Navigation of Turtlebot using AR Marker

Shiekh Zeeshan
Department of Computer Science
University at Albany
State University of New York
New York, USA
zeeshansheikh15@gmail.com

***Abstract*—In this paper, design of an autonomous tracking of AR marker be turtlebot is discussed along with the actions associated with it for different AR markers. Initially the purpose of the turtlebot has been determined and then the required software and packages to implement the purpose has been obtained. After installing the required software and implementing the project important tests have been carried out and some solutions have been proposed for the problems encountered during the tests. The result obtained and experience achieved in the implementation of the project have been quite satisfactory and will be a guiding light for future works.**

I. INTRODUCTION

Today's robot system is a complex hardware device with different types of sensors and computers which have special software's to control them. Robots are able to navigate and perform important and challenging tasks in changing environment and conditions. Robots have a various of tasks and expectations associated with them for each task. They can achieve and perform functions and tasks which normally are done manually. As robots started to be more accurate and efficient than their human counterpart their demand was realized in more and more complex fields and was received with much enthusiasm [22].

All these led to a need for development of special software's and mechanism that can help us realize the tasks and expectation of the consumers. The use of AR markers was one such mechanism in which the robot detects and identifies different AR markers. After detecting the AR marker, it takes appropriate actions. This mechanism has a lot of scope for future implementations. It could be used for automatic landing of drones and other flying machines or it could be used to avoid automatic driving cars from an obstacle [21].

Even though there are lot of software platforms for simulation and controlling the robots but for the most part the ROS operating system is used. ROS helps in building efficient and reliable robot control and navigation software. It could be used to create a virtual simulation environment and then launch the robot in the hardware using remote control. It gives the users better control of the robot without going through the manual process of working on the real robot.

In this paper we are going to discuss one such useful implementation of the robot using ROS and AR marker. We design a software to detect and track the AR marker and take different actions according to the reading of AR marker. We also discuss the design and implementation of an environment for development and simulation of mobile robots using ROS.

The contribution of this paper is that the content of paper can be used as a tutorial for building automatic tracking of the AR marker and other related things associated with it. It could also be used to make an automatic obstacle avoidance mechanism and self-driving cars which take directions based on the reading of AR markers.

In this paper I have included on how to implement a turtlebot that can detect an AR marker and how it reacts after detecting it. The next step after the implementation of this was to make the turtlebot identify the ar marker and avoid it or move forward or backward according to the definitions we have given for the different reading and values of the AR markers.

The next step was to make a follower turtlebot which can follow a particular AR marker or any AR markers. Based on the reading it starts following the AR marker and moves towards it. If this AR marker is attached to a another turtlebot then if the other turtlebot moves using the teleoperation command usimg remote control or manual movement then this follower turtlebot also moves towards the leading turtlebot. It does so by detecting and tracking the AR markers attached to it.

The final implementation of this paper includes detailed description of two parts. The first has a turtlebot which detects an AR marker and moves forward and takes right and for another AR marker it turns right and moves forward. The second part has another turtlebot which follows the first turtlebot also known as the leader turtlebot. The leadet turtlebot moves according to the AR marker and the follower turtlebot moves according to the leader turtlebot.

The paper is organized as follows. The next section is going to be about ROS. It deals with the detailed description of the operating system and its associated functionalities and uses. It also discuses the role of the operating system in understanding the turtlebot. The following section deals with the description of the turtlebot and the kinds of robots to which this paper applies. It starts by discussing the hardware of the turtlebot and the functions associated with each hardware present. The next section includes the description of the AR marker and the meaning of the values associated with the readings. It also points out the importance of AR marker in our project. The fourth section discusses the implementation of the paper and the different software's used and the scripts and functions used for it. It discussed in detail the value and meaning of different functions and their effect on the turtlebot.

The remaining sections of the paper consists of the conclusion and references to all the papers and journals used for the implementation of the project. It also discusses the possible future works that can be associated with it and the modifications that could be required for that scope.

