Design and Implementation of a Path Finding Robot Using Flood Fill Algorithm

1. Scopus Index
2. Manuskrip awal
3. Email penerimaan manuskrip
4. Manuskrip perbaikan pertama
5. Email penerimaan dengan perbaikan minor
6. Form hasil Reviewer
7. Email Penerimaan
8. Surat penerimaan
9. Manuskrip akhir

Scopus

Source details

International Journal of Mechanical Engineering and Robotics Research

Years currently covered by Scopus: from 2016 to 2025

Publisher: International Journal of Mechanical Engineering and Robotics Research

E-ISSN: 2278-0149

Subject area: Engineering: Mechanical Engineering · Engineering: Control and Systems Engineering · Computer Science: Artificial Intelligence

Source type: Journal

CiteScore 2023
2.8

SJR 2023
0.263

SNIP 2023
0.473

View all documents > Set document alert Save to source list

CiteScore CiteScore rank & trend Scopus content coverage

CiteScore 2023

$$2.8 = \frac{\text{1,433 Citations 2020 - 2023}}{\text{518 Documents 2020 - 2023}}$$

Calculated on 05 May, 2024

CiteScoreTracker 2024

$$3.1 = \frac{\text{1,065 Citations to date}}{\text{349 Documents to date}}$$

Last updated on 05 March, 2025 • Updated monthly

CiteScore rank 2023

Category	Rank	Percentile
Engineering — Mechanical Engineering	#331/672	50th
Engineering — Control and Systems Engineering	#172/321	46th

View CiteScore methodology > CiteScore FAQ > Add CiteScore to your site

Design and Implementation of a Path Finding Robot using Flood Fill Algorithm

Semuil Tjiharjadi
Computer Engineering Dept., Maranatha Christian University, Bandung, Indonesia
Email: semuiltj@gmail.com

Erwin Setiawan
Computer Engineering Dept., Maranatha Christian University, Bandung, Indonesia

Abstract

Autonomous robot is a robot that can perform certain work independently without the human help. Autonomous of navigation is one of the capabilities of autonomous robot to move from one point to another. Implementation of Autonomous robot navigation to explore an unknown environment, requires the robot to explore and map the environment and seek the path to reach a certain point.
Path Finding Robot is a mobile robot which moves using wheels with differential steering type. This path finding robot is designed to solve a maze environment that has a size of 5 x 5 cells and it is based on the flood-fill algorithm. Detection of walls and opening in the maze were done using ultrasonic range-finders. The robot was able to learn the maze, find all possible routes and solve it using the shortest one. This robot also use wall follower algorithms to correct the position of the robot against the side wall maze, so the robot can move straight. After several experiments, the robot can explore and map of maze and find the shortest path to destination point with a success rate of 70%.

Index Terms—**flood fill algorithm, path finding, maze, wall folower algorithm**

I. INTRODUCTION

Autonomous navigation is an important feature of mobile robotics. It allows the robot to independently move from a place to target location without a tele-operator. There are several techniques and algorithms have been developed for this purpose, each of them having their own merits and shortcomings [1,4-7].

Path Finding robot is using a structured technique and controlled implementation of autonomous navigation which is sometimes preferable in studying specific aspect of the problem [4]. This paper discusses an implementation of a small size mobile robot designed to solve a maze based on the flood-fill algorithm.

The path finding task is where robots try to solve a maze in the least time possible and using the most efficient way. A robot must navigate from a corner of a maze to the center as quickly as possible. It knows where the starting location is and where the target location is, but it does not have any information about the obstacles between the two. The maze is normally composed of 256 square cells, where the size each cell is about 18 cm × 18cm. The cells are arranged to form a 16 row × 16 column maze. The starting location of the maze is on one of the cells at its corners, and the target location is formed by four cells at the center of the maze. Only one cell is opened for entrance. The requirements of maze walls and support platform are provided in the IEEE standard .

Footnotes: 8-point Times New Roman font;

Manuscript received July 1, 2012; revised August 1, 2012; accepted September 1, 2012.

Copyright credit, project number, corresponding author, etc.

II. LITERATURE REVIEW

2.1. Breadth First Search

Breadth First Search uses First In First Out queue. It is used when space is not a problem and few solutions may exist and at least one has shortest path. It works poorly when all solutions have long path length or there is some heuristic function exists. It has large space complexity [9].

2.2. Depth First Search

Depth First Search uses Last In First out queue and are recursive in algorithm. It is simple to implement. But major problem with Depth First Search is it requires large computing power, for small increase in map size, runtime increases exponentially [9].

2.3. Heuristic Function

Heuristic function maps problem state descriptor to a number which represents degree of desirability. Heuristic function has different errors in different states. It plays vital role in optimization problem [9].

2.4. Genetic Algorithm

Genetic algorithm is used to find approximate optimal solution. It is inspired by evolutionary biology such as inheritance, mutation, crossover and selection [3]. Advantages of this algorithm are it solves problem with multiple solutions, it is very useful when input is very large. Disadvantages of Genetic algorithm are certain optimization problems cannot be solved due to poorly known fitness function, it cannot assure constant optimization response times, in Genetic algorithm the entire population is improving, but this could not be true for an individual within this population [9].

2.5. A* algorithm

A* combines feature of uniform-cost search and heuristic search. It is BFS in which cost associated with each node is calculated using admissible heuristic [1]. For graph traversal, it follows path with lowest known heuristic cost. The time complexity of this algorithm depends on heuristic used. Since it is Breadth First Search drawback of A* is large memory requirement because entire open-list is to be saved [9].

2.6. Flood Fill Algorithm

Robot maze problems are an important field of robotics and it is based on decision making algorithm [10]. It requires complete analysis of workspace or maze and proper planning [11]. Flood fill algorithm and modified flood fill are used widely for robot maze problem [2]. Flood fill algorithm assigns the value to each node which is represents the distance of that node from centre [9]. The flood fill algorithm floods the the maze when mouse reaches new cell or node. Thus it requires high cost updates [5]. These flooding are avoided in modified flood fill [1].

III. HARDWARE DESIGN

Mobile robot base construction was made using miniQ 2WD robot chassis. It was a product from DFRobot as shown in Figure 1. In the product consists of 1 robot chassis with a diameter of 122mm. 2 wheels with a diameter of 42mm, 1 piece ball caster and 2 DC motors which have been furnished by the gearbox as well as two pieces of the DC motor bracket to pair on the chassis.

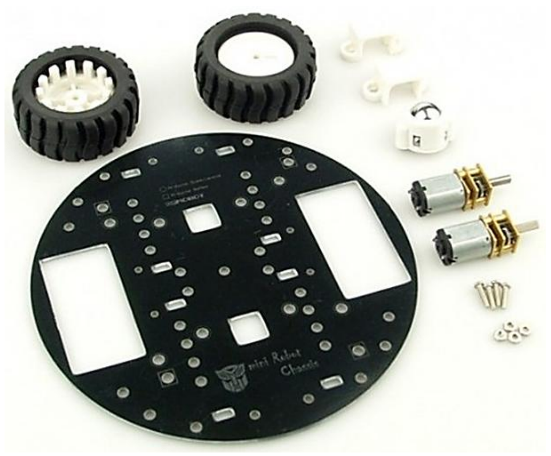

Figure 1. 1*2WD miniQ robot chassis*.

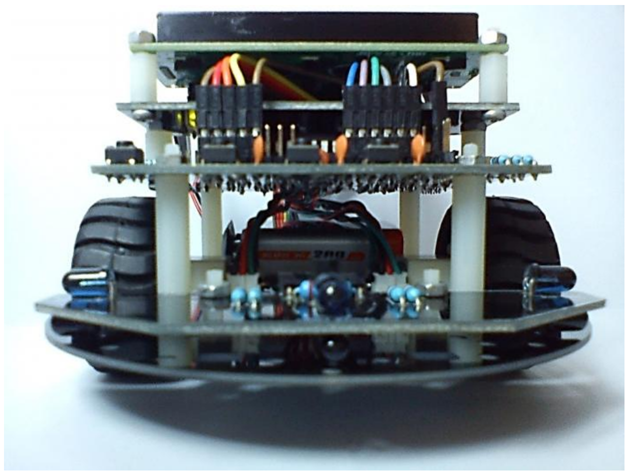

Figure 2. Mobile Robot from side view.

In this maze solving robot had 2 pieces rotary encoder. Rotary encoder used is miniQ robot chassis encoder which is also a product from DFRobot. Rotary encoder is compatible with 2WD products miniQ robot chassis. Rotary encoder attached to the DC motor to calculate the rotation of the wheel as shown in Figure 2.

The whole hardware system of this mobile robot can be seen in the block diagram at Figure 4 and Figure 5 shows the main program. Mobile robot used three infrared sensors to detect maze wall at right, left and front position. Driver L293D controled 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, spun to the right, turned to the left, and rotates reverse. ATmega324 microcontroller serves to process the signal-sinyalinput, perform processing algorithms, and generates output signals to control a robot. Information about all actions that had been taken by the robot, would be displayed on the LCD 16 x 2.

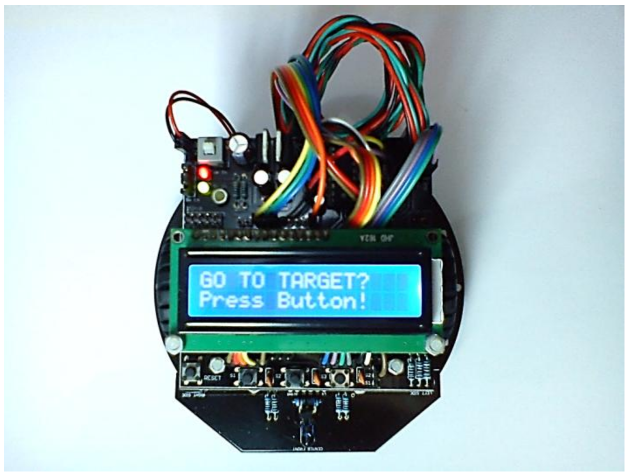

Figure 3. Mobile Robot from above view.

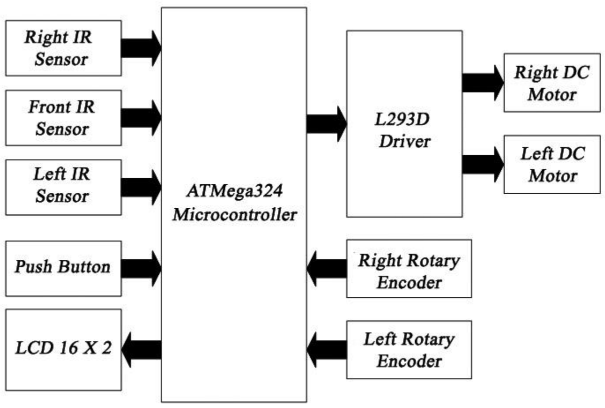

Figure 4. Block Diagram of Mobile Robot.

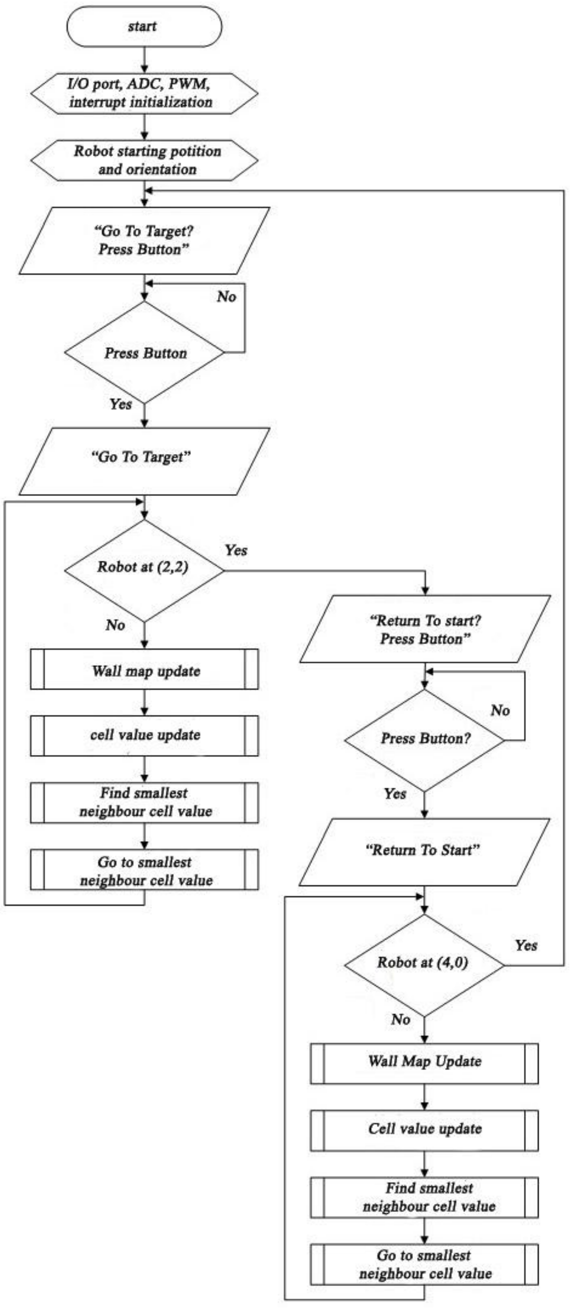
start
I/O port, ADC, PWM, interrupt initialization
Robot starting potition and orientation
"Go To Target? Press Button"
No
Press Button
Yes
"Go To Target"
Robot at (2,2)
Yes
No
Wall map update
cell value update
Find smallest neighbour cell value
Go to smallest neighbour cell value
"Return To start? Press Button"
No
Press Button?
Yes
"Return To Start"
Robot at (4,0)
Yes
No
Wall Map Update
Cell value update
Find smallest neighbour cell value
Go to smallest neighbour cell value

Figure 5. Flowchart of the main program.

