ROS Based, Simulation and Control of a Wheeled Robot using Gamer's Steering Wheel

Rajesh Kannan Megalingam, Deepak Nagalla, Ravi Kiran Pasumarthi, Vamsi Gontu, Phanindra Kumar Allada,
Department of Electronics and Communication Engineering,
Amrita Vishwa Vidyapeetham,
Amritapuri, India
rajeshkannan@ieee.org, nagalla.deepak@gmail.com, ravikiran.p3637@gmail.com, vamsi.gontu2308@gmail.com, alladaphanindrakumar@gmail.com

***Abstract*—Search and rescue robots to be deployed in disaster prone areas, are trending in the field of humanitarian research. As these robots are expected to maneuver through multiple terrains and avoid profuse obstacles, its behavior has to be reckoned prior to implementation of the robot in real-time scenarios. A basic robot has been created in solid works and simulated in Gazebo of Robot Operating System (ROS). A gamer's steering wheel is used to control the movement and speed of the robot in various schemas. The triumphant integration of robot with Gazebo needs a well-defined robot which describes all the transforms from ROS base_link to all other ROS child_links**.

Keywords—Wheeled robot, steering wheel, Gazebo, URDF, ROS, Simulation, Teleoperation, Nodes, Topics, Rqt_graph, Transforms, frames, closed loop control.

I. INTRODUCTION

Gamer's use steering wheel for many simulation games like Need for speed, Blur, etc. The use of steering instead of conventional joysticks makes the game interactive and interesting. This methodology is also being used in real-time toy car control. The gamer's steering wheel consumes low power and the system can also be deployed in any two-wheeled robot. This interface is compatible with both simulation and hardware using the ROS platform. This is not just plain sailing but also convenient for the controller. Many interfaces are developed to control the robots but with some snags like high power consumption, complex to set up and many more. The gamer's steering wheel based controlling method described in this paper is a plain sailing and exhilarating interface, which is compatible with any robot whose movement is dependent on two drive wheels provided the dimensions of the robot are known.

Robotic Operating System (ROS) with a vast collection of well-defined packages along with a prodigious community has been handy in development of multitudinous robotic systems that are most complicated. Gazebo is one of the ROS compatible simulation plugins where a 3-dimensional model, which exactly resembles the real-time scenario of any robot can be created. Not only the robot but also a world or the terrain in which the robot has to be deployed can be created. With these provisions, the behaviour of the robot can be tested without actually mangling the real robot. To test a robot in Gazebo, either a robot has to be designed in the software or a unified robot description format (URDF) file has to be configured to be launched with Gazebo.

II. MOTIVATION AND PROBLEM STATEMENT

Every day several robots malfunction while undergoing a series of tests. One of the major reasons behind this is, testing the robot directly in real time without a precise estimation of its behavior in a simulation platform. There are situations where the robot behavior is as expected but the control mechanism could be complicated. Faulty controllers might give an erroneous command to the robots which could lead to damage of property and sometimes if the behavior is unexpected, it can be a threat to human life. So, testing the robot in simulated environments with a user-friendly control interface prior to real-time testing is always propounded. Simulation is imitating a real-time system to assess its behavior using a virtual environment.

There are several ROS integrated simulation tools like Gazebo which acts as a tool for 3 - dimensional simulation of robots and robotic arms. For gazebo, ROS provides an interface with the robot for controlling. Although controlling the robot by programming is highly developed, testing robot frequently causes the damage due to non-observance of mechanical or electronic devices. Various strategies can be evaluated on our robot without any harm to the physical system. Advantages of using simulators for robots are that they can be cost-effective, can test any part of the system, determines whether robots meet the required expectation. The steering wheel-based control mechanism is implemented in this work as it stimulates the user-robot interaction and makes the control more realistic.

III. RELATED WORKS

The basic properties which decides the stability of the design and linear feedback control techniques are presented in [1]. Designing and simulating the robot using the solid works and nonlinear dynamics to control the simulated robot are presented in [2]. The algorithm of the simulation control of robots and the behavior of the robot during simulation is presented in the research article [3]. Paper [4] deals with the types of controllers and designing the controllers according to the requirements needed. Working principle of controllers and the new form of feedback to controllers, easy understanding of the control system is presented in [5]. A real-time application of the steering controller and some aspects that a joystick control can perform is presented in [6]. Interface of simulation platform and the ROS, familiarization is presented in the research paper [7]. Designing a unique controller and building the interface between the controller and the robot platform is presented in the paper [8]. Controlling of a robotic platform by designing the controller in the similar miniature form of the robot and designing the controller is presented in [9]. Wireless communication between the master controller and the slave robots, an algorithm for a wheeled robot is presented in [10]. The differential drive motor control is taken from learning robotics using python is presented in [11].

