Autonomous Security Robot using Jetson Nano

by Krishna Vamsee Duggaraju, Master of Science

A Design Project Submitted in Partial
Fulfillment of the Requirements
for the Degree of
Master of Science
in the field of Mechanical Engineering

Advisory Committee:

Dr. Nima Lotfi
Dr. Mingshao Zhang
Dr. Arman Dabiri

Graduate School
Southern Illinois University Edwardsville
May 2, 2022

ABSTRACT

MASTER OF SCIENCE

DEPARTMENT OF MECHANICAL ENGINEERING

SOUTHERN ILLINOIS UNIVERSITY EDWARDSVILLE

DESIGN PROJECT REPORT

Project Title: Developing an Autonomous security robot using Jetson Nano.

Author: Krishna Vamsee Duggaraju

Abstract: The goal of this project report is to build an Autonomous Security Robot that can travel in indoors and survey the area using the Jetson Nano as its brain, Lidar as its primary sensor and a Camera module used for Object detection. The OS used is ROS, and the SLAM principle is used for 2D map generation. The ROS Navigation stack is utilized for navigation since it includes the Adaptive Monte-Carlo Localization (amcl) node. This robot can stream, detect and recognize different objects using a Raspberry Pi V2 camera and pre-trained DNN models on 90-class MS-COCO dataset.

ACKNOWLEDGEMENTS

Firstly, my deepest thanks go out to my academic advisor. Dr. Nima Lotfi for his valuable encouragement, guidance and mentoring in carrying out this project work. Also, I would like to take the opportunity to recognize and credit my parents, family members and friends for their support.

TABLE OF CONTENTS

ABSTRACT 2

ACKNOWLEDGEMENTS 3

LIST OF FIGURES 5

LIST OF TABLES 5

Chapter

1. INTRODUCTION 6

2. OVERVIEW. 7
   - 2.1. Perception 8
     - 2.1.1. Localization 8
     - 2.1.2. Simultaneous Localization and Mapping 9
   - 2.2. Path Planning 11
     - 2.2.1. Global Path Planning 11
     - 2.2.2. Local Path Planning 12
   - 2.3. Object Detection 12
     - 2.3.1. Two-Stage Detectors 13
     - 2.3.2. Single-Stage Detectors 14

3. METHODOLOGY 15
   - 3.1. SLAM Gmapping 16
   - 3.2. Amcl 17
   - 3.3. Path Planning 17
   - 3.4. Kinematic Model of Differential Drive Robot 18
   - 3.5. Remote Control of Mobile Robot 19
   - 3.6. Object Detection using SSD 20

4. IMPLEMENTATION 20
   - 4.1. Design 20
   - 4.2. Communication Protocol 22
   - 4.3. Software Architecture 22

5. EXPERIMENTATION 24

6. RESULTS 25

7. CONCLUSION 29

8. APPENDIX 29

REFERENCES. 30

LIST OF FIGURES

Figure Page

2.1 Control scheme of robot Navigation 7

3.1 Overview of robot setup 16

3.2 Two wheeled differential drive robot 18

4.1 Autonomous Mobile Robot 21

4.2 ROS /tf transforms of the robot 23

4.3 ROS nodes graphs 24

5.1 Layout of Autonomous Mobile Robot 24

5.2 Map generated using SLAM Gmapping 25

6.1 Robot at Point-A 26

6.2 Robot at Point-A in rviz 26

6.3 Robot generating a trajectory 26

6.4 Robot navigating to Point-B 27

6.5 Robot reached Point-B 27

6.6 Obstacle added to the environment 28

6.7 Trajectory generated avoiding the obstacle 28

6.8 Robot reached Point-B safely 28

6.9 Object detection using SSD 29

LIST OF TABLES

Table Page

3.1 Parameters of Differential drive robot 19

6.1 Shows the results obtained till now 25

6.2 Shows the final result 29

1. INTRODUCTION

The study of safety robots with keen models to battle crime has provoked the curiosity of various educational institutions and organizations. Barrier evaluations, intruder detection, developing virtual terrains or maps, stopping explosives, and recognizing odd human ways of behaving have all been finished with autonomous mobile robots for quite a long time. In hazardous areas, such robots have been built to counter dangers, decrease worker risks, and reduce staffing needs. In this way, security robots serve as mobile security agents tasked with safeguarding properties, precious things, and people. However, despite the fact that such robots can save money, they will never be able to replace a human security expert due to the high cost of humans [1]. In this project, we have developed a mobile robot that can navigate autonomously while surveying and detecting the area.

Robots come in a variety of shapes and sizes. It may be classified into three primary groups. First, a robot that is manually operated by a human being. Second, a semi-automatic robot is one that can make decisions on its own while having certain parts controlled by humans. Third, an automated robot that can work on its own [2]. Robots, such as delivery robots, food servers, and security robots, can navigate and avoid obstacles on their own inside the map.

A controller, which serves as the robot's brain, as well as a power management system, sensors, and actuators, are all included in mobile robots. Sensors vary depending on the robot's requirements and use, while actuators in mobile robots are usually DC motors. Autonomous Mobile Robots are mobile robots that move around unattended in their surroundings to fulfill a specific purpose (AMR). We can accomplish this by equipping the robot with sensors that allow it to sense its surroundings, as well as artificial intelligence and machine learning algorithms that allow it to plan its path and navigate accordingly. They can avoid obstacles in their path by re-routing its path using machine learning algorithms [3].

Due to its vast variety of applications and recent technical developments, object detection have attained an interest in recent years. Security monitoring, self-driving cars, traffic surveillance, drone scene analysis, and robotic vision are just a few of the real-world uses. Among the several factors and activities that have contributed to the rapid proliferation of object recognition systems, the development of deep convolutional neural networks and GPU processing capabilities should be acknowledged as major contributions. Deep learning is presently used extensively in all aspects of computer vision.

In past, many mobile robots have been developed. For example, during the Pathfinder mission to investigate Mars in summer 1997, the mobile robot Sojourner was utilized; Plustech built the first application-driven walking robot; Pioneer, a robot designed to explore the Sarcophagus at Chernobyl; HELPMATE, a mobile robot used in hospitals for transportation tasks; Alfred Kärcher GmbH & Co., Germany's BR 700 industrial cleaning robot and the RoboCleaner RC 3000 consumer robot; Alfred Kärcher GmbH & Co., Germany's BR 700 industrial cleaning robot and the RoboCleaner RC 3000 consumer robot; Alfred Kärcher GmbH & Co. Alice is one of the tiniest completely autonomous robots, while KHEPERA is a small mobile robot for research and education. It is approximately 2x2x2 cm, has an 8-hour battery life, and navigates via infrared distance sensors, tactile whiskers, or even a small camera [3].

Embedded systems and microprocessors have made low-cost solutions possible as a result of recent technological advancements. Different sensors and configurations are employed depending on the application, and different algorithms are developed. They can be controlled via IP addresses or IoT technology through the internet [4].

In this project, the security robot uses a Nvidia Jetson Nano microcomputer with ROS melodic OS and an Arduino UNO microcontroller to control two 12V DC motors with an L298N motor driver and a 12V power source. The Lidar sensor allows it to create a map using SLAM Gmapping algorithm and navigate in that environment using the ROS Navigation stack. For surveying, the robot uses a Monocular RGB camera as a sensor and pre-trained Deep neural network models for Object detection.

2. OVERVIEW

For mobile robot navigation, there are 4 main building blocks. They are Perception, localization, path planning, and motion control. The mobile robot's control scheme is as follows [3]:

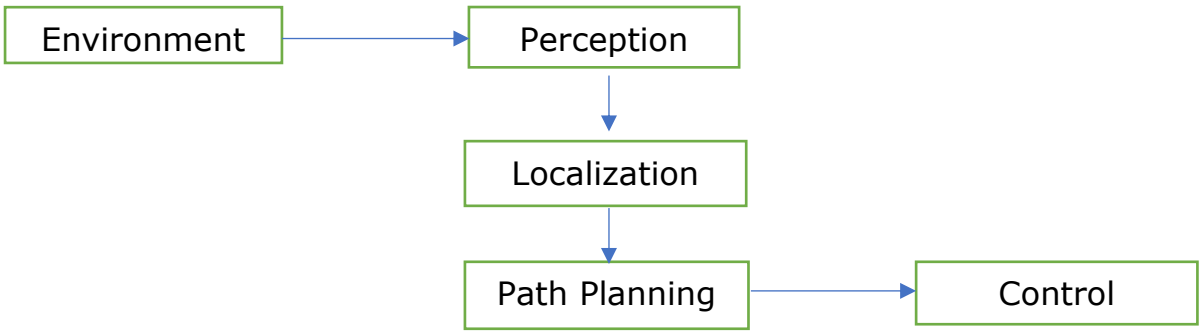

Fig. 2.1. Control scheme of Robot navigation

2.1 PERCEPTION

For any advanced robot, perception is an important feature. Robots ranging from automated vacuum cleaners to self-driving automobiles, robot arms to humanoid robots all require this feature. This perception is divided into two parts: understanding the surroundings and navigating through it. Knowing the surroundings entails constructing a map based on several sensors.