The last part of the paper which includes the future works is quite important as it discusses the things which could be implemented in the future and act as a standard.

II. Robotics Operating System

ROS is collection of libraries, drivers and tools for the effective development and implementation of robotics and its applications. It has a linux like command tool, communication between inter processes and various other application related packages [10]. The ROS has a Node which is the executable process and it has an publish and subscribe which are interprocess communication methods. One or more process can be published by the publisher and its content will be received by the processes which subscribe to a certain topic.

The ROS has one more advantage that sets it apart from its counterparts and that is it is platform independent. It is implemented in C++, python and lisp. It also has experimental libraries in java.

There are various versions of ROS out there and one of them is indigo. We are going to use this version of ROS since this is the most compatible of ROS versions with our hardware so far and also has a lot packages that help in the implementation of our project. It includes many sensor drivers and navigation tools and environment mapping, path planning, interprocess communication visualization tool, as well as a 3D environment visualization tool and many others.

ROS is well known for its property of scaling. It is very suitable for large runtime systems and for large development processes. ROS is framework which is distributive in nature which means that nodes or executables can be individually designed and then it can be loosely coupled at runtime so as to form one single unit. These processes are grouped under one package and stack.

ROS allows us to develop new robotic systems and when used it with a simulation the time for development gives a reliable and high performance. This helps us get more robotic control and software can be made better. There are various versions of ROS but we have included some of the most used versions below. The differences is due to the compatibility issues they have with different operating systems and environment.

A. Indigo

ROS indigo is primarily for Ubuntu 14.04 LTS (Trusty) release, but it also is supported by other operating systems to some degrees. We are using this version because it is the default version installed in our turtlebot and the operating system matches with the given version [18].

Fig. 1 Indigo Version of ROS operating system [18]

Indigo also has support for releasing testing and integration testing of catkin-based packages only. It is because it wants to provide long term support of this distribution. This gives us a good chance that our implementation need not undergo changes or updates and can work under the same environment for a long time.

B. Kinetic

ROS indigo is primarily for Ubuntu 16.04 (Xenial) release, but it also is supported by other operating systems to some degrees [17].

Fig. 2 Kinetic Version of ROS operating system [17]

C. Lunar

ROS indigo is primarily for Ubuntu 17.04 (Zesty) release, but it also is supported by other operating systems to some degrees [20].

ROS also has a lot of packages which are commonly used in the implementation of our project. Some of the packages are (i)*turtlebot_bringup_minimal_launch*, (ii)*turtlebot_teleop* These two functions along with other are used for starting the turtlebot and making some manual movements with the help of keyboard. The first one is required before the start of any program because it is the main launch file that connects the turtlebot and the laptop or the hardware and the software [15].

ROS has many other goals of sharing and collaboration. It is designed to be as thin as possible which means it is designed to be easily integrated with other robot software frameworks. A few examples of other frameworks with which it has already been integrated are OpenRAVE, Orocos, and Player.

The second goal is to develop ROS-agnostic libraries which means that it writes libraries with clean functional interfaces. It also has a language independence concept which means that the ROS framework can be easily implemented in any modern programming language. It has been already implemented in languages like python and C++. It also has some presence in languages like java and lisp in the form of libraries.

It also has easy testing functionality which offers great advantage to users. It has a unit called rostest which is an built in unit and integration test framework that makes it easy to bringup and tear down test fixtures. It provides functionalities like hardware abstraction, low level device control, message passing between processes, and package management.

III. TURTLEBOT

Turtlebot is based on open source and is very cheap robot kit that is made for personal use. It could be used to drive around the house and also use simulators that are available in the market to simulate them in the 3D environment and then drive them in real world [1].

Fig. 3 Turtlebot [1]

The kit which is mostly available in the market consists of a mobile base, a 2D, 3D distance sensor, a laptop or any kind of computer as long as it has all the necessary software's installed in it to make it run properly. It also has a mounting hardware kit and in addition to that the users can download the turtlebot SDK from the official website of ROS. One of the main advantages that turtlebot offers as compared to other counterparts available in the market is that it is easy to buy build and assemble and also its parts can be created from standard materials [12].

