ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 5, Issue 9, September 2016

Path Finding and Turning with Maze Solving Robot

Yee Mon Nyein, Nu Nu Win

***Abstract*— A maze solving robot is a self-contained fully autonomous mobile robot, it can capable of transporting itself to the target of an unknown arbitrary two-dimensional maze. The robot must find its way to the target of the maze simply to return again and again in increasingly more courageous attempts in order to certain the fastest run time. The individual components of maze solving robot system consist of the motor control system, navigation sensor array, and a mapping system or algorithm for navigating the maze intelligently. The aim of this research is to develop and implement a maze solving robot and find the possible path from the starting point to the target by reducing the collision. This system can also reduce the collision errors which are caused by unbalancing the speed of the two motors. The main objective is to build fully maze solving robot systems with complex environment. There are a number of techniques which have been used for solving the maze by robot. In this research, flood fill algorithm is used as path finding method to reach the target of the maze.**

***Index Terms*— Collision Avoidance, Flood Fill Algorithm, Maze Solving Robot, Path finding**

I. Introduction

Maze Solving Robot also called micromouse is a robot designed to get to the target of the maze, unaided. The robot essentially comprises of a drive motor, steering and turning method to move the robot. It has sensors to detect the wall and control logic to control the activity of the robot and find the possible path. The robot is powered by a battery and has to find its way from a predetermined starting point to the target of the maze unaided. The shorter the time period it takes, the better the decision making algorithm. The robot has to keep consider of where it is and detect once it reaches the finishing point. The mouse will need to keep consider of where it is and detect when it has reached the goal. Autonomous robots have wide reaching applications, as reviewed in, from Bomb sniffing to finding humans in wreckage to home automation. It is possible that a Micromouse can save lives and can be argued that not only saving a life of human but also saving the world. This is why we have contributed to researches and advanced robotic studies. The micromouse objective is to find the target of a 6 by 6 cell maze for some time; the micromouse will attempt to make its fastest run from the starting point to the destination cells.

Early Micromouse was far less technologically and electronically advanced compared to those today. Moreover, Several of the other mice at the competition did not include microprocessors in their design, instead opting to use simple IC logic. In contrast, today's Micromouse is extremely evolved, electronically refined robots. Current microprocessor technologies allow the mice to perform computations not conceivable thirty two years ago. As a result, the robots can be programmed to use more sophisticated algorithms to find the target of the maze. Motors, sensors, integrated circuits, and other components have greatly improved features to assist the robot designer. Overall, micromouses are now smaller, faster, and smarter than their earlier counterparts. Design and construction of a Micromouse requires a broad range of engineering skills such as electronics design, mechanical design, program design and how the student approach complex engineering problems. This combined with an open design process makes the Micromouse research a very practical and challenging design research.

The Micromouse project is to build the autonomous robot that will perform maze solving algorithm and performs smooth movement while situated in the maze. The scope of the project will be producing a robot with good interactive between the microcontroller with the mechanical elements, signal controlling and software efficiency. The Micromouse will gain information from the working environment which means it will memorize the maze information from the wall it encounters on each cell it visited. This autonomous robot will perform the maze. Maze solving algorithm to the target of the Maze and the size of the Maze will be 36 cells with 30cm x 30cm each. It will also avoid colliding with the maze walls by automatic balancing system and performing all the basic smooth movement which are straight, clockwise or anticlockwise 90 degree turning and also 180 degree turning.

II. Hardware Design of Maze Solving Robot

In this research, the two-wheel differential drive maze solving robot is composed of two 6V DC motors with each optical encoder. It was shown in Figure 1. In the product consists of 1 robot chassis with length 12.5cm and 13 cm width, 2 wheels with a diameter of 66 mm, 1 piece ball caster and 2 6V DC motors which have been furnished by the gearbox as well as two pieces of the DC motor bracket to pair on the chassis.