2.1.1 LOCALIZATION

Localization refers to a robot that can move and can keep track of its position and orientation while moving in a given environment. This is done with the help of odometry and other sensor data such as GPS, Laser sensor, Camera, and so on. This procedure is divided into two parts:

- Prediction Update
- Perception Update

The robot uses odometry data to estimate its position in Prediction update. However, as a result of the error due to odometry, the robot's uncertainty increases and gets accumulated over time.

The onboard sensors, such as the camera, laser sensor, and GPS, rectify this uncertainty, which is referred to as Perception Update. The robot calculates its position and distance from the right wall using a range finder. As a result of the difference with the location from the Prediction update phase, the correction occurs in the Perception update phase. As a result, the error is reduced. The following is a mathematical representation of this:

Let the Robot's estimated position be $\boldsymbol{p} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{\theta} \end{bmatrix}$

The position of the differential-drive at a given time is estimated from it's formal position by adding the incremental distances $(\Delta x: \Delta y: \Delta\theta)$, where,

$$\boldsymbol{\Delta x = \Delta s \cos\left(\theta + \frac{\Delta\theta}{2}\right)}$$

$$\boldsymbol{\Delta y = \Delta s \, sin\left(\theta + \frac{\Delta\theta}{2}\right)}$$

$$\boldsymbol{\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b}}$$

$$\boldsymbol{\Delta s = \frac{\Delta s_r + \Delta s_l}{2}}$$

The new position p’ can be calculated as

$$p' = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} \cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2} sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix}$$

Where, distance between the wheels is denoted by *b*.

For robot localization, there are two alternative ways. They are the probabilistic and autonomous map-building approaches, respectively. Markov localization, Kalman filter-based localization, and other techniques fall within the first group. Second, SLAM techniques for automatic map building during localization and the use of RFID are discussed.

Markov Localization

It addresses three major issues. Markov localization can locate itself from an unknown location, track several positions, and efficiency. However, in order to update the probability of potential positions, the state-space must be given in a discrete manner. This representation is a map or a grid, and the memory required for map size is limited.

Kalman Filter based Localization

It is an ideal sensor fusion strategy that can efficiently and precisely solve the position tracking problem. It tracks the robot from its initial point. It's a variant of Markov localization that doesn't rely on an arbitrary function. Rather, Gaussian is used to express the robot's assumption.

There are no of additional probabilistic ways, but they all require a hand-drawn environment map with various features such as beacons, walls and other landmarks that are used for localization. Localization is extremely challenging in dynamically changing contexts, as it is constantly evolving. Automatic map-building approaches are used to overcome this challenge. According to it, the robot starts at a random location and explores the surroundings on its own, using its sensors to create a map. This aids the robot in locating itself in relation to the map [7].

2.1.2 SIMULTANEOUS LOCALIZATION AND MAPPING

SLAM recovers both the environment map and the robot path using data from the robot's primary and secondary sensors. The information gathered is based on the robot's displacement as determined by odometry, as well as features such as lines, edges, corners, and planes. These characteristics are retrieved from images captured by an ultrasonic sensor, a lidar or a

camera. In unfamiliar situations, SLAM technology makes the robot to create or update map while simultaneously tracking their locations. It's written in a probabilistic manner, with the current position and robot map computed using a probability distribution. The method of knowing the surroundings at a spot without any prior knowledge is the goal of SLAM. This calculates the robot's trajectory and maps its surroundings. These observations are recorded and updated at regular intervals, which lowers trajectory inaccuracies. The difficulties are described mathematically as follows.

Assuming there is given a progression of controls $\boldsymbol{u_t}$ and sensor perceptions $\boldsymbol{o_t}$ throughout a discrete time steps t, the SLAM issue is to process an estimate of the agent's state $\boldsymbol{x_t}$ and a map of the environment. All are generally probabilistic, so the goal is to calculate:

$$\boldsymbol{P(m_{t+1}, x_{t+1}|o_{1:t+1}, u_{1:t})}$$

By applying Baye's rule gives a structure for consecutively updating the rear ends, given a map and a change function:

$$\boldsymbol{P(x_t, x_{t+1}|)},$$

$$P(x_t|o_{1:t}, u_{1:t}, m_t) = \sum_{m_{t-1}} P(o_t|x_t, m_t, u_{1:t}) \sum_{x_{t-1}} P\,(x_t, x_{t+1}|) P(x_{t-1}|m_{t-1}, o_{1:t-1}, u_{1:t})/Z$$

Similarly, the map is sequentially updated by

$$\boldsymbol{P(m_t|x_t, o_{1:t}, u_{1:t}) = \sum_{x_t} \sum_{m_t} P(m_t|x_t, m_{t-1}, o_t, u_{1:t}) P\left(m_{t-1}, x_{t,}|o_{1:t-1}, m_{t-1}, u_{1:t}\right)}$$

Kalman filters and Particle filters are the techniques used to approximate these equations. They provide an estimate of the probability function for the robot's pose as well as the map's parameters.

SLAM can be used with a variety of sensors, each with its own set of algorithms. There are two sorts of sensor models. They are raw-data and Landmark-based techniques. The position is estimated using landmarks and GPS and Wi-Fi connection points. Whereas raw-data uses $\boldsymbol{P(o_t, x_t|)}$ as function of location.

Optical sensors include 1D or 2D or 3D LiDAR sensors, 2D or 3D sonars, and one or more two-dimensional cameras. Researchers have proposed a variety of SLAM approaches that use these sensors. It largely relies on the robot sensor you're using. We employ Visual SLAM, ORB-SLAM, ORB-SLAM2, ORB-SLAM3, EKF-SLAM, RGBD-SLAM, and other algorithms if it's a

camera. If Lidar is used, Hector SLAM, Gmapping, and Cartographer are used. Gmapping uses the particle filter pairing procedure, while Hector SLAM uses the scan matching technique to build the 2D map. The scan matching approach is used with loop detection by Cartographer. For our interest, we are using SLAM Gmapping algorithm in our project [18,19].

2.2 PATH PLANNING

It is the process of determining an optimal path from a robot's current location to its desired location. This is only possible if you know the robot's present and desired positions. The above-mentioned factors must be adequately defined in order to solve this challenge.

Path planning, in theory, solves three difficulties. They're aiming for goal position, avoiding obstacles, and finding the optimal path. However, it all depends on the precision of the sensors utilized, which generates errors in the data. As a result, two new path planning methods have been created to address this issue. Local path planning, which focuses on avoiding obstacles, and global path planning, which seeks for the quickest path. We can obtain accurate navigation by integrating both of them [8].

2.2.1 GLOBAL PATH PLANNING

It generates a global plan for the robot to accomplish a task. For path planning, it employs the *A** algorithm. It uses the cost function used for path to the destination is written mathematically as,

$$\boldsymbol{f(n) = g(n) + h(n),}$$

Where n is the next target point, *f(n)* is the valuation function of node, *g(n)* is the starting point's real cost to the current point, and *h(n)* is the current node's predicted cost to the end point. The performance of the algorithm is determined by the value of *h(n).* The Manhattan distance between two focuses (x_1, y_1) and(x_2, y_2) is utilized in the *A** method and is as follows:

$$\boldsymbol{D_{Manhattan} = |x_1 - x_2| + |y_1 - y_2|}$$

2.2.2 LOCAL PATH PLANNING

The DWA procedure is utilized for local route planning, and it requires robot mathematical simulation calculations on the robot's path. In the y-axis direction, the two-wheeled differential drive robot doesn't move. The robot travels a little distance at speed v and at a point t to the x-axis throughout

some period of time; hence, the development increases x and y of the robot, can be acquired:

$$\boldsymbol{\Delta x = x + v\Delta t\cos(\theta_t)},$$

$$\boldsymbol{\Delta y = y + v\Delta t\sin(\theta_t)}.$$

The robot trajectory is given by

$$\boldsymbol{x_{t+1} = x_t + v\Delta t\cos(\theta_t)},$$

$$\boldsymbol{y_{t+1} = y_t + v\Delta t\sin(\theta_t)},$$

$$\boldsymbol{\theta_{t+1} = \theta_t + \omega\Delta t},$$

Where angular velocity of the robot is denoted by w.

