONCLUSION AND FURTURE WORKS

The main contribution of this project is the development of an autonomous medical robot able to reach the patient in an indoor environment and to recognize it. This project addresses the problem of indoor navigation with the use of the LiDAR sensor on the TurtleBot3 and indoor positioning using Bluetooth technology. ROS was the primary software for the development of the whole project. This project undertakes indoor navigation using SLAM. Indoor positioning can be used on a map of the environment obtained using SLAM methods where the robot will travel and self-locate with a position error of 0.3 metres.

This project has a great potential for growth. Although all the objectives have been successfully completed, some future improvements can be considered in order to improve this robot as a real autonomous robot.Position Control, Patrol,Follower,Panorama,Auto Parking and some other objectives can be implemented further more in future by using the developed robot and evaluate the performance accordingly.