In this maze solving robot had 2 pieces rotary encoder. Rotary encoder attached to the DC motor to calculate the rotation of the wheel. The whole hardware system of this mobile robot can be seen in the block diagram. Mobile robot

***Yee Mon Nyein**, Mechatronics Department, Mandalay Technological University, Mandalay ,Myanmar*
***Nu Nu Win,** His Mechatronics Department, Mandalay Technological University,Mandalay, Myanmar*

used three Ultrasonic sensors to detect maze wall at right, left and front position. Driver L298 module is controlled the direction of rotation and speed of a DC motor .Rotary encoder is used to calculate the rotation of the right and left wheels. Push button was used to instruct the robot to start. The system output would drive two DC motors that served as actuators to move the right and left wheels, so that the robot can move forward, turn to the right, turned to the left, and rotates reverse. Arduino Mega microcontroller serves to process the signal input from the three ultrasonic distance sensors, perform processing algorithms that is solved the possible path, and generates output signals to control a robot.

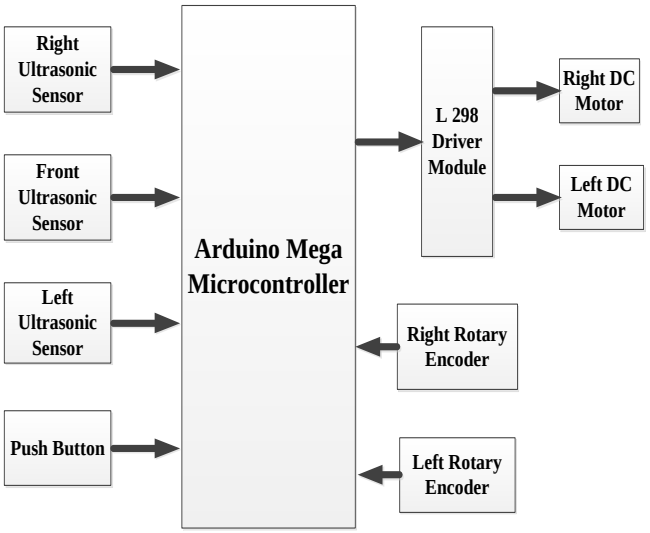

Fig 1.Operation Block Diagram of the Maze Solving Robot

A. Maze Solving Design Approach

In our design, it is important to focus our attention to find ways in solving the maze to generate our own algorithms in combination with other types of algorithms that may be used for programming of our microcontroller. We have generated a method in solving a maze that combines 3 ultrasonic sensors in trying all possible paths through the maze in order to find the destination. When does the robot decide to turn? How does the robot know what direction to turn? And how can we recognize the paths that are dead-end and omit them after the first round of Navigation?

Initially our robot is moving centered between the two side walls. The left, right distance sensors detect these walls as the mouse is navigating forward as shown in fig 2. These sensors on the left and right also control the mouse to navigate through the center of the maze.

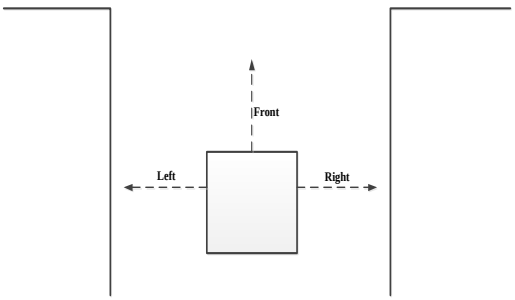

Fig 2.Straight Forward Movement of the robot

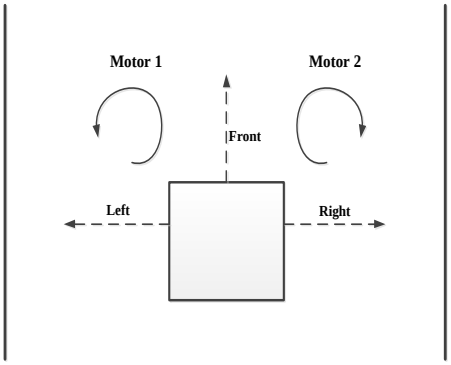

Fig 3.Two Motor with Forward Movement

At the straight forward movement, the two DC motors drive same forward direction with the same velocity. To move the robot straight stably between two walls, the distance from the right sensor and left sensor need not to exceed more than 8 cm. If the left sensor exceed more than 8 cm from the left ultrasonic sensor, the robot will move to the right to get stable move within the two walls. Similarly, if the right sensor exceeds more than 8 cm from the right ultrasonic sensor, the robot will move to the left to get stable move within the two walls.