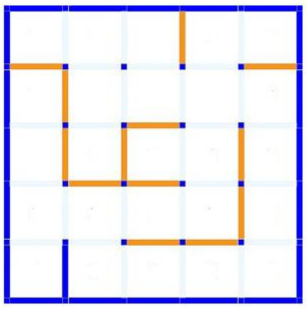

Figure 6. Design of the maze.

The maze designed for the robot to solve is of the size of 5×5 cells as shown in Figure 6. The actual maze constructed, as shown in Figure 7, has a physical size of about 1.32 m^2. 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 (Figure 2) must decide which one of the paths is shorter and solve the maze through that path.

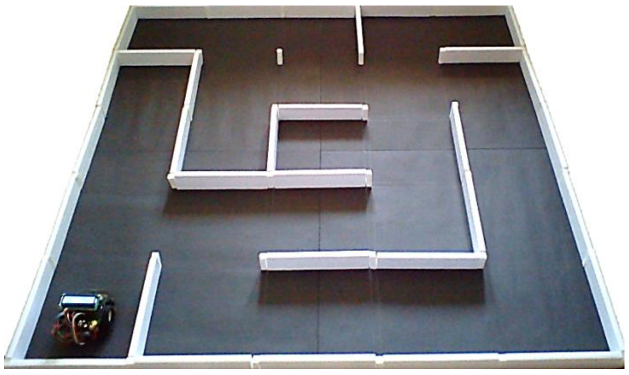

Figure 7. The maze.

IV. ALGORITHM

Choosing an algorithm for the maze robot is critical in solving the maze. In this exercise, flood-fill algorithm was chosen to solve the maze due to its balance in efficiency and complexity.

Mapping the maze which has size of 5x5 cells is accomplished by using two-dimensional memory array with a size of 5x5. Artificial intelligence program requires two memory arrays 5x5. The first memory array is used to store information in each cell walls of the maze. The second array of memory function is used to store the cell value information in each cell. The position of the robot in the program expressed by the coordinates (row, column). The movement of the robot in the array is done to position the robot as in Figure 8.

If the robot moves one cell to the south, then the coordinates of the line increases 1. If the robot moves one cell to the West, then the coordinates of the column will be reduced by 1. If the robot moves one cell to the North, then the coordinates of the line will be reduced by 1. If the robot move one cell to the East, the coordinates of the column will increase 1. The initial conditions of the robot, already has information about the initial position, the initial orientation, the size of the maze, and the existence of the outer walls of the maze.

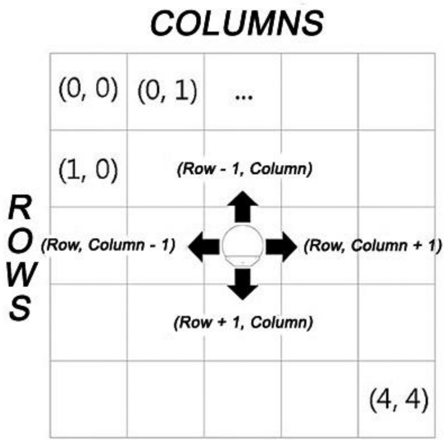

Figure 8. Array of Robot Movement

There are four main steps in the algorithm; wall data updates, cell value updates, the smallest neigbour cell calculation, and moving to the smallest neighbour cell.

4.1 Wall data update

If robot decides where it wants to move to, it will check if it is surrounded by any walls in any of the three directions: front, right nad left. The robot will read the distance of any obstacle at each direction and check if the distance in each is more than 20 cm. The ones that exceed 20 cm are updated as "wall" on their respective side. It shows by the flowchart in Figure 9. Robot also needs to know which direction it is facing. There are four orientations for the robot: north, south, east or west, as shown in table 1. Initial orientation was set at start and the robot keeps tracking of any changes.

Table 1. Robot detection when it detect wall.

Robot Orientation	Detection Sensor		
	Right	Front	Left
South	West wall	South wall	East wall
West	North wall	West wall	South wall
North	East wall	North wall	West wall
East	South wall	East wall	North wall

4.2 Cell value update

Update value of the cell (restocked every cell with the new value) serves to adjust the value in each cell of the position of the wall that has been updated by the robot. The value stored in the array 2 dimensions of memory cell with size 5x5. Update the value of the cell is done by using the flood fill algorithm.

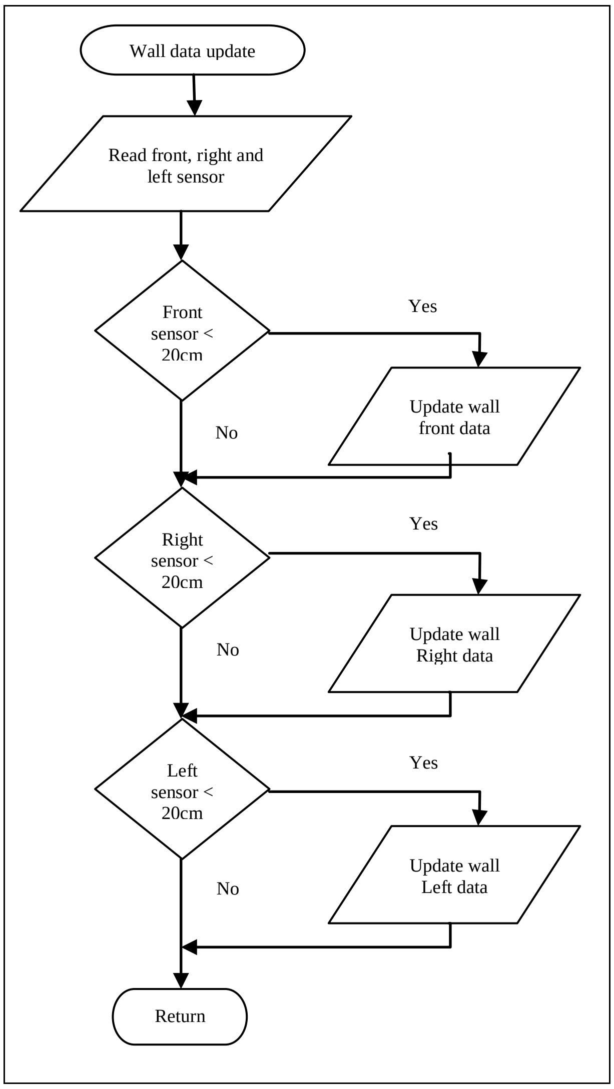

Figure 9. Flowchart for updating wall location at each cell

Update cell values subroutine works by resetting the values of the previous cell, then it will give a value of 255 in each cell, then fill in the values of these cells gradually, start value (level) 0 to all the cells filled grades. The cells that will be updated is the current_level array while neighboring cells will be inserted into the next_level array. After value fill in process is completed, then the cells are in next_level array will be moved to an array of fill in current_level to do next value. The update process will be complete if the value of the cell array next_level empty.

4.3 The smallest neigbour cell calculation

Subroutine specify the smallest neighboring cells function to search for a neighboring cell which has the smallest value. The smallest neighboring cell search is done on a priority basis, so that if there is more than one neighboring cell that has the smallest value, then the selected cells are cells that have a higher priority.

Prioritization is based on the movement of the robot is moving forward one cell has the first priority, move one cell to the right has a second priority, move one cell to the left has a third priority, and moving backward one cell has the fourth priority. For example, if the robot were facing the South, the South cells have a first priority, the second priority of the West has a cell, the cell has a third priority East and North cells have fourth priority as in Figure 10. If the robot was facing the East, the East cells have a first priority, South cells have second priority, the North has a third priority cells, and cells West has fourth priority.

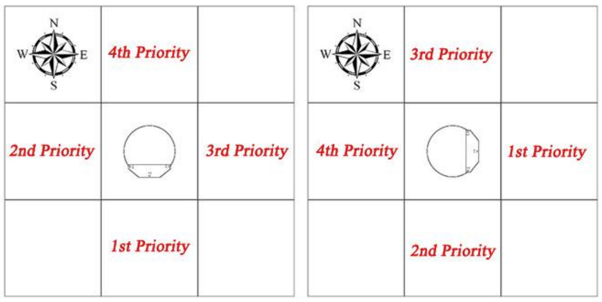

Figure 10. Priority of Neighbour cell

4.4. Moving to the smallest neighbour cell

Subroutine moves to the smallest neighboring cells function to move the robot towards neighboring cells which have the smallest value, after the robot finds the neighboring cells. To perform movement to the cell, the robot should know the location of the cell. Furthermore, the robot will move to the cells by observing the orientation. For example, if the South cell is the smallest cell and orientation of the robot was facing west, then to move to the position of the cell, the robot must be turning left, then move forward as in Figure 11. If the South cell is the smallest cell and robot orientation was facing East, then to move to the position of the cell, the robot must be spinning right, then move forward.

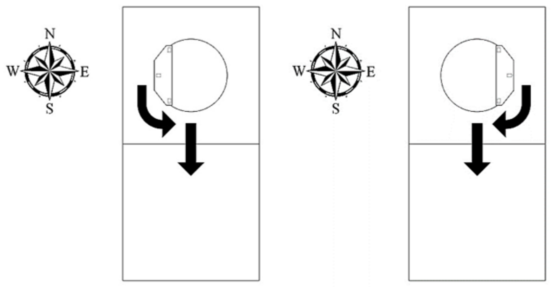

Figure 11. Moving to smallest neighbour cell.

V. RESULTS AND DISCUSSION

In this experiments, Robot will learn to find the shortest path from the starting cell (line 4, column 0) to the destination cell (row 2, column 2) and then back again to the initial cell. The initial orientation of the robot is facing the North.

The maze simulator program aims to facilitate the observation on how the flood fill algorithm. Figure 12 is a view maze simulator program. Maze blue wall is a wall that position known to the robot. While the maze walls are colored orange wall position is not known by the robot.

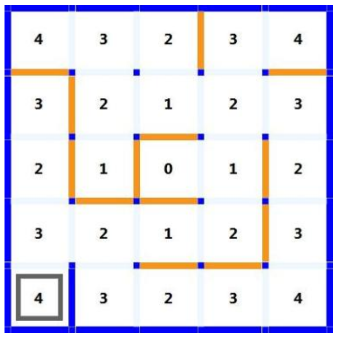

Figure 12. Simulation search path to cell (2,2)

Robot will perform a search of the initial cell lines (4.0) to the destination cell (2, 2). Flood fill algorithm simulation results when a search of the cell lines (4, 0) to the cell (2, 2) are shown in Figure 12 to 22.

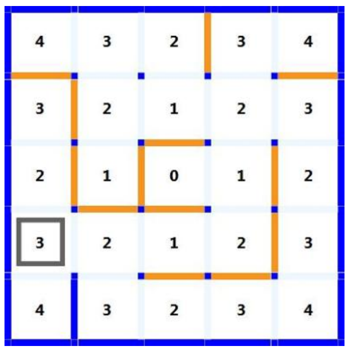

Figure 13. Simulation search path to cell (2,2)

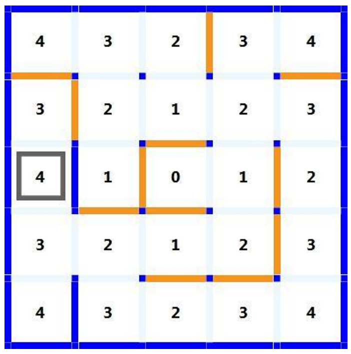

Figure 14. Simulation search path to cell (2,2)

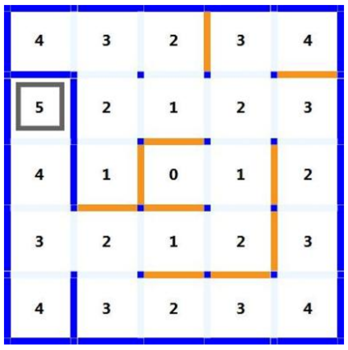

Figure 15. Simulation search path to cell (2,2)

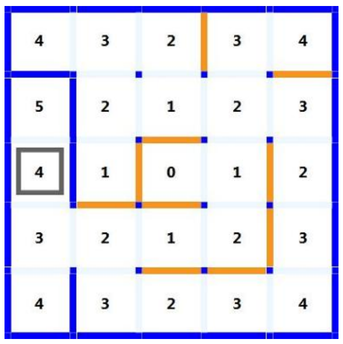

Figure 16. Simulation search path to cell (2,2)

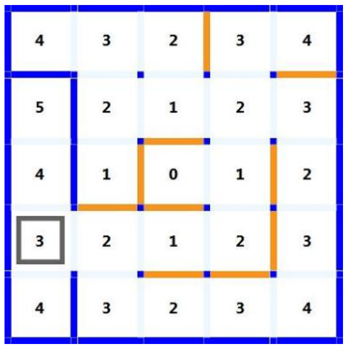

Figure 17. Simulation search path to cell (2,2)

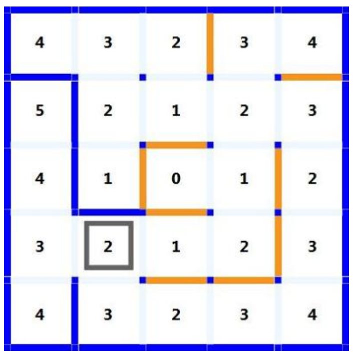

Figure 18. Simulation search path to cell (2,2)