The turtlebot can be efficiently used to do realtime obstacle avoidance and autonomous navigation with the help of the inbuilt components that it comes with. Turtlebot is equipped to run standard Simultaneous localization and mapping (SLAM) algorithms to build a map and has the ability to be remotely controlled from a laptop or android based smart phone. There also have been examples of turtlebot following other turtlebots or people and any objects which move. The inbuilt sensors detect the motion and simultaneously react whether to move forward or backward depending upon the leader that it is following. These kind of examples are available to run as a demo in the official documentation and can be run with little or no knowledge of turtlebots and ROS programming. All the user has to do is to run a couple of simple commands and the turtlebot is good to go [2].

The turtlebot has various other demos available at the official ROS site and can be used and practiced to make it better or just for understanding purpose. The turtlebot official guide in ROS comes with one such demo which is called the turtlebot teleop command which runs with the help of keyboard and can be remotely controlled using SSH from another computer. In this demo the turtlebot is commanded to move in different directions and at different speeds using some simple keys.

The guide and the codes available for the turtlebot serves as good starting point for various users and developers all around the world. They use it to either run and learn the simple working of the turtlebot or they are used by advances developers to make it better and something that could be used more efficiently and has a better scope of being used in the real life.

Turtlebot also has been known to have accessories such as an arm attachment that can be attached to it and used for manipulating objects and other things that needs to be lifted or removed. It also can act as an alternative to the obstacle avoidance method and can remove the obstacle instead of diverting it from the path.

As can be seen that the turtlebots are a beginning to endless possibilities that can help the world in a numerous ways and can even act as an alternative for most of the manual works and maneuvering techniques known to us so far in driving or other places.

The turtlebot which I am going to use for this project is the turtlebot 2. This turtlebot consists of an YUJIN Kobuki base , a 2200mAh battery pack , a Kinect sensor. It also has an Asus 1215N laptop with a dual core processor. The turtlebot is equipped with a fast charger and a hardware mounting kit attaching everything together and adding future sensors.

A. YUJIN Kobuki base

It is a low cost mobile research bse designed for education and research on state of art robotics so as to be compatible with the turtlebot and other robots. One special feature of this base is its continuous operation and it is supported by power supplies for and external computer and the additional sensors that come attached to it. The kobuki base gives good navigation due to the fact that it has high accurate odometry and it uses factory calibrated gyroscope [11].

Fig. 4 Kobuki base of turtlebot [11]

Some of the specifications are that it has a maximum translational velocity of 70 cm/s and a maximum rotational velocity of 180 degree/s. It can carry a maximum payload of 5 kg and it cannot drive off a cliff with a depth greater than 5 cm. The expected operating time is 3/7 hours and expected charging time is 2-3 hours [11].

The kobuki is a mobile base which means that it has sensors and motors and power sources. For the base to be functional a user has to build its platform above its shell. On the hardware side it has a netbook or an embedded board which acts as an computational core for the system and it also has a few extra sensors which make it truly functional.

B. *Sensor*

The sensor used in the turtlebot 2 is Asus Xtion PRO. They are infrared sensors and it also uses adaptive depth detection technology to capture the real time body movement of users and other objects which are kept in the view of the sensor. It is this hardware that is used in the turtlebot following demo available in the ROS tutorial of turtlebot.

Fig. 5 ASUS Xtion PRO sensor of turtlebot [3]

There are a set of developer tools that helps the users to make their own gesture-based applications. For this purpose, they don't need to write complex programming algorithms. Some of the functions provided by these sensors are as following.

- *Gesture detection* – The Xtion PRO development solutions has the ability to track peoples hands motion without any delay and in real time effect which makes our hand an efficient controlling device. The various motions carried out by hand helps in achieving different functions at the same time.
- *Whole body detection* – The Xtion PRO development solutions has made the sensors in such a way that it can track a users whole body movement which has a series of useful benefits that can be made out of it. It could be used in gaming , health and other related fields to analyze the behavior of the human body without manually going through it.

