SLAM USING LIDAR IN DIFFERENTIAL DRIVE ROBOT UNDER ROS

BY

RAJESH.S

MTECH-AI&ROBOTICS

SASTRA UNIVERSITY

**Table of Contents**

LIST OF TABLES…………………………………………………………5

LIST OF FIGURES………………………………………………………..6

ABSTRACT……………………………………………………………….7

NOTATIONS.................................................................................................7

1. INTRODUCTION AND LITERATURE SURVEY

1.1. General………………………………………………………....…...8

1.2. Literature Survey…………………………………….....……….......9

1.3. Research Gap…………………………………………………….....9

2. METHODOLOGY

2.1. Design……………………………………………………………....10

2.2. Describing Parameters………………………………………...…...11

2.3. Exporting to ROS……………………………………………...........13

2.4. Attaching Gazebo Plug-ins………………………………………....15

3. WORKING WITH CODE

3.1. Differential Drive Code………………………………………….....17

3.2. LiDAR Sensor Code………………………………………….....….18

4. RESULTS AND DISCUSSION

4.1. Mapping……………………………………………………….....…20

4.2. Configuration Discussion………………………………….........…..21

5. CONCLUSION AND FUTURE SCOPE……………...……..........….....22

REFERENCES………………………………………………………...…..23

**List of Figures**

Figure 2.1. FUSION360 Model .

Figure 2.2. Component names and dimensions.

Figure 2.3. URDF\_Exporter Window in FUSION 360.

Figure 2.4. Exported URDF file sample.

Figure 2.5. ROS workspace file structure.

Figure 2.6. Gazebo world and plug-in .

Figure 2.7. Placing Bot inside the created world.

Figure 3.1. Differential Drive Code.

Figure 3.2. LiDAR Sensor Code.

Figure 3.3. Laser rays from LiDAR sensor.

Figure 3.4.GMapping code.

Figure 4.1 Mapping process and final generated map.

Figure 4.2. Gazebo and Rviz Windows .

**ABSRACT**

Simulators are the tools used to study the behavior of an entity, in this case Wheeled MobileRobots. These simulators played a major role in robotics research as a tool for effective testing of algorithms, concepts and strategies. One such widely used simulator in the field of robotics is Gazebo. This paper is an overview of how a differential drive robot is simulated using Gazebo inROS environment.

ROS is an open-source robot operating system. As, ROS is open-source, a lot of people have contributed to build robot software systems. These software systems can be used for variousresearch aspects and other requirements accordingly. Gazebo enables simulation of the world,physical model, sensors and control system through the Unified Robot Description Format(URDF) file. ROS is interfaced with Gazebo to allow the implementation and usage of various

robotic software systems and tools on the simulated robot.

In this report, we focus on how a Wheeled Mobile robot is autonomously navigated i.e., how Localization and Mapping are being implemented on the wheeled mobile robot in the simulation environment.

Keywords: Gazebo, ROS, Mobile robot, Simulation, Autonomous navigation.

**NOTATIONS**

Simultaneous Localization and Mapping (SLAM)

Position And Orientation (POSE)

Robot Operating System(ROS)

Light Detection and Ranging(Lidar )**CHAPTER 1: INTRODUCTION AND LITERATURE SURVEY**

**1.1. General :**

The human race has evolved in many ways in terms of work. The aspect of

work/working style has always been improving for better. From the beginning of

industrial revolution till the present automation processes involved in the present

generation industries and the fast progress towards the near future “Intelligent

Industrialisation” which involves complex artificially intelligent systems. In all the

aforementioned developments, machines were employed as mere tools to accomplish highly tedious tasks/repetitive tasks. One such category of machines involve robots which are at present heavily employed in most of the manufacturing industries. But, the future of Artificially Intelligent systems/ Intelligent robots should always and only be an extension to human beings to accomplish their tasks in much more efficient way and not a replacement to human beings. Therefore, it is the task of the roboticist or developer who is working in the robotics field to develop an accountable and human friendly robot so as to work in a collaborative fashion not just with fellow robots but with humans as well. Mobile robots are one such class of robots which are not stationary in nature or working and can freely move around in order to accomplish the given task. But before spending huge investments on the developments/prototyping of the mobile robots, a roboticist would be keen on knowing how his/her ideal robot concept would fare in real world with all the proposed/intended capabilities the roboticist wants in that robot. One such method that could drive this beforehand task of checking the feasibility of the robot prior hardware development is the designing and simulation environment.