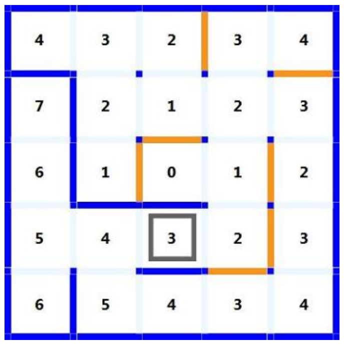

Figure 19. Simulation search path to cell (2,2)

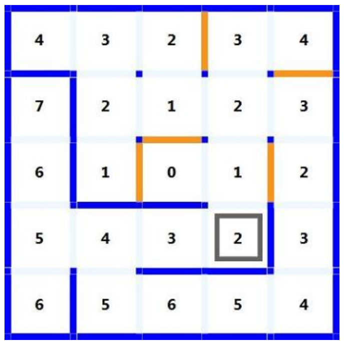

Figure 20. Simulation search path to cell (2,2)

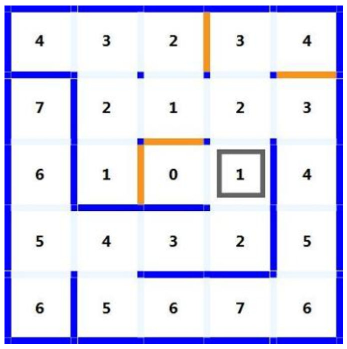

Figure 21. Simulation search path to cell (2,2)

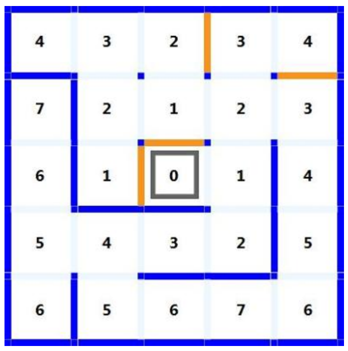

Figure 22. Simulation search path to cell (2,2)

After robot run the search and update his wall data, then it knows the shortest path to go to cell (2,2). It is shown in table 2.

Table 2. First and second routes of robot experiment

	Routes	Number of steps
First run	(4,0) →(3,0) → (2,0) → (1,0) → (2,0) → (3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	10
Return home	(2,2) → (2,3) → (3,3) → (3,2) → (3,1) → (3,0) → (4,0)	6
Second run	(4,0) →(3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	6

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 foward 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.

VI. CONCLUSION

This design and implementation of the robot is a study about the ability to equip a small mobile robot with the ability to learn how to navigate in unknown environment based on its own decisions. The flood-fill algorithm was found to be an effective tool for maze-solving of a moderate size. For the robot to make its decisions it relies on inputs from several sensors, namely the ultrasonic range sensors and wheel rotation decoders.

The robot has successfully able to map the maze in the first, return home and second runs. In its second run it reaches its target cell through the shortest route it has mapped in the previous first run and return home.

Future works may include to studying the robot's maze solving capability in a bigger and more complex maze. In order to improve the quality in wall detection, better object sensor, such as a laser range finder, is needed. It is much more costly but it have ability to scan its surrounding at a wirde angle plane, so it will help a lot in search ability at bigger and more complex maze.

REFERENCES

[1] Bekti, Samudra Harapan. *Pencarian Shortest Path Dinamik dengan Algoritma Bellman Based Flood Fill dan Implementasinya pada Robot Micromouse*: Institut Teknologi Bandung. 2009.

[2] Braunl, Thomas. *Embedded Robotics*. Berlin: Springer. 2006.

[3] Cook, David. *Intermediate Robot Building*. New York: Apress. 2010.

[4] Elshamarka, Ibrahim danAbu Bakar Sayuti Saman. *Design and Implementation of a Robot for Maze-Solving using Flood-Fill Algorithm*: Universiti Teknologi Petronas. 2012.

[5] Elshamarka, I. And A.B.S. Saman, "Design and Implementation of a Robot for Maze-Solving Using Flood-Fill Algorithm", in International Journal of Computer Applications Volume 56-No.5, pp.8-13, October 2012.

[6] Ansari, A., M.A. Sayyed, K. Ratlamwala and P. Shaikh, "An Optimized Hybrid Approach For Path Finding", in International Journal in Foundations of Computer Science & Technology (IJFCST), Vol. 5 No. 2, pp. 47-58, March 2015.

[7] Sharma, K. And C. Munshi, "A Comprehensive and Comparative Study of Maze-Solving Techniques by Implementing Graph Theory", in IOSR Journal of Computer Engineering, Vol. 17, Issue 1, Ver. IV, pp. 24-29, 2015.

[8] Lucas, G. W. *A Tutorial and Elementary Trajectory Model for the Differential Steering System of Robot Wheel Actuators*. http://rossum.sourceforge.net/ papers/DiffSteer/, dikunjungi Juni 2014.

[9] Sreekanth, R.K., "Artificial Intelligence Algorithms", IOSR Journal of Computer Engineering (IOSRJCE), volume 6, issue 3 September-October, 2012.

[10] Magnusson, Per. *Design of an H-Bridge*. http://axotron.se/index_en.php?page=34, dikunjungi Juni 2014.

[11] Mazidi, Muhammad Ali, Sarmad Niami, dan Sepehr Niami. *The AVR Microcontroller and Embedded System*. New Jersey: Prentice Hall. 2011.

[12] Rizqiawan, Arwindra. *Sekilas Rotary Encoder*. http://konversi.wordpress.com/ 2009/06/12/sekilas-rotary-encoder/, dikunjungi Juni 2014.

[13] Scherz, Paul.*Practical Electronics for Inventors*. New York: McGraw-Hill. 2000.

[14] Schildt, Hebert. *The Complete Reference C++*. Osborne: McGraw-Hill. 2013.

Full paper submission to ICAMD 2015-D05 Inbox ×

icamd 2016 <icamd@asr.org>
to me, tjiharjadi

Oct 30, 2015, 4:55 PM

Dear Mr. Semuil Tjiharjadi and Erwin Setiawan,

Have a nice day!

Thank you very much for your interests in ICAMD 2016.

Your full paper entitled "Design and Implementation of a Path Finding Robot using Flood Fill Algorithm" is well received. And your paper ID is **D05**. Thank you very much for your submission!

Pls take a note for the following tips;
1. The reference No.s are out of order in the text, please rewrite them in accordance with order.
2. The figure 3 is missing in the text; pls add it in the correct position.

Pls revise your paper according to the above. And please send it back to this email address as soon as possible after you finished!

Anything confused, pls feel free to contact us via this email address. We will contact you within 2 working days. More information about ICAMD 2016, pls visit the conference website:http://www.icamd.org/.

Kiko Xu(Ms.)
Conference secretary of ICAMD 2016
Conference Website: www.icamd.org
Conference email address: icamd@asr.org
ASR official conference website: www.asr.org(http://www.asr.org/list-15-1.html)

Design and Implementation of a Path Finding Robot using Flood Fill Algorithm

Semuil Tjiharjadi
Computer Engineering Dept., Maranatha Christian University, Bandung, Indonesia
Email: semuiltj@gmail.com

Erwin Setiawan
Computer Engineering Dept., Maranatha Christian University, Bandung, Indonesia

Abstract

Autonomous robot is a robot that can perform certain work independently without the human help. Autonomous of navigation is one of the capabilities of autonomous robot to move from one point to another. Implementation of Autonomous robot navigation to explore an unknown environment, requires the robot to explore and map the environment and seek the path to reach a certain point.
Path Finding Robot is a mobile robot which moves using wheels with differential steering type. This path finding robot is designed to solve a maze environment that has a size of 5 x 5 cells and it is based on the flood-fill algorithm. Detection of walls and opening in the maze were done using ultrasonic range-finders. The robot was able to learn the maze, find all possible routes and solve it using the shortest one. This robot also use wall follower algorithms to correct the position of the robot against the side wall maze, so the robot can move straight. After several experiments, the robot can explore and map of maze and find the shortest path to destination point with a success rate of 70%.

Index Terms—**flood fill algorithm, path finding, maze, wall folower algorithm**

I. Introduction

Autonomous navigation is an important feature of mobile robotics. It allows the robot to independently move from a place to target location without a tele-operator. There are several techniques and algorithms have been developed for this purpose, each of them having their own merits and shortcomings [1-5].

Path Finding robot is using a structured technique and controlled implementation of autonomous navigation which is sometimes preferable in studying specific aspect of the problem [2]. This paper discusses an implementation of a small size mobile robot designed to solve a maze based on the flood-fill algorithm.

The path finding task is where robots try to solve a maze in the least time possible and using the most efficient way. A robot must navigate from a corner of a maze to the center as quickly as possible. It knows where the starting location is and where the target location is, but it does not have any information about the obstacles between the two. The maze is normally composed of 256 square cells, where the size each cell is about 18 cm × 18cm. The cells are arranged to form a 16 row × 16 column maze. The starting location of the maze is on one of the cells at its corners, and the target location is formed by four cells at the center of the maze. Only one cell is opened for entrance. The requirements of maze walls and support platform are provided in the IEEE standard .

II. LITERATURE REVIEW

2.1. Breadth First Search

Breadth First Search uses First In First Out queue. It is used when space is not a problem and few solutions may exist and at least one has shortest path. It works poorly when all solutions have long path length or there is some heuristic function exists. It has large space complexity [6].

2.2. Depth First Search

Depth First Search uses Last In First out queue and are recursive in algorithm. It is simple to implement. But major problem with Depth First Search is it requires large computing power, for small increase in map size, runtime increases exponentially [6].

2.3. Heuristic Function

Heuristic function maps problem state descriptor to a number which represents degree of desirability. Heuristic function has different errors in different states. It plays vital role in optimization problem [6].

2.4. Genetic Algorithm

Genetic algorithm is used to find approximate optimal solution. It is inspired by evolutionary biology such as

Footnotes: 8-point Times New Roman font;
Manuscript received July 1, 2012; revised August 1, 2012; accepted September 1, 2012.
Copyright credit, project number, corresponding author, etc.

inheritance, mutation, crossover and selection [7]. Advantages of this algorithm are it solves problem with multiple solutions, it is very useful when input is very large. Disadvantages of Genetic algorithm are certain optimization problems cannot be solved due to poorly known fitness function, it cannot assure constant optimization response times, in Genetic algorithm the entire population is improving, but this could not be true for an individual within this population [6].

2.5. A* algorithm

A* combines feature of uniform-cost search and heuristic search. It is BFS in which cost associated with each node is calculated using admissible heuristic [1]. For graph traversal, it follows path with lowest known heuristic cost. The time complexity of this algorithm depends on heuristic used. Since it is Breadth First Search drawback of A* is large memory requirement because entire open-list is to be saved [6].

2.6. Flood Fill Algorithm

Robot maze problems are an important field of robotics and it is based on decision making algorithm [8]. It requires complete analysis of workspace or maze and proper planning [9]. Flood fill algorithm and modified flood fill are used widely for robot maze problem [10]. Flood fill algorithm assigns the value to each node which is represents the distance of that node from centre [6]. The flood fill algorithm floods the the maze when mouse reaches new cell or node. Thus it requires high cost updates [3]. These flooding are avoided in modified flood fill [1].

III. HARDWARE DESIGN

Mobile robot base construction was made using miniQ 2WD robot chassis. It was a product from DFRobot as shown in Figure 1. In the product consists of 1 robot chassis with a diameter of 122mm. 2 wheels with a diameter of 42mm, 1 piece ball caster and 2 DC motors which have been furnished by the gearbox as well as two pieces of the DC motor bracket to pair on the chassis.

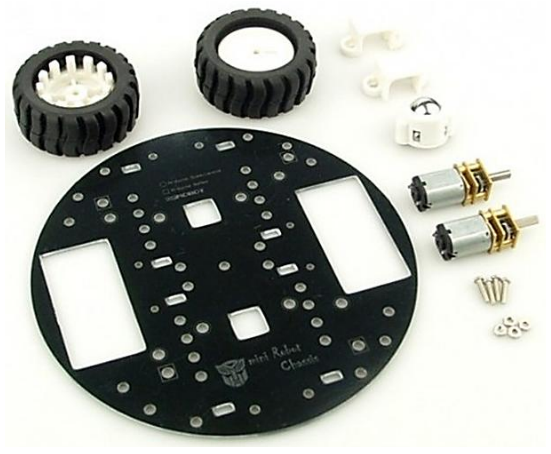

Figure 1. 1*2WD miniQ robot chassis*.

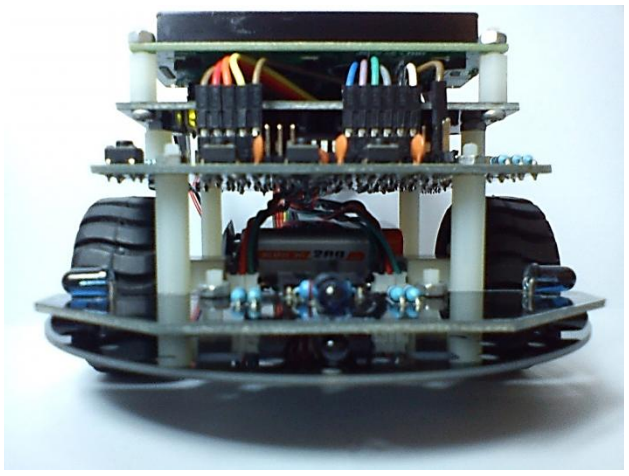

Figure 2. Mobile Robot from side view.