These sensors are the worlds first and exclusive professional PC motion sensing software development solution. It has a SDK which is compatible to OPNI NITE middleware which helps in making the sensing applications and the development of games far easier as before [3].

The kit is widely open to development and modifications. Any user can develop their own applications for business or for making peoples lives more convenient and intuitive. The developers can apply the motion sensing technology to wide variety of fields and use them to achieve significant results.

IV. AR MARKER

Augmented reality markers or in short AR markers are visual cues which trigger the display of the visual information [9]. AR markers are usually normal images which are made in such a way that they can be identified by the camera and can give some useful meaning. It could be used by reading its id or other property which depends upon the size, width and other factors present in the image [4].

After a marker is recognized, properties like its scale and the angle of rotation or other properties are converted into virtual information and passed to the appropriate destination which could be the turtlebot or other system for which the AR marker is being recognized [7].

ROS has a library associated with it to create and detect AR markers. It is known as ar_track_alvar. After installing this library, it can be called to generate AR markers of various size and shape to be detected by the system. The AR markers are usually generated by the thickness of the lines which are present in the black square. The lines are white in colour and the thickness gives every marker a unique id. This id helps in the identification process [8].

The square shape of these tags helps in the real estimation of the position and the orientation of the AR marker. The camera detects the size in the image and distortion and then calculates the projections associated with it and then determines accurately the tag position. AR tags have different shapes and patterns depending upon the number that we have associated while we were generating the marker. The command used to generate the tag has a field of a numeral which is the main factor that influences the shape and size of the AR marker [6].

It should be noted that not all AR markers are detected by the turtlebot. The AR marker generated should be of some particular size. While printing the AR marker care should be taken that the AR marker is not to big and neither too small else it will not be detected

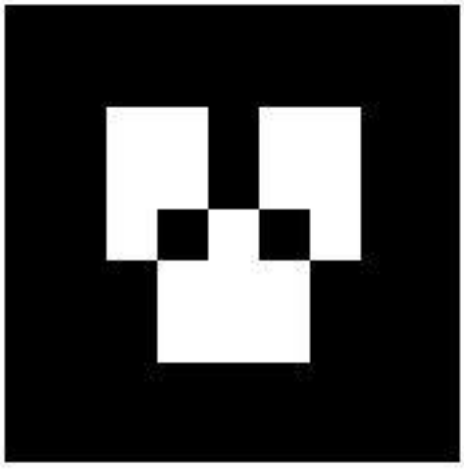

Fig. 6 AR Marker [5]

As we can see above we have a sample AR marker which was generted by the same ar_track_alvar library of ROS. There are various other scripts which can generate the AR marker without the need of installing the library but it is recommended to use the library for this purpose.

V. METHODOLOGY

The methodology we have used is divided into a number of steps which includes from installation of the ROS to implementation of the complete project comprising of the leader and follower turtlebots.

A. *Installation*

We begin by analyzing all the software's required for the successful implementation of our project. The first requirement is a compatible environment in which the turtlebot can run and function efficiently. The environment which we are going to use is the ROS indigo. After installing it we almost have all the packages needed for the implementation of our project. There are few packages that still need to be installed separately. Some of them are the ar_track_alvar package.after installing it we also need to install OpenCV. This library is crucial since it deals with the detection of the AR marker.

There is also another packages like bringup and teleop which needs to be installed separately if the ROS operating system is installed separately from the official Ubuntu 14.04. If the installation is from the turtebot documentation available in the official ROS website then we don't need to install the other packages separately since all of them are preinstalled with the Ubuntu 14.04 version of ROS. The ar_track_alvar package still needs to be installed separately because it is not included in it.

B. *Implementation*

The implementation initially starts with two phases in which we first try to generate the AR marker and then try to detect the same AR marker from the turtlebot.