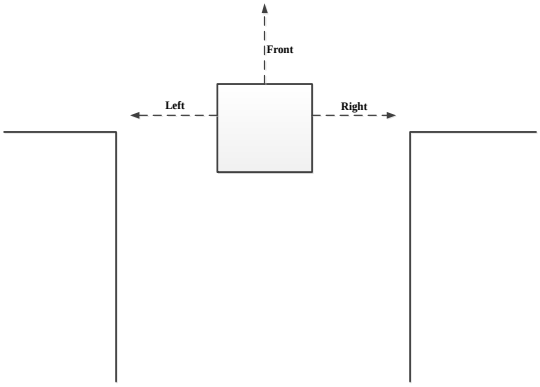

Fig 4.Robot reaches the Intersection Point

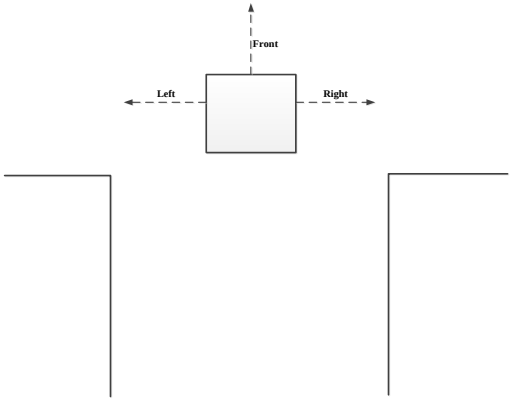

Fig 5.Decision Making Step to turn for Robot

The next step is that the mouse will navigate forward until it reaches the intersection or the position at the left sensor and right sensors cannot detect any walls as shown in fig 4. When this happens, the left and right sensors will send a signal to the microcontroller telling the microcontroller to reduce the speed of the motors through the process of Pulse Width Modulation.

As our robot navigates forward, it will eventually reach a point where all the two side sensors (the left and right) does not detect any walls as shown in the fig 5. At this point, the sensors send a signal to the microcontroller to reduce the speed of the robot to 0 and begin the turning process at 90 degree angle. Notice that the programming we are doing does not stop the robot immediately, rather it will reduce the speed to 0 and turns 90 degrees toward the right or right according to the decision of path finding algorithm.

If the microcontroller decided to turn the left for the robot based on the algorithm, the motor at the left is moved reverse movement and the motor at the right moves forward movement. That movement makes the robot to left direction turns as discuss in fig 6.

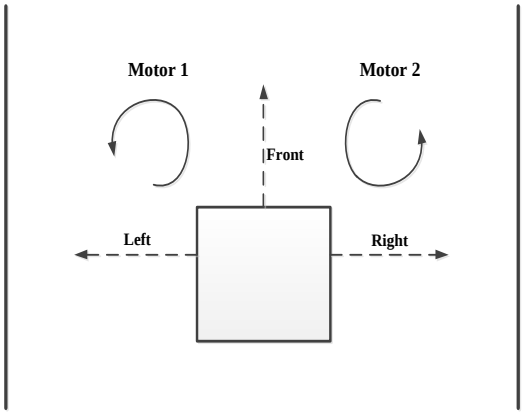

Fig 6.Robot Turning Left

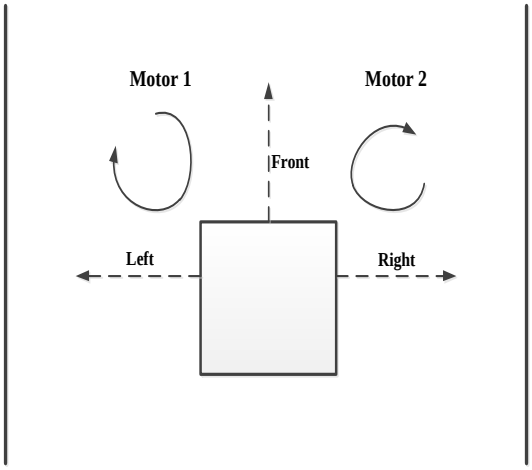

Fig 7.Robot Turning Right

Similarly, the microcontroller is decided to turn the right for the robot based on the algorithm, the left motor drive forward direction movement and reverse direction movement is driven at the right motor of the robot. Therefore, the maze solving robot turns to right direction.

