**Maze Solving Robot Using ROS**

1. Introduction

Robotic field has caught a big attention to our world as it conveniently increases efficiency of especially monotonous work. This field does include the industrial robots and mobile robots that is used for various purposes. As the fields become open for everyone, a lot of competition such as MicroMouse competition started to build up as to making the best robot. In the MicroMouse competition, it normally uses 256 square units of maze. Micromice or the robot will compete to solve the maze without the needs of any manual assistance. In order to do so, suitable algorithm should be use to solve the maze in the least time possible.

1. Overview

In the Industry, there are certain places where the robot can collect the processed objects and place those objects in the warehouse. So here in this object collection process, the robot needs to enter into the machine processing area and it should come out of the machine processed area. Entering the robot to the machine area essentially like a room entrance and coming out of the robot from the machine area is also called a room exit.

So here in this project, you are going to develop a robot application that should enter the room in the prescribed entrance place and also identify the distances of the objects/walls in the robot entering the path and come out of the room by measuring the distances with the prescribed exit place.

For finding distances we generally use Lidars. By integrating the Hokuyo Laser Distance finder into the robot you will be able to find the distance of objects or walls. Here you need to develop a python algorithm to enter and exit the room on the Gazebo simulator.

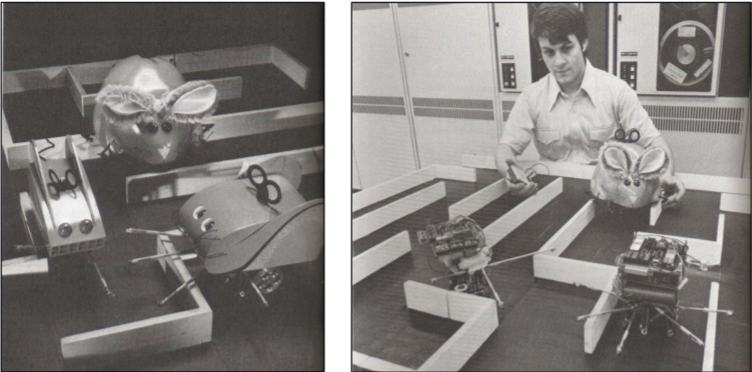
1. Purpose

In the real world, a robot does not know in advance what the map looks like. Splitting apart the map and the robot allows you to work with the notion of the robot having to figure out its surroundings without being able to "cheat" by asking the map where it is and what's around the corner. This is what you'd expect in both real robot software and robot software running in a simulated world. Put another way, robots need to build a model of the world as they go. Think of this as a really gentle, simple introduction. For serious robots, you use SLAM. (Yes, some self-driving cars use high definition maps to help. But still someone has to build the maps, which amounts to building a model of the world, and at run-time the cars can only use the maps as strong hints, since the world might have changed since the last map update.)

1. Literature Survey

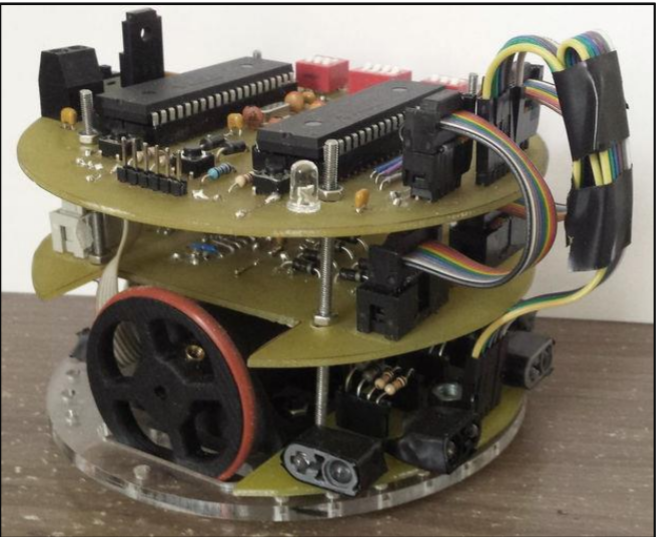
In the middle of the 20th century, Maze solving problems become an important field of robotics [1]. In the year of 1972, editors of IEEE Spectrum magazine came up with the concept of micromouse which is a small microprocessor controlled vehicle with self-intelligence and capability to navigate a critical maze [2][3]. Then in May 1977, the fast US Micro mouse contest, called “Amazing Micromouse Maze Contest” was announced by IEEE Spectrum. From then, this type of contest became more popular, and many type of maze solving robots are developed every year.