In this maze solving robot had 2 pieces rotary encoder. Rotary encoder used is miniQ robot chassis encoder which is also a product from DFRobot. Rotary encoder is compatible with 2WD products miniQ robot chassis. Rotary encoder attached to the DC motor to calculate the rotation of the wheel as shown in Figure 2. [11]

The whole hardware system of this mobile robot can be seen in the block diagram at Figure 3 and Figure 4 shows the main program. Mobile robot used three infrared sensors to detect maze wall at right, left and front position. Driver L293D controled the direction of rotation and speed of a DC motor [12]. 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, spun to the right, turned to the left, and rotates reverse [13]. ATmega324 microcontroller serves to process the signal-sinyalinput, perform processing algorithms, and generates output signals to control a robot [9]. Information about all actions that had been taken by the robot, would be displayed on the LCD 16 x 2 at Figure 5.

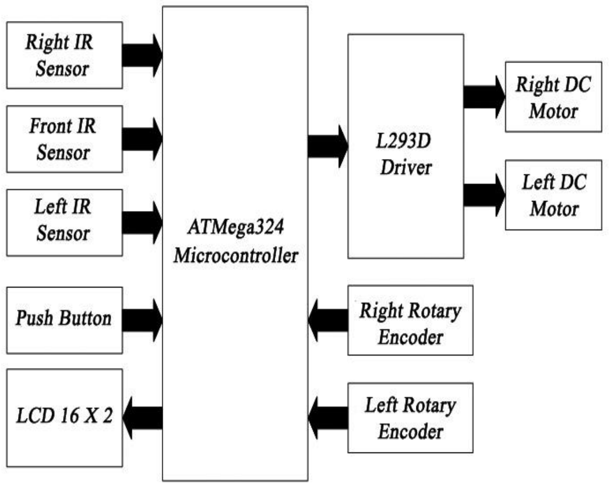

Figure 3. Block Diagram of Mobile Robot.

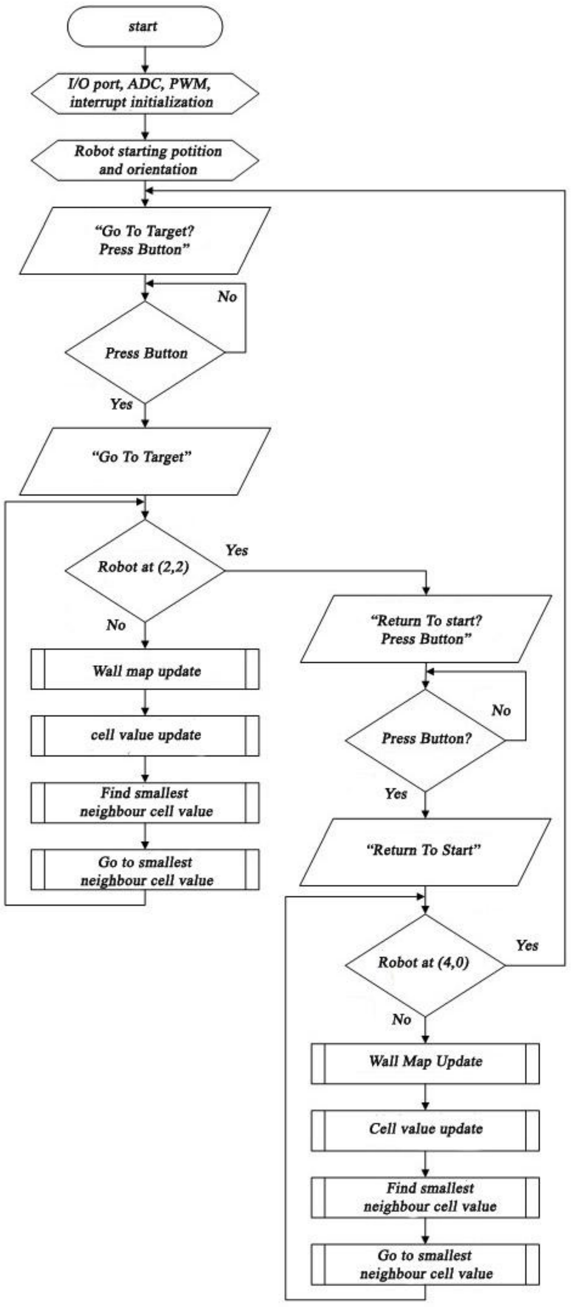
start
I/O port, ADC, PWM, interrupt initialization
Robot starting potition and orientation
"Go To Target? Press Button"
Press Button
No
Yes
"Go To Target"
Robot at (2,2)
Yes
No
Wall map update
cell value update
Find smallest neighbour cell value
Go to smallest neighbour cell value
"Return To start? Press Button"
Press Button?
No
Yes
"Return To Start"
Robot at (4,0)
Yes
No
Wall Map Update
Cell value update
Find smallest neighbour cell value
Go to smallest neighbour cell value

Figure 4. Flowchart of the main program.

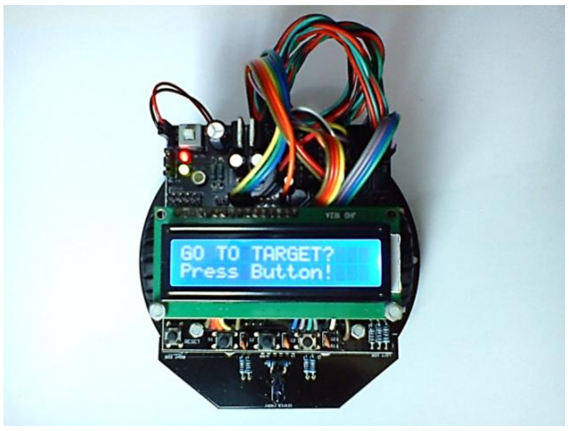

Figure 5. Mobile Robot from above view.

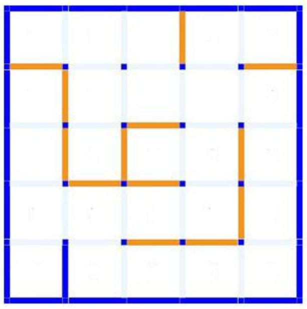

Figure 6. Design of the maze.

The maze designed for the robot to solve is of the size of 5×5 cells as shown in Figure 6. The actual maze constructed, as shown in Figure 7, has a physical size of about 1.32 m^2. 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 (Figure 2) must decide which one of the paths is shorter and solve the maze through that path.

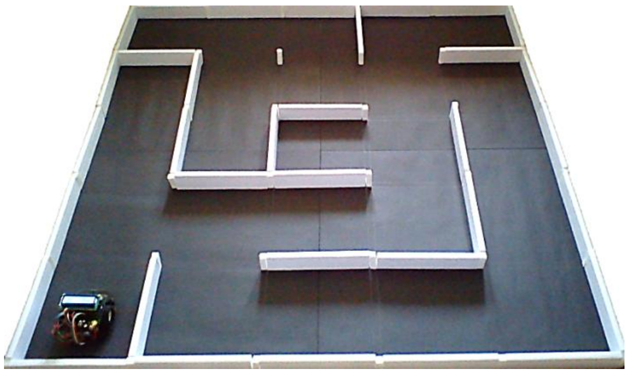

Figure 7. The maze.

IV. ALGORITHM

Choosing an algorithm for the maze robot is critical in solving the maze. In this exercise, flood-fill algorithm was chosen to solve the maze due to its balance in efficiency and complexity.

Mapping the maze which has size of 5x5 cells is accomplished by using two-dimensional memory array with a size of 5x5. Artificial intelligence program requires two memory arrays 5x5. The first memory array is used to store information in each cell walls of the maze. The second array of memory function is used to store the cell value information in each cell. The position of the robot in the program expressed by the coordinates (row, column). The movement of the robot in the array is done to position the robot as in Figure 8.

If the robot moves one cell to the south, then the coordinates of the line increases 1. If the robot moves one cell to the West, then the coordinates of the column will be reduced by 1. If the robot moves one cell to the North, then the coordinates of the line will be reduced by 1. If the robot move one cell to the East, the coordinates of the column will increase 1. The initial conditions of the robot, already has information about the initial position, the initial orientation, the size of the maze, and the existence of the outer walls of the maze.

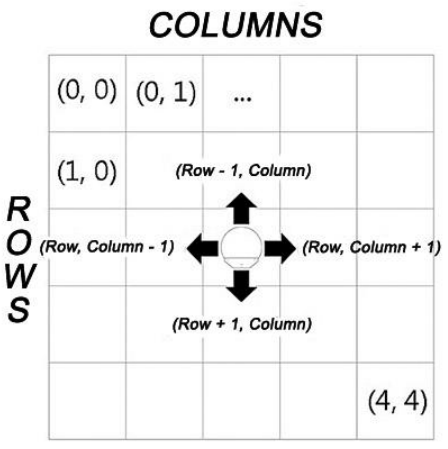

Figure 8. Array of Robot Movement

There are four main steps in the algorithm; wall data updates, cell value updates, the smallest neigbour cell calculation, and moving to the smallest neighbour cell.

4.1 Wall data update

If robot decides where it wants to move to, it will check if it is surrounded by any walls in any of the three directions: front, right nad left. The robot will read the distance of any obstacle at each direction and check if the distance in each is more than 20 cm. The ones that exceed 20 cm are updated as “wall” on their respective side. It shows by the flowchart in Figure 9. Robot also needs to know which direction it is facing. There are four orientations for the robot: north, south, east or west, as shown in table 1. Initial orientation was set at start and the robot keeps tracking of any changes.

Table 1. Robot detection when it detect wall.

Robot Orientation	Detection Sensor		
	Right	Front	Left
South	West wall	South wall	East wall
West	North wall	West wall	South wall
North	East wall	North wall	West wall
East	South wall	East wall	North wall

4.2 Cell value update

Update value of the cell (restocked every cell with the new value) serves to adjust the value in each cell of the position of the wall that has been updated by the robot. The value stored in the array 2 dimensions of memory cell with size 5x5. Update the value of the cell is done by using the flood fill algorithm.

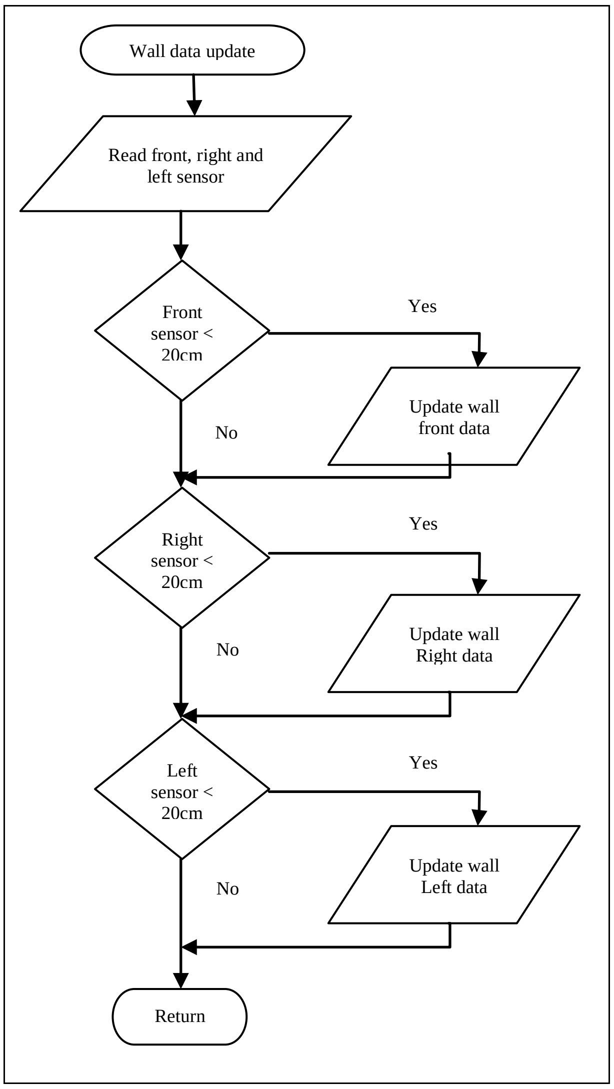

Figure 9. Flowchart for updating wall location at each cell

Update cell values subroutine works by resetting the values of the previous cell, then it will give a value of 255 in each cell, then fill in the values of these cells gradually, start value (level) 0 to all the cells filled grades. The cells that will be updated is the current_level array while neighboring cells will be inserted into the next_level array. After value fill in process is completed, then the cells are in next_level array will be moved to an array of fill in current_level to do next value. The update process will be complete if the value of the cell array next_level empty.

4.3 The smallest neigbour cell calculation

Subroutine specify the smallest neighboring cells function to search for a neighboring cell which has the smallest value. The smallest neighboring cell search is done on a priority basis, so that if there is more than one neighboring cell that has the smallest value, then the selected cells are cells that have a higher priority.

Prioritization is based on the movement of the robot is moving forward one cell has the first priority, move one cell to the right has a second priority, move one cell to the left has a third priority, and moving backward one cell has the fourth priority. For example, if the robot were facing the South, the South cells have a first priority, the second priority of the West has a cell, the cell has a third priority East and North cells have fourth priority as in Figure 10. If the robot was facing the East, the East cells have a first priority, South cells have second priority, the North has a third priority cells, and cells West has fourth priority.