The robot may navigate through a route that is dead-end. In this case, all three side sensors, the left and right sensors as well as the front sensor detects wall. When this scenario occurs, there will be a robot will stop and brakes before hitting the wall in the range of sensor detection we have specified in our design. Then, the robot stops, it will make a 180 degree turn to turn around. The robot is capable of making the 180 degree turns easily. At that condition, the two motor move forward and reverse direction.

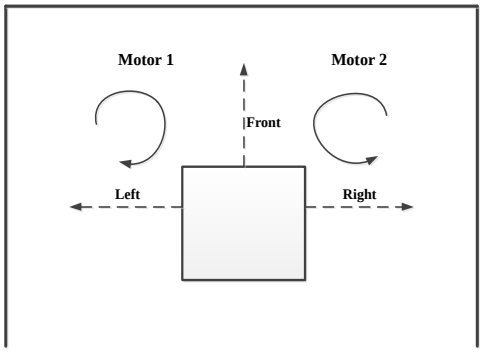

Fig 8.Robot turning back

B. Turning Characteristics of wheel

By changing the speeds of the two drive wheels independently each of the required manoeuvres can be performed. When both motors are rotating at the same speed the robot will theoretically travel in a straight line. The relationship between the two distances travelled by each of the wheels during turning is shown in fig 9. Using this method a simple mathematical formula can be employed to calculate the ratio of the outside wheel velocity to the inside wheel velocity using the ratio of the distance travelled by each of the wheels.

$$Ratio = \frac{2\pi(R+W)}{2\pi R} = \frac{R+W}{R} \quad (1)$$

As the circumference of each wheel is theoretically identical, this ratio corresponds direly to the velocity of rotation for each of the wheels and motors. The greater the ratio, the sharper the turning curve performed by the micromouse. Using this ratio, precise turning control of the robot can be achieved by altering the velocities of the motors driving the wheels.

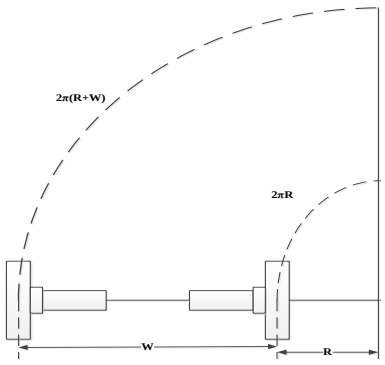

Fig 9.Turning Characteristics of wheel

C. Revolution for Grid in the Maze

The electric equivalent circuit of the armature and the free–body diagram of the rotor are shown in the Fig 10 below. The input of the system is voltage source (*V)* that applied to the motor's armature, while the output is the angular velocity of the shaft ω. Then rotor and shaft are to be rigid. The friction torque T is directly proportional to armature current i.

$$T \propto i \quad (2)$$

The torque on the rotor is proportional to the armature current i(t) and can be expressed as

So, $T = K_t i(t)$

K (t) is a constant of proportionality called the torque constant. It should be noted that the electric constant and the motor constant. The back emf, v_e is directly proportional to the angular velocity of the shaft by a constant factor K_e

$$v_e = K_e \dot{\theta} \quad (3)$$

The following equations can be derived based on Newton's 2*nd* law and Kirchhoff's voltage law.

$$J\ddot{\theta} + B\dot{\theta} = K_t i \quad (4)$$

$$L\frac{di}{dt} + Ri = v - K_e\dot{\theta} \quad (5)$$

$$P(s) = \frac{\dot{\theta}(s)}{v(s)} = \frac{\omega(s)}{v(s)} = \frac{K_t}{(Js+B)(Ls+R)+K_tK_e} \quad (6)$$

Where, P (*s*) is the transfer function of DC motor.

