CHAPTER 1

Introduction

An automated guided vehicle or automatic guided vehicle (AGV) is a portable robot that follows markers or wires in the floor, or uses vision, magnets, or lasers for navigation. The aim of this project is to build a navigation system for an AGV that finds its application in warehouses ,AGVs used in warehouses and distribution centres logically move loads around the warehouses and prepare them for shipping/loading or receiving or move them from an induction conveyor to logical storage locations within the warehouse.

Simultaneous localization and mapping (SLAM) was originally developed by Hugh Durrant-Whyte and John J. Leonard, it is concerned with the problem of building a map of an unknown environment by a mobile robot while at the same time navigating the environment using the map. Currently simulations have been conducted in ROS (Robot Operating System) and the operation of the navigation system is tested. Mecanum wheels have been used for Omni directional movement. The prototype of the model is built using BeagleBone Blue which is an all-in-one Linux-based computer for robotics. It has an on-board Wifi and Bluetooth module along with various other sensors.

CHAPTER 2
Methodology and Working

Robot Description

The modelling of the robot has been done using solidworks, Fig1 shows the solid works model of the robot. The model consists of a chassis to which four motors with quadrature encoders have been attached. Mecanum wheels have been used to provide Omni-directional movement, a BeagleBone Blue is placed on the chassis, and the LIDAR is mounted on the top.

The URDF (Universal Robot Description Format) model is a collection of files that describe a robots physical description to ROS. These files are used by ROS to tell the computer what the robot actually looks like in real life. URDF files are needed in order for ROS to understand and be able to simulate situations with the robot before a researcher or engineer actually acquires the robot. It is the standard ROS XML representation of the robot model.Here individual parts have been designed and assembled as a whole, and once the model is ready a SW2URDF addin was installed for exporting the model to a URDF file (Unified Robot Description Format), this file is used to perform simulation in ROS.

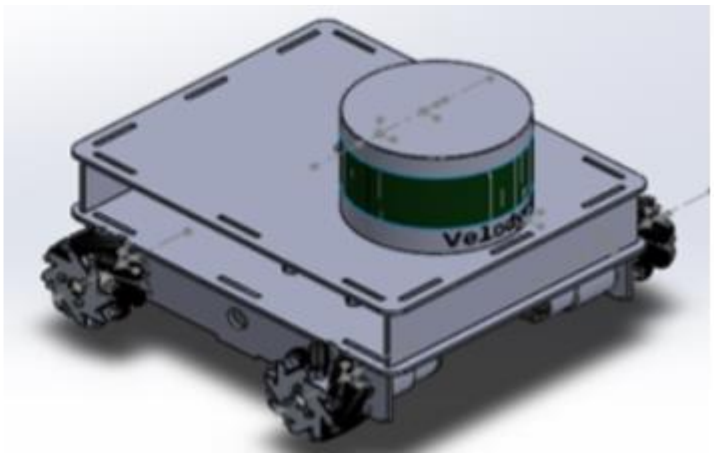


Fig1: Solidworks model of the robot

ROS

Robot Operating System or ROS is an open source flexible framework used for the development of robotic software. Even though ROS is not an Operating System it acts as a robotic middleware for heterogeneous computer cluster such as message-passing between processes, implementation of commonly used functionality, hardware abstraction, low-level device control, and package management. It is a set of libraries, tools, and conventions that facilitates in simplifying the complexity of mathematics and mechanics required to design the robot. It serves as a common platform to many robots in the world.

ROS supports various planner softwares and simulators like MoveIt!, Rviz, and Gazebo for various robotic applications like arm motion planner, navigation, environment mapping, camera calibration, point cloud mapping, etc. ROS by itself is not a real-time framework but it is possible to integrate ROS with real-time code. ROS supports libraries development through roscpp and rospy.

The URDF is extracted from the soildworks model and imported into ROS. The URDF is only a description of the robot in order to obtain the robot necessary changes are to be made such as:

- **Inertia matrix**: The inertia matrix has to be changed such that the robot gains stability in the Gazebo world, meshlab has been used to determine the inertia matrix.
- **Gazebo tags**: To operate the robot in Gazebo it requires the use of 'Gazebo Tags' this has to written in the URDF.
- **Material** : The robot is to be coloured to gain visual appeal
- **Sensors**: Gazebo tags are added to recognize the LIDAR as a sensor and necessary parameters are included in according to the requirements.
- **Diffrential drive controller**: Controller for differential drive wheel systems. This is necessary for locomotion of the robot.