- *Generating the AR marker*- In this part we start by generating the AR marker with the help of simple commands available to us in the documentation of the ar_track_alvar package. We can generate various AR markers of different shape and sizes depending on the numeral value that we attach to the command. It should be noted that all the AR markers generated are different in pattern and have a unique id associated with them.

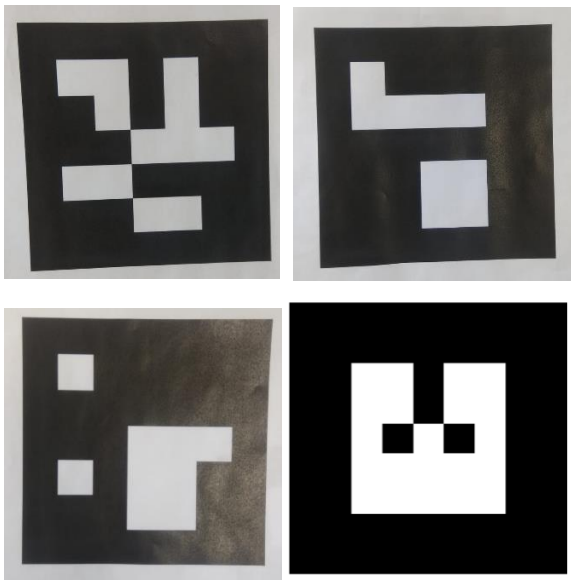

Fig. 7 AR Markers generated for this project

- *Detecting the AR marker*- In the AR marker detection part we can simply run the command associated with the ar_track_alvar official documentation page or we can write our own script to detect it. There are numerous scripts available online and in official ROS documentation which do the same thing. Some of the examples of ROS have show how to follow an AR marker or other demos which are related to detection of AR markers.

Fig. 8 AR Marker on the Leader Turtlebot

After we have successfully implemented the AR marker generator and detector, now it is time to do some actions based on the id of the AR marker.

Once we have the Id of the AR marker we can take various actions and command the turtlebot to act differently on detecting different AR marker. Now our task again has two phases. One is to command a turtlebot to take different actions on detecting different AR marker and the other is to make a turtebot follow the ar marker. The first one is called the leader in our implementation and the second one is called the follower since it follows the first turtlebot which has an AR marker attached to its back as shown in figure [8].

1) Leader turtlebot: In case of the leader turtlebot we have to implement a turtlebot which can take different actions on detecting different turtlebot. The implementation is quite related to the demos available to us in the official documentation of the turtlebot and ROS. The demos gives us a idea on how the turtlebot make movement. We can figure out after looking at the scripts that the motions are divided into two parts. One is the horizontal motion and the other is the rotational motion. The function we use for this purpose is called the twist function.We make the turtebot detect the AR marker and after it is detected we try to get its id and then print it on the console. After this step we can see that the different AR markers have different ids which can now be used to carry out different actions according to our purpose. The two actions are first to move forward and then turn right and then again move forward. The other is to take a left turn or turnfor 90 degrees and then move forward.
To implement this we have to first make the choice of the forward movement using the twist function. Once we detect

the AR marker of choice we start moving forward with a velocity of 10 cm /s. For this to be achieved we make two movements.

The first one is the move_cmd and the other is the turn_cmd. In the first one we make the horizontal speed of the twist function as 10cm/s and the rotational speed to be 0 rad/s. In this way we achieve the straight line motion without any rotation.

The second one is the turn_cmd which has a horizontal speed of 0 cm/s and an angular speed of 10 rad/s. This is because we don't want any horizontal movement in this function. This is only for rotational movement.

After we have made both the functions we now just need to call them in order to make it work. The turtlebot first starts by capturing frames in an infinite loop. As soon as a marker is detected it captures the frame and detects the id of the AR marker. After we get the id we see if the id is matching the id we specified on the script for first movement.