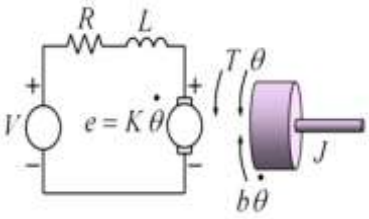

Fig 10.DC motor armature

The voltage induced in the motor armature can be calculated for any rotational speed by multiplying the K_e (Voltage Constant) and the rotational speed of the armature.

$$v_e = K_e\omega = 0.03 \times \omega \quad (7)$$

The voltage drop across the armature resistance can be calculated using Ohm's law.

$$v_{drop} = I.R_{armature} = 200mA \times 1.617\Omega = 0.3234V \quad (8)$$

The battery voltage is a variable that can be determined by user. In this case, select to calculate the rotational speed of the motor armature at 6 volts. The battery voltages $V_{battery}$ equal to voltage induced in the armature plus the back emf voltage induced in the winding.

$$v_{battery} = v_e + v_{drop} = 0.03 \times \omega + 0.3234 \quad (9)$$

$$\omega_{motorshaft} = 189.22\, rad/\sec \quad (10)$$

The transmission (reduction) ratio is 1:48. Calculate the output shaft speed by dividing the armature speed by the transmission ratio.

$$\omega_{outputshaft} = \frac{\omega_{motorshaft}}{48} = 3.9\, rad/\sec \quad (11)$$

$$v = \omega \times r = 0.13\, m/\sec \quad (12)$$

Therefore, velocity of the robot is come out. The robot moves 0.13 m at one second. After that, Calculate how many distance travel by the robot in one pulse.

Distance traveled per pulse=wheel circumference/number of pulse per revolution

Circumference = Diameter x π = 18.84 cm

Distance traveled per pulse=18.84/8=2.355 cm/pulse

Revolutions = Distance/ Circumference=1.6 revolution / grid

The robot moves 1.6 revolutions for a grid and 12 pulses is also represent a grid is passes through by the robot.

III. The Maze and the Robot

The maze designed for the robot to solve is of the size of 6 x 6 cells as shown in fig 11. The maze was designed so that it will have two paths in order for it to be solved. One of the paths is longer than the other. The robot must decide which one of the paths is suitable and solve the maze through that path.

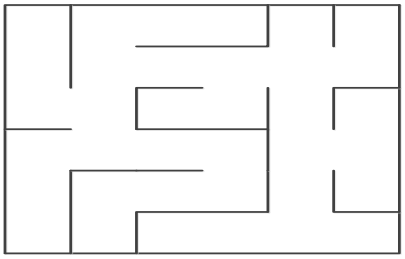

Fig 11.Maze Design

IV. Algorithm

Choosing an algorithm for the maze robot is critical in solving the maze. In this research, flood-fill algorithm was chosen to solve the maze due to its balance in efficiency and complexity. There are four main steps in the algorithm: Mapping, Flooding Updating and Turning; which are described in the following sub-sections.

A. Mapping the maze

For the robot to be able to solve the maze, it has to know how big the maze is and virtually divides them into certain number of cells that can be used later in calculating the possible path to the destination. In this research, a maze of 6 x 6 cells is used. Between two cells there can be a wall. Thus, in a row of six cells, there are five walls in between them. In total, in a row, there are eleven units of cells or walls. This information is stored in an 11 x 11 array. The white units are the cells which the robot can be placed inside. The orange units are the locations for potential walls. The black units indicate wall intersections which are ignored by the algorithm. The external borders of the maze are also ignored as they are fixed boundaries of the maze. Both cells (white) and walls (orange) are set to zero in as their initial conditions.