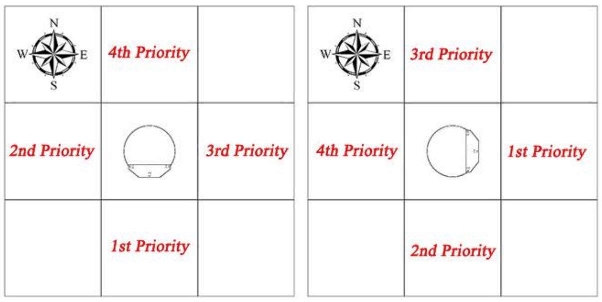

Figure 10. Priority of Neighbour cell

4.4. Moving to the smallest neighbour cell

Subroutine moves to the smallest neighboring cells function to move the robot towards neighboring cells which have the smallest value, after the robot finds the neighboring cells. To perform movement to the cell, the robot should know the location of the cell. Furthermore, the robot will move to the cells by observing the orientation. For example, if the South cell is the smallest cell and orientation of the robot was facing west, then to move to the position of the cell, the robot must be turning left, then move forward as in Figure 11. If the South cell is the smallest cell and robot orientation was facing East, then to move to the position of the cell, the robot must be spinning right, then move forward.

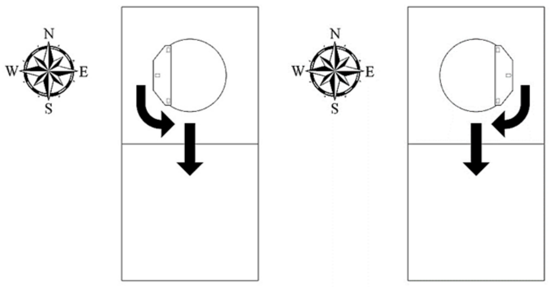

Figure 11. Moving to smallest neighbour cell.

V. RESULTS AND DISCUSSION

In this experiments, Robot will learn to find the shortest path from the starting cell (line 4, column 0) to the destination cell (row 2, column 2) and then back again to the initial cell. The initial orientation of the robot is facing the North.

The maze simulator program aims to facilitate the observation on how the flood fill algorithm. Figure 12 is a view maze simulator program. Maze blue wall is a wall that position known to the robot. While the maze walls are colored orange wall position is not known by the robot.

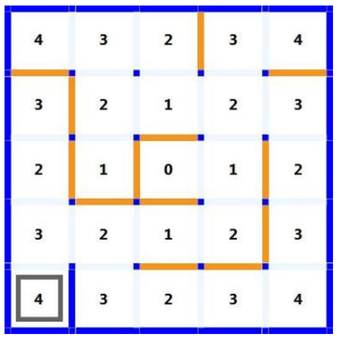

Figure 12. Simulation search path to cell (2,2)

Robot will perform a search of the initial cell lines (4.0) to the destination cell (2, 2). Flood fill algorithm simulation results when a search of the cell lines (4, 0) to the cell (2, 2) are shown in Figure 12 to 22.

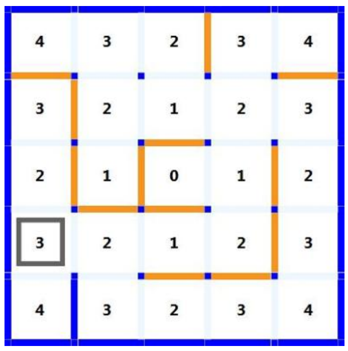

Figure 13. Simulation search path to cell (2,2)

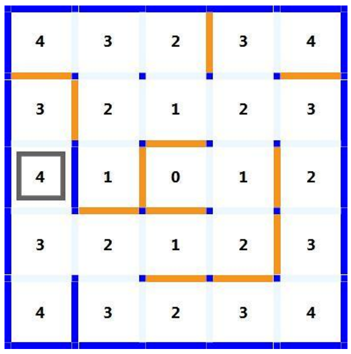

Figure 14. Simulation search path to cell (2,2)

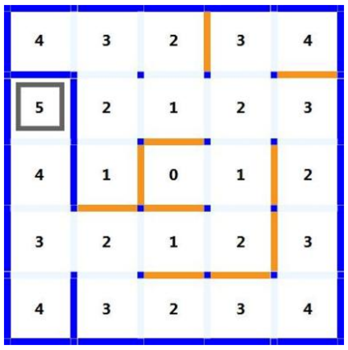

Figure 15. Simulation search path to cell (2,2)

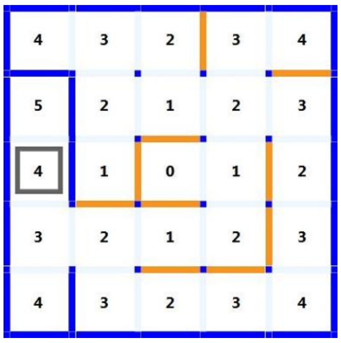

Figure 16. Simulation search path to cell (2,2)

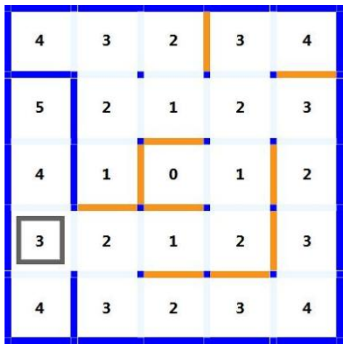

Figure 17. Simulation search path to cell (2,2)

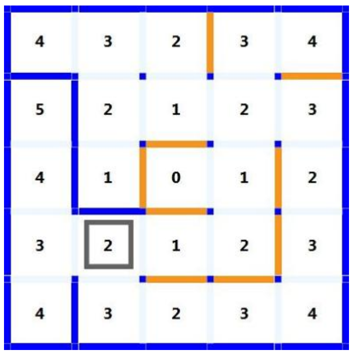

Figure 18. Simulation search path to cell (2,2)

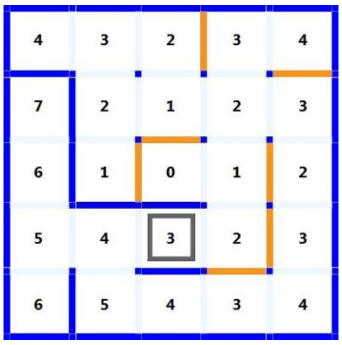

Figure 19. Simulation search path to cell (2,2)

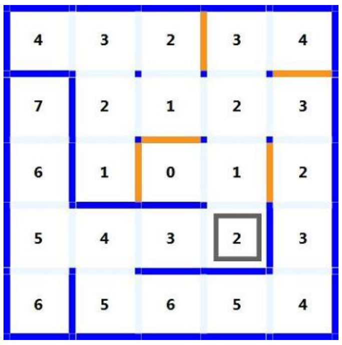

Figure 20. Simulation search path to cell (2,2)

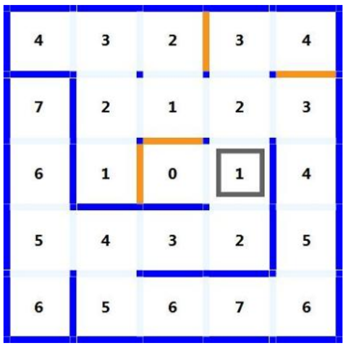

Figure 21. Simulation search path to cell (2,2)

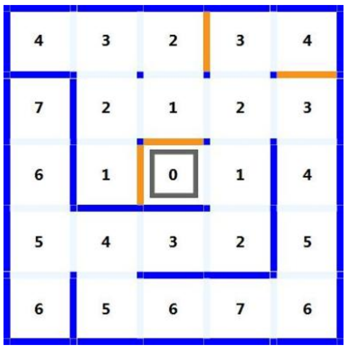

Figure 22. Simulation search path to cell (2,2)

After robot run the search and update his wall data, then it knows the shortest path to go to cell (2,2). It is shown in table 2.

Table 2. First and second routes of robot experiment

	Routes	Number of steps
First run	(4,0) → (3,0) → (2,0) → (1,0) → (2,0) → (3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	10
Return home	(2,2) → (2,3) → (3,3) → (3,2) → (3,1) → (3,0) → (4,0)	6
Second run	(4,0) → (3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	6

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 foward 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.

VI. CONCLUSION

This design and implementation of the robot is a study about the ability to equip a small mobile robot with the ability to learn how to navigate in unknown environment based on its own decisions. The flood-fill algorithm was found to be an effective tool for maze-solving of a moderate size. For the robot to make its decisions it relies on inputs from several sensors, namely the ultrasonic range sensors and wheel rotation decoders.

The robot has successfully able to map the maze in the first, return home and second runs. In its second run it reaches its target cell through the shortest route it has mapped in the previous first run and return home.

Future works may include to studying the robot's maze solving capability in a bigger and more complex maze. In order to improve the quality in wall detection, better object sensor, such as a laser range finder, is needed. It is much more costly but it have ability to scan its surrounding at a wirde angle plane, so it will help a lot in search ability at bigger and more complex maze.

REFERENCES

[1] Bekti, Samudra Harapan. *Pencarian Shortest Path Dinamik dengan Algoritma Bellman Based Flood Fill dan Implementasinya pada Robot Micromouse*: Institut Teknologi Bandung. 2009. 1

[2] Elshamarka, Ibrahim danAbu Bakar Sayuti Saman. *Design and Implementation of a Robot for Maze-Solving using Flood-Fill Algorithm*: Universiti Teknologi Petronas. 2012. 2

[3] Elshamarka, I. And A.B.S. Saman, "Design and Implementation of a Robot for Maze-Solving Using Flood-Fill Algorithm", in International Journal of Computer Applications Volume 56-No.5, pp.8-13, October 2012. 3

[4] Ansari, A., M.A. Sayyed, K. Ratlamwala and P. Shaikh, "An Optimized Hybrid Approach For Path Finding", in International Journal in Foundations of Computer Science & Technology (IJFCST), Vol. 5 No. 2, pp. 47-58, March 2015. 4

[5] Sharma, K. And C. Munshi, "A Comprehensive and Comparative Study of Maze-Solving Techniques by Implementing Graph Theory", in IOSR Journal of Computer Engineering, Vol. 17, Issue 1, Ver. IV, pp. 24-29, 2015. 5

[6] Sreekanth, R.K., "Artificial Intelligence Algorithms", IOSR Journal of Computer Engineering (IOSRJCE), volume 6, issue 3 September-October, 2012. 6

[7] Cook, David. *Intermediate Robot Building*. New York: Apress. 2010. 7

[8] Magnusson, Per. *Design of an H-Bridge*. http://axotron.se/index_en.php?page=34, dikunjungi Juni 2014. 8

[9] Mazidi, Muhammad Ali, Sarmad Niami, dan Sepehr Niami. *The AVR Microcontroller and Embedded System*. New Jersey: Prentice Hall. 2011. 9

[10] Braunl, Thomas. *Embedded Robotics*. Berlin: Springer. 2006. 10

[11] Rizqiawan, Arwindra. *Sekilas Rotary Encoder*. http://konversi.wordpress.com/ 2009/06/12/ sekilas-rotary-encoder/, Juni 2014. 11

[12] Scherz, Paul.*Practical Electronics for Inventors*. New York: McGraw-Hill. 2000. 12

[13] Lucas, G. W. *A Tutorial and Elementary Trajectory Model for the Differential Steering System of Robot Wheel Actuators*. http://rossum.sourceforge.net/ papers/DiffSteer/, Juni 2014. 13

ICAMD2016-Notification of Acceptance-D05 Inbox ×

icamd 2016 <icamd@asr.org>
to me, tjiharjadi

Fri, Dec 4, 2015, 9:50 AM

Dear Mr. Semuil Tjiharjadi and Erwin Setiawan,
Have a nice day!
Here is a good news for you. Our reviewers has reviewed your paper entitled "Design and Implementation of a Path Finding Robot using Flood Fill Algorithm", and sent us the review comments back. And congratulations! Your paper has been accepted. Here are the attachments (Notification of Acceptance, Review Form).
Please revise your paper according to the review comments. Please finish the registration process, according to the instruction of the notification of acceptance.
We are looking forward to meeting you in Singapore!
Anything confused, pls feel free to contact us via this email address. We will contact you within 2 working days. More information about ICAMD 2016, pls visit the conference website:http://www.icamd.org/.

Kiko Xu(Ms.)
Conference secretary of ICAMD 2016
Conference Website: www.icamd.org
Conference email address: icamd@asr.org
ASR official conference website: www.asr.org(http://www.asr.org/list-15-1.html)

Review Form of ICAMD 2016

January 13-14, 2016. Singapore

http://www.icamd.org/

Evaluation:					
	Poor	Fair	Good	Very Good	Outstanding
Originality	○	○	√	○	○
Innovation	○	√	○	○	○
technical merit	○	○	√	○	○
applicability	○	○	○	√	○
Presentation and English	○	○	○	√	○
Match to Conference Topic	○	○	○	√	○
Recommendation to Editors					
	Strongly Reject	Reject	Marginally Accept	Accept	Strong Accept
Recommendation	○	○	○	√	○

Paper ID : D05

Paper Title : Design and Implementation of a Path Finding Robot using Flood Fill Algorithm

I. Review

A. *Suitability of topic*

1. Is the topic appropriate for publication?

   √ Yes ◻ Perhaps ◻ No

2. Is the topic important to colleagues working in the field?

   √ Yes ◻ Perhaps ◻ No

B. *Contents*

1. Is the paper technically sound? If no, why not?

   √ Yes ◻ No

2. Is the coverage of the topic sufficiently comprehensive and balanced?
   √ Yes
   □ Important Information is missing or superficially treated.
   □ Treatment somewhat unbalanced, but not seriously so.
   □ Certain parts significantly overstressed.
3. How would you describe the technical depth of the paper?
   □ Superficial
   □ Suitable for the non-specialist
   √ Appropriate for the generally knowledgeable individual working in the field
   □ Suitable only for an expert
4. How would you rate the technical novelty of the paper?
   □ Novel √ Somewhat Novel □ Not Novel

C. Presentation

1. How would you rate the overall organization of the paper?
   √ Satisfactory □ Could be improved □ Poor
2. Are the title and abstract satisfactory?