ROS computation graph

ROS works on peer-to-peer network of ROS graphs and processes that are processing data together. The basic and important concepts used in ROS Computation Graph are nodes, master, messages, services, topics, parameter server, and bag all of which provide Computation Graph in many ways.

- **Node**: Nodes are different processes that perform computation. For a complex robotic control system, there can be any number of nodes and all of which can communicate with each other. For example, in a robot, the kinematics can be computed in a node, navigation in other node, finding coordinates of object using camera integration through other, and so on. The nodes for ROS can be written using ROS client library such as roscpp or rospy.
- **Master**: The ROS Master provides name registration and lookup to the rest of the Computation Graph. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.
- **Parameter Server**: The Parameter Server allows data to be stored by key in a central location. It is currently part of the Master
- **Messages**: Nodes communicate with each other by passing messages. A message is simply a data structure, comprising typed fields. Standard primitive types (integer, floating point, boolean, etc.) are supported, as are arrays of primitive types. Messages can include arbitrarily nested structures and arrays (much like C structs).
- **Topics**: Messages are routed via a transport system with publish/subscribe semantics. A node sends out a message by publishing it to a given topic. The topic is a name that is used to identify the content of the message. A node that is interested in a certain kind of data will subscribe to the appropriate topic. There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics. This is depicted in the Fig2. In general, publishers and subscribers are not aware of each other's existence.
- **Services:** The Publish/Subscribe model is a very flexible communication paradigm, but its many-to-many, one-way transport is not appropriate for request/reply

interactions, which are often required in a distributed system. Request/reply is done via services, which are defined by a pair of message structures: one for the request and one for the reply. A providing node offers a service under a name and a client uses the service by sending the request message and awaiting the reply.

- **Bags**: Bags are a format for saving and playing back ROS message data. Bags are an important mechanism for storing data, such as sensor data, that can be difficult to collect but is necessary for developing and testing algorithms.

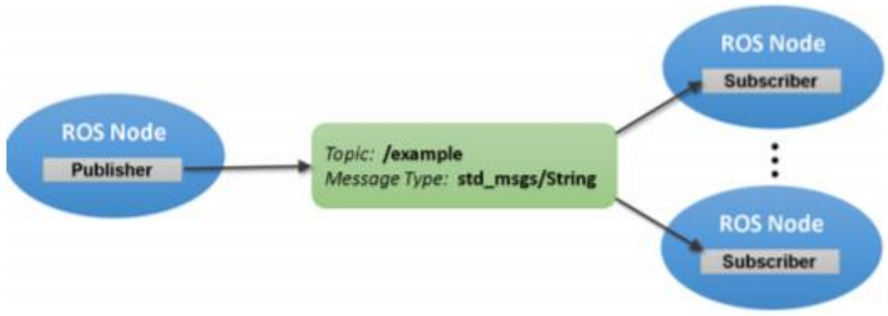


Fig2: ROS Publish/Subscribe

MoveIt! Rviz Plugin

MoveIt! comes with a plugin for the ROS Visualizer (RViz). The plugin allows to setup scenes in which the robot will work, generate plans, visualize the output and interact directly with a visualized robot. Once the robot description (URDF and SRDF) are generated, the robot is launched into RViz to help visualise and interact with the robot. Fig4 shows the robot in Rviz.

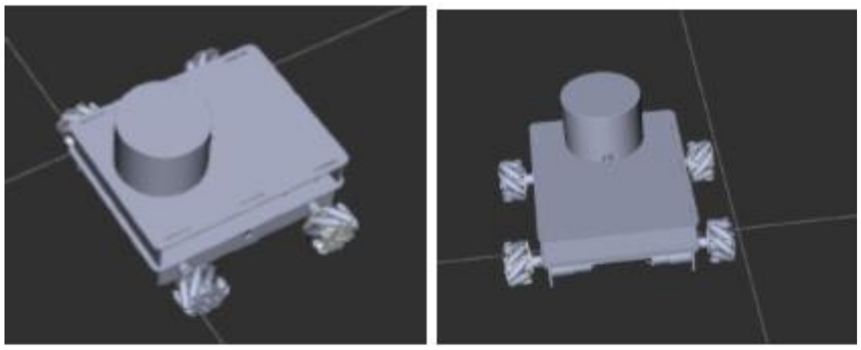


Fig4: Robot in Rivz

Gazebo

Gazebo is a simulator to simulate the robot before actually implementing it on hardware to prevent any catastrophic damages to the robot. Gazebo has its own engine to compute the physics for the robot world. The “gazebo ros pkgs” is a set of ROS packages that provide the necessary interfaces to simulate a robot in the Gazebo 3D rigid body simulator for robots. It integrates with ROS using ROS messages, services and dynamic reconfigure. In order to use Gazebo in simulating the robot there has to be Gazebo tags in the robot description. URDF file.