Late 1970s the designs of the maze solving robots designs were used to have huge physical shapes that contain many blocks logic gates. Figure show the example of early the maze solving robots (micro mouse). Due to technological development the physical size of the robot becomes smaller and the features of the robot becomes modern.



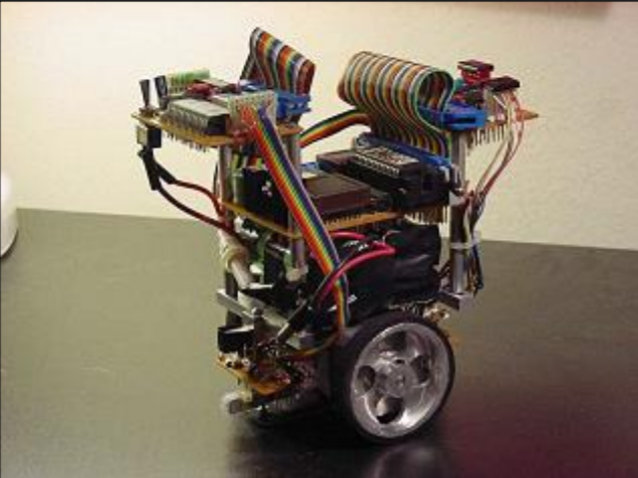
**Micromouse using Wall Following Algorithm by University of East London, 1999 [4].**

In the year 1999, Michael Gims, Sonja Lenz and Dirk Becker from University of East London developed a micro mouse. They used a non-graph theory algorithm, Wall Following Algorithm. But their robot did not move intelligent in the map and it could not solve maze with loop.

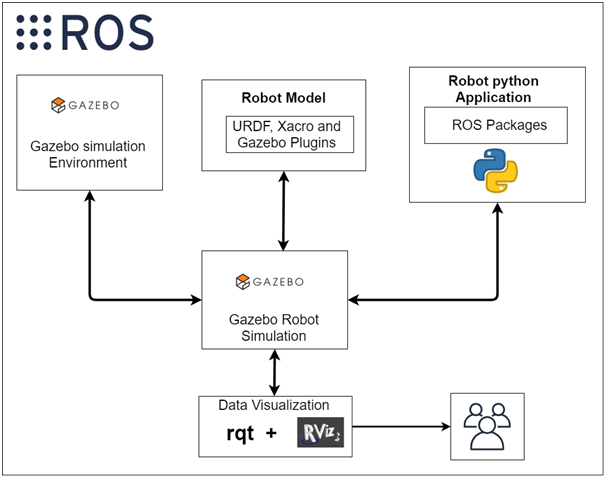


**Micromouse Maze Solving Robot by Chang Yuen Chung, UTM. 2008 [5]**

Chang Yuen Chung, a student of Universiti Teknologi Malaysia (UTM) designed a micromouse using Flood Fill Algorithm. His robot was designed in three layers so that the robot looks more compact and smaller size. But it was very hard to troubleshoot if there is circuit faulty. Flood Fill Algorithm is one of the graph theory mazes solving algorithm. In paper [8], Chang Yuen Chung claimed that this Flood Fill Algorithm is able to find the shortest path but more memory is required for execution.



1. Theoretical Analysis
2. Block diagram



1. Hardware/Software designing

Virtual Box - Oracle VM **VirtualBox** is cross-platform virtualization software. It allows users to extend their existing computer to run multiple operating systems including Microsoft Windows, Mac OS X, Linux, and Oracle Solaris, at the same time.