The determination of the ideal direction depends on a direction that satisfies the speed necessity, and the evaluation function is given by:

$$\boldsymbol{G(v,\omega) = \sigma\big(\alpha head(v,\omega)\big) + \beta\,\mathrm{dis}(v,\omega) + \gamma vel(0,\omega)},$$

where $head(v,\omega)$ denotes the difference of angle between the predicted end of the path and the goal; *dis(v,w),* as mentioned before, is the minimal length from the obstruction to the intended trajectory; The instant speed evaluation is denoted by *vel(v,w);* the smoothing function is denoted by **σ** (.); and the evaluation coefficients are denoted by α, β, γ>0. [9,18].

2.3 OBJECT DETECTION

Object detection is a computer vision technique that deals with photographs and videos for examples for recognizing and classifying things of a specific class (like individuals, structures, and vehicles). A portion of the well-informed object discovery spaces incorporate multi-classes identification, edge location, notable thing recognition, act recognition, scene text identification, face identification, and walker discovery. Object detection is currently utilized in an assortment of fields, including security, military, transportation, medication, and life sciences. Previous area explicit object detectors are divided into two classifications: two-stage detector and single-stage detectors. Two-stage detectors produce great localization and object detection accuracy, while one-stage detectors give great speed. The RoI (Region of Interest) pooling layer isolates two-stage locators into two phases [20].

2.3.1 TWO-STAGE DETECTORS

1) R-CNN

R-CNN is an area-based CNN identifier. The R-CNN indicator is comprising of four sections. The primary module generates region ideas that are class free. The subsequent module takes every area idea and extracts a fixed-length feature vector. A series of class-explicit straight SVMs is utilized in the third module to order the items in a single picture. For exact bounding box prediction, the last module is a bounding box regressor.

2) Fast R-CNN

R-CNN consumes a lot of time to group SVMs since it does a ConvNet forward pass for every area suggestion without sharing calculation. Fast R-CNN takes highlights from an entire input picture and then gives the fixed size features to bounding box regression and classification fully connected layer through RoI pooling layer. Another change is that Fast R-CNN separates a decent size include map from region recommendations of different sizes utilizing a RoI pooling layer. This interaction doesn't require distorting regions and recoveries the spatial data of area proposition's qualities.

3) Faster R-CNN

Fast R-CNN proposes RoI utilizing selective search, which is slow and requires same amount of time. Faster R-CNN replaces it with another RPN (regional proposal network), which is a full convolutional network that predicts area recommendations with a wide variety of sizes and viewpoint proportions. Since it shares fully-image convolutional features and a common arrangement of convolutional layers with the detection network, RPN speeds up the generation of region suggestions.

4) Mask R-CNN

Mask R-CNN uses Faster R-CNN with ResNet – FPN network. FPN contains both a bottom up and a hierarchical pathway with lateral connections. A back bone ConvNet computes feature maps at many scales with a scaling step of two. The hierarchical procedure gives improved goal highlights by up examining topographically coarser however semantically more grounded include maps from higher levels. At start, the result of the last convolutional layer catches the top pyramid include mappings. All links joins feature maps from a similar spatial size bottom up and hierarchical pathways. While include maps arrive in a variety of sizes, the 11 convolutional layers can change the size. After a lateral connection operation, another pyramid level will frame, and expectations will be created independently on each level. The feature pyramid network gathers key qualities in light of the fact that

higher-goal include maps are valuable for identifying little items and lower-goal highlight maps are rich in semantic data.

2.3.2 SINGLE-STAGE DETECTORS

1) YOLO

YOLO (You Look Only Once) is a one stage object identifier. The YOLO pipeline splits the info picture into a SxS matrix, with every framework cell tasked with identifying the item whose center falls inside it. P(object) determines the probability of the box having an item, and IOU (intersection over union) shows how exactly the box holding that object is. B bounding boxes (x,y,w,h) and confidence scores for them are predicted for every grid cell, as well as C-dimensional conditional class probabilities for C categories. The feature extraction network has 24 convolutional layers and two completely connected layers. The main 20 convolutional layers, a normal pooling layer, and a fully connected layer are utilized while pre-training on the ImageNet dataset. For more performance, the whole network is utilized for detection. There are different versions of YOLO v2, YOLO v3 which utilizes multi-mark characterization to adjust more perplexing datasets containing additional covering names.

2) SSD

SSD, a single shot detector for different classifications in a single stage that predicts class scores and box offsets for a given arrangement of default bounding boxes of varying scales at each point in various feature maps with fluctuated scales. Each feature map's default bounding boxes have unmistakable perspective proportions and sizes. The scale of default bounding box is figured in various feature maps with a normal separating between the highest and lowest layers, where each feature map figures out how to be sensitive to the size of the articles. It forecasts the offsets and confidences for all item classifications for each default box.

3) DSSD

Deconvolutional Single Shot Detector is a modified adaptation of SSD (Single Shot Detector) that incorporates a forecast module and uses ResNet-101 as its base. To enhance includes, the deconvolution module raises the resolution of feature maps. Every convolution layer, which is followed by a prediction module, is intended to predict a wide scope of objects of different sizes.

4) RetinaNet

The R-CNN object detector is a conventional two-stage object detector. The first stage generates a small number of area ideas, while the second stage sorts each possible site into categories. Two-stage detectors achieve more

precision than single stage detectors that suggest a dense list of candidate locations. Focal loss focuses on the difficult training instances while avoiding the large no of easy negative examples that would otherwise overwhelm the detector during training. It takes the speed of one-stage detectors while considerably reducing the difficulty of training unbalanced positive and negative samples with one-stage detectors.

There are many other classical detectors like M2Det and RefineDet. But in our project, we are using SSD for our interest [20,21].

3. METHODOLOGY

ROS is an open-source platform that has a number of packages that can be used directly in robots. It supports a wide range of drivers and employs the most widely used programming languages, C++ and Python. ROS has nodes that can send and receive data and take action based on it.

We can perform autonomous navigation on a differential drive robot utilizing the ROS Navigation stack. The essential requirement for the robot to navigate autonomously is a map. The SLAM software creates the map, which is then stored in the map server. The robot must then be localized, for which the map server gives the map, the laser scan sensor provides scan data, and the wheel encoders supply odometry data. The robot may then generate its course using navigation stack, avoiding obstacles with the help of global and local planners. The robot's linear and angular velocities, which must follow the specified course, are then calculated, converted to individual wheel velocities, and transmitted to the differential drive controller. Fig. 3.1. depicts the aforesaid setup in its entirety [9,11,14].

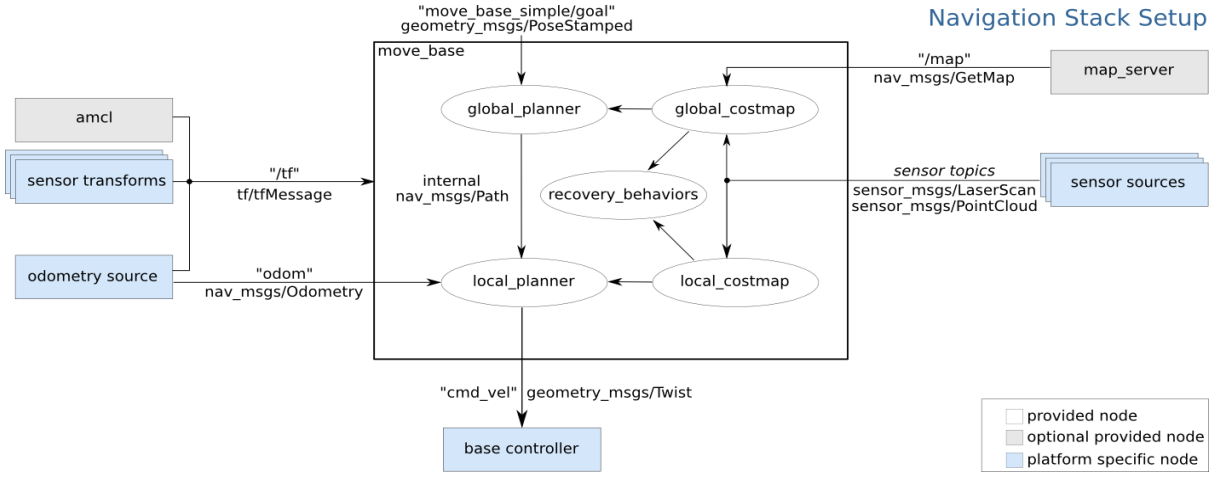

Fig. 3.1. Overview of robot setup

3.1 SLAM GMAPPING

Gmapping is a widely used laser-based grid mapping algorithm and is available in openslam.org. It uses grid mapping to decrease complexity and space usage and is based on the Rao-Blackwellized particle filter (RBPF). The following are the steps of SLAM Gmapping:

1) **Sampling:** In Gmapping, every particle contains a piece of data about the map and robot pose. The initial particles are odometry information, which are contrasted with the laser examine by the sweep matcher. In the event that the particle distribution is good, those particles are utilized for pose estimation; in any case, new recommendations are tested around the laser filter and Gaussian noise is utilized to further develop the particle distribution.
2) **Weighting**: The relevance of each particle is indicated by the value assigned to it. It can be calculated by,

$$\boldsymbol{\omega_t^{(i)}} = \frac{\boldsymbol{P}\left(\boldsymbol{x_{1:t}^{(i)}} \middle| \boldsymbol{z_{1:t}}, \boldsymbol{u_{1:t-1}}\right)}{\boldsymbol{\pi}\left(\boldsymbol{x_{1:t}^{(i)}} \middle| \boldsymbol{z_{1:t}}, \boldsymbol{u_{1!t-1}}\right)}$$

3) **Adaptive Resampling:** It is used to determine whether to resample particles.

$$\boldsymbol{N_{e_{ff}}} = \frac{\mathbf{1}}{\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} (\tilde{\boldsymbol{\omega}}^{(i)})^{\mathbf{2}}}$$

4) **Map Estimation:** After all these steps, the map estimation is done.

ROS Gmapping optimizes the parameters to configure the algorithm. The map is saved in map server once it has been created. This map server offers the map to amcl, as well as the move base for path planning, as described in the next sections [8,17,18].

3.2 AMCL

In this we are utilizing Adaptive Monte-Carlo Localization (amcl) approach, a part of ROS Navigation stack which utilizes a particle filter to track the robot against a map. Amcl is a probabilistic limitation framework for a robot moving in 2D.

The particles, similar to the real position of the robot with a specific weight, have their own direction and orientation values. The weight value is characterized as the absolute difference between the robot's actual pose and the particle's expected posture. The more the particle's weight, the more approximate it indicates the robot's pose.

As the robot goes about and gives new sensor information, particles are re-examined. With every re-test, low-weight particles are removed, while high-weight particles remain. After a couple of patterns of the AMCL

strategy, the particles will combine and compute a estimate of the robot's position. As an outcome, this calculation decides the robot's direction and orientation. [10,15,17,18].

3.3 PATH PLANNING

The velocity commands are broadcasted to the /cmd_vel topic after localization is completed using the amcl node. To navigate, the robot must predict a path that avoids all obstacles and leads to the desired destination. The move_base node, which is part of the ROS Navigation stack, can help with this. It has both global and local_planners to help you find the best path.

- **Global_Planner:** This package includes nodes for calculating the best path from the robot's current location on the map to the desired location. It uses Dijkstra's, A*, and other methods to find the shortest path to the goal.
- **Local_planner**: This package includes nodes for calculating the shortest path avoiding obstacles from the robot's current location on the map to the desired location. It uses DWA and other methods to find the shortest path to the goal.
- **Recovery_behaviors:** By making a 360-degree rotation, this node assists the robot in avoiding dynamic obstructions. If the obstacles continue along the course with no possible path after rotation, the robot will come to a stop with an error.
- **Global and local_costmaps:** This node uses point cloud of /laser_scan data to recognize obstacles and construct a map. For worldwide navigation, we use global_costmap, while for local navigation, we use local_costmap.

Once the path planning is done, the required linear and angular velocities to reach the goal are sent to base_controller. Base_controller converts the robot velocities from move_base to individual wheel velocity commands. The base_controller of a differential drive robot is explained in next section [12,18].

3.4 KINEMATIC MODEL OF DIFFERENTIAL DRIVE ROBOT

The robot is a two-wheel non-holonomic robot and the third wheel is a castor wheel for stability. The two wheels are separate driving wheels that are parallel to one another. They are propelled forward and backward by two DC motors and can rotate at varying RPMs for steering. The heading angle of the robot is calculated by dividing the linear velocity of the robot's right and left wheels v_r and v_l, respectively. Although this is a complicated

arrangement, it is commonly employed in numerous research studies in this field. The key benefit of this method is that the robot can rotate with a limited radius, which is useful in many applications such as mapping a specific area.

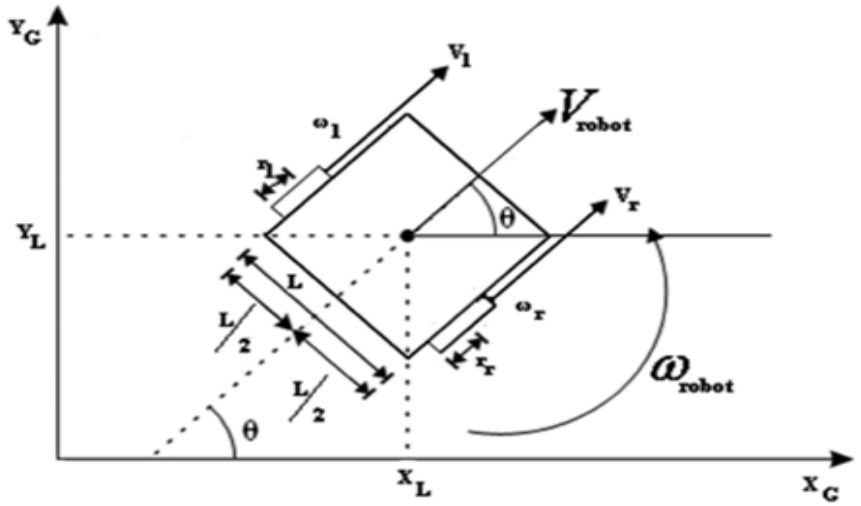

Fig. 3.2. Two wheeled differential drive robot

TABLE 3.1. PARAMETERS OF DIFFERENTIAL DRIVE ROBOT

Quantity	Description
v_r	Right wheel's linear velocity
v_l	Left wheel's linear velocity
L	Length between the wheels
ω	Robot's angular velocity
V	Robot's linear velocity
θ	Robot's orientation
r	Wheel radius

Consider the rate of movement of the left and right wheels, v_r, and v_l,, respectively, in the x and y planes, with their orientation/steering angle. The direction of the robot's movements is indicated by this. The kinematic equations of a differential drive robot are as follows, given the above constraints.

$$\dot{x} = \frac{r}{2}(v_r + v_l)\cos(\theta)$$

$$\dot{y} = \frac{r}{2}(v_r + v_l)\sin(\theta)$$

$$\dot{\theta} = \frac{r}{L}(v_r - v_l)$$

This model uses wheel encoders to send odometry data to the main computer, which is needed to control the robot's velocity and orientation

with a microcontroller. Complex sensors and advanced control algorithms are required for the robot to move in any unknown area [13].

3.5 REMOTE CONTROL OF MOBILE ROBOT

After you've set up your differential drive robot, you'll need to control it remotely. This is required in order to use the SLAM program to construct a map of your surroundings. As a result, we can operate our robot using a variety of methods, including a remote controller, an IP address, or a ROS node. Teleop twist keyboard is a popular ROS module that many researchers utilize for their robots.

Twist command is used by the Teleop twist keyboard, which takes inputs from the keyboard. The differential drive controller gets the velocity commands after being subscribed to the /cmd_vel topic in ROS, and the robot moves according to the command given.

3.6 Object detection using SSD

The SSD model is a network that identifies objects in a advance. It utilizes Faster R-anchoring CNN's component that blends the possibility of YOLO, and mirrors the qualities of different scale perspectives. The SSD model purposes a multi-scale target highlight extraction methodology, which permits it to perceive targets speedier than Faster R-CNN and with more noteworthy exactness than YOLO.