2) Follower turtlebot: In this part we have to implement a follower which on detecting a particular AR marker makes a forward movement towards the AR marker. Now in this case we have to first detect the AR marker and then print out the id of the AR marker on the console. After we got the expected id of the marker we start acting accordingly. Now we have two steps for the implementing of the follower turtlebot. They are move forward and rotate and search

Fig. 9 Follower Turtlebot

a) Rotate and search: In this step we start by making rotations in an infinite loop as soon as the turtlebot runs the script. Once it starts rotating infinitely it also starts capturing the frames from the camera. In this rotational motion if suddenly it detects an AR marker it captures the frame and starts the analysis of the detected AR marker. If the detected AR marker is of the expected id then we take action and move on to the next step which is move forward. The implementation of this part has been carried out by two twist functions. The move cmd and the turn cmd . Both of them have a horizontal component and a angular component. Since in this part we are only rotating the turtlebot we will only use the turn cmd and ignore the move cmd function for now. The turn cmd function will have a angular speed of 40-50 rad/s and a horizontal velocity of 0 cm/s as we donot want any horizontal movement in this case .This is just to rotate the turtlebot and search for the AR marker.

Fig. 10 AR Marker on the Leader Turtlebot and the Follower Turtlebot

b) Move forward: In this case we need to make a forward movement after detecting the AR marker. If the detected marker id is valid we stop the rotational movement and start moving towards the AR marker. This is achieved by using the move cmd function which gets called when the AR marker frame gets captured and the detected id is that of the expected range or number. There is a special case in this movements and that is as we have mentioned that the AR marker is attached to the leader turtlebot which is also moving in some direction so in this case there is a possibility that the AR marker can go out of view of the follower turtlebots camera. When something like this happens , the follower is being programmed to start rotating againg and stop moving forward as soon as the AR marker goes out of view. It again starts the cycle of rotating and searching for the AR marker till it finds it.

VI. LIMITATIONS

After the implementation of the project I started to test it by running it for some times. There were few limitations that I realized were present in the project.

After the leader turtlebot starts running and follows the directions given by the AR markers it goes right and then left as directed by the AR marker. The follower turtlebot is supposed to follow the movement of the leader turtlebot and copy its directions. But there was some lagging behind due to the fact that the camera was not able to detect the AR marker when the rotational motion was fast. The rotational motion was kept in such a way that it is able to detect a moving AR marker but for this the rotate and search operation became very slow and resulted in lagging behind the leader.

Another limitation was that the turtlebot was not able to stop at a certain distance before the AR marker. It kept moving in the direction of the AR marker as long as it could see the AR marker. This again is due to the fact that I was using the laptop camera instead of the Kinect camera on the turtlebot which could have given the distance of the AR marker and we could have stopped the turtlebot at a distance from the marker.

VII. Conclusion

The aim of this project was to gain first hand experience on working with the turtlebot and understanding ROS and AR markers. The project goal was achieved as I successfully made a leader and a follower turtlebot which could follow a leader turtlebot which has an AR marker attached to it. There were some lagging and some limitations of the movement and it could have been more efficient if the sensor and turtlebot camera was used. The leader was efficient and the response time after detecting the AR marker was also fast. The follower was also accurately able to follow the leader turtlebot and reach the destination.

References

Below are some of the references used for the implementation of the project.

[1] "TurtleBot 2 - Open source personal research robot", *Clearpath Robotics*, 2019. [Online]. Available: https://www.clearpathrobotics.com/turtlebot-2-open-source-robot/. [Accessed: 09- May- 2019].

[2] "TurtleBot", *Turtlebot.com*, 2019. [Online]. Available: https://www.turtlebot.com/. [Accessed: 09- May- 2019].

[3] [8]"Xtion PRO | 3D Sensor | ASUS", *ASUS*, 2019. [Online]. Available: https://www.asus.com/me-en/3D-Sensor/Xtion_PRO/. [Accessed: 09- May- 2019].

[4] "What are augmented reality markers ? - AnyMotion GmbH", *AnyMotion GmbH*, 2019. [Online]. Available: https://anymotion.com/en/wissensgrundlagen/augmented-reality-marker. [Accessed: 09- May- 2019].