**1.2. Literature Survey:**

The designing of the mobile robot provides space for conceptualising and

implementing appropriate sensors and motors which are needed to support the

working of the robot. Whereas, the simulation part helps the developers to know

about the feasibility of the applied algorithms and also to visualise how their

respective design would impact the environment in which the robot would be

working. For this task, the open source platform for robotics called ROS(Robotic

Operating System) and Gazebo simulator which is purely a physics based simulator has been employed to carry out the autonomous navigation simulation task. ROS is developed and maintained by open source robotics foundation (OSRF) and it is also maintained and updated by the developer community. Gazebo on the other hand is also supported by OSRF and is maintained and updated by active community members.The gazebo is based on ODE (Open Dynamics Engine) and as per Torres- Torriti et.al.,[3], the ODE takes care of all the related kinematics and dynamics solving part pertaining to the robot and its an open source engine powering gazebo from backend. Overall, simulation plays an important role in estimating the developments needed in order to make an efficient robot with the necessary hardware and software components. This also helps in testing or comparing the results later on with the actual robot too. This is emphasised by Tayaka K et.al.,[4], where at first simulations are performed on ready made on Pioneer3-DX and PeopleBot models, and finally the results are verified with the real world experiments. ROS also supports URDF, and specially gazebo supports SDF file formats using which a roboticist can develop a model using simple XML commands. This method is employed by Megalingam RK et.al.,[5] and Dennis Chikurtev [6] in their work, where both the works almost employ the same algorithms and obtain similar results with only changes being in the design of the robot and the simulations environments being used. Even pre-built open source robots like Turtlebot which is developed by ROBOTIS company, could also be used for the autonomous navigation task as shown in the work of M.S. Hendriyawan Achmad et.al.,[7].Thanks to widespread adoption of open source platforms by both users and developers, because of which, even 3D CAD modelling software such as SOLIDWORKS, AUTODESK Fusion 360, etc., are supporting URDF file formats in the form of either built-in module or as an additional plug-in. This makes the work of researchers even more easier to quickly test the algorithms and their simulation efficiencies. This method is employed by Darshan KT et.al.,[8] wherein, the model of quad rotor and LIDAR sensor is obtained from GrabCAD(Open source CAD library) and is combined together using SOLIDWORKS CAD designing software and exported to ROS using URDF file format add-on.

**1.3. Research Gap:**

Overall, from the literature survey performed, it is understood that either the

simulations are performed with the help of XML coded robots, or readymade versions such as TURTLEBOT are used, or existing CAD models are combined together in order to simulate and analyse a robot’s/algorithm’s performance. Therefore, this paper focuses on addressing both the design and simulation aspects of the mobile robots, all while stressing on discussions about the intermediate steps involved between designing in a FUSSION360 software from scratch and exporting the designed robot and the final implementation of all the a forementioned algorithms for localisation and path planning of the mobile robot .

**CHAPTER 2: METHODOLOGY**

The process of designing and developing a mobile robot begins with the

generation of the design idea and formation of a hand sketch of the same.

Furthermore, based on the requirements and desired tasks that are to be performed by the robot, the required sensors and additional accessories/tools are added as part of the design elements of the robot. Therefore, the overall stages involved from the design of the robot till the implementation and execution of algorithms into the robot are described as below:

• Design

• Describing Parameters

• Exporting to ROS

• Attaching Gazebo Plug-ins

**2.1. Design:**

The 2D design or the hand sketch of the robot is prepared and space

has been allocated in the design so that the relevant sensor such as LIDAR.

Although the considered controllers, sensors, and motors are subjective to change

based on actual making/development of the robot or as per the desired task from the robot. But, for the current simulation, the above said apparatus is considered and accordingly design space has been allocated. Furthermore, a robotic arm[12] has been attached to the robot, to carry out goods transfer operation.

The software considered for the above said designing task is “AUTODESK FUSION 360”. The particular software used provides ease-of-access to designs from any devices as it is cloud based. Furthermore, the above said CAD software makes robot exporting part easier as there is no coding or manually describing joint types and relevant parameters are required as opposed to other popular CAD designing softwares such as “SOLIDWORKS”. All the exporting part is made simpler by installing the required “fusion2urdf” plug-in which is easily available in the internet for the FUSION 360 users for free. More in detail about describing parameters and exporting is discussed in the upcoming section.