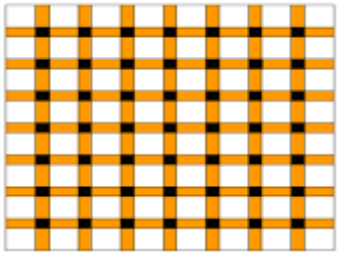

Fig 12.Maze-mapping Array

B. Updating the wall data

Before the robot decides where it wants to move to, it has to check if it is surrounded by any walls in any of the three

directions: right, left and front. The robot reads the distance of any obstacle at each direction and check if the distance in each is more than 8 cm. The ones that exceed 8 cm are updated as "wall" on their respective side. For the robot to update the correct wall data, it has to know first which direction it is facing. There are four orientations for the robot to be facing: north, south, east or west. Initial orientation was set at start and the robot keeps tracking of any changes.

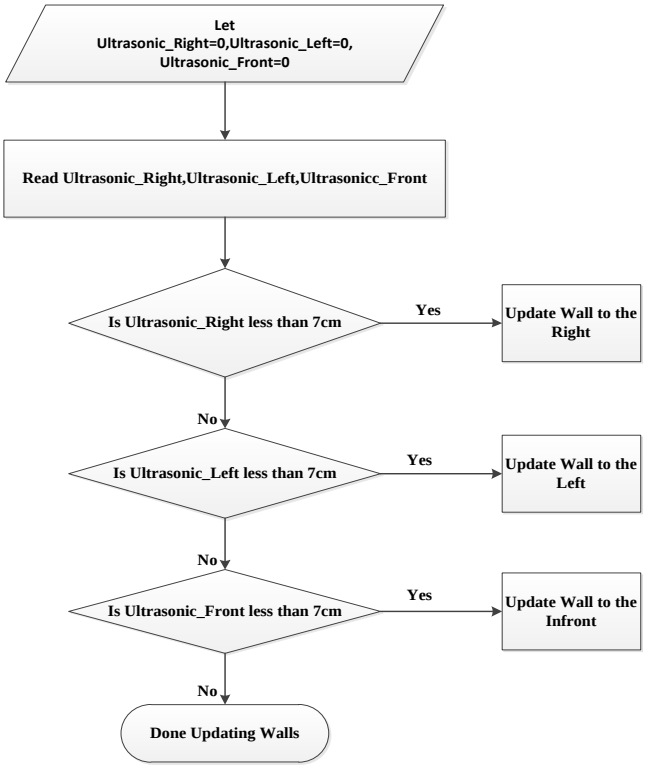

Fig 13.Flowchart for updating wall location at each cell

C. Checking for Turns

After the robot has decided which direction it will go next; it returns the amount of degrees it needs to turn in order to go to the cell intended. After turning the robot, the algorithm updates the new orientation of the robot, i.e. facing north, south, east or west.

TABLE 1.VALUES RETURNED BY THE ALGORITHM AND ITS RESPECTIVE TURN

Degrees	Turn needed
0	No Turn
90	Turn Right
180	Turn Back
270	Turn Left

V. TEST AND RESULT

Wall map data will be updated when the robot go to cells that have not been visited before. Flood fill algorithm will update the value of the cell based on the position of the wall that has been mapped out by the robot. Robots always perform movement to neighboring cells which have the smallest value. If there is more than one neighboring cell that has the smallest value, then the cell selection will be done on a priority basis. Go forward has first priority, turn to the right has the second priority, turn to the left has a third priority, and move backwards has a fourth priority. The value is changed in accordance with the position of the wall that has been mapped out by the robot. Cell values represent the cell distance to the destination cell.

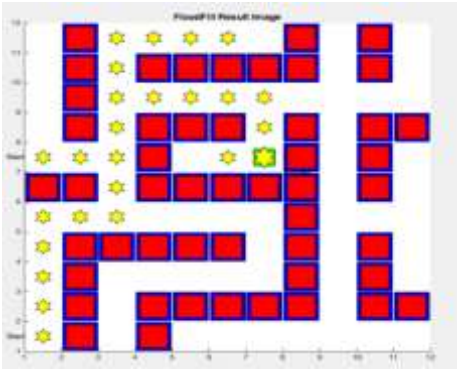

Fig 14.Simulation Result for Maze Solving Robot

TABLE 2.NUMBER OF STEPS FOR SOLVING THE MAZE

Start Point	Target Point	Routes	Number of Steps
1,1	7,7	(1,1),(1,2),(1,3),(1,4),(1,5),(2,5),(3,5),(3,6),(3,7),(2,7),(1,7),(3,8),(3,9),(3,10),(3,11),(4,11),(5,11),(6,11),(4,9),(5,9),(6,9),(7,9),(7,8),(7,7)	32

