



Engineering Journal

All Egypt Micromouse Competition

Organized by E-JUST and IEEE E-JUST SB

In collaboration with EME Innovation Hub-Borg

Team Name: Master Mouse

Team member names & roles:

1st member (Captain): Mostafa Osama - Mechanical

2nd member (Co-captain): Abdullah Amr Mohamed Saleh - Coding

3rd member: Mohannad Ahmed Elsayed Hafez - Coding

4th member: Mohamed Khaled Fathy Masoud - Electrical

5th member: Mariam Ayman Moustafa Mahmoud – Coding

Faculty: Engineering

Governorate: Alexandria

Table of Contents

Abstract.....	4
Introduction	4
A. Team background & Experience.....	4
Strategy.....	5
A. Discussing special used mechanisms	5
I. Time of Flight Sensor (ToF)	5
II. HC-020K Encoder:.....	6
Components.....	7
Design.....	9
A. Analyze design aspects	9
B. Discuss design efficiency and optimization	10
Coding Exempts & Algorithm	11
Discussion.....	13
A. Share your AEMC 2021 journey and experience	13
B. Include your results and findings	13
C. Bugs & Errors, and overcoming them.....	14
Conclusion	14
Acknowledgements.....	14
References	15

Abstract

In any maze competition, excellent Micromouse performance needs good sensing ability, optimized intelligent algorithm and suitable bearable structure in the Micromouse design and manufacture. This journal describes a successful integration of our Micromouse project. Micromouse is a computer controlled autonomous mobile robot that is able to find a predetermined destination when placed in an unknown maze. The goal of this journal is to report our complete design example solution for the described maze in the competition. The software code is written in Python language with over 650 lines listed in three main files. The components, design, and algorithms used will also be highlighted. In addition, we will discuss the challenges faced us while implementing our solution and how we succeeded to overcome them following special techniques and strategies.

Introduction

A. Team background & Experience

All the team members are graduates of STEM school and have a great experience working on Capstone projects and dealing with robotics, coding, and many others. Also, some of the members have participated before in Minesweepers International Competition in which they succeeded to reach the final international qualification. This is in addition to each member's unique background in different programming languages, coding, machine learning, and so many others gained from enrolling in lots of online and offline courses.

Strategy

A. Discussing special used mechanisms

I. Time of Flight Sensor (ToF)

While choosing a certain sensor to measure the distance, it was a must to give a high concern to the accuracy of it; the most famous sensor is the ultrasonic sensor; the ultrasonic can measure the distance accurately, but sometimes gave inaccurate readings for a very small distances, but it still works most accurately when it comes to measure a relatively bigger distance.

ToF (Time of Flight) sensor has a high accuracy, fast response speed, ability to be consistent in ranges of small distances and unaffected by dust or water, and all these advantages are due its working principle. Compared to the ultrasonic sensor, ToF is able to measure distances accurately in short timing and unaffected by humidity, air pressure and temperature, making it suitable for use all the time. ToF sensors are able to measure long distances and ranges accurately with great accuracy and precision; ToF can detect distances near objects of various shapes and sizes. Regarding the safety of ToF sensors, the laser coming out of the ToF sensors can never harm the human eyes as they use low-power infrared laser light, and these sensors reach Class 1 Laser Safety Standard which ensures its safety to human eyes.

Compared to other 3D depth range scanning technology, ToF sensors are relatively much cheaper when compared to them.

Although ToF has so many benefits, it also might have some limitations:

- If there are too many bright surfaces located near the ToF sensor, it might receive too much reflected light, and the sensor only requires the light to be reflected just once for the measurement.
- In corners and concave shapes, light might be reflected more than just one time, causing the sensor to be distorted and give inaccurate readings.

In places that have too much light with high intensity like a sunny day, the sensor can't sometimes be able to detect the light reflected from the object that's needed to measure the distance off it.

II.HC-020K Encoder:

Generally, encoders are some sensors that are used to control the motion of some robots and machines; they can be found in the most of the machines and industries.

Encoders are simple devices that convert motion to an electrical signal that can be read by some type of control devices to use in a motion control system; it gives a signal that provides the ability to determine the position, count, speed or direction; then another control device can use that information to send a command for a particular function; for example:

- They're used in elevators as encoders tell the controller when the car has reached the correct floor. That is, encoder motion feedback to the elevator's controller ensures that elevator doors open level with the floor. Without encoders, you might find yourself climbing in or out of an elevator, rather than simply walking out onto a level floor.
- Encoders can give motion feedback to the robots; this might mean ensuring the robotic welding arms have the correct information to weld in the correct positions.