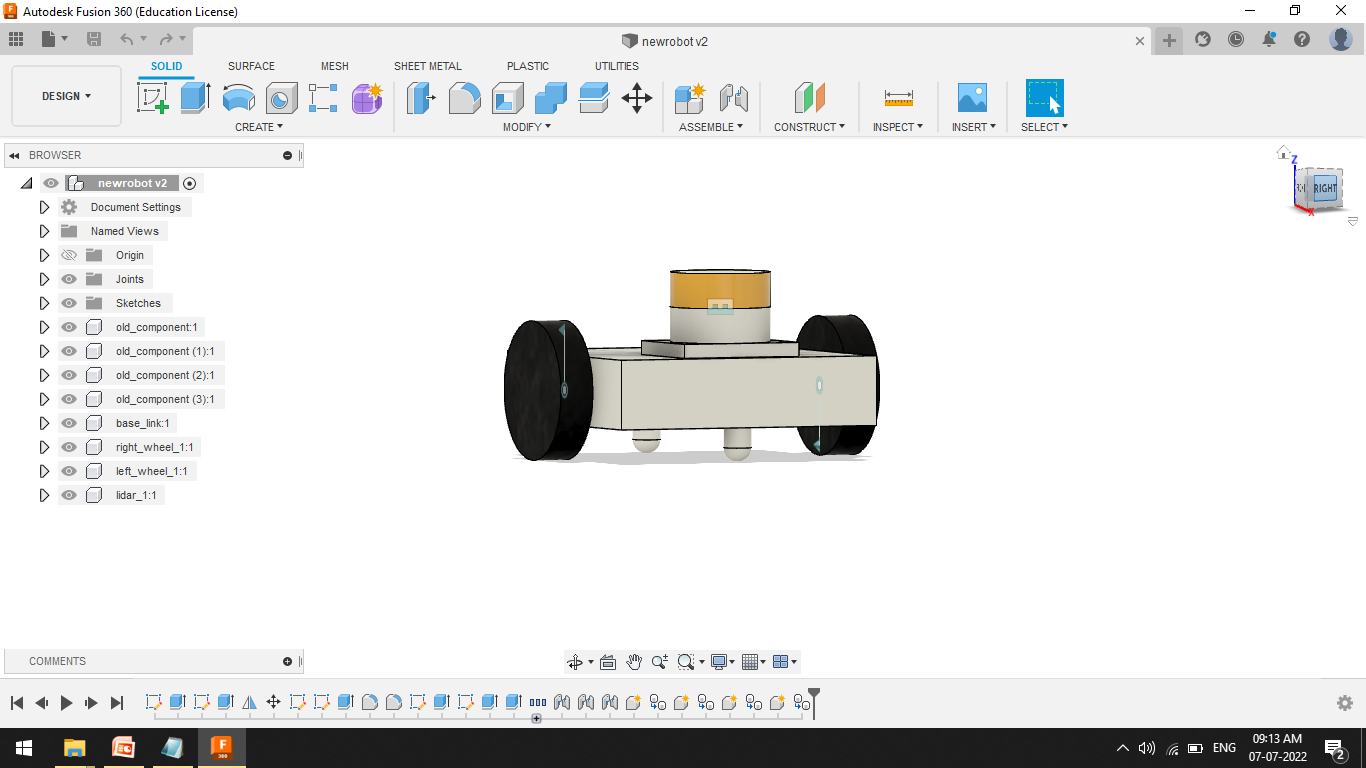


Figure 2.1. FUSION360 Model .

**2.2. Describing Parameters:**

Although a 3D model is good as a visual model of the robot design, the

implementation of the same in gazebo environment would produce errors(such as,

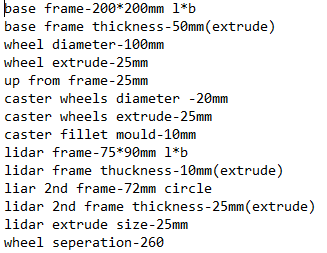
robot model collapsing or not being able to establish with parent link) or would

require much more detailing so as to perform simulation. According to Worcester

Polytechnic Institute’s report[14]., the above said problem is valid and thus, requires a collision model which is an approximated and simple model as opposed to complicated visual model. Also, the physics calculations involved in considering such complex models is high and thus, the processor power required for calculations are also equally higher. These complex visual models could be approximated into simple shapes and all are connected to the base\_link(i.e., the base/main frame of the robot)through the respective joint configurations such as revolute joints(for wheels), rigid joints(for upper plate holdings, front and back casters, LiDAR model). Thus, the complex parts in the robot such as the wheels,

LiDAR sensor are all approximated to a collision model But, as gazebo is a physics based simulator software, therefore, mere connection of components to the base frame/base\_link alone is not enough. This is because, in reality, a robot possess mass not just of its own body but also of the components it carries on it/with it. This intrinsic physical properties of mass, density, volume, and centre of mass of each component is all important for simulation as well. Therefore, this is done by altering the physical properties of the collision model and assigning the

needed/relevant material so that the FUSSION360 software can automatically calculate theabove said physical properties based on the measurements of the model, andfurthermore, the URDF exporter uses this information to create appropriate gazebo and rviz files.These physical material properties are mentioned in Table 2.2.



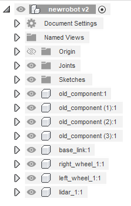


Figure 2.2. Component names and dimensions.

**2.3. Exporting to ROS:**

After creating the respective files with the said physical materials,

component names and joint name. the “fusion2urdf” plug-in

which is added to FUSION 360 software creates the final URDF files. This

“fusion2urdf” is accessed using the “ADD-INS toolbar” and the FUSION 360 makes this conversion process as easier as a click of a “run” button as shown in Figure 2.3.

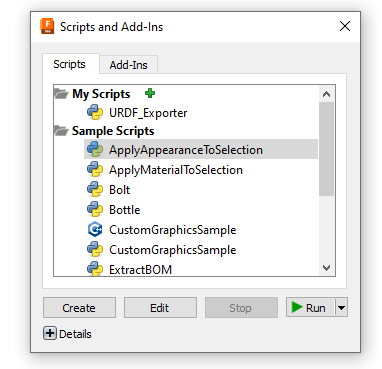


Figure 2.3. URDF\_Exporter Window in FUSION 360.