IV. Design and Implementation

Nodes are a part of ROS in which they communicate with the other nodes in order to transfer the data or information. These nodes are generally connected by a bridge called topic. Topic is a unidirectional data channel. The communication between these nodes and topics can be viewed through graphs(Rqt_graph) or frames. Teleop_twist_keboard node is connected to gazebo through cmd_vel which can be seen in Fig.1.

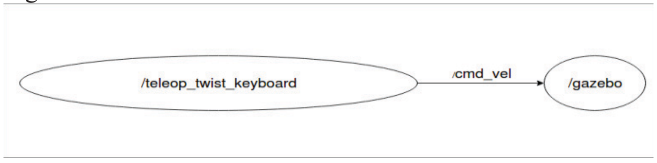

Fig. 1. Connecting keyboard with Gazebo

Simulator consists of launch and configuration files. Virtual robot can be built in Gazebo which produces a xacro file that has all the links connected to parent link. Similar to Gazebo tool there are many other platforms (v-rep, webots, solid works) in which Solid works is most prominent for designing.

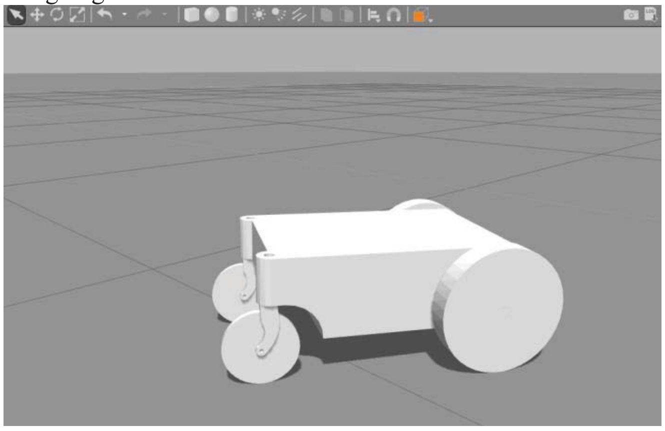

Fig. 2. Gazebo Bot

In Fig.2. shows the designed bot in Solidworks and placed it in Gazebo. Solid works to URDF exporter has been used to generate a URDF file from the solid works model, that includes inertia on the body, the torque on the wheel links, parent links, velocity. For the obtained URDF file adding the differential-drive [11] motor controls the velocity of the wheels independently. Forward kinematics equations are used in differential-drive to solve the problems obtained during the slippery surfaces controlling velocity.

$$V = r\omega \qquad (1)$$

Equation (1) is the basic fundamental equation for this differential-drive. Once this URDF file is added into a ROS package and connected to the master, one can navigate the robot in any environment. When the body is computationally ready with all the connections within the code, the controller launches all the required files using ROS commands. These launch files communicate through the nodes to publish or access the required data. The topic of joint states published by Gazebo node and subscribed to the keyboard node using command velocity(/cmd_vel) topic, can be seen in fig. 1.