For HC-020K encoder, it is a module of wide voltage, high resolution, and short response. It is widely applied to the production of such high precision kind of encoders.




In every application, the same process is done; the encoder generates a count that's sent to the controller, which sends a signal to the machine to perform a function.

Encoder Mechanism:

There are different types of technologies that encoders use to create signals, which are mechanical, magnetic, resistive and optical with optical being the most common and used one (the encoder gives a signal based on the interruption of light).

Components

Item	Photo	Price	Shipping link
VL53L0X Time of Flight Sensor “ToF”		190	https://store.fut-electronics.com/products/vl53l0x-time-of-flight-sensor-precision-distance-measurements?_pos=1&_sid=e101799c2&_ss=r
HC-020k Encoder		100	https://ram-e-shop.com/product/speed-measuring-module-encoder-disk-set-for-smart-car-chassis/
12v battery		450	https://ram-e-shop.com/product/battery-li-12v-3000mah/
L298 Motor Driver		55	Owned
Dc Geared Motor + Wheel		80	https://makerselectronics.com/product/dc-geared-motor-with-wheel
N20 Free wheel		25	-----

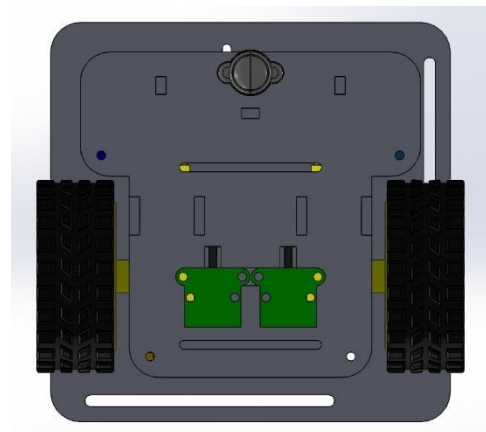
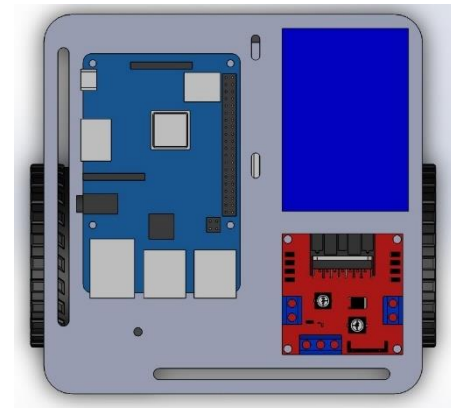
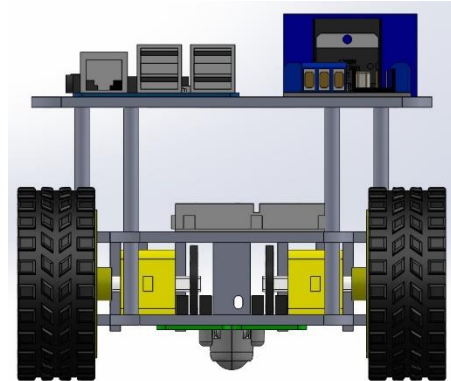
Logic Level Converter		15	https://makerselectronics.com/product/4-channel-logic-level-converter-bi-directional
12v to 5v DC step down		80	https://ram-e-shop.com/product/dc-dc-step-down-converter-variable-3a-24v-12v-to-5v-5a-power-module/
Raspberry Pi model 3 B+		1250	Owned
Total		2270	

Design

A. Analyze design aspects

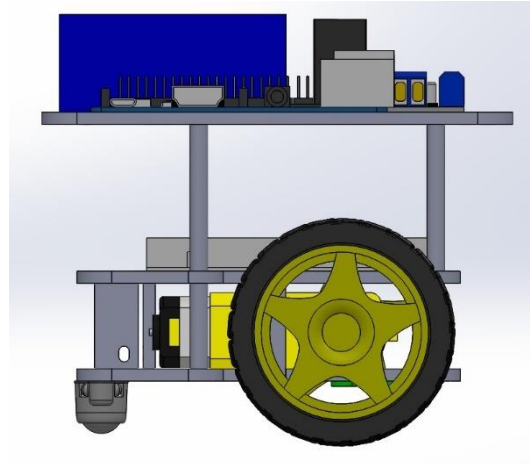
The main challenge in our design was fitting all the components in the recommended dimensions which are 13.5 cm * 13.5 cm, to successfully encounter this challenge we carefully chose all of the components to make sure every single component can be placed in the robot and that its wires can be connected to it and to the power supply and the controller in an organized manner, and of course we made sure that the motors are small to fit in, moreover to exploit as much area as we can we made three plates of wood, the base of the robot containing the motors, encoders and ToF sensor, the mid plate containing the breadboard and the 12v to 5v DC converter, and the top plate containing the raspberry pi, the motor driver and the battery.