After the conversion, the design files are sliced into separate components and is

exported to the set destination folder as “newrobot\_description” file with the mesh

diagrams being stored as a shown in Figure 2.4

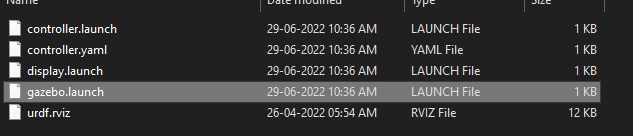
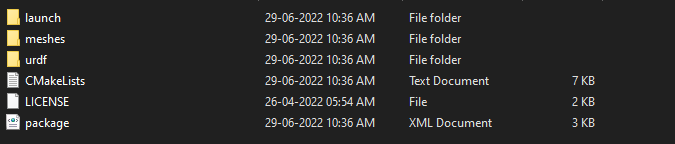


Figure 2.4. Exported URDF file sample.

Thereafter, the “newrobot\_description” is moved/copied to the workspace folder

of theROS environment in Ubuntu. Here, Ubuntu 20.04 with ROS (Noetic Version) has been installed and the respective model is stored in the file, “Home > Desktop > catkin\_ws > src >newrobot\_description”. Although, the name of the workspace could be anything or the file can also be copied into an entirely new workspace. The file structure is shown in Figure2.4.





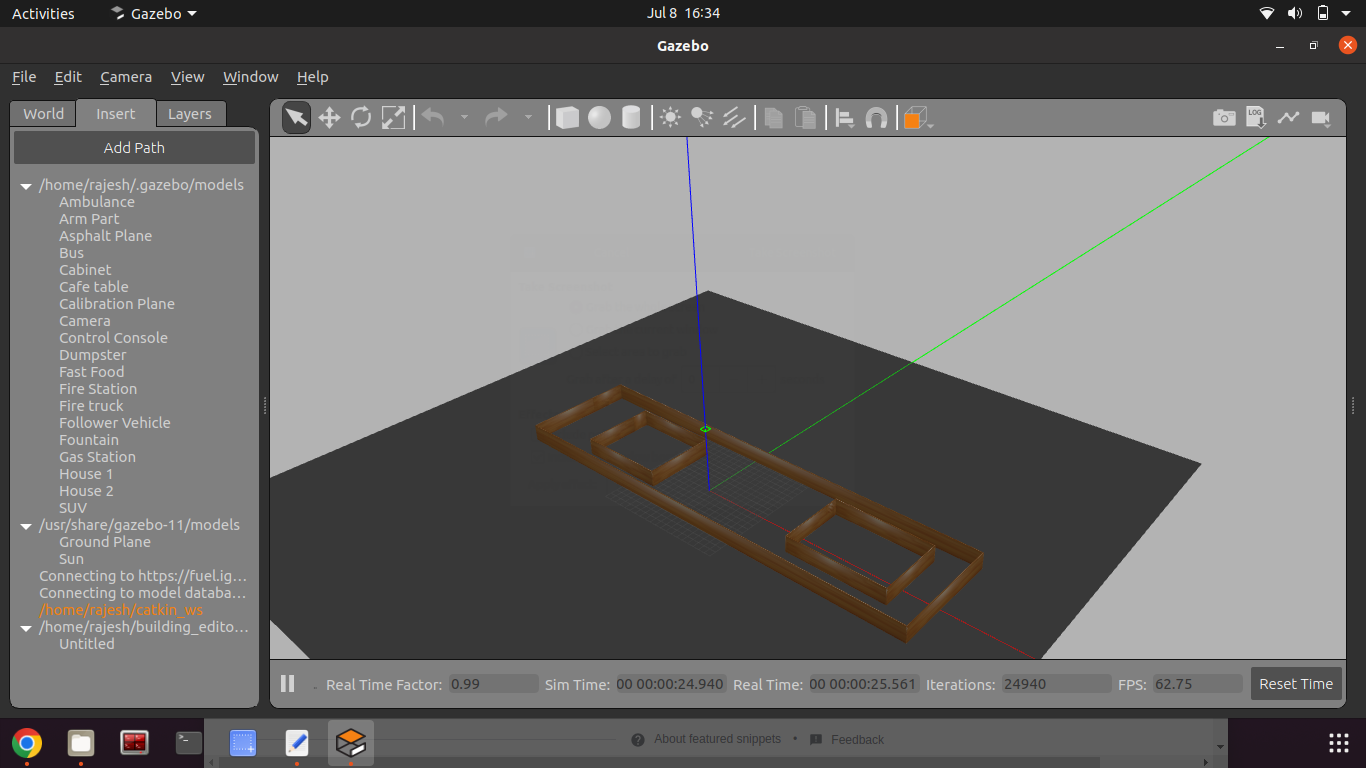
Figure 2.5. ROS workspace file structure

**2.4. Attaching Gazebo Plug-ins:**

After transferring/exporting the URDF to ROS environment in Ubuntu Linux OS, a world is created for the robot to be able to spawn and navigate inthe gazebo environment. One corner of the world is created as a closure boundary to test the manoeuvrability of the robot. The created gazebo 3D world is shown in Figure 2.6 in both perspective and orthogonal view. The code modules related to the gazebo for spawning the robot in gazebo environment with the created world and for attaching the joint states of the robot is shown in Figure 2.7. This file is stored in the URDF folder of the “newrobot\_description” folder with the name ”gazebo.launch”. All the gazebo plugins are available for use form the open source gazebo website “http://gazebosim.org/tutorials? tut=ros\_gzplugins".From the gazebo plugins website various plugins such as differential drive motion, camera, LiDAR, and other sensors could be borrowed to attach with the robot with

easy-to-modify user tags and it is encoded in XML file format. More on the coding

part is discussed in chapter 3.