   √ Yes □ No

3. Is the length of the paper appropriate? If not, recommend how the length of the paper should be amended, including a possible target length for the final manuscript

   √ Yes □ No

4. Are symbols, terms, and concepts adequately defined?

   √ Yes □ Not always □ No

5. How do you rate the English usage?

   □ Satisfactory √ Needs improvment □ Poor

6. How do you rate the list of references?

   √ Satisfactory □ Unsatisfactory

D. Overall rating (circle appropriate rating)

1. How would you rate the technical contents of the paper?

□ Excellent [✓ Good] □ Fair □ Poor

2. How would you rate the novelty of the paper?

□ Highly Novel □ Sufficiently Novel [✓ Slightly Novel] □ Not Novel

3. How would you rate the "literary" presentation of the paper?

[✓ Totally Accessible] □Mostly Accessible □Partially Accessible □Inaccessible

4. How would you rate the appropriateness of this paper for publication?

□ Excellent Match [✓ Good Match] □ Weak Match □ Poor Match

II. RECOMMENDATION

□ Publish unaltered
[✓ Publish In Minor, Required Changes]
□ Review Again After Major Changes
□ Reject (Paper Is Not Of Sufficient Quality Or Novelty To Be Published In This Journal)
□ Reject (A Major Rewrite Is Required; Encourage Resubmission)
□ Reject (Paper Is Seriously Flawed; Do Not Encourage Resubmission.)

III. COMMENTS

Please state the reason you gave the recommendation above. Please give the author specific guidance regarding revisions, differentiating between optional and mandatory changes.

This paper, to solve a maze environment, designed a path finding robot using flood fill algorithm. The detection technology was based on ultrasonic range-finders. Authors also performed several experiments showing that the robot can explore and map of maze and find the shortest path to destination point with a success rate of 70%. It can be of interest to people involved in Autonomous robot study. Some minor improvement on this paper can be made based on the following:

1. Make it clear the time-cost of the robot on the condition of finding the shortest path or reaching the destination.
2. As you mentioned the flood fill algorithm requires high cost updates, and then what is the better methodology to solve this problem.
3. Please check carefully about your vocabularies and grammars.
4. Pls make sure the format of your paper is totally matched with the journal template.

Attention; Official language is English in paper writing and presenting

D05-Notification of Acceptance&Review Form-ICAMD2016 Inbox ×

icamd 2016 <icamd@asr.org>
to me, tjiharjadi

Dec 8, 2015, 4:00 PM

Notification of Acceptance

2016 International Conference on Advanced Mechanical Design

(ICAMD2016)

Singapore, January 13-14, 2016.

http://www.icamd.org/

Dear Semuil Tjiharjadi and Erwin Setiawan,

We are pleased to inform you that the review processes for **2016 International Conference on Advanced in Mechanical Design (ICAMD 2016)** has been completed. The conference received submissions from nearly 7 different countries and regions, which were reviewed by international experts, and about 10 papers have been selected for presentation and publication. Based on the recommendations of the reviewers and the Technical Program Committees, we are pleased to inform you that your paper identified above has been accepted for publication and oral presentation. You are cordially invited to present the paper orally at ICAMD 2016 to be held during **January 13-14, 2016 in Singapore** at **Quality Hotel**. (Click)

Paper ID:	**D05**
Paper Title:	**Design and Implementation of a Path Finding Robot using Flood Fill Algorithm**

After reviewing, your above paper will be published in **Conference Proceedings (ISSN: 2261-236X),** which will be indexed by SCOPUS, Ei Compendex and other academic databases; or accepted papers will be recommended to be published in the **IJMERR** ***International Journal of Mechanical Engineering and Robotics Research (ISSN: 2278-0149)*** which will be indexed by Index Copernicus, ProQuest, UDL, Google Scholar, Open J-Gate; etc.

Publication in both Conference Proceedings and Journal:

Firstly, papers will undergo the peer review system of the conference committee, and accepted papers will be published into Conference Proceedings after registration. Then the authors are supposed to add at least 30% new content and resubmit the papers to *icamd@asr.org* within 30 days after the conference for further peer review to get published by IJMERR, and no extra fee will be charged for publishing if accepted.

ICAMD Organizing Committees

Registration Instruction

***So in order to register the conference and have your paper included in the proceeding successfully, you must finish following SIX steps**

1. Revise your paper according to the Review Comments carefully. (**Attached**)

2. Format your paper according to the Template carefully.

IJMERR-Template: http://www.etpub.com/down/Journal-template.doc;

Conference Proceeding: http://www.icamd.org/prods_template_A4.doc.

3. Download and complete the Registration Form.
http://www.icamd.org/reg.docx

4. Finish the payment of Registration fee. (The information can be found in the Registration form)

5. Finish the Copyright Form.
IJMERR: http://www.ijmerr.com/uploadfile/2015/0326/20150326112301988.pdf
Conference Proceeding: http://www.matec-conferences.org/doc_journal/copyright/mateconf_copyright.pdf

6. Send your **final papers (both .doc and .pdf format), filled registration form (.doc format), copyright form (.jpg format) and the scanned payment (.jpg format)** to us at icamd@asr.org. (Before December 20, 2015)

***If you pay by on-line Credit Card Payment, please fill your confirmation number in the registration form after paying.**
***If you pay by bank transfer, please scan the payment slip as the payment proof for checking.**

Maybe some unforeseeable events prevents a few authors from attending the event to present their papers, so if you and your co-author(s) could not attend ICAMD 2016 to present your paper for some reasons, please inform us. And we will send you the official receipt of registration fee and journal after ICAMD 2016 conference free of charge.

Please strictly adhere to the format specified in the conference template while preparing your final paper. If you have any problem in preparing the final paper, please feel free to contact us via *icamd@asr.org*. For the most updated information on the conference, please check the conference website at http://www.icamd.org/. The Final Conference Program will be available at the website around **December, 2015**.

Finally, we would like to further extend our congratulations to you and we are looking forward to meeting you in Singapore.

Kiko Xu(Ms.)
Conference secretary of ICAMD 2016
Conference Website: www.icamd.org
Conference email address: icamd@asr.org
ASR official conference website: ***www.asr.org****(http://www.asr.org/list-15-1.html)*

American Society for Research

Notification of Acceptance

2016 International Conference on Advanced Mechanical Design (ICAMD2016)

Singapore, January 13-14, 2016.

http://www.icamd.org/

Dear Semuil Tjiharjadi and Erwin Setiawan,

We are pleased to inform you that the review processes for **2016 International Conference on Advanced in Mechanical Design (ICAMD 2016)** has been completed. The conference received submissions from nearly 7 different countries and regions, which were reviewed by international experts, and about 10 papers have been selected for presentation and publication. Based on the recommendations of the reviewers and the Technical Program Committees, we are pleased to inform you that your paper identified above has been accepted for publication and oral presentation. You are cordially invited to present the paper orally at ICAMD 2016 to be held during **January 13-14, 2016 in Singapore** at **Quality Hotel**. (Click)

Paper ID:	**D05**
Paper Title:	**Design and Implementation of a Path Finding Robot using Flood Fill Algorithm**

After reviewing, your above paper will be published in **Conference Proceedings (ISSN: 2261-236X),** which will be indexed by SCOPUS, Ei Compendex and other academic databases; or accepted papers will be recommended to be published in the **IJMERR** ***International Journal of Mechanical Engineering and Robotics Research (ISSN: 2278-0149)*** which will be indexed by Index Corpernicus, ProQuest, UDL, Google Scholar, Open J-Gate; etc.

Publication in both Conference Proceedings and Journal:

Firstly, papers will undergo the peer review system of the conference committee, and accepted papers will be published into Conference Proceedings after registration. Then the authors are supposed to add at least 30% new content and resubmit the papers to *icamd@asr.org* within 30 days after the conference for further peer review to get published by IJMERR, and no extra fee will be charged for publishing if accepted.

American Society for Research
ASR
2000

ICAMD Organizing Committees
http://www.icamd.org/
Singapore

Registration Instruction

***So in order to register the conference and have your paper included in the proceeding successfully, you must finish following SIX steps**

1. Revise your paper according to the Review Comments carefully. (**Attached**)

2. Format your paper according to the Template carefully.

   IJMERR-Template: http://www.etpub.com/down/Journal-template.doc;

   Conference Proceeding: http://www.icamd.org/prods_template_A4.doc.

3. Download and complete the Registration Form.
   http://www.icamd.org/reg.docx

4. Finish the payment of Registration fee. (The information can be found in the Registration form)

5. Finish the Copyright Form.
   IJMERR: http://www.ijmerr.com/uploadfile/2015/0326/20150326112301988.pdf
   Conference Proceeding: http://www.matec-conferences.org/doc_journal/copyright/matecconf_copyright.pdf

6. Send your **final papers (both .doc and .pdf format), filled registration form (.doc format), copyright form (.jpg format) and the scanned payment (.jpg format)** to us at icamd@asr.org. (Before December 20, 2015)

***If you pay by on-line Credit Card Payment, please fill your confirmation number in the registration form after paying.**
***If you pay by bank transfer, please scan the payment slip as the payment proof for checking.**

Maybe some unforeseeable events prevents a few authors from attending the event to present their papers, so if you and your co-author(s) could not attend ICAMD 2016 to present your paper for some reasons, please inform us. And we will send you the official receipt of registration fee and journal after ICAMD 2016 conference free of charge.

Please strictly adhere to the format specified in the conference template while preparing your final paper. If you have any problem in preparing the final paper, please feel free to contact us via *icamd@asr.org*. For the most updated information on the conference, please check the conference website at http://www.icamd.org/. The Final Conference Program will be available at the website around **December, 2015**.

Finally, we would like to further extend our congratulations to you and we are looking forward to meeting you in Singapore.

Design and Implementation of a Path Finding Robot using Flood Fill Algorithm

Semuil Tjiharjadi
Computer Engineering Dept., Maranatha Christian University, Bandung, Indonesia
Email: semuiltj@gmail.com

Erwin Setiawan
Computer Engineering Dept., Maranatha Christian University, Bandung, Indonesia

Abstract—**Autonomous robot is a robot that can perform certain work independently without the human help. Autonomous of navigation is one of the capabilities of autonomous robot to move from one point to another. Implementation of Autonomous robot navigation to explore an unknown environment, requires the robot to explore and map the environment and seek the path to reach a certain point.**

Path Finding Robot is a mobile robot which moves using wheels with differential steering type. This path finding robot is designed to solve a maze environment that has a size of 5 x 5 cells and it is based on the flood-fill algorithm. Detection of walls and opening in the maze were done using ultrasonic range-finders. The robot was able to learn the maze, find all possible routes and solve it using the shortest one. This robot also use wall follower algorithms to correct the position of the robot against the side wall maze, so the robot can move straight. After several experiments, the robot can explore and map of maze and find the shortest path to destination point with a success rate of 70%.

Index Terms—**flood fill algorithm, path finding, maze, wall folower algorithm**

I. Introduction

Autonomous navigation is an important feature of mobile robotics. It allows the robot to independently move from a place to target location without a tele-operator. There are several techniques and algorithms have been developed for this purpose, each of them having their own merits and shortcomings [1-5].

Path Finding robot is using a structured technique and controlled implementation of autonomous navigation which is sometimes preferable in studying specific aspect of the problem [2]. This paper discusses an implementation of a small size mobile robot designed to solve a maze based on the flood-fill algorithm.

The path finding task is where robots try to solve a maze in the least time possible and using the most efficient way. A robot must navigate from a corner of a maze to the center as quickly as possible. It knows where the starting location is and where the target location is, but it does not have any information about the obstacles between the two. The maze is normally composed of 256 square cells, where the size each cell is about 18 cm × 18cm. The cells are arranged to form a 16 row × 16 column maze. The starting location of the maze is on one of the cells at its corners, and the target location is formed by four cells at the center of the maze. Only one cell is opened for entrance. The requirements of maze walls and support platform are provided in the IEEE standard .

II. Literature Review

A. *Breadth First Search*

Breadth First Search uses First In First Out queue. It is used when space is not a problem and few solutions may exist and at least one has shortest path. It works poorly when all solutions have long path length or there is some heuristic function exists. It has large space complexity [6].

B. *Depth First Search*

Depth First Search uses Last In First out queue and are recursive in algorithm. It is simple to implement. But major problem with Depth First Search is it requires large computing power, for small increase in map size, runtime increases exponentially [6].

C. *Heuristic Function*

Heuristic function maps problem state descriptor to a number which represents degree of desirability. Heuristic function has different errors in different states. It plays vital role in optimization problem [6].

D. *Genetic Algorithm*

Genetic algorithm is used to find approximate optimal solution. It is inspired by evolutionary biology such as inheritance, mutation, crossover and selection [7]. Advantages of this algorithm are it solves problem with multiple solutions, it is very useful when input is very large. Disadvantages of Genetic algorithm are certain optimization problems cannot be solved due to poorly known fitness function, it cannot assure constant optimization response times, in Genetic algorithm the entire population is improving, but this could not be true for an individual within this population [6].