The SSD is separated into three segments: the backbone network, the first bounding box creation, and the convolution forecast. The essential network and the strengthening highlight extraction layer can be isolated from the backbone network. Object forecast and area expectation are remembered for the convolution forecast. The calculation's essential advances are as per the following: The photographs are initial sent into the organization, which utilizes profound brain organizations to separate data. Second, make a few default boxes for extricating highlight maps at different sizes. Third, the default casings' qualities are separated to expect the objective's caring and position. Finally, the non-maximal suppression method (NMS) is utilized to pick the forecast outcome that best match designated information [21].

4. IMPLEMENTATION

On a low-cost computer platform, this robot includes the architecture, design, and implementation of SLAM and navigation. The source code for the project as well as the necessary instructions have been uploaded to the github repository.

4.1 DESIGN

The real-time Autonomous mobile robot is shown in Fig. 4.1.

COMPONENTS

1) Nvidia Jetson NANO

Nvidia's Jetson Nano is a compact, powerful computer for embedded applications and AI IoT. It has an Nvidia Maxwell architecture GPU with 128 Nvidia Cuda cores and an ARM Cortex-A57MPCore quad-core processor. It contains 4GB of 64-bit LPDDR4 memory with a clock speed of 1600MHz. It also has HDMI and USB interfaces, as well as GPIO pins, I2C, I2S, and UART connection.

2) Arduino Uno

We'll need a microcontroller that can provide PWM signals to the motor driver to control the robot's velocity and orientation. The ATmega328P microprocessor is used in the Arduino Uno. It comes with 16 digital I/O pins, 6 analog I/P pins, a 16MHz ceramic resonator, and 32KB of flash memory. It has the ability to calculate encoder readings (odometry data) and communicate them to ROS.

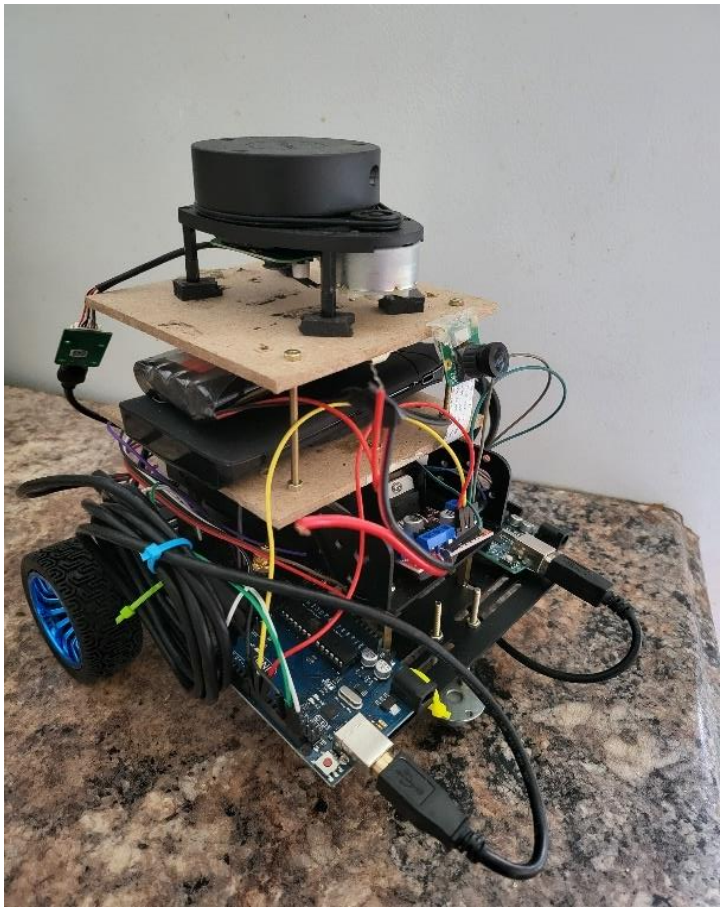

Fig. 4.1. Autonomous Mobile Robot

3) RPLIDAR A1

The SLAM technique generates a map and computes it using a laser range finder. The RPLidar A1 is a 360-degree laser range scanner that may be used both indoors and outdoors. It has a range of 12 meters and can generate 8000 times sample rate. As a result, laser data can be produced with reasonable precision.

4) Raspberry PiBMP (V2) Camera module with 160-degree FoV Camera lens attachment

The Raspberry Pi HD Camera Module V2 incorporates 8 megapixels of excellent imaging, diminished picture tainting and programmed control capacities, for example, openness control, white equilibrium and luminance discovery. It is attached with 160-degree field of view lens can show a considerable range of clarity and helps to enhance the appeal of the picture. The sensor size of 160-degree angle lens is 1/2.7''. TTL: 23.6mm. EFL: 2.6mm. F.NO: 2.0. MBFL: 3.8mm.

5) L298N Motor driver

The L298N is a double H-Bridge engine driver which permits speed and course control of two DC engines all the while. It can run DC motors with voltages ranging from 5 to 35V and peak currents of up to 2A.

6) DC Motors

Two 6V DC motors with wheel encoders are used to transfer odometry data for localization purposes.

7) Battery

The motor driver is powered by a 12V NiMH rechargeable battery, while the Jetson Nano is powered by an 10000mAh power bank.

8) Chassis

For the robot, a simple, sturdy structure is required. So, I 3D printed the chassis and used standoffs and screws to attach all of the components. It has three levels: a base with two Arduino Unos and a motor driver, a middle floor with batteries and extra cable connections, and a top floor with the Jetson Nano and Lidar. The motors are mounted in motor housings that are affixed to the base. Figure shows a clear photograph of the robot, and 3D printable files have been published to github.

9) Electrical Connections

The github repository contains the electrical connection diagram for this robot.

4.2 COMMUNICATION PROTOCOL

The overall design of robot is in a distributed way. In this case, the robot serves as a slave, collecting data while the server performs the computation (master). Wi-Fi is used to connect between the robot and the server, allowing us to communicate with the robot remotely. Wi-Fi provides more connection bandwidth, which is critical for navigating in an environment. As a result, Wi-Fi connectivity is being used in this project.

The robot is a differential drive robot, which means it requires a microcontroller to interpret the individual wheel velocity commands and move it. So, we're using two Arduino Unos connected through i2c communication bus0 and bus1 to the Jetson Nano. A USB connector is also used to attach a lidar sensor to the Jetson nano. Two-wheel encoders are linked to the two Arduinos, and an L298N motor driver is used to operate the motors using the pwm signals from the Jetson nano.

4.3 SOFTWARE ARCHITECTURE

For this robot, many nodes for SLAM and navigation have been designed and implemented on various devices.
1) move_base: To execute path planning and motion control, this node is subscribed to topics /tf and /scan and sends messages to topic /cmd_vel.
2) base_controller: This node handles motor control as well as data from encoders. It subscribes to /cmd_vel for motor control and sends odometry data to /odom.
3) base_frame_laser: There are a number of different coordinate frames depending on various sensors. Within this node, coordinate frame alterations are performed. The converted data is published to the /tf topic.
4) amcl: This node is useful for robot localization. It follows the /laser_scan, /odom, and /tf topics.
5) slam_gmapping: This node receives odometry laser scan data by subscribing to /tf and /scan. The laser scan data will correct the robot's pose. The map is published to topic /map.
6) map_server: The map generated by slam_gmapping is saved on this node. It subscribes to the subject /map and publishes it.
7) rviz: Rviz, a ROS program which is used to dynamically present visualizations of the postures and the created map.

The /tf transforms and rqt_graph used in this robot are shown in Fig. 4.2. and 4.3. respectively.