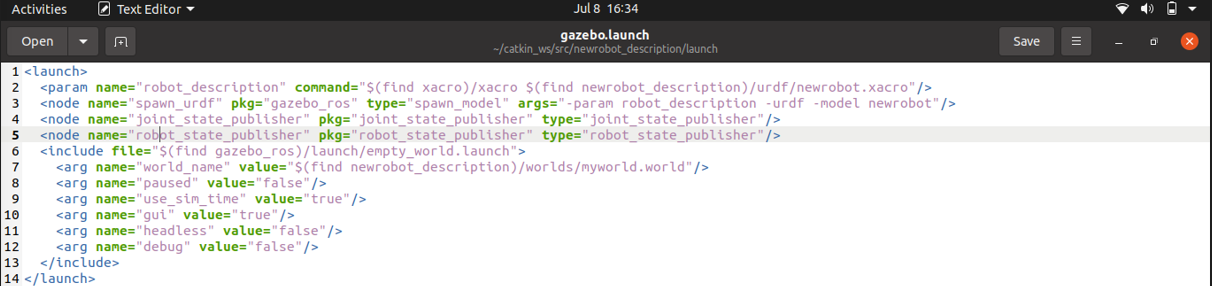


Figure 2.6. Gazebo world and plug-in .

This “gazebo.launch” file is created by the URDF exporter itself, therefore, the usercan edit by adding only the world folder location alone in the argument(arg) tag (as shown in the grey colour comment line) in Figure 2.7.

Thereafter, subsequent plugins could be added in “scobo.gazebo” file which is also

located in the URDF folder only. These plugins are discussed in chapter 3.

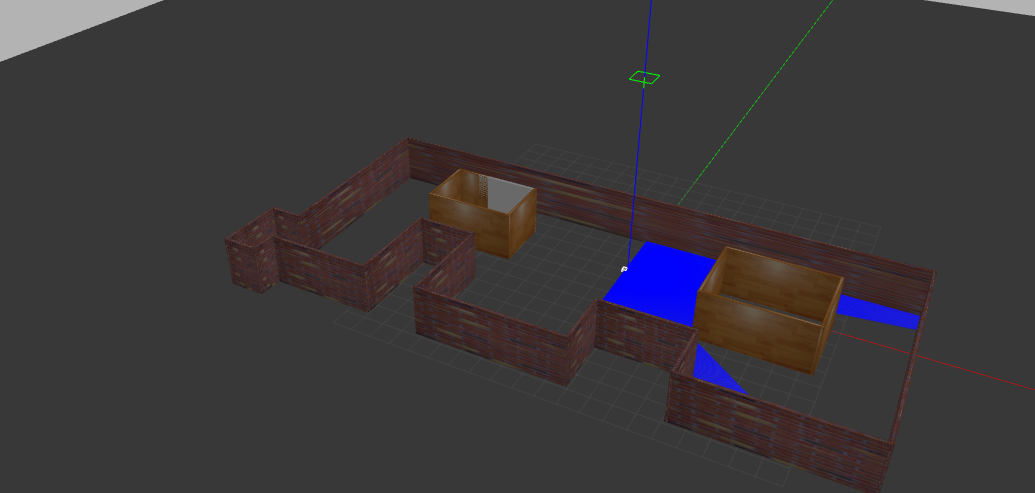


Figure 2.7. Placing Bot inside the created world.

**CHAPTER 3: WORKING WITH CODE**

The code for a physical robot is given using C++ language for communication purpose, as it is more convenient for the controller of the robot to communicate withthe motor drivers and sensor modules as C++ is a compiler based language. Thus,encode the human written code into computer binary language much faster. Even,python could also be coded into the robot for the similar tasks. But, the only drawback of using python as primary robotic coding language is that python is an interpreter language and this increases the time complexity in terms of code execution. But still,ROS provides freedom and flexibility to the user to use either of the languages as per convenience.In terms of simulation, this process of using code to provide functionality to the robot is much more simplified. This is done with the help of XML and YAML file formats which describe the physical and parameter attributes of the robot respectively. Thus, making it easier for roboticist to test his/her robots with various algorithms and codes by just changing the necessary parameters with respect to robot alone.Although, these XML and YAML act as a surface layer code formats, where the user pre-requisites pertaining to the coding language is drastically reduced, the backend implementations of the codes pertaining to the motion model, localisation, path planning, and navigation are all built using C++ and python with C++ being given much importance by the open source community for the aforementioned reasons.Therefore, the necessary implementations for the robot is done using the above saidXML and YAML formats with the codes being adopted from the open source community supported ROSWiki and gazebo plugins sources.