E. *A* algorithm*

A*combines feature of uniform-cost search and heuristic search. It is BFS in which cost associated with each node is calculated using admissible heuristic [1]. For graph traversal, it follows path with lowest known heuristic cost. The time complexity of this algorithm depends on heuristic used. Since it is Breadth First Search drawback of A* is large memory requirement because entire open-list is to be saved [6].

F. Flood Fill Algorithm

Robot maze problems are an important field of robotics and it is based on decision making algorithm [8]. It requires complete analysis of workspace or maze and proper planning [9]. Flood fill algorithm and modified flood fill are used widely for robot maze problem [10]. Flood fill algorithm assigns the value to each node which is represents the distance of that node from centre [6]. The flood fill algorithm floods the maze when mouse reaches new cell or node. Thus it requires high cost updates [3]. These flooding are avoided in modified flood fill [1].

III. Hardware Design

Mobile robot base construction was made using miniQ 2WD robot chassis. It was a product from DFRobot as shown in Figure 1. In the product consists of 1 robot chassis with a diameter of 122mm. 2 wheels with a diameter of 42mm, 1 piece ball caster and 2 DC motors which have been furnished by the gearbox as well as two pieces of the DC motor bracket to pair on the chassis.

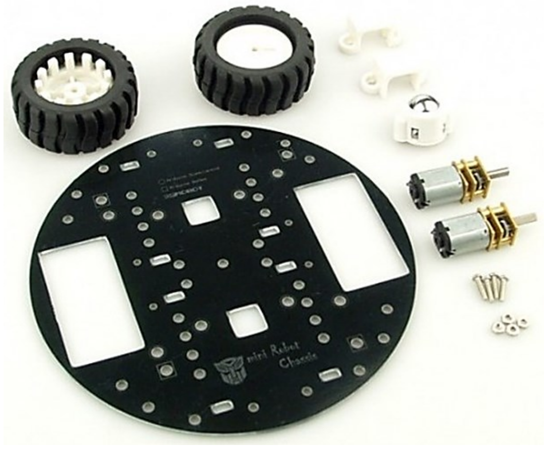

Figure 1. 12WD miniQ robot chassis.

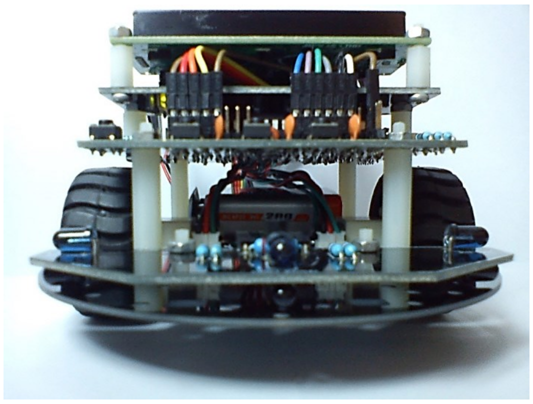

Figure 2. Mobile robot from side view.

In this maze solving robot had 2 pieces rotary encoder. Rotary encoder used is miniQ robot chassis encoder which is also a product from DFRobot. Rotary encoder is compatible with 2WD products miniQ robot chassis. Rotary encoder attached to the DC motor to calculate the rotation of the wheel as shown in Figure 2. [11]

The whole hardware system of this mobile robot can be seen in the block diagram at Figure 3 and Figure 4 shows the main program. Mobile robot used three infrared sensors to detect maze wall at right, left and front position. Driver L293D controled the direction of rotation and speed of a DC motor [12]. 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, spun to the right, turned to the left, and rotates reverse [13]. ATmega324 microcontroller serves to process the signal-sinyalinput, perform processing algorithms, and generates output signals to control a robot [9]. Information about all actions that had been taken by the robot, would be displayed on the LCD 16 x 2 at Figure 5.

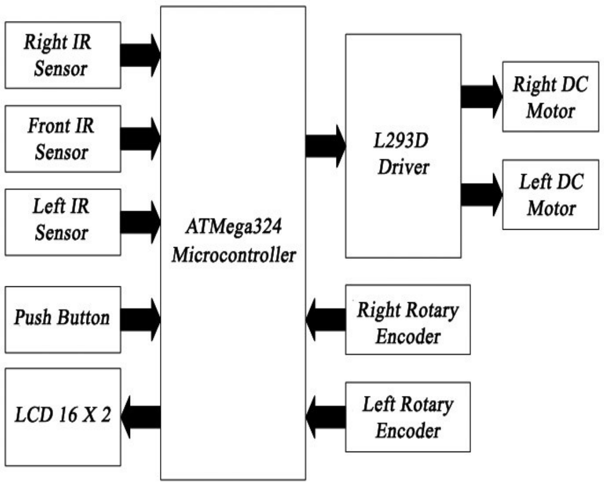

Figure 3. Block diagram of mobile robot.

The maze designed for the robot to solve is of the size of 5×5 cells as shown in Figure 6. The actual maze constructed, as shown in Figure 7, has a physical size of about 1.32 m2. 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 (Figure 2) must decide which one of the paths is shorter and solve the maze through that path.

IV. Algorithm

Choosing an algorithm for the maze robot is critical in solving the maze. In this exercise, flood-fill algorithm was chosen to solve the maze due to its balance in efficiency and complexity.

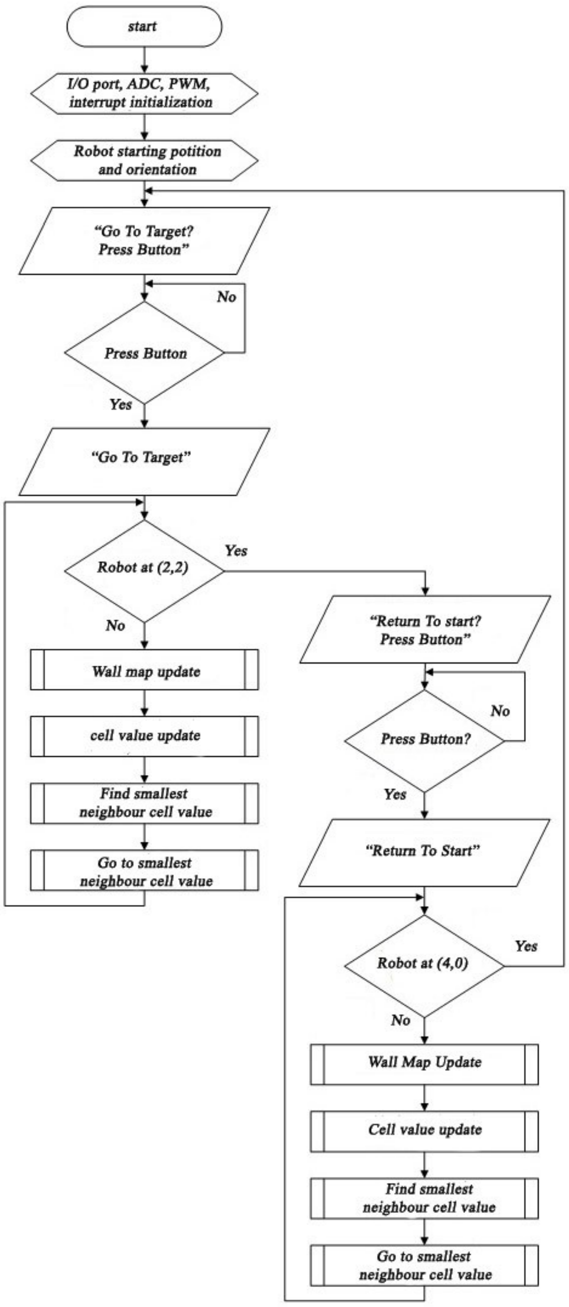
start
I/O port, ADC, PWM, interrupt initialization
Robot starting potition and orientation
"Go To Target? Press Button"
Press Button
No
Yes
"Go To Target"
Robot at (2,2)
Yes
No
Wall map update
cell value update
Find smallest neighbour cell value
Go to smallest neighbour cell value
"Return To start? Press Button"
Press Button?
No
Yes
"Return To Start"
Robot at (4,0)
Yes
No
Wall Map Update
Cell value update
Find smallest neighbour cell value
Go to smallest neighbour cell value

Figure 4. Flowchart of the main program.

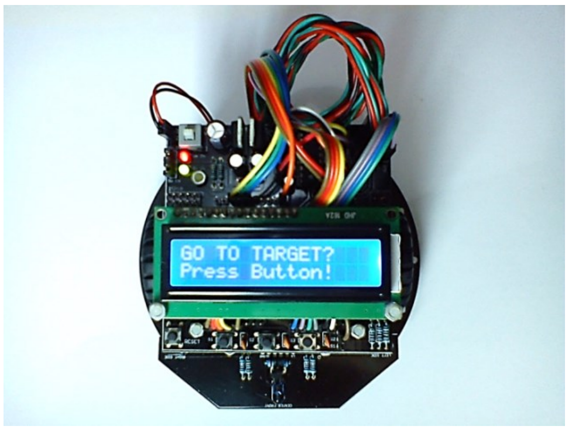

Figure 5. Mobile robot from above view.

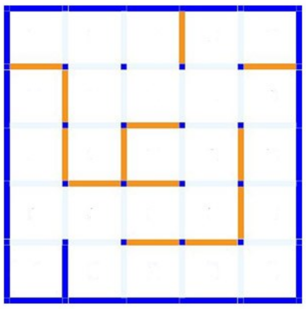

Figure 6. Design of the maze.

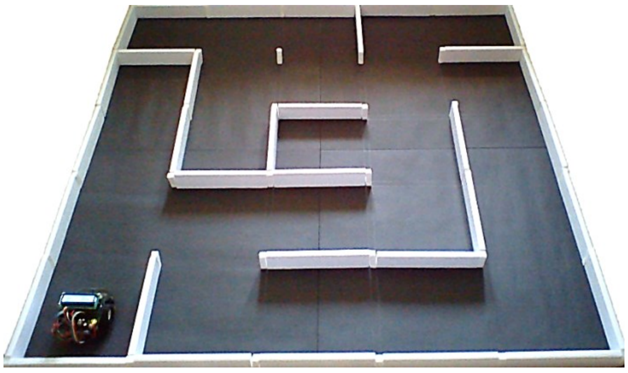

Figure 7. The maze.

Mapping the maze which has size of 5x5 cells is accomplished by using two-dimensional memory array with a size of 5x5. Artificial intelligence program requires two memory arrays 5x5. The first memory array is used to store information in each cell walls of the maze. The second array of memory function is used to store the cell value information in each cell. The position of the robot in the program expressed by the coordinates (row, column). The movement of the robot in the array is done to position the robot as in Figure 8.

If the robot moves one cell to the south, then the coordinates of the line increases 1. If the robot moves one cell to the West, then the coordinates of the column will be reduced by 1. If the robot moves one cell to the North, then the coordinates of the line will be reduced by 1. If the robot move one cell to the East, the coordinates of the column will increase 1. The initial conditions of the robot, already has information about the initial position, the initial orientation, the size of the maze, and the existence of the outer walls of the maze.

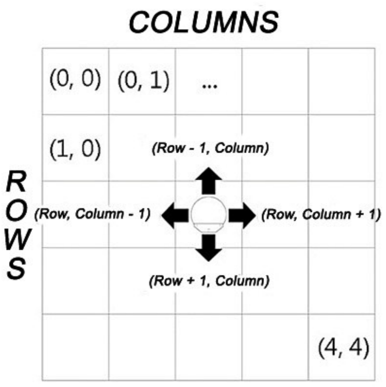

Figure 8. Array of robot movement

There are four main steps in the algorithm; wall data updates, cell value updates, the smallest neigbour cell calculation, and moving to the smallest neighbour cell.

A. Wall Data Update

If robot decides where it wants to move to, it will check if it is surrounded by any walls in any of the three directions: front, right nad left. The robot will read the distance of any obstacle at each direction and check if the distance in each is more than 20 cm. The ones that exceed 20 cm are updated as "wall" on their respective side. It shows by the flowchart in Figure 9. Robot also needs to know which direction it is facing. There are four orientations for the robot: north, south, east or west, as shown in table 1. Initial orientation was set at start and the robot keeps tracking of any changes.

TABLE I. ROBOT DETECTION WHEN IT DETECT WALL.

Robot Orientation	Detection Sensor		
	Right	Front	Left
South	West wall	South wall	East wall
West	North wall	West wall	South wall
North	East wall	North wall	West wall
East	South wall	East wall	North wall

B. Cell Value Update

Update value of the cell (restocked every cell with the new value) serves to adjust the value in each cell of the position of the wall that has been updated by the robot. The value stored in the array 2 dimensions of memory cell with size 5x5. Update the value of the cell is done by using the flood fill algorithm.

Update cell values subroutine works by resetting the values of the previous cell, then it will give a value of 255 in each cell, then fill in the values of these cells gradually, start value (level) 0 to all the cells filled grades. The cells that will be updated is the current_level array while neighboring cells will be inserted into the next_level array. After value fill in process is completed, then the cells are in next_level array will be moved to an array of fill in current_level to do next value. The update process will be complete if the value of the cell array next_level empty.

C. The Smallest Neigbour Cell Calculation

Subroutine specify the smallest neighboring cells function to search for a neighboring cell which has the smallest value. The smallest neighboring cell search is done on a priority basis, so that if there is more than one neighboring cell that has the smallest value, then the selected cells are cells that have a higher priority.