Ubuntu 18.04 - Ubuntu is a complete Linux operating system, freely available with both community and professional support.

Robot Operating System (ROS) - It is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex

and robust robot behaviour across a wide variety of robotic platforms. ROS is a framework that sits on top of an existing operating system such as GNU/Linux. Packages are provided for Ubuntu Linux to help get your robot up and rolling. We are using this framework to create an automation gmapping and navigation system.

Python - It is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation.

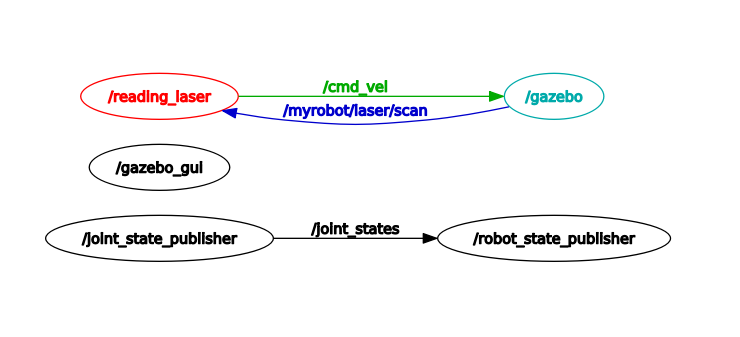
We are choosing python because it:

1. Readily available and Open source
2. Huge Libraries
3. Easy to Understand and learn
4. Big developer Communities
5. Fewer code lines and larger Functionalities

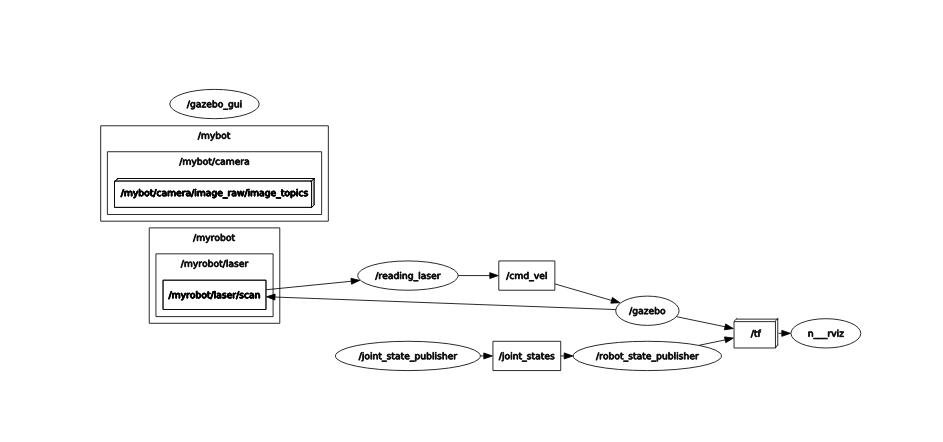
Gazebo - Gazebo offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. With the help of this we are able to simulate our robot in a virtual world.