Navigation

The term SLAM is as stated an acronym for Simultaneous Localization and Mapping. It was originally developed by Hugh Durrant-Whyte and John J. Leonard based on earlier work by Smith, Self and Cheeseman. Durrant-Whyte and Leonard originally termed it SMAL but it was later changed to give a better impact. SLAM is concerned with the problem of building a map of an unknown environment by a mobile robot while at the same time navigating the environment using the map. SLAM is applicable for both 2D and 3D motion, but we have only considered 2D motion.

The implementation of SLAM requires the use of a distance measurement device, a LIDAR is used for this purpose. The robot is first traversed through the environment and a map is generated, this map helps the robot to gain familiarity of the environment. Locomotion, LASER scanning, Map generation, Localization and Path Planning are the steps for implementing SLAM.

Locomotion

Locomotion is to travel from one place to another, here the locomotion is performed in Cartesian co-ordinate system.

odom_pub

odom_pub is a python script for the localization of the robot within the coordinate space of the Gazebo simulation. It takes the tf information published by the URDF between the chassis link and odom frames to calculate the position of the robot. Also, aids in setting the velocity of the robot by the Differential Drive controller.

Differential Drive Controller

Controller for differential drive wheel systems. Control is in the form of a velocity command that is split then sent on the two wheels of a differential drive wheel base. The controller works with a velocity twist from which it extracts the x component of the linear velocity and the z component of the angular velocity. Velocities on other components are ignored. The controller works with wheel joints through a velocity interface.

Robot Controller

This is a python script which sends velocity commands which are then handled by the Differential Drive Controller, which is part of the URDF of the robot. Thereafter rotation and translation is handled by the differential drive controller. Specific x,y co-ordinates are given in space, to which the robot first rotates to minimize the angle difference towards the destination. Then, a specific velocity is given to the velocity interface.

LASER scan and Map generation

The “gmapping” package provides laser-based SLAM (Simultaneous Localization and Mapping), as a ROS node called slam gmapping. Using slam gmapping, and create a 2-D occupancy grid map (like a building floorplan) from laser and pose data collected by a mobile robot.

To use slam gmapping, a mobile robot that provides odometry data and is equipped with a horizontally-mounted, fixed laser range-finder is required. The slam gmapping node will attempt to transform each incoming scan into the odom (odometry) tf frame. Rao-Blackwellized particle filters have been introduced as effective means to solve the simultaneous localization and mapping (SLAM) problem. This approach uses a particle filter in which each particle carries an individual map of the environment. Accordingly, a key question is how to reduce the number of particles. We present adaptive techniques to reduce the number of particles in a Rao Blackwellized particle filter for learning grid maps. We propose an approach to compute an accurate proposal distribution taking into account not only the movement of the robot but also the most recent observation. This drastically decrease the uncertainty about the robot’s pose in the prediction step of the filter. Furthermore, we apply an approach to selectively carry out re-sampling operations which seriously reduces the problem of particle depletion.

Localization

Localization is the ability of the robot to determine its orientation and its position in the map. Path Planning involves determining (if it exists) a path to reach a given goal location given a localized robot and a map of traversable regions.
The map generated is sufficient for the robot to navigate through the environment given that the environment is static, the map fully reflects the environment and there are no errors in estimate, but in reality the environment changes (e.g. opening/closing doors), It is dynamic (things might appear/disappear from the perception range of the robot) and the data obtained is noisy.

Thus there is a need to complement our ideas deal design with other components that address these issues, namely Obstacle-Detection/Avoidance Local Map Refinement, based on the most recent sensor reading.

ROS Navigation Stack is used, the “Map Server” provides the map. ROS implements the Adaptive Monte Carlo Localization algorithm (AMCL). AMCL uses a particle filter to track the position of the robot, Each pose is represented by a particle, the rplidar scans the environment and it is matched against the already available map and the position of the robot is estimated, the odometry data provides the direction of motion.