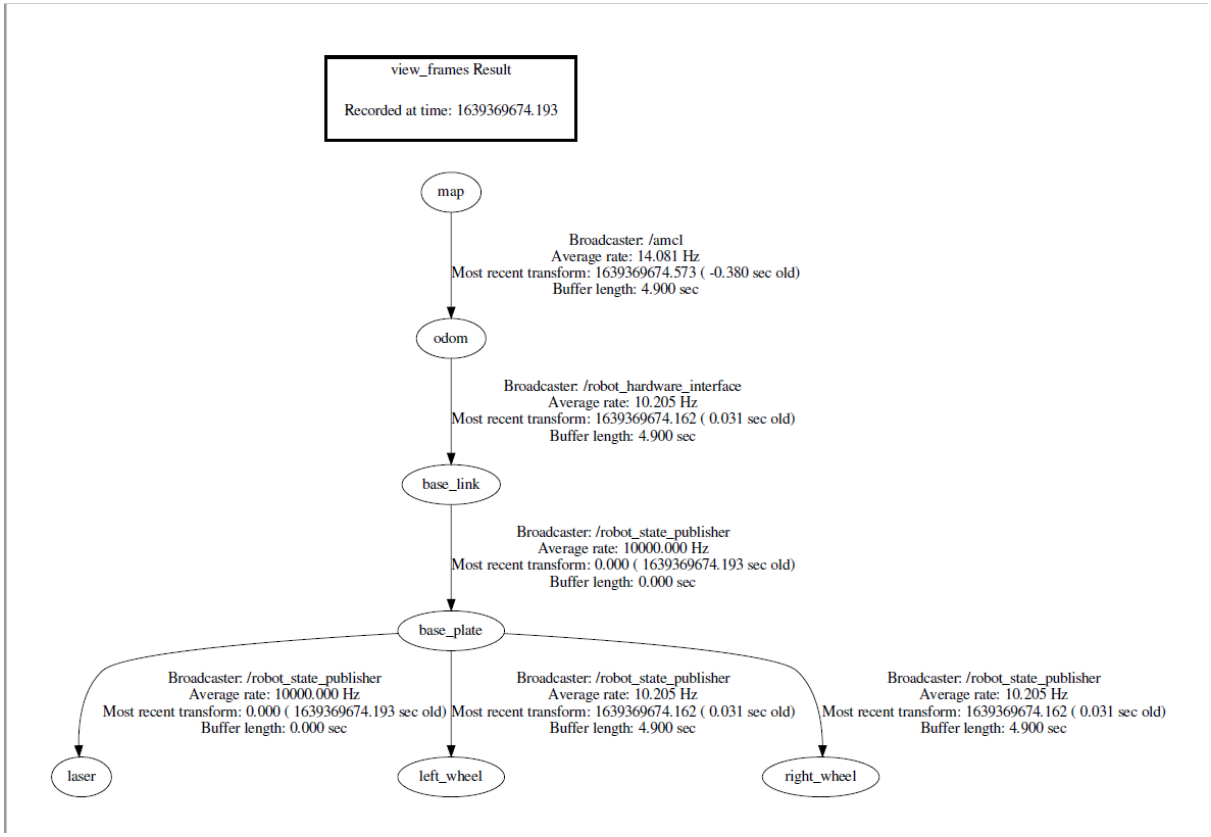

Fig. 4.2. ROS /tf transforms of the Robot.

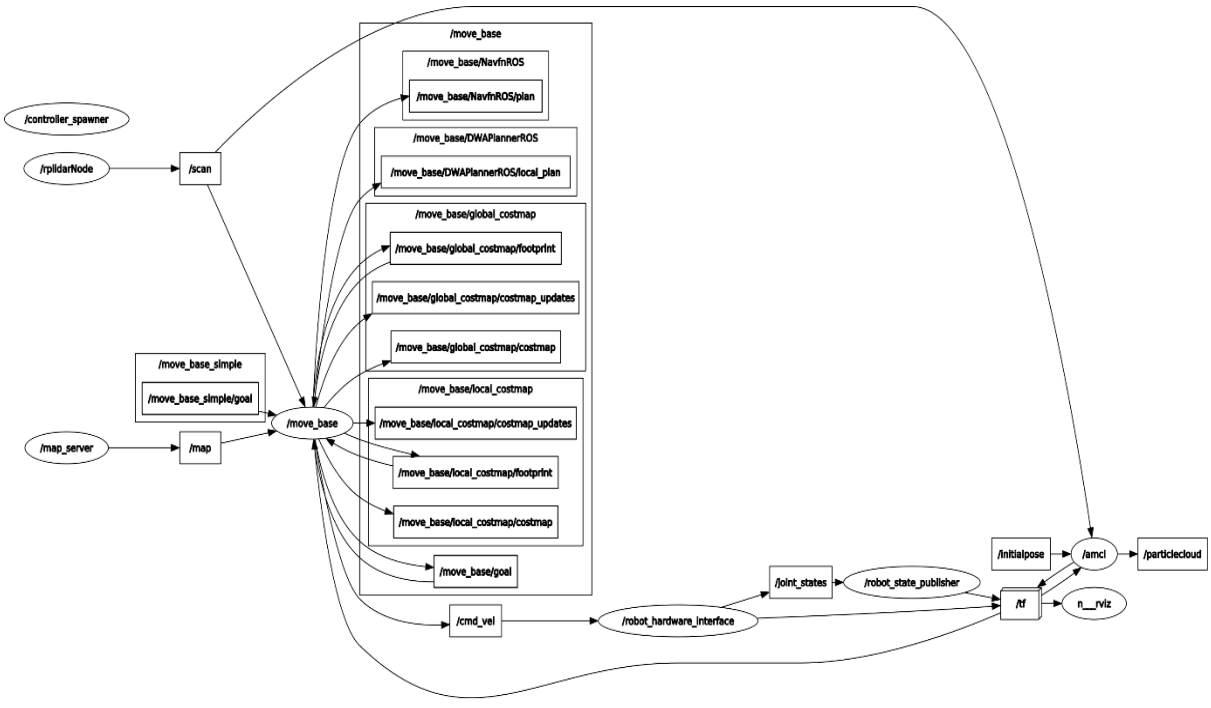

Fig. 4.3. ROS nodes graph

5. EXPERIMENTATION

It's time to test the robot after it's been assembled, electrically connected, and software configured. The robot is made to move in a certain environment that has two spiders which resembles two points, Point-A and Point-B and is also simulated using Rviz in order to test it. The robot is first made to move around the area using the teleop twist keyboard, while a map is created using the laser scan at the same time. The layout of the robot is shown in Fig. 5.1. The generated map is shown in Fig. 5.2. After that, a grid map is created. The generated map is visualized in rviz. The dark borders of the maps represent obstacles, while the light areas represent empty space where the robot can navigate. It is saved in map_server using ROS map_saver node.

Fig. 5.1. Layout for Autonomous Mobile Robot

Fig. 5.2. Map generated using SLAM Gmapping.

6. RESULTS

The mobile robot in this project can explore an unknown area and avoid obstacles on its own. At the same time, it may use data from the lidar sensor to localize and create a 2D map. After map of the unknown environment is created, the robot may use the known map and path planning to locate itself and drive independently to the goal.

Table. 6.1. Shows the obtained results till now

Objective	**Result**
Moving the robot in the given environment using teleoperation.	PASSED
With the lidar data, ROS can create a useful 2D map of the test location.	PASSED
Localization of the robot	PASSED
The robot is capable of path planning and real-time navigation.	In-Progress

Now the Robot has to navigate from Point-A (spider-1) to Point-B (spider-2) autonomously. So, using 2D pose estimate in rviz as shown in Fig. 6.2, Localization is done and using 2D Nav goal, the goal position is given on the map to the robot. Then using local and global planners a green trajectory is generated as shown in Fig. 6.3 and robot follows that trajectory and reaches the goal position in real time as shown in Fig 6.4 and 6.5.