**3.1. Differential Drive Code:**

The motion model employed for the robot robot is a velocity model in “Differential Drive” configuration as the ScoBot is designed to have dual motors and wheels. Further more this XML code pertaining to the differential drive is modified as per the robot frame parameters and the joint names as shown in Figure 3.1. This differential drive module is used for teleoperation purpose of the robot and this is helpful in mapping and overall movement of the robot in gazebo.



Figure 3.1. Differential Drive Code

**3.2. LiDAR Sensor Code:**

The LiDAR sensor equipped with the robot is powered using the gazebo plugin from the source “http://gazebosim.org/tutorials?tut=ros\_gzplugins". This gazebo plug-in acts as an emulator for the real-life LiDAR sensor. The performancemetrics, gaussian noise parameters(with zero mean and standard deviation of 0.03m),sensor range(with minimum range set to 10mm and maximum range set to 20m),sensor resolution, sampling rates, and the respective scan topic name(/scan) which is useful for activating mapping node are all modified in the code as shown in Figure3.2.In the code given in Figure 3.2, the visualisation tag is set to false, whereas if set to true the laser lines emitting out of the LiDAR sensor could be visualised for estimation of range and to get a general intuition of how the sensor scans/performs.This is represented in Figure 3.3. Furthermore, the maximum and minimum angle of the sensor(set in code as radians) coverage is modified because of the robot’s structure



Figure 3.2. LiDAR Sensor Code

which would interrupt the laser rays and would lead to erroneous readings while mapping the environment with the help of LIDAR sensor. Therefore, the sensor angle is set as 115° based on the design analysis of the robot.

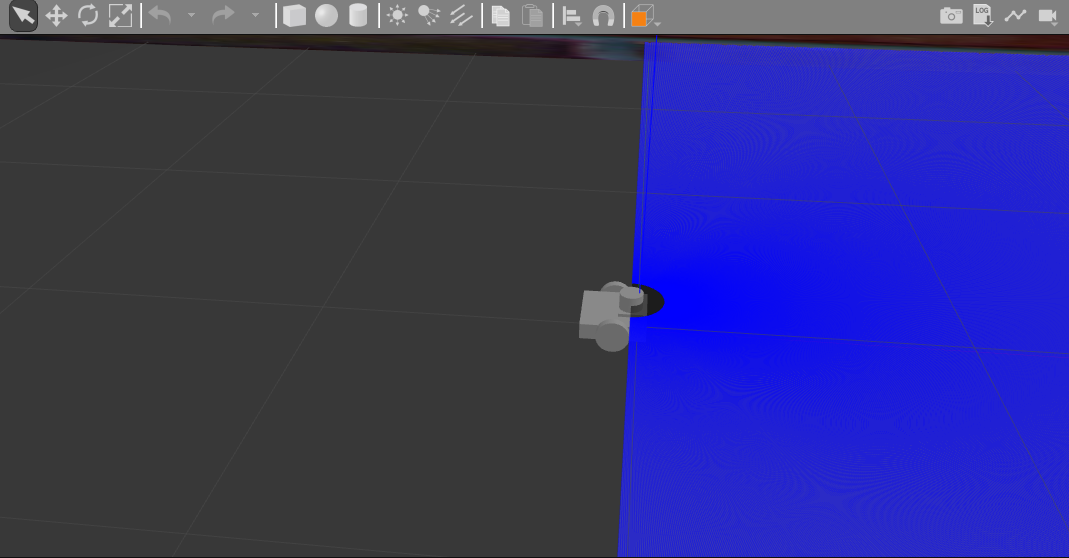


Figure 3.3. Laser rays from LiDAR sensor

**CHAPTER 4: RESULTS AND DISCUSSION**

The laptop/machine used for the simulation is configured with intel core i7

processor, 8 GB RAM and 500 GB hard disk. The real time factor of simulation is

nearer to 1 which is a good sign and thanks to collision model which helps in reducing the load on the CPU and also helps in simulation smoothness during execution.