A. Steering Wheel Control

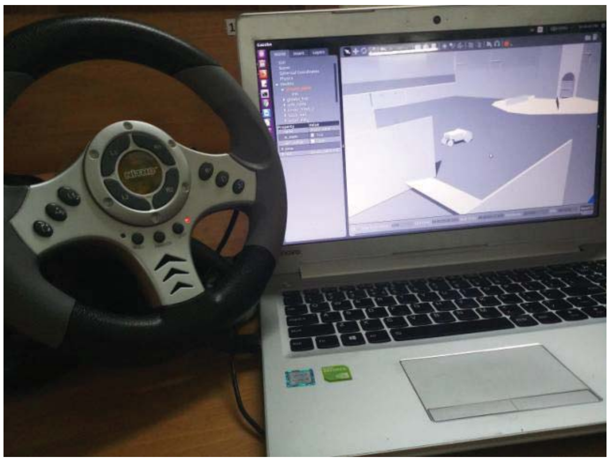

Fig. 3. Steering Wheel

In Fig.3. shows the Nitho Drive Pro steering wheel which is compatible with PC2, PC3 and PC. It helps us to control the wheeled robot using teleoperation. In the steering wheel, the joint_state_values are published in the terminal which is in turn subscribed by the robot, that helps in teleoperation. This robot simulation helps us to get the virtual feedback in Gazebo, so the operator knows what is happening just by looking in the simulation without the need to physically monitor the robot. By using the steering wheel, speed and also the direction can be adjusted using axis values. Different values for buttons ranging from real number 0 to 1 can be assigned. Values of the axes can be changed to turn based on our need. The steering wheel system consists of two different controls for the forward and backward motion: one is pedal motion and the other is by using buttons on the steering wheel.

Nitho Drive Pro does not require much force to rotate its wheel and it can be programmed in such a way that speed can be increased or decreased by pressing the designated buttons. One can also find out which buttons are assigned for what number and its functions. Proper commands can be given to access the control system of the steering wheel and find out the mapping of the steering wheel and be able to change its calibration. In the mapping option of the steering wheel, one can find out which button is assigned to which number and change the controls accordingly. In calibration, default values are assigned to the buttons and axes for movement. Calibration values can be changed manually or by altering the values. Mode of controlling of the axes can also be changed. If the PS/MODE button is on it means that the steering wheel gives the continuous values of the axes. If it is in a STEADY mode it means that the axis is in button state and it behaves just like a regular button and gives a single value. If it is in OFF mode, then the steering wheel is powered-off.

V. Architectural Diagram

Fig. 4. represents the interconnection between different blocks of the system, which play a vital role in the control mechanism and visualization.

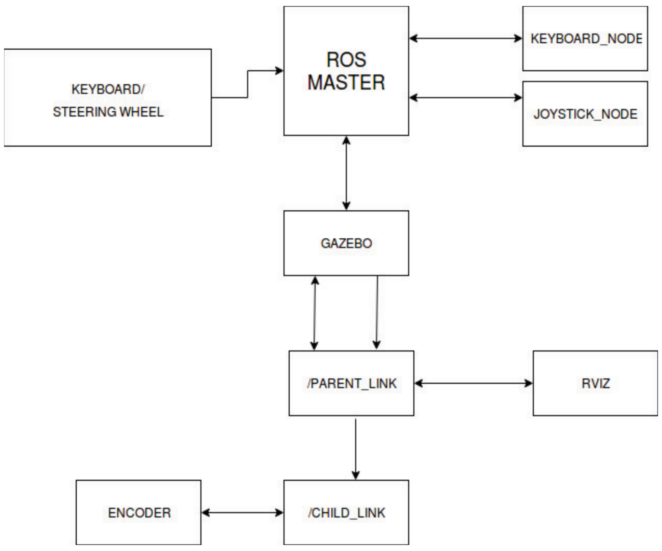

Fig. 4. Architectural Diagram.

A. ROS Master

Every device that is working based on ROS needs a master which sets the environment for using ROS commands. The master acts as a common module for all the nodes and topics. It can be configured to communicate with ROS master in some other system and creates a bridge for the nodes in one system to communicate with the nodes in other. Unless ROS master is available for the host system, no task can be accomplished using ROS.

B. Keyboard/Steering Wheel

The input to ROS master is given from either keyboard or steering wheel, which is connected to the processor through I/O peripherals. Keys like ‘i’, ‘,’, ‘j’, ‘l’ in the keyboard are used for the movement of the robot in forward, backward, right, left respectively. Speed control can also be done using the keyboard. The steering wheel can be used for the same and the control resembles driving a car. These devices should be accessed through either keyboard_node or joystick_node to be integrated with ROS master.

C. Keyboard_node or Joystick_node

These nodes form a bridge between I/O devices and ROS. Each key of these devices has a unique ID in these nodes. So, when a key is pressed, the respective ID gets active and this, in turn, can be assigned with a function to be performed. The values or commands generated in this function are published over a topic which can be accessed by the Gazebo node, where the robot is being simulated in different environments.