BeagleBone Blue

BeagleBone Blue is the affordable and complete robotics controller built around the popular BeagleBone open hardware computer. Linux-enabled, Blue is community-supported and fully open-source. High-performance, flexible networking capabilities are coupled with a real-time capable Linux system and a compelling set of peripherals for building mobile robots quickly and affordably. Utilizing the pre-configured WiFi access point, starting your code development is as simple as connecting a battery and opening your web browser. Fig5 show the BeagleBone Blue.

It consist of a Octavo OSD3358 1GHz ARM Cortex-A8 processor, with 512MB DDR3 RAM, 4GB 8-bit on-board flash storage, 32-bit 200-MHz programmable real-time units (PRUs),On board flash programmed with Linux distribution. It supports Debian, ROS. Fig6 shows the BeagleBone Blue with all sensors labelled.

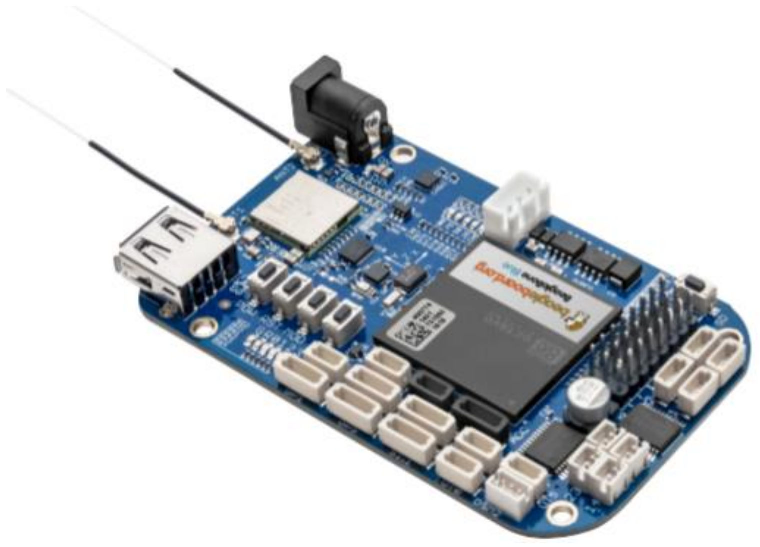


Fig5: BeagleBone Blue

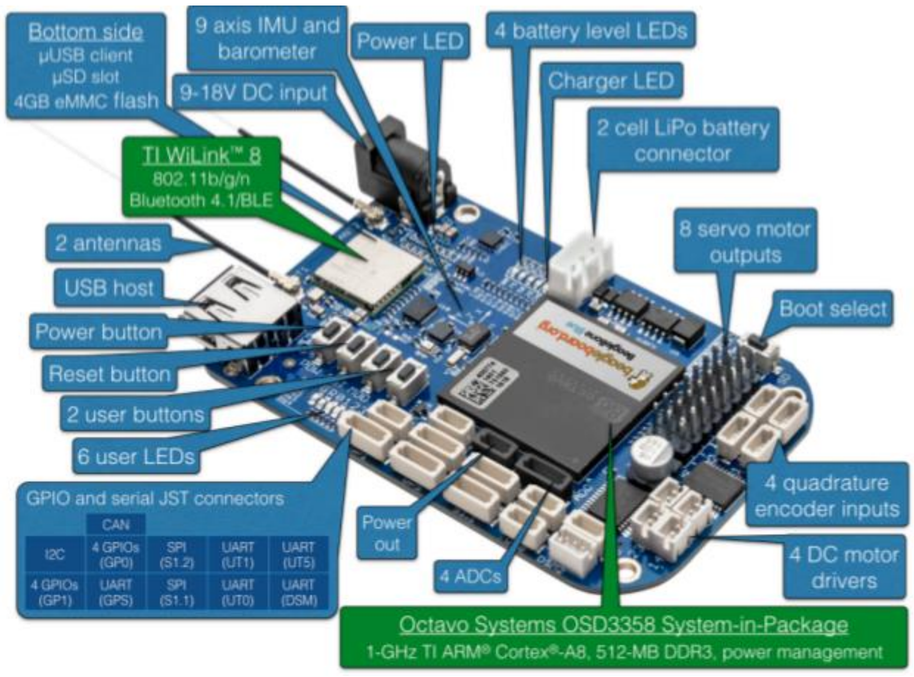


Fig6: BeagleBone Blue with all sensors labelled.