1. Experimental Investigations

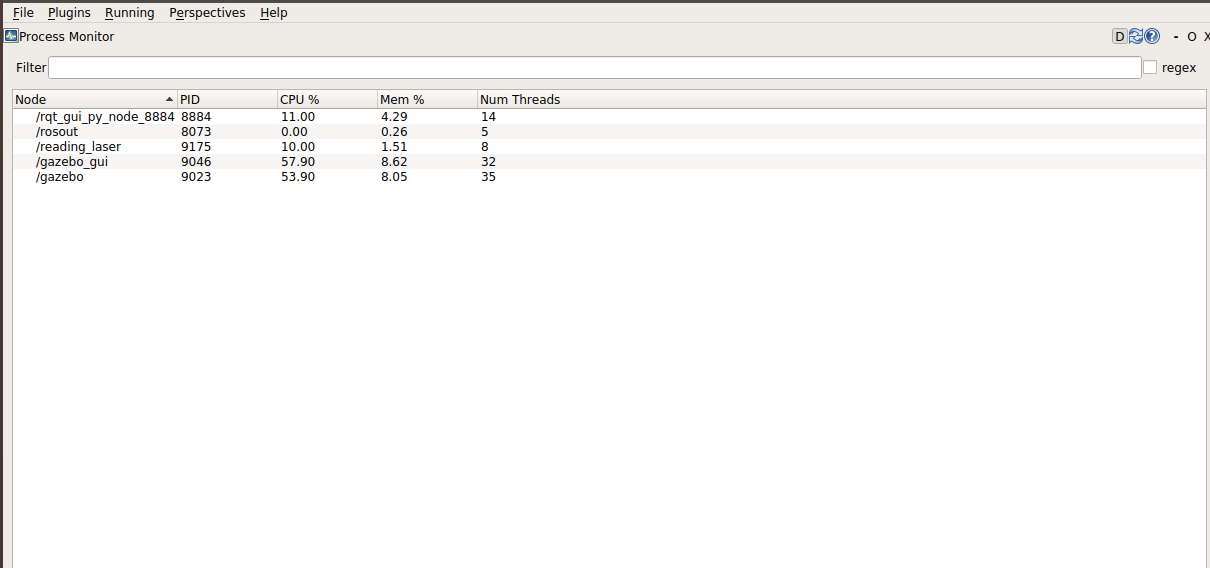
**Node Graph**



**Active Node Graph**

****

**Process Monitoring**



1. Flowchart



1. Result

We see that the robot moves autonomously by considering the laser scan values inside the Gazebo simulator. Users can visualize a robot model moving in the Gazebo environment in ROS.

1. Advantages & Disadvantages

Advantage: The maze solving robot is designed to find a path without any assistance or help. As a type of autonomous **robot**, it has to decode the path on its own to solve the maze successfully.

Disadvantage: If any thing is there in the path way then it will stop there.

1. Applications

Mobile robots can be implemented for home automation such as autonomous vacuum cleaners which need to navigate itself in the house and clean it at the same time. For most mobile robot applications, path planning is an important task. Any structure of multiple paths can be called a maze. A path between two specific locations existed in the

maze should be well planned and the planned path should be

collision-free and the shortest one. Applications of maze

solving systems include intelligent traffic control that helps

ambulances, fire fighters, or rescuing robots to find accurately

their shortest path to their destination.

1. Conclusion

As a conclusion, to complete the maze-solving robot project we requires knowledge on ROS, Gazebo Simulator, Basic Linux Commands, Python programming and XML programming, Basic Nomenclator. The tuning of both sensor and motor are very important as it determines how smooth the robot can explore inside the maze. Therefore, the motor should be made zero-error free and the ultrasonic should be tested and then the codes should be adjusted according to the range of detection of the sensor.

1. Future Scope
   1. In future the robot can be upgraded to include a better algorithm as well as memory so it can try remember the last maze it solved.
   2. The size of robot can be reduced.
   3. The speed of robot can be increased.
2. Bibliogrophy

[1] R. Zheng, R. Hill and M. Gardener, Engaging older adults with modern technology,

1st ed. Hershey, PA: Information Science Reference, 2013.

[2] "Maze Solvers Archives - cyberneticzoo.com", cyberneticzoo.com, 2017. [Online].

Available: http://cyberneticzoo.com/category/mazesolvers/. [Accessed: 12- Apr2017].

[3] "Spectral lines: Announcing the Amazing Micro-Mouse Maze Contest - IEEE Xplore

Document", Ieeexplore.ieee.org, 2017. [Online]. Available:

http://ieeexplore.ieee.org/document/6367601/. [Accessed: 12- Apr- 2017].

[4] MICHAEL GIMS, S. L. D. B. 1999. Micromouse-Microprocessor Controlled Vehicle. Bachelor of Engineering, University of East London.

[5] CHANG, Y. C. 2009. Micromouse Maze Solving Robot. Bachelor of Engineering, Universiti Teknologi Malaysia.

1. Appendix
   * + 1. Source code

<https://github.com/smartinternz02/SI-GuidedProject-4797-1627035365>

* + - 1. Robot output Screenshot