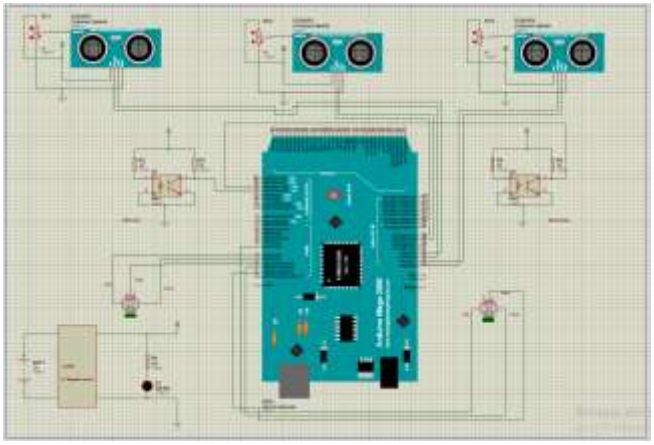

Fig 15.Simulation Test of Maze Solving Robot

VI. CONCLUSION

In this paper, Flood fill algorithms can provide the possible path with the maze among other algorithms. The most important equipment inside the micro mouse is controller that must be programmed with software to ensure micromouse making wise decision and navigating smoothly. The path finding algorithm receives inputs from the sensors and then calculates the best move. Developing algorithm for the micro mouse ultimately controls the autonomy positioning of the robot. The micro mouse needs to map the maze through intelligent exploration, and then track the optimal path. Finally, the movement algorithm outputs commands to the motor drivers for finding the target of the maze in the minimal amount of time.

ACKNOWLEDGMENT

The author is very thankful to Dr.Myint Thein, Rector of Mandalay Technological University, for his encouragement, invaluable permission and his kind support in carrying out this paper work.

The author is deeply grateful to Dr.Wut Yi Win, Associate Professor and Head, Department of Mechatronic Engineering, Mandalay Technological University for supplying all necessary things.

The author also wishes to thank to supervisor Dr.Nu Nu Win, Associate Professor, Department of Mechatronic Engineering, Mandalay Technological University, accomplished guidance, her willingness to share her ideas and, helpful suggestions and for her patience, continuous supervision and encouragement during a long period of this paper.

The author especially appreciates and thanks all her teachers for paper support, and guidance during theoretical study and paper preparation durations.

REFERENCES

[1] Sandeep Yadav, Kamal Kumar Verma, Swetamadhab Mahanta, "The Maze Problem Solved by Micro mouse",International Journal of Engineering and Advanced Technology (IJEAT),ISSN: 2249 – 8958, Volume-1, Issue-4, April 2012

[2] Ibrahim Elshamarka, Abu Bakar Sayuti Saman, "Design and Implementation of a Robot for Maze-Solving using Flood-Fill Algorithm", International Journal of Computer Applications (0975 – 8887),Volume 56– No.5, October 2012

[3] Wyard-Scott, L. and Q.H.M. Meng, "A potential maze solving algorithm for a micromouse robot", Communications, Computers, and Signal Processing, IEEE Pacific Rim Conference on 1995

[4] George Law, "Quantitative Comparison of Flood Fill and Modified Flood Fill Algorithms",International Journal of Computer Theory and Engineering, Vol. 5, No. 3, June 2013

[5] Babak Hosseini Kazerouni, Mona Behnam Moradi and Pooya Hosseini Kazerouni, "Variable Priorities in Maze- Sloving Algorithms for Robot's Movement", 2003.

[6] Sandeep Yadav, Kamal Kumar Verma, Swetamadhab Mahanta, "A New Method for a Micromouse to Find its Way through a Maze Unaided", International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-1, Issue-4, April 2012

[7] Chang Yuen Chung, "Micromouse maze solving robot" (Universiti Teknologi Malaysia, May 2009)