D. Gazebo

Gazebo_node acts as both subscriber and publisher. It subscribes to the commands published by keyboard or joystick nodes and to the transforms published by URDF node. This data is processed to estimate the joint states and fake the movement of the robot. This data is used by RViz to mimic the robot in Gazebo. The joint states of the robot are published over Odom topic which is used by the differential drive plugin of the gazebo to avoid deviation from the actual path.

E. RViz

RViz is a visualization tool which helps to analyze different aspects of the robot-like transforms. Transforms play a dominant role in avoiding collisions and in establishing coordination amongst the different frames and regulating the movement of all the joints. Along with the transforms of physical systems the odometry frame can also be calculated. This is not a simulation tool but a visualization interface where one can visualize both simulated robots and a real-time robot.

F. Parent_link/Parent_frame

Parent frame is the reference frame for all the transformations of the robot. This frame has to be set at the point where the entire translation and rotation of the robot is concentrated. Unless this frame is precise, the robot will misbehave during simulation if the control is a closed loop. The movement of each child frame is calculated with respect to the parent frame. Also, the position of each frame of the robot is estimated with respect to this as a frame of reference.

G. Child_link/Child_frame

Each child frame reflects one of the many joints that have to undergo transformations for movement of the robot. These frames are connected to the parent frame and publishes all the data into the reference frame. The coordination between each frame is vital for the robot to behave as expected. The data from these frames help in locating the robot in the environment or world.

H. Encoders

The encoder is an electromechanical device through which the distance traveled by the robot can be estimated in real-time. But this data is published over Odom topic by Gazebo plugin while simulating the robot by faking the joint calibrations. This data acts as feedback for this closed loop control system.

VI. Experiments And Results

A directory for the simulation of the robot in gazebo was created initially. Implementing the differential drive in the URDF file helps in controlling the robot in Gazebo. After successful control of the robot with the keyboard, a directory to help us control the robot using joystick was created. Using the Quantum joystick to control did not provide the live interaction needed to control a robot. So using of Nitho Drive Pro steering wheel proved to be helpful for the handicapped people because it does not require much force to rotate the steering wheel. Simulation of the robot in Gazebo is done by testing the code using the keyboard to move the robot in any of the four directions. It was successful while controlling the robot using the steering wheel as well. As it was a success of controlling the robot in gazebo one can create our own URDF of a new robot and added the differential drivers to it. Later it was programmed in such a way that one can increase or decrease the speed by pressing the designated buttons. TABLE I describes the frequencies at which data is being published on different topics in Hz and from the results obtained, it can be inferred that the ROS system is stable irrespective of the terrain the test is conducted.

TABLE I. Frequencies at Which Different Topics are Published in Different Terrains

S.NO	Topic		Test			
		Test-1	*Test-2*	*Test-3*	*Test-4*	*Test-5*
1.	Joy	24.14	24.53	24.2	24.09	24.29
2.	cmd_vel	24.94	24.92	24.4	24.26	24.42
3.	joint_states	9.986	10.0	10.0	10.0	9.99
4.	tf	55.0	54.91	55.01	54.88	54.89

The robot was also tested on different terrains to analyze compatibility in different conditions. Some of the conditions are explained below.

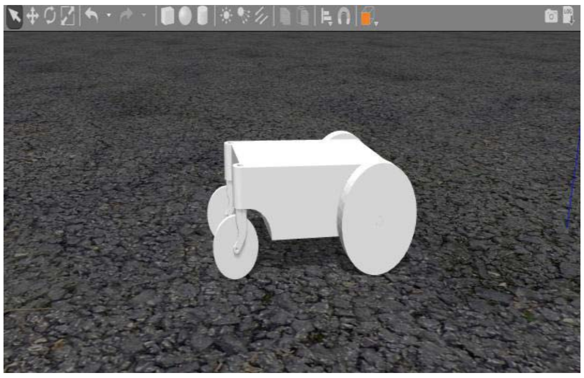

Fig. 5. Asphalt Plane

First, the platform which was chosen is an Asphalt plane because it is deployed in many industries and airports. Our robot was tested on it to find out the compatibility and movement of the robot. Depending on the speed of the robot the body movement takes place. When given the same speed as when giving in the ground plane as shown in Fig.5, the robot does not move due to friction. But when the speed is increased, the body started to move slowly. So, one can conclude that more speed is required to move the body in Asphalt plane than the ground plane.