CHAPTER 3

Results

Simulations have been conducted in ROS and the BeagleBone Blue has been programmed to receive data through Bluetooth, the following section gives a detailed description of the results.

Robot Model

Robot model was designed in Solidworks and the URDF for the robot was successfully developed and integrated with ROS. The robot model can be used in Gazebo simulator and Rviz. The presence of the robot in the Gazebo world can be seen in Fig7 and Fig8.

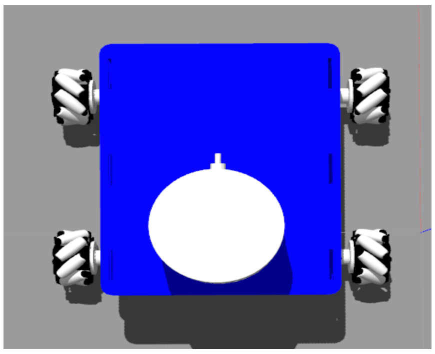


Fig7: Top view of the robot

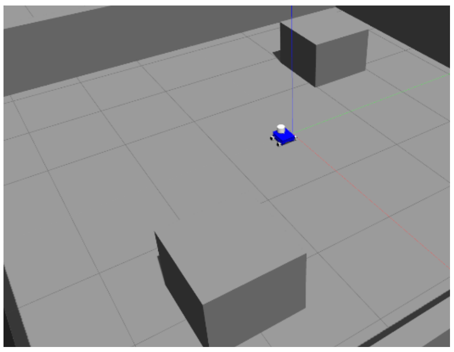


Fig8: Robot in Gazebo environment

SLAM

The gmapping package provides laser-based SLAM (Simultaneous Localization and Mapping), as a ROS node called slam gmapping. Using slam gmapping, a 2-D occupancy grid map can be created (like a building floorplan) from laser and pose data collected by a mobile robot. The Robot scanning the Gazebo world is shown in Fig9 and the LASER scan results is shown in Fig10.As the robot traverses the environment and the map is generated the process of map generation is shown in Fig11 and the resultant map generated when the robot completes traversing the entire world is shown in Fig12.