**4.1. Mapping:**

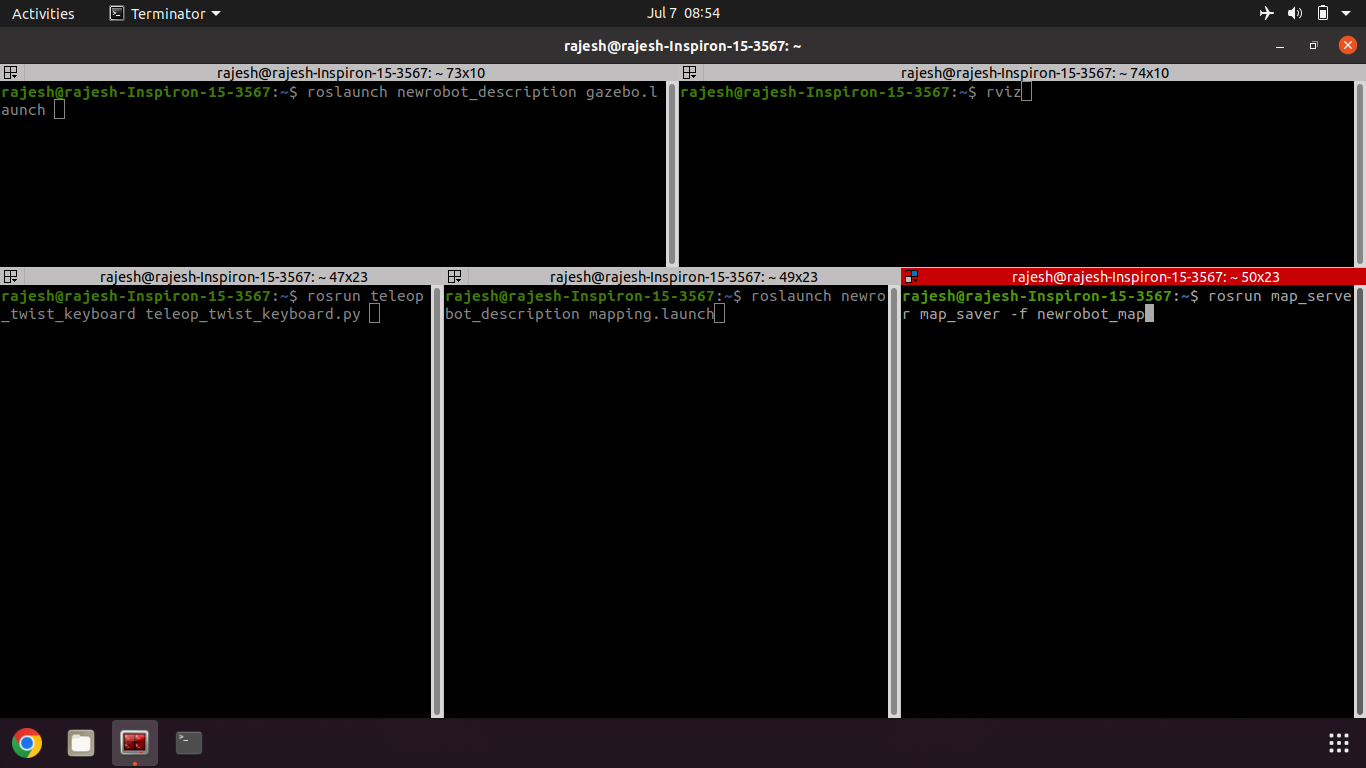
The mapping is done after installing the gmapping package from ROSWiki

page “http://wiki.ros.org/gmapping". Furthermore, the gmapping node is run while thegazebo and rviz pages are kept open so as to visualise and map the environment

simultaneously with the help of keyboard teleoperation and finally map(Occupancy

Grid Map) is saved to the 2dnav\_configuration. The process and the final map is

shown in Figure 4.1.



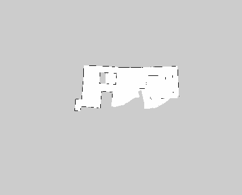


Figure 4.1 Mapping process and final generated map.

**4.2. Configuration Discussion:**

The entire implementation of the navigation stack and the spawning process of

robot in gazebo and rviz is simplified with just three terminal commands that are

executed in the subsequent terminal tabs by calling”Source/Devel/Setup.bash” file or it’s preferred alias as “sdsb”. The first command in terminal opens up gazebo and the second terminal command opens up rviz as shown in Figure 4.2.