[5] "Detecting and Tracking AR Tags", *Reality Bytes*, 2019. [Online]. Available: https://realitybytes.blog/2017/06/02/detecting-and-tracking-ar-tags/. [Accessed: 09- May- 2019].

[6] "AR Markers - TIME SQUATTERS", *TIME SQUATTERS*, 2019. [Online]. Available: https://www.timesquatters.com/ar-markers/. [Accessed: 09- May- 2019].

[7] Y. Levski, "Markerless vs. Marker Based Augmented Reality", *Appreal-vr.com*, 2019. [Online]. Available: https://appreal-vr.com/blog/markerless-vs-marker-based-augmented-reality/. [Accessed: 09- May- 2019].

[8] "Augmented reality", *En.wikipedia.org*, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Augmented_reality#Navigation. [Accessed: 09- May- 2019].

[9] [7]"What is augmented reality (AR)? - Definition from WhatIs.com", *WhatIs.com*, 2019. [Online]. Available: https://whatis.techtarget.com/definition/augmented-reality-AR. [Accessed: 09- May- 2019].

[10] "Robots/TurtleBot - ROS Wiki", *Wiki.ros.org*, 2019. [Online]. Available: http://wiki.ros.org/Robots/TurtleBot. [Accessed: 09- May- 2019].

[11] "About | KOBUKI", *KOBUKI*, 2019. [Online]. Available: http://kobuki.yujinrobot.com/about2/. [Accessed: 09- May- 2019].

[12] *Generationrobots.com*, 2019. [Online]. Available: https://www.generationrobots.com/media/TurtleBot-2-UserManual.pdf. [Accessed: 09- May- 2019].

[13] R. Advance, "TurtleBot 2 (complete & assembly) - TURTLEBOT", *Robot-advance.com*, 2019. [Online]. Available: https://www.robot-advance.com/EN/art-turtlebot-2-complete-assembly-1189.htm. [Accessed: 09- May- 2019].

[14] "REP 119 -- Specification for TurtleBot Compatible Platforms (ROS.org)", *Ros.org*, 2019. [Online]. Available: http://www.ros.org/reps/rep-0119.html. [Accessed: 09- May- 2019].

[15] "turtlebot_bringup/Tutorials/indigo/TurtleBot Bringup - ROS Wiki", *Wiki.ros.org*, 2019. [Online]. Available: http://wiki.ros.org/turtlebot_bringup/Tutorials/indigo/TurtleBot%20Bringup. [Accessed: 09- May- 2019].

[16] "turtlebot/turtlebot2e", *GitHub*, 2019. [Online]. Available: https://github.com/turtlebot/turtlebot2e/blob/master/README.md. [Accessed: 09- May- 2019].

[17] "kinetic - ROS Wiki", *Wiki.ros.org*, 2019. [Online]. Available: http://wiki.ros.org/kinetic. [Accessed: 09- May- 2019].

[18] "indigo - ROS Wiki", *Wiki.ros.org*, 2019. [Online]. Available: http://wiki.ros.org/indigo. [Accessed: 09- May- 2019].

[19] "ROS/Tutorials - ROS Wiki", *Wiki.ros.org*, 2019. [Online]. Available: http://wiki.ros.org/ROS/Tutorials#Core_ROS_Tutorials. [Accessed: 09- May- 2019].

[20] "lunar - ROS Wiki", *Wiki.ros.org*, 2019. [Online]. Available: http://wiki.ros.org/lunar. [Accessed: 09- May- 2019].

[21] "Simulation environment for mobile robots testing using ROS and Gazebo - IEEE Conference Publication", *Ieeexplore.ieee.org*, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/7790647. [Accessed: 09- May- 2019].

[22] 2019. [Online]. Available: https://www.researchgate.net/publication/321234318_Design_of_an_autonomous_mobile_robot_based_on_ROS. [Accessed: 09- May- 2019].