Fig. 6.1. Robot at Point-A

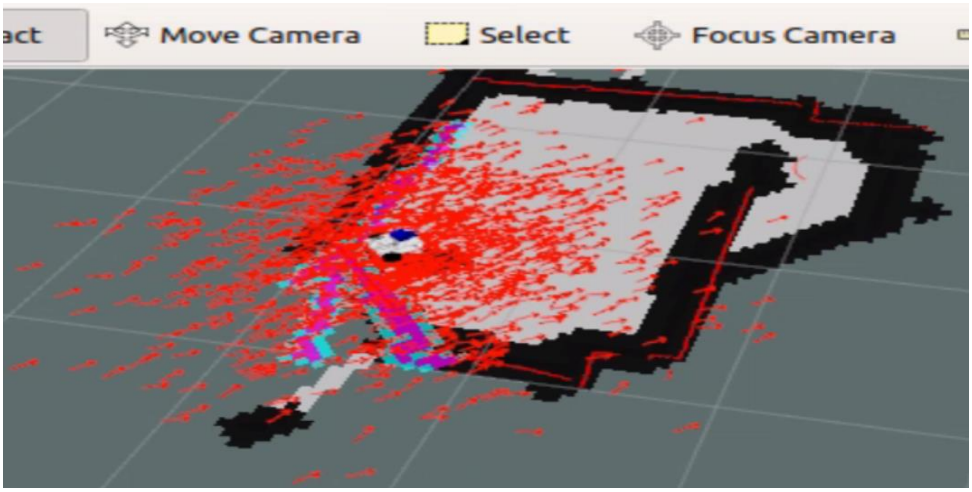

Fig. 6.2. Robot at Point-A in rviz

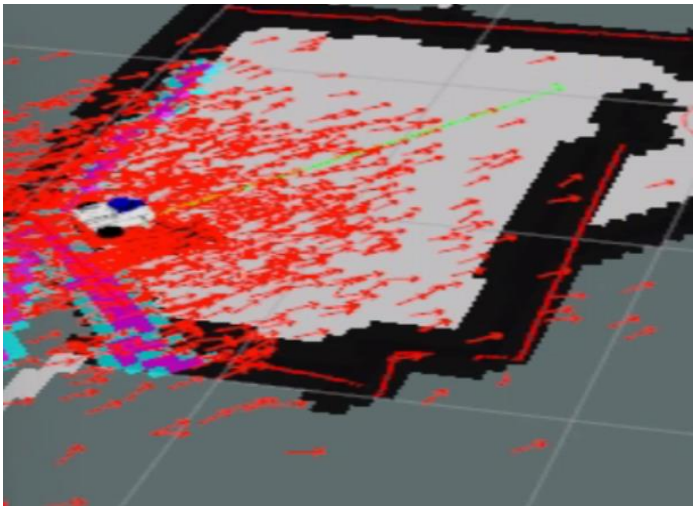

Fig 6.3. Robot generating a trajectory

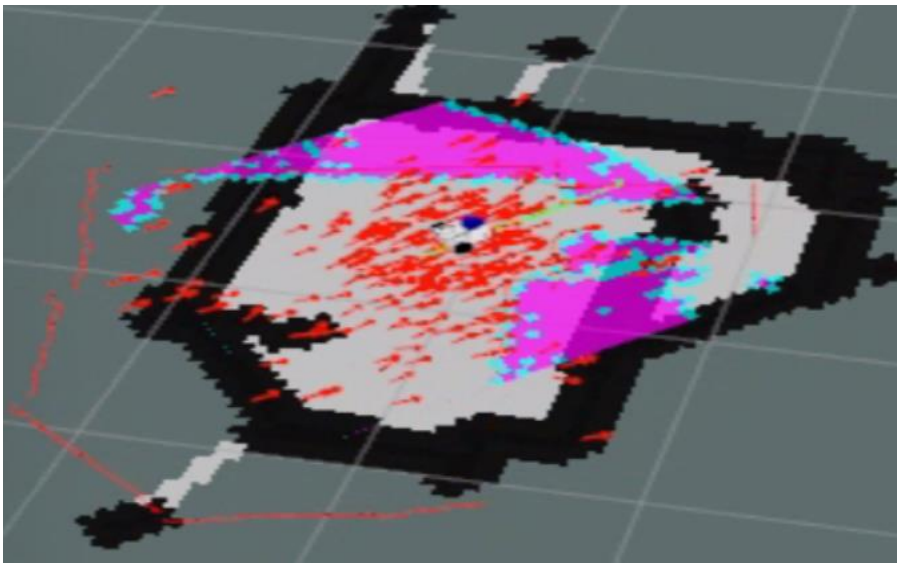

Fig. 6.4. Robot navigating to Point-B

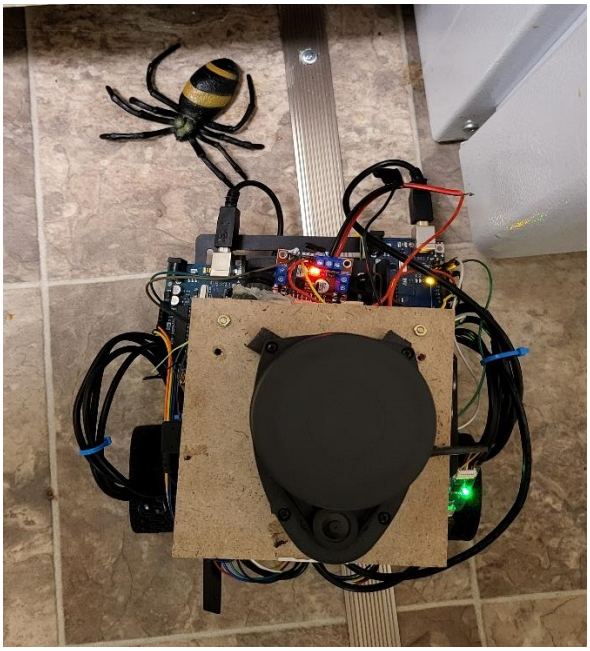

Fig. 6.5. Robot reached Point-B

At first, the robot is tested in a simple environment where it creates a 2D-Map, detect walls and avoiding obstacles it can navigate to goal position. Now the environment is made a bit complex by adding an obstacle in the previous environment as shown in Fig. 6.6. Using 2D-Pose estimation, the robot is localized and using 2D navigation a green trajectory has been generated avoiding the obstacle in rviz which is shown in fig 6.7. The robot has followed the generated path and reached the goal position safely in real-time by avoiding the obstacle in between as shown in Fig 6.8.

Fig. 6.6. Obstacle added to the environment

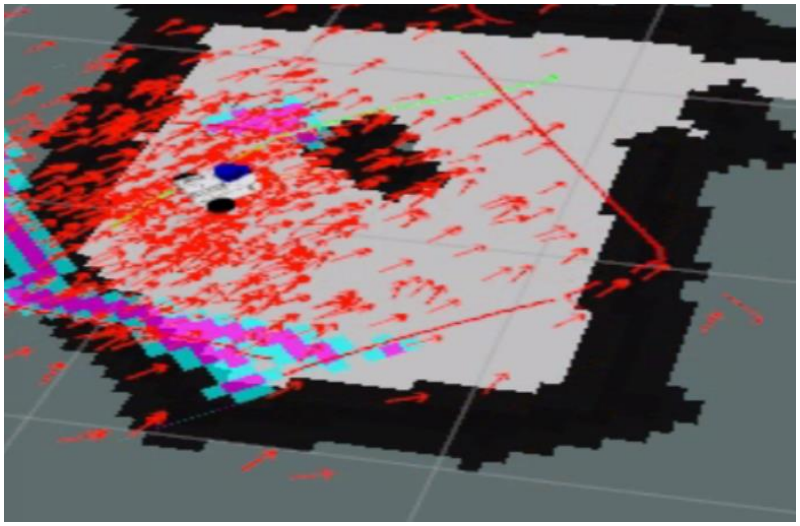

Fig 6.7. Trajectory generated avoiding the obstacle

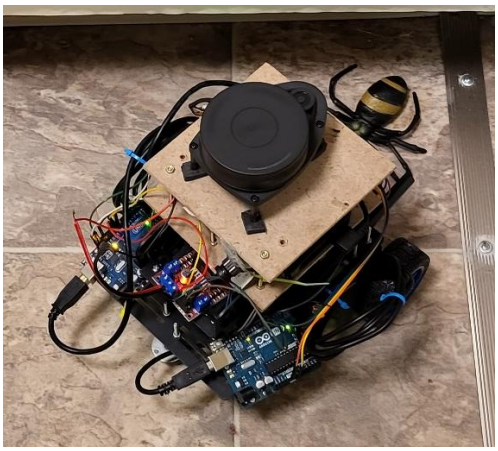

Fig 6.8. Robot reached Point-B safely

Hence the results show us that the mobile robot can successfully avoid obstacles and able to navigate in any given environment. But the DWA algorithm took quite a bit of time to figure out new path that needs to be generated when an obstacle is introduced in the environment. Hence the algorithm needs further optimization so that the results obtained will be optimum.

The next main feature that to be tested is Object detection. We are using Single shot detection (SSD) network pre-trained on standard COCO dataset. Hence the robot can be able to detect around 91 objects and it can display the accuracy of detecting each object as shown in Fig 6.9. But the constraint is that the robot should be in well luminated environment and the camera we are using can be able to detect objects which are in distance up-to 8inch radius. To detect objects far from the robot, it needs a better camera. Hence the robot has reached all the objectives and can be deployed in any environment.