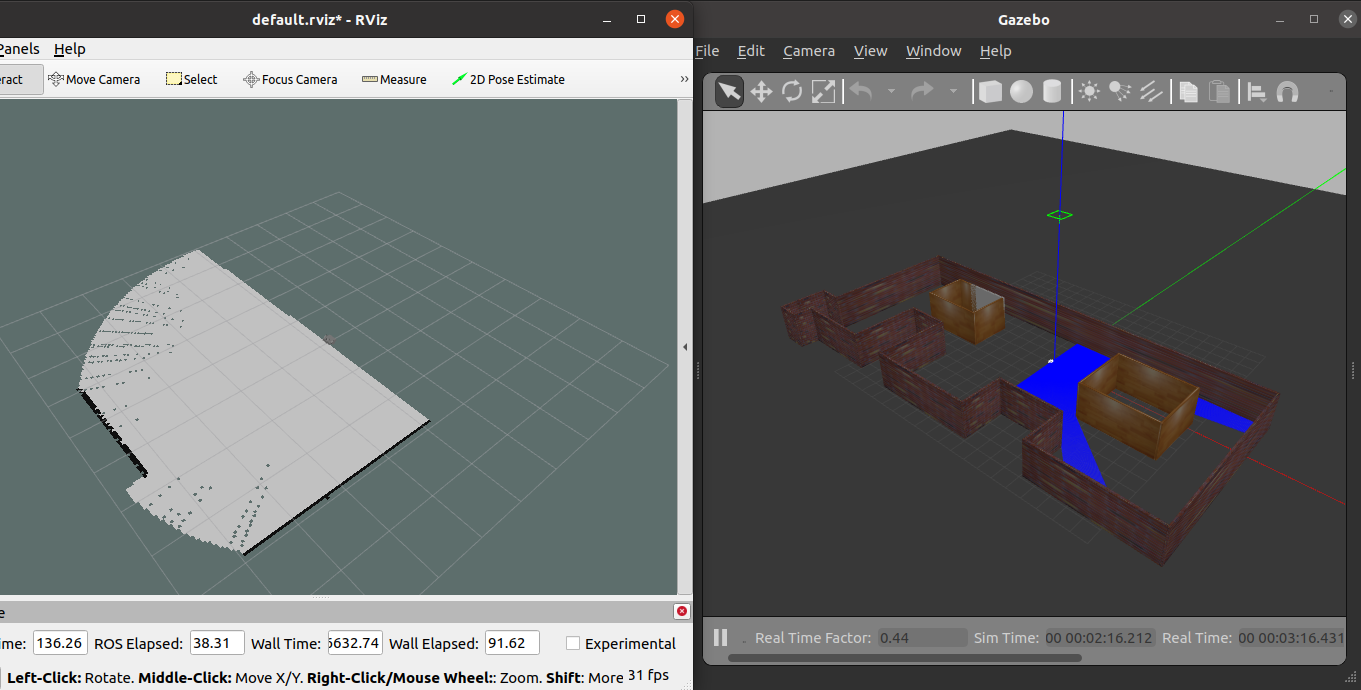


Figure 4.2. Gazebo and Rviz Windows

Furthermore, the 2D navigation stack is called upon in the third terminal tab which

sets up the stages and the aforementioned parameters needed for the robot to perform autonomous navigation in the gazebo simulation.

**CHAPTER 5: CONCLUSION AND FUTURE SCOPE**

Overall, the robot has been designed from FUSSION360. The process is ROS environment is also streamlined and is made suitable by tuning and modifying the parameters pertaining to the robot and the overall navigation stack is implemented and the Rviz parameters are also placedas per the requirement.The robot starts moving and also the convergence rate is also faster. This shows that the

fine tuning performed by the community members are perfect to be readily adopted

into any simulation projects.Moving ahead, the robot could also be physically developed for real-time operations such as goods transfer either in a rugged industrial environment, domestic environments(such as airports, malls, hospitals, grocery stores, etc), and even for personal use for performing in-house operations as well.Also, the motor and sensors could also be equipped/modified as per the application domain, this makes this particular design efficient, compact and also flexible in terms of applications. All the changes could also be performed in simulation environment as discussed in the above sections.

**REFERENCES**

1. Open Source Robotic Foundation. (2016) ROS/Introduction. [Online]. Available:

http://wiki.ros.org/ROS/Introduction

2. Robot simulation made easy. (2016)[Online]. Available: http:// gazebosim.org/

3. Torres-Torriti M, Arredondo T, Castillo-Pizarro P. Survey and comparative study

of free simulation software for mobile robots. Robotica. 2016

Apr;34(4):791-822.

4. Takaya K, Asai T, Kroumov V, Smarandache F. Simulation environment for

mobile robots testing using ROS and Gazebo. In2016 20th International

Conference on System Theory, Control and Computing (ICSTCC) 2016

Oct 13 (pp. 96-101). IEEE.

5. Megalingam RK, Teja CR, Sreekanth S, Raj A. ROS based autonomous indoor

navigation simulation using SLAM algorithm. International Journal of Pure

and Applied Mathematics. 2018;118(7):199-205.

6. Chikurtev D. Mobile Robot Simulation and Navigation in ROS and Gazebo.

In2020 International Conference Automatics and Informatics (ICAI) 2020

Oct 1 (pp. 1-6). IEEE.

7. Achmad MH, Nuryadi S, Fadlun W, Daud MR. ROS-based 2-D Mapping Using

Non-holonomic Differential Mobile Robot. JURNAL INFOTEL. 2018

Jul 10;10(2):75-82.

8. Darshan K T, Rakesh M, 2019, Simulation of an Autonomous Navigation System

using Quadrotor Model in Robot Operating System, INTERNATIONAL

JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT)

NCRACES – 2019 (Volume 7, Issue 10).

9. Konstantin Klimov. (2018, August 6) rp-lidar a1. [Online]

https://grabcad.com/library/rp-lidar-a1-inventor-reassembly-1

10. Prakhar Singh. (2020, September 18) raspberry-pi-3. [Online]

https://grabcad.com/library/raspberry-pi-model-3-b-1

11. Anjay V. (2019, February 10) n20-micro-metal-gear-motor. [Online]

https://grabcad.com/library/n20-motor-with-magnetic-encoder-1

12. Alex Guacho. (2021, May 5) openmanipulator-x-frameset. [Online]

https://grabcad.com/library/openmanipulator-x-frame-set-rm-x52-1

13. Toshinori Kitamura (2021) fusion2urdf [Source code].

https://github.com/syuntoku14/fusion2urdf