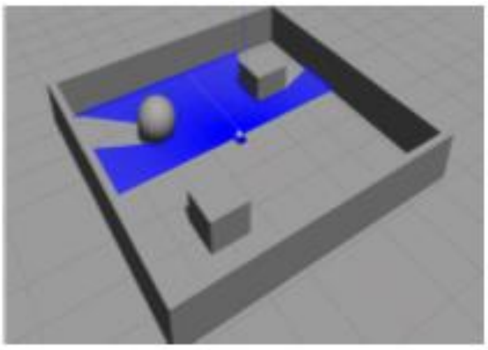


Fig9: Robot scanning the Gazebo

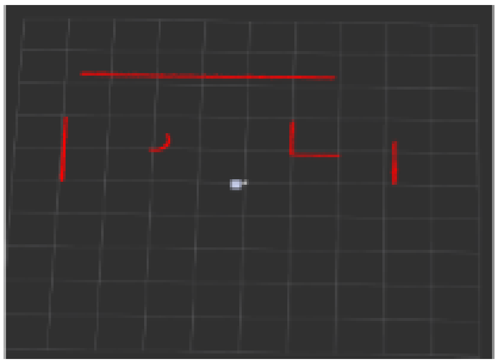


Fig10: The laser scan of the environment

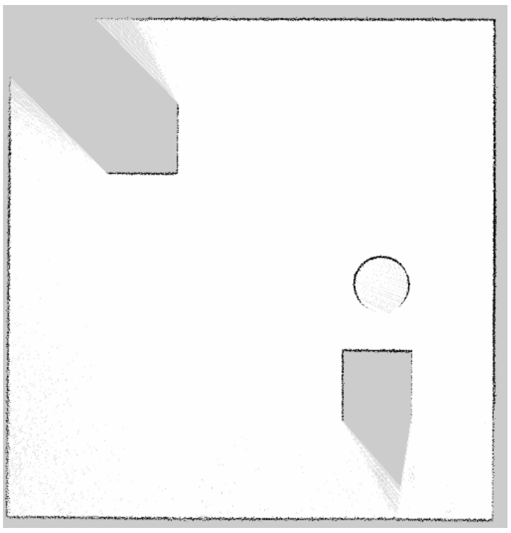


Fig11: Map under construction

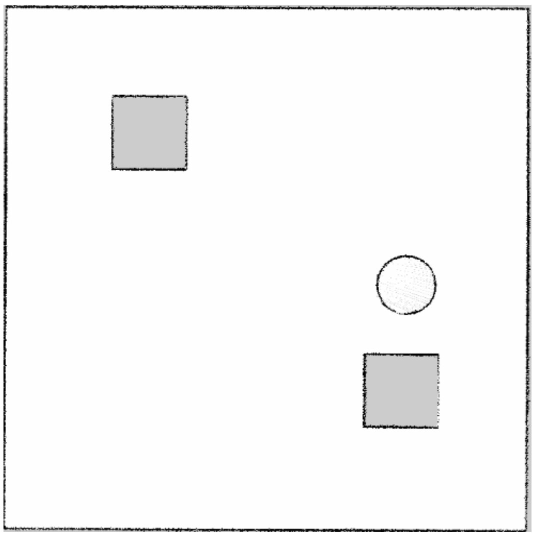
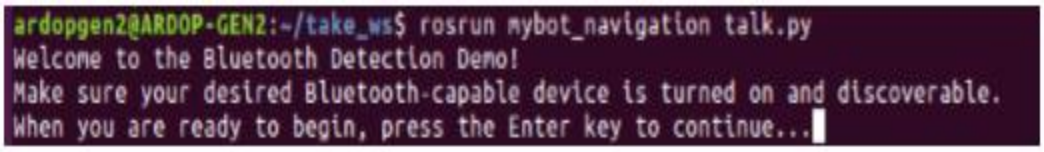


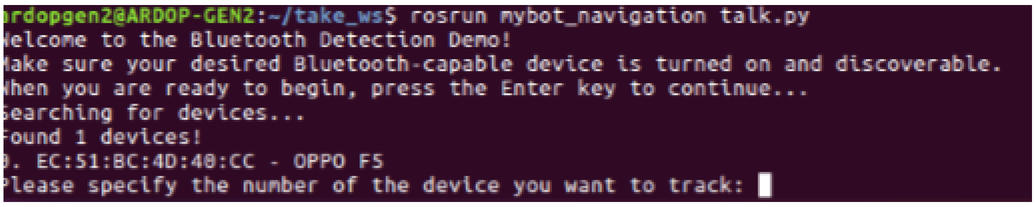
Fig12: Map after complete generation

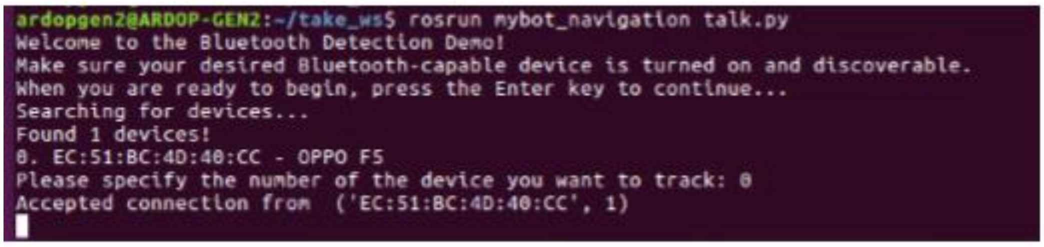
It can be seen from the above figures that all the visible edges of the objects in the sight of the robot can be seen in Fig 10, and as the robot traverses through the environment at sees all the edges and the complete map can be generated as shown in the Fig12.

Remote Control of Locomotion in Simulation Using Mobile Phone

A topic named /cmd_vel is published to the /Gazebo node and this topic publishes Twist messages to the wheels. The real world scenario where the map will be generated by moving the robot using a Bluetooth module was replicated in simulation. This was achieved pairing the Bluetooth module with the computer and a python script named talk.py is run it connects to the mobile app and the data is received, The process of connection is shown in Fig13, Fig14, Fig15.


```
ardopgen2@ARDOP-GEN2:~/take_ws$ rosrun mybot_navigation talk.py
Welcome to the Bluetooth Detection Demo!
Make sure your desired Bluetooth-capable device is turned on and discoverable.
When you are ready to begin, press the Enter key to continue...
```
Fig13: Enter key is to be pressed to start searching for a Bluetooth device