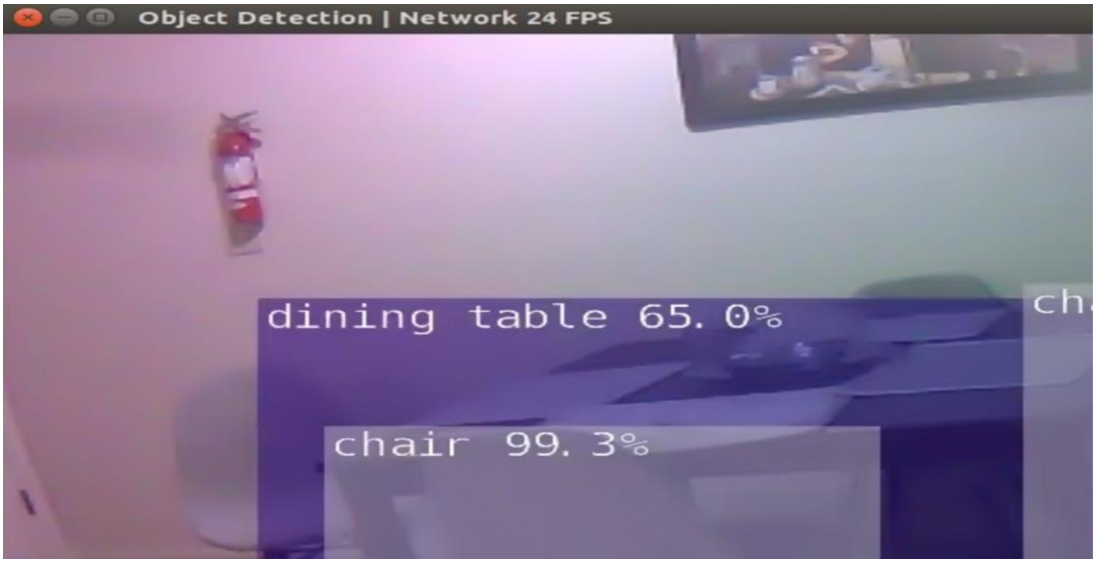

Fig. 6.9. Object detection using SSD

Table 6.2. Shows the final result

Objective	**Result**
Moving the robot in the given environment using teleoperation.	PASSED
With the lidar data, ROS can create a useful 2D map of the test location.	PASSED
Localization of the robot	PASSED
The robot is capable of path planning and real-time navigation.	PASSED
Object detection	PASSED

7. CONCLUSION

This project describes the hardware and software systems of a self-driving mobile robot with object detection. Path planning and motion control are now complete on the SLAM mobile robot. To obtain a more accurate environment map, the settings of SLAM Gmapping are tweaked. A simulated scenario is constructed, and the grid map that results precisely reflects the robot and its surroundings. The visualization is made using Rviz. The robot effectively avoids obstacles and reaches the desired position in the experiments. It can detect various objects using camera and the accuracy is displayed in rviz. The possibility of employing an embedded platform to create indoor 2D SLAM is demonstrated by this low-cost mobile robot.

8. APPENDIX

The github repository link for all the code, electrical connections and commands: https://github.com/krishnavamsee96/Jetbot

REFERENCES

[1] T. Theodoridis and H. Hu, "Toward Intelligent Security Robots: A Survey," in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 42, no. 6, pp. 1219-1230, Nov. 2012, doi: 10.1109/TSMCC.2012.2198055.

[2] C. Joochim, "Autonomous Security Robot Base on SLAM," 2021 18th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2021, pp. 959-962, doi: 10.1109/ECTI-CON51831.2021.9454695.

[3] X. Dai, H. Zhang and Y. Shi, "Autonomous Navigation for Wheeled Mobile Robots-A Survey," Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007), 2007, pp. 551-551, doi: 10.1109/ICICIC.2007.192.

[4] Jianqiang Jia, Weidong Chen and Yugeng Xi, "Design and implementation of an open autonomous mobile robot system," IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004, 2004, pp. 1726-1731 Vol.2, doi: 10.1109/ROBOT.2004.1308073.

[5] D. Chung Tran and V. Huy Le, "Safe Navigation for Indoor Mobile Robot Based on IoT Information," 2020 Applying New Technology in Green Buildings (ATiGB), 2021, pp. 142-145, doi: 10.1109/ATiGB50996.2021.9423136.

[6] Panigrahi, Prabin Kumar, and Sukant Kishoro Bisoy. "Localization Strategies for Autonomous Mobile Robots: A review." *Journal of King Saud University-Computer and Information Sciences* (2021).

[7] W.A.S Norzam *et al* 2019 *IOP Conf. Ser.: Mater. Sci. Eng.* 705 012037.

[8] R. K. Megalingam, A. Rajendraprasad and S. K. Manoharan, "Comparison of Planned Path and Travelled Path Using ROS Navigation Stack," 2020 International Conference for Emerging Technology (INCET), 2020, pp. 1-6, doi: 10.1109/INCET49848.2020.9154132.

[9] Xuexi Zhang, Jiajun Lai, Dongliang Xu, Huaijun Li, Minyue Fu, "2D Lidar-Based SLAM and Path Planning for Indoor Rescue Using Mobile Robots", Journal of Advanced Transportation, vol. 2020, Article ID 8867937, 14 pages, 2020.

[10] D. Talwar and S. Jung, "Particle Filter-based Localization of a Mobile Robot by Using a Single Lidar Sensor under SLAM in ROS Environment," 2019 19th International Conference on Control, Automation and Systems (ICCAS), 2019, pp. 1112-1115, doi: 10.23919/ICCAS47443.2019.8971555.

[11] Q. Xu, J. Zhao, C. Zhang and F. He, "Design and implementation of an ROS based autonomous navigation system," 2015 IEEE International

Conference on Mechatronics and Automation (ICMA), 2015, pp. 2220-2225, doi: 10.1109/ICMA.2015.7237831.

[12] M. Liao, D. Wang and H. Yang, "Deploy Indoor 2D Laser SLAM on a Raspberry Pi-Based Mobile Robot," 2019 11th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2019, pp. 7-10, doi: 10.1109/IHMSC.2019.10097.

[13] L. A. Yekinni and A. Dan-Isa, "Fuzzy Logic Control of Goal-Seeking 2-Wheel Differential Mobile Robot Using Unicycle Approach," 2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS), 2019, pp. 300-304, doi: 10.1109/I2CACIS.2019.8825082.

[14] Y. Li and C. Shi, "Localization and Navigation for Indoor Mobile Robot Based on ROS," 2018 Chinese Automation Congress (CAC), 2018, pp. 1135-1139, doi: 10.1109/CAC.2018.8623225.

[15] W. -H. Wang, Y. -H. Chien, H. -H. Chiang, W. -Y. Wang and C. -C. Hsu, "Autonomous Cross-Floor Navigation System for a ROS-Based Modular Service Robot," 2019 International Conference on Machine Learning and Cybernetics (ICMLC), 2019, pp. 1-6, doi: 10.1109/ICMLC48188.2019.8949176.

[16] Auat Cheein, F.A., Lopez, N., Soria, C.M. *et al.* SLAM algorithm applied to robotics assistance for navigation in unknown environments. *J NeuroEngineering Rehabil* 7, 10 (2010).

[17] S. Oajsalee, S. Tantrairatn and S. Khaengkarn, "Study of ROS Based Localization and Mapping for Closed Area Survey," 2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR), 2019, pp. 24-28, doi: 10.1109/ICMSR.2019.8835455.

[18] R. K. Megalingam, A. Rajendraprasad and S. K. Manoharan, "Comparison of Planned Path and Travelled Path Using ROS Navigation Stack," 2020 International Conference for Emerging Technology (INCET), 2020, pp. 1-6, doi: 10.1109/INCET49848.2020.9154132.

[19] Pengtao Qu *et al* 2021 *J. Phys.: Conf. Ser.* 2024 012056

[20] J. Kang, S. Tariq, H. Oh and S. S. Woo, "A Survey of Deep Learning-Based Object Detection Methods and Datasets for Overhead Imagery," in IEEE Access, vol. 10, pp. 20118-20134, 2022, doi: 10.1109/ACCESS.2022.3149052.

[21] Q. Shuai and X. Wu, "Object detection system based on SSD algorithm," 2020 International Conference on Culture-oriented Science & Technology (ICCST), 2020, pp. 141-144, doi: 10.1109/ICCST50977.2020.00033.