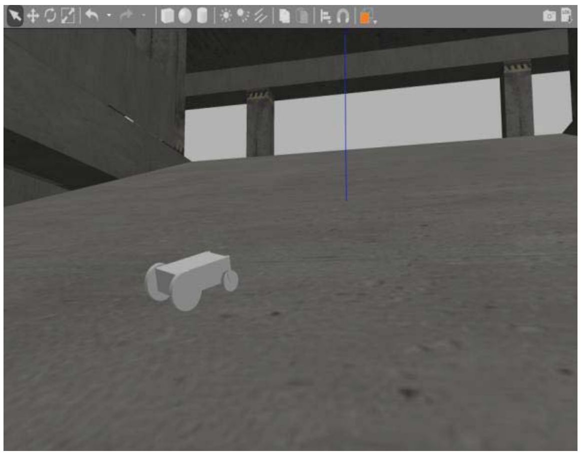

Fig. 6. Slope

Next, the chosen platform was a slope as it is used in daily life like parking garages as shown in the Fig.6. While testing on the slope more speed was required than in the asphalt plane to move the body as it has to carry the weight of the body also. But when the increased speed is more than a certain value the body started to topple which is due to the torque on the center of mass of the body. But the range of the speed required to move the body on the slope without toppling was determined through multiple tests. If one measures the speed on a scale of 0-100 the range that is required to move on a slope is 55-70.

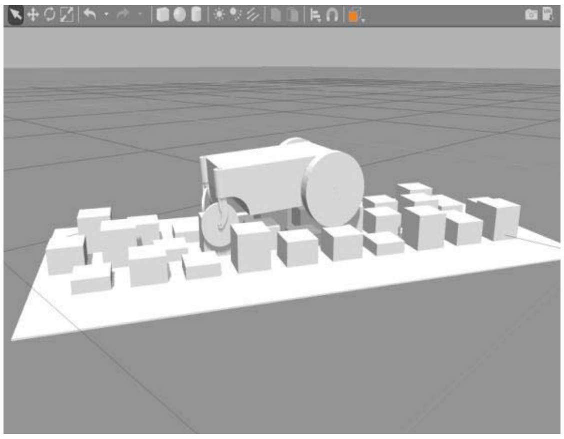

Fig. 7. Rough Terrain

After testing on the Asphalt plane and the slope which are smooth it was tested on a rough plane as there is no such surface as smooth in the real life. Wanting to take a worst-case scenario of the surface so that the body can handle any type of situations, the above Fig.7. and Fig.8. are the rough terrains which were considered for this test. While operating the body in the rough terrain speed higher than in any other terrain was used because if the robot goes slowly it could get stuck in any of the holes. Even if it gets stuck one has to assess the terrain and control accordingly. Rather than speed, controlling of the body played a key role in the movement of the robot.