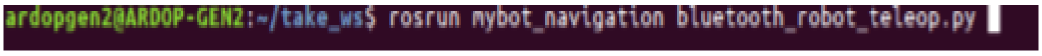

```
rdopgen2@ARDOP-GEN2:~/take_ws$ rosrun mybot_navigation talk.py
elcome to the Bluetooth Detection Demo!
ake sure your desired Bluetooth-capable device is turned on and discoverable.
hen you are ready to begin, press the Enter key to continue...
earching for devices...
ound 1 devices!
. EC:51:BC:4D:40:CC - OPPO F5
lease specify the number of the device you want to track:
```
Fig14: The search is complete and the mobile is found



```
ardopgen2@ARDOP-GEN2:~/take_ws$ rosrun mybot_navigation talk.py
Welcome to the Bluetooth Detection Demo!
Make sure your desired Bluetooth-capable device is turned on and discoverable.
When you are ready to begin, press the Enter key to continue...
Searching for devices...
Found 1 devices!
0. EC:51:BC:4D:40:CC - OPPO F5
Please specify the number of the device you want to track: 0
Accepted connection from  ('EC:51:BC:4D:40:CC', 1)
```

Fig15: The connection is established and the computer can receive data.

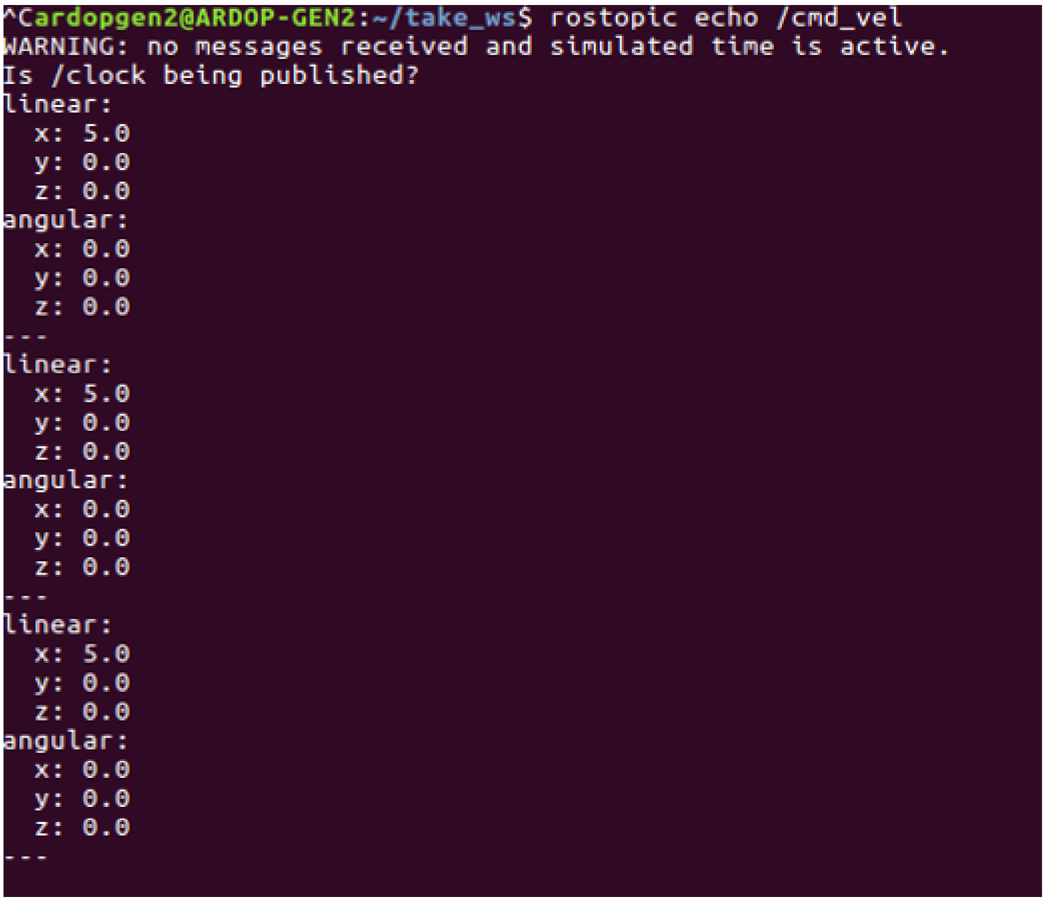
Once the connection is established and Blue Term (mobile App) is cappable of sending data across, a subscriber is to be created such that it is cappable of receiving this data and publishing the /cmd_vel topic to the /Gazebo node, to achieve this a subscriber named bluetooth_robot_teleop.py is created and run as shown in Fig16.



```
ardopgen2@ARDOP-GEN2:~/take_ws$ rosrun mybot_navigation bluetooth_robot_teleop.py
```

Fig16: Running the sunscriber as a node

The /cmd_vel consist of Twist messages these provide linear and angular velocity to the robot,the differential drive controller accepts linear velocities in the “x” direction and angular velocities around the “z” axis, all other components of llinear and angular velocities are rejected.The results are shown in Fig17.