Prioritization is based on the movement of the robot is moving forward one cell has the first priority, move one cell to the right has a second priority, move one cell to the left has a third priority, and moving backward one cell has the fourth priority. For example, if the robot were facing the South, the South cells have a first priority, the second priority of the West has a cell, the cell has a third priority East and North cells have fourth priority as in Figure 10. If the robot was facing the East, the East cells have a first priority, South cells have second priority, the North has a third priority cells, and cells West has fourth priority.

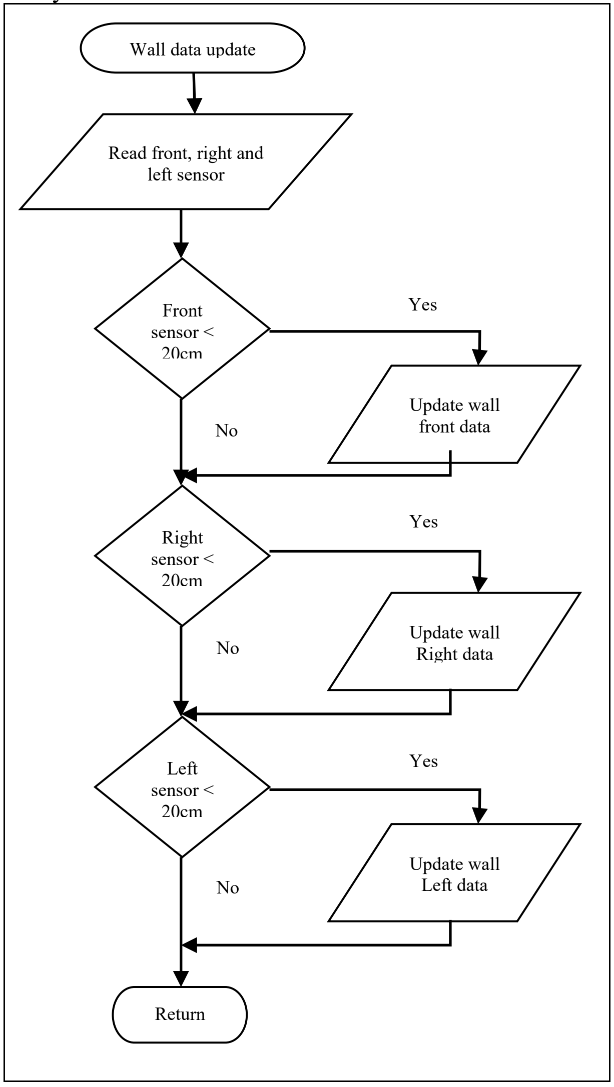

Figure 9. Flowchart for updating wall location at each cell

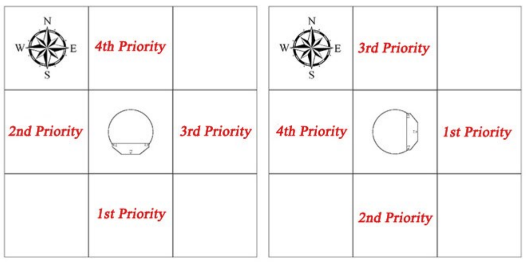

Figure 10. Priority of Neighbour cell

D. Moving to the Smallest Neighbour Cell

Subroutine moves to the smallest neighboring cells function to move the robot towards neighboring cells which have the smallest value, after the robot finds the neighboring cells. To perform movement to the cell, the robot should know the location of the cell. Furthermore, the robot will move to the cells by observing the orientation. For example, if the South cell is the smallest cell and orientation of the robot was facing west, then to move to the position of the cell, the robot must be turning left, then move forward as in Figure 11. If the South cell is the smallest cell and robot orientation was facing East, then to move to the position of the cell, the robot must be spinning right, then move forward.

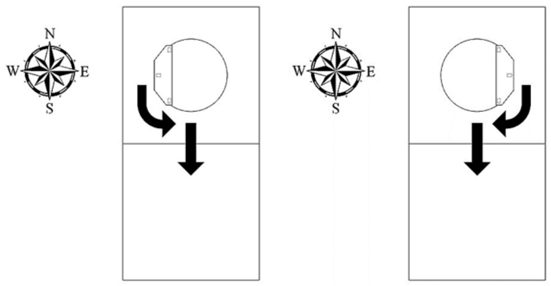

Figure 11. Moving to smallest neighbour cell.

I. Results and Discussion

In this experiments, Robot will learn to find the shortest path from the starting cell (line 4, column 0) to the destination cell (row 2, column 2) and then back again to the initial cell. The initial orientation of the robot is facing the North.

The maze simulator program aims to facilitate the observation on how the flood fill algorithm. Figure 12 is a view maze simulator program. Maze blue wall is a wall that position known to the robot. While the maze walls are colored orange wall position is not known by the robot.

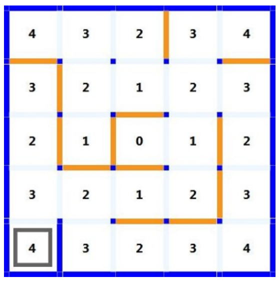

Figure 12. Simulation search path to cell (2,2)

Robot will perform a search of the initial cell lines (4.0) to the destination cell (2, 2). Flood fill algorithm simulation results when a search of the cell lines (4, 0) to the cell (2, 2) are shown in Figure 12 to 22.

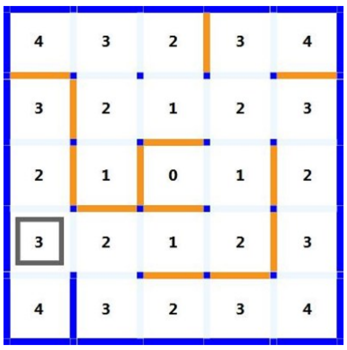

Figure 13. Simulation search path to cell (2,2)

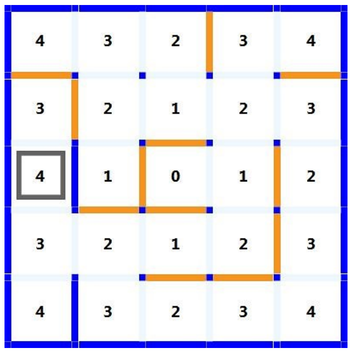

Figure 14. Simulation search path to cell (2,2)

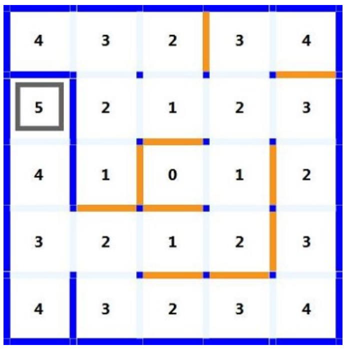

Figure 15. Simulation search path to cell (2,2)

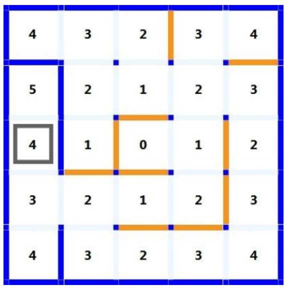

Figure 16. Simulation search path to cell (2,2)

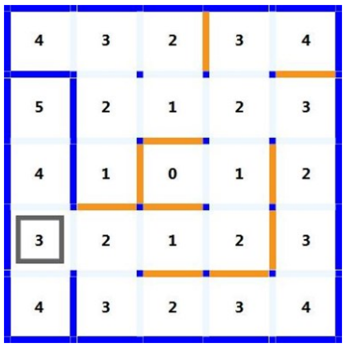

Figure 17. Simulation search path to cell (2,2)

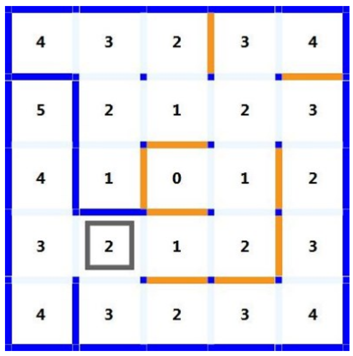

Figure 18. Simulation search path to cell (2,2)

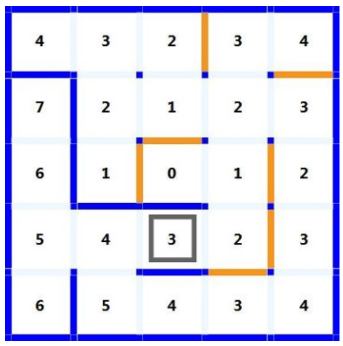

Figure 19. Simulation search path to cell (2,2)

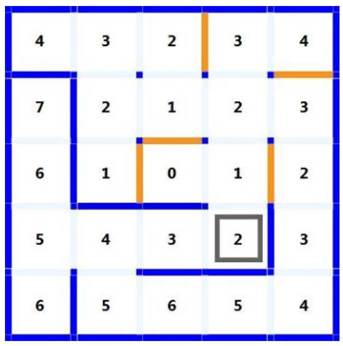

Figure 20. Simulation search path to cell (2,2)

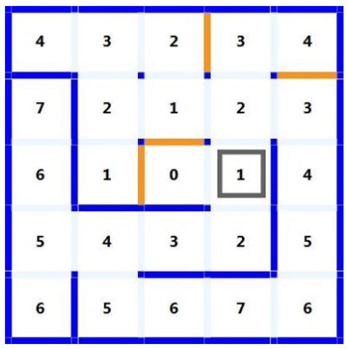

Figure 21. Simulation search path to cell (2,2)

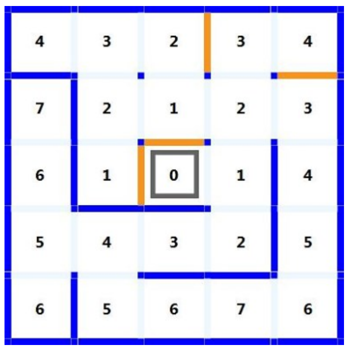

Figure 22. Simulation search path to cell (2,2)

After robot run the search and update his wall data, then it knows the shortest path to go to cell (2,2). It is shown in table 2.

TABLE II. FIRST AND SECOND ROUTES OF ROBOT EXPERIMENT

	Routes	**Number of steps**
First run	(4,0) →(3,0) → (2,0) → (1,0) → (2,0) → (3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	10
Return home	(2,2) → (2,3) → (3,3) → (3,2) → (3,1) → (3,0) → (4,0)	6
Second run	(4,0) →(3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	6

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 foward 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.

V. CONCLUSION

This design and implementation of the robot is a study about the ability to equip a small mobile robot with the ability to learn how to navigate in unknown environment based on its own decisions. The flood-fill algorithm was found to be an effective tool for maze-solving of a moderate size. For the robot to make its decisions it relies on inputs from several sensors, namely the ultrasonic range sensors and wheel rotation decoders.

The robot has successfully able to map the maze in the first, return home and second runs. In its second run it reaches its target cell through the shortest route it has mapped in the previous first run and return home.

Future works may include to studying the robot's maze solving capability in a bigger and more complex maze. In order to improve the quality in wall detection, better object sensor, such as a laser range finder, is needed. It is much more costly but it have ability to scan its surrounding at a wirde angle plane, so it will help a lot in search ability at bigger and more complex maze.

REFERENCES

[1] Bekti, Samudra Harapan. *Pencarian Shortest Path Dinamik dengan Algoritma Bellman Based Flood Fill dan Implementasinya pada Robot Micromouse*: Institut Teknologi Bandung. 2009. 1

[2] Elshamarka, Ibrahim danAbu Bakar Sayuti Saman. *Design and Implementation of a Robot for Maze-Solving using Flood-Fill Algorithm*: Universiti Teknologi Petronas. 2012. 2

[3] I. Elshamarka, And A. B. S. Saman, "Design and Implementation of a Robot for Maze-Solving Using Flood-Fill Algorithm", in International Journal of Computer Applications Volume 56-No.5, pp.8-13, October 2012. 3

[4] A. Ansari, M. A. Sayyed, K. Ratlamwala and P. Shaikh, "An Optimized Hybrid Approach For Path Finding", in International Journal in Foundations of Computer Science & Technology (IJFCST), Vol. 5 No. 2, pp. 47-58, March 2015. 4

[5] K. Sharma, And C. Munshi, "A Comprehensive and Comparative Study of Maze-Solving Techniques by Implementing Graph Theory", in IOSR Journal of Computer Engineering, Vol. 17, Issue 1, Ver. IV, pp. 24-29, 2015. 5

[6] R. K. Sreekanth, "Artificial Intelligence Algorithms", IOSR Journal of Computer Engineering (IOSRJCE), volume 6, issue 3 September-October, 2012. 6

[7] Cook, David. *Intermediate Robot Building*. New York: Apress. 2010. 7

[8] Magnusson, Per. *Design of an H-Bridge*. http://axotron.se/index_en.php?page=34, dikunjungi Juni 2014. 8

[9] Mazidi, Muhammad Ali, Sarmad Niami, dan Sepehr Niami. *The AVR Microcontroller and Embedded System*. New Jersey: Prentice Hall. 2011. 9

[10] Braunl, Thomas. *Embedded Robotics*. Berlin: Springer. 2006. 10

[11] Rizqiawan, Arwindra. *Sekilas Rotary Encoder*. http://konversi.wordpress.com/ 2009/06/12/ sekilas-rotary-encoder/, Juni 2014. 11

[12] Scherz, Paul.*Practical Electronics for Inventors*. New York: McGraw-Hill. 2000. 12

[13] G. W. Lucas, *A Tutorial and Elementary Trajectory Model for the Differential Steering System of Robot Wheel Actuators*. http://rossum.sourceforge.net/ papers/DiffSteer/, Juni 2014. 13