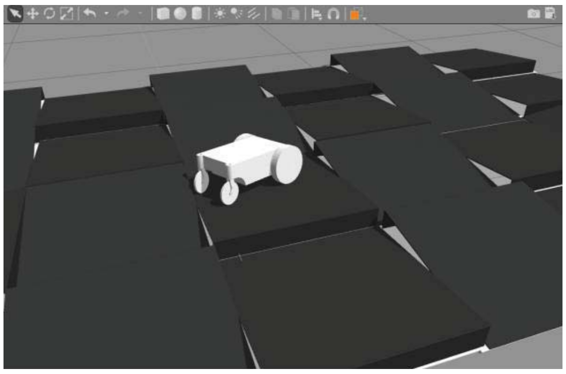

Fig. 8. Ramps

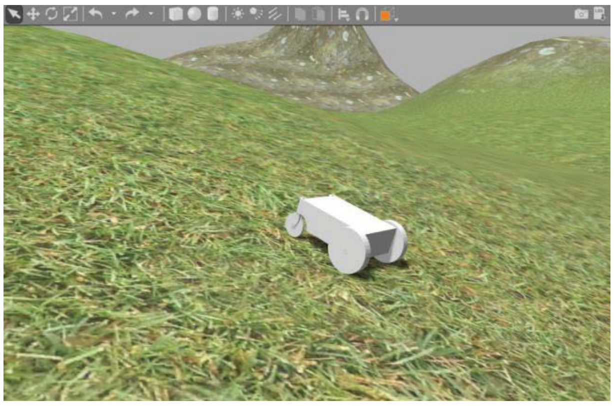

Fig. 9. Driving Terrain

After testing all the smooth and rough terrains the robot was tested in a real-time scenario. So, considering the case of a mountain terrain which is slippery and bumpy made it perfect for a real-time scenario. While testing our robot in this terrain the robot wheels use to rotate at their own position without moving forward and if high speed is applied the body suddenly jumped and toppled. To avoid this situation, a medium speed is applied to control the body. While applying the medium speed the body goes perfectly in a straight line but if one wants to rotate the body it would slip sideways. To avoid this, a low speed is used and rotating the body worked perfectly. In this situation speed and control of the body played a key role in the movement of the body.

VII. FUTURE IMPLEMENTATION

The simulation software is used to estimate the real-time behavior of the robot in varied terrains or environments. As all the transforms are set and connected efficiently, the behavior of robot is intact with the expectations. This, in turn, elevates the fact that the behavior of a real-time robot could be as expected. So, with these stable results and performance of the robot in simulated arenas, we are looking forward to implementing the same ROS system in a real-time robot which is robust but stable. We are also expecting to simulate the multi-terrain robot with tracked wheels and flipper mechanism, which will be more efficient in multiple terrains.

VIII. CONCLUSION

In this work, a four-wheeled robot whose movement depends on two drive wheels has been simulated and tested in different terrains under different conditions like variations in speed, friction and various other conditions using gamer's steering wheel. The deviations from the actual path have been resolved by using differential drive plugin of Gazebo. Also, a communication has been established between the simulated robot and the I/O devices through this plugin. The robot is rigid and stable even when clambering over an uneven terrain which has various obstacles. Hence a successful simulated testing of the robot has been achieved through this work.

ACKNOWLEDGEMENT

It is gladdening to thank Humanitarian Technology Labs and the Department of Electronics and Communication Engineering, Amrita Vishwa Vidyapeetham, Kollam who provided us with a highly encouraging work environment and lab facilities, which was backbone for completion of this project.

REFERENCES

[1] El-Dessoky, M.M, Alzahrani, E.O, Almohammadi, N.A, "Chaos control and function projective synchronization of novel chaotic dynamical system" ,Journal of Computational Analysis and Applications 27(1), pp. 162-172, 2018.

[2] Yousuf, L.S.Email Author,"Experimental and simulation investigation of nonlinear dynamic behavior of a polydyne cam and roller follower mechanism",Mechanical Systems and Signal Processing 116, pp. 293-309, 2018.

[3] Altarazi, S.A. Ammouri, M.M.,"Concurrent manual-order-picking warehouse design: a simulation-based design of experiments approach",International Journal of Production Research pp. 1-19, 2017.

[4] Nobutake Hiraoka, Katsuhiko Inagaki, "A study of a new controller interface for omnidirectional robots", 10th Asian Control Conference (ASCC), 2015.

[5] Ashley L. Guinan;Nathaniel A. Caswell;Frank A. Drews, "William R. ProvancherA video game controller with skin stretch haptic feedback", 2013 IEEE International Conference on Consumer Electronics (ICCE), 2013.

[6] Masayoshi Wada, "A Joystick Steering Control System with Variable Sensitivity For Stable High Speed Driving" , IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society, 2013.

[7] Xudong Ma Fang Fang;Kun Qian;Can Liang, "Networked robot systems for indoor service enhanced via ROS middleware", 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2018.

[8]]Rajesh Kannan Megalingam;Sarath Sreekanth; Govardhan;Chinta Ravi Teja;Akhil Raj, "Wireless gesture controlled wheelchair", 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS), 2017.

[9] Rajesh Kannan Megalingam;Sricharan Boddupalli;K. G. S. Apuroop, "Robotic arm control through mimicking of miniature robotic arm", 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS), 2017.

[10] Rajesh Kannan Megalingam;Deepak Nagalla;Pasumarthi Ravi Kiran;Ravi Teja Geesala;Katta Nigam, "Swarm based autonomous landmine detecting robots", 2017 International Conference on Inventive Computing and Informatics (ICICI), 2017.

[11] Lentin Joseph, "learning Robotics using python", Packt Publishing, 2015.