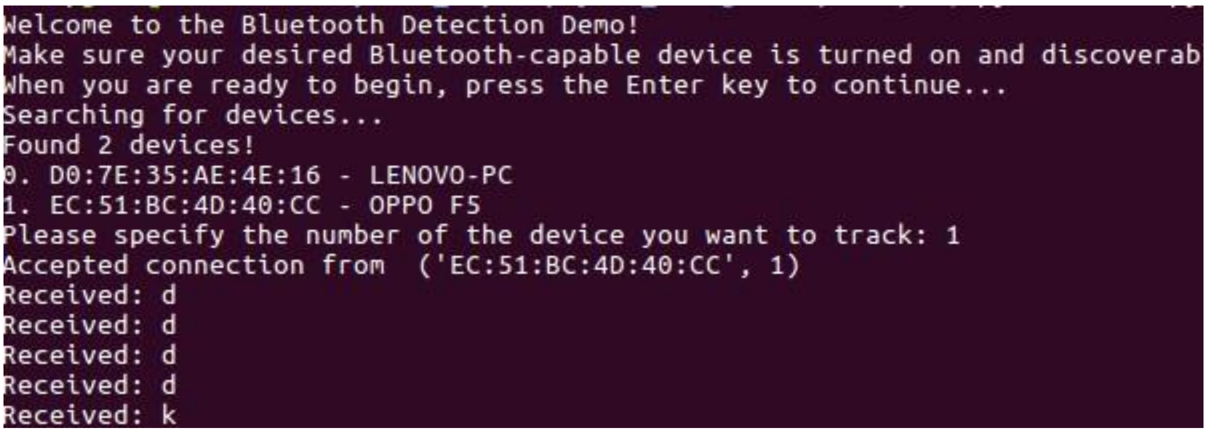


```
^Cardopgen2@ARDOP-GEN2:~/take_ws$ rostopic echo /cmd_vel
WARNING: no messages received and simulated time is active.
Is /clock being published?
linear:
  x: 5.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
linear:
  x: 5.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
linear:
  x: 5.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
```

Fig17: Publishing the /cmd_vel topic to the /Gazebo node.

BeagleBone Blue

The BeagleBone Blue is used to receive the data sent through Blue Term (an Android APP), this data received will provide direction and velocity to the robot. Currently the BeagleBone Blue is capable of reading the data that is sent through the App, the data received is published and the subscriber controls the motors .The results are shown in Fig18.



```
Welcome to the Bluetooth Detection Demo!
Make sure your desired Bluetooth-capable device is turned on and discoverab
When you are ready to begin, press the Enter key to continue...
Searching for devices...
Found 2 devices!
0. D0:7E:35:AE:4E:16 - LENOVO-PC
1. EC:51:BC:4D:40:CC - OPPO F5
Please specify the number of the device you want to track: 1
Accepted connection from  ('EC:51:BC:4D:40:CC', 1)
Received: d
Received: d
Received: d
Received: d
Received: k
```

Fig18: BeagleBone Blue reading data from Blue Term

Conclusion

The entire AGV was integrated with ROS and was extensively developed on it. Simulations have been conducted on Gazebo, these simulations can be carried out even in absence of hardware and SLAM has been implemented, the map of the gazebo environment is successfully generated and localization of the robot has been achieved using "Adaptive Monte Carlo Localization algorithm (AMCL)", locomotion of the AGV can be controlled through the mobile through Bluetooth.

References

1) http://moorerobots.com/blog/post/3
2) https://moveit.ros.org/documentation/concepts/
3) http://wiki.ros.org/ROS/Concepts
4) http://wiki.ros.org/gmapping
5) 2D SLAM Quality Evaluation Methods arxiv:1708.02354v1 [cs.RO] 8 AUG 2017
6) Introduction to navigation using ROS -- Giorgio Grisetti
7) http://wiki.ros.org/navigation
8) http://wiki.ros.org/navigation/Tutorials
9) http://beagleboard.org/blue
10) http://beagleboard.org/p/cw-early/simple-bluetooth-device-detection-0d2469