Another thing that we had to put into our consideration was the weight as the small motors we chose do not bear that much torque so we had to make sure that the robot weight is light, to do that we chose our 12v 3000mA battery which is about 160 grams, it can deliver 90 minutes of testing and that was sufficient, at last we stuck a symmetric design to make it easier in rounds and turns as the robot will navigate autonomously and a symmetric design with the sensors placed in their best positions will lead to a perfect navigation and that is what we aimed for.



B. Discuss design efficiency and optimization

After finishing our design we performed multiple tests to make sure it is qualified to do its task. Firstly, the robot could bear the weight of all the components while moving fast and smooth which is great for solving the whole maze in a short time period. Secondly, the placing of the sensors was effective to help us explore the maze and navigate through it without any mistake or wrong readings. Lastly, the dimensions of the robot helped us navigate any place in the maze without getting stuck between walls or being forced to touch a wall while performing a rotation or any other problem that may bother us or hold us up from completing our mission.



Maybe AEMC is not the competition where you are asked to build a complicated robotic arm or a sophisticated mechanism but it challenges you to perform your task in the tiniest dimensions you can work in while being as efficient as possible in the field by finishing the needed task in the shortest time. We managed to achieve all our thoughts of the design and even better than we expected and we are thrilled to share our findings with you in our trials and in the presentation.

Coding Exempts & Algorithm

The idea of the algorithm is close to flood-fill algorithm that defines for each cell the distance to the destination. The difference is our algorithm start the flood-fill from the start cell and then return to it and inverse the array values to get a nearly map of the maze.

I. Motion:

The machine can't understand this is a maze or how it's moving, all it knows the array 16x16, its location [y][x] and its set. The array represents the maze, the location represents where the robot stands, the set is how the robot stands either it's facing up, down, left or right. By making a list of the cells surrounding the robot cell and giving the set 4 values from 1 to 4 with each one represents where the robot forward direction is. The next move (decided by the algorithm in next part) is either right "r" or left "l" or forward "f" or dead end "d" each one changes the position x and y and the robot set.

II. Maze Solving:

➤ Maze Exploring:

The initial set for the maze with all cells set to -1 except for the 4 middle cells is set to 0 as they are the destination. As the robot moves it leave a trace behind it by counting number of cells walked from the initial cell.

By reaching a dead end the cell value = -5 as it's a not preferred cell. if the next cell has a cell value less than the count reached the count is reset to the cell value to continue counting from it.

The robot favors the cells with values -1 (undiscovered cell with no trace), if 2 available cells have the value -1, it calculates the shortest to the destination and if they are equal, it randomizes the move. With no available cells with value -1, the robot selects the max value from the given possibilities to reach the end of the trace and start discovering, however, this will lead for loop if no -1 cells have shown, so after it visit the cell 4 times it starts to increase the cell value and search for the minimum value to escape the loop.

➤ Back to Start:

The rules are simple, return to the start with no human interruption, a bonus time is given. And also the time of the return is not calculated, so we should try to return autonomously, at least we can find a new path that shorten the distance.

Same is applied, a cell with -1 is preferred, but the reverse for cells with no -1 cell, the cell with minimum value is preferred to find the end of the trace as the counter this time is decreasing till it reaches the start cell. the cells visited more than 5 times is multiplied and the max cell is reached.

If the counter is decreased it has 2 scenarios, good one it reaches the end with value more than 1 so the new path is shorter than the other, bad one that the counter is 0 and the start cell is not reached, so the robot stops counting and find the shortest way to the 0,0 cell and start the run.

Every time it returns the available cells, it sees the distance to 0,0 and if the distance = 10 or less it prefers this short way rather than discovering new cells.

Walking out from this phase we have a nearly reverse flood fill for the maze array, so we have to inverse it and delete the ways that leads to dead ends to avoid it in the run.

➤ Run:

We need to do 2 things before deciding the path and start to run:

- Inverse: by subtracting the value of the middle cells from all other cells except dead ends (-5) and loop cells (300) it gives them a value of -10 to make the robot avoid choosing these cells. At the end we get a flood fill made by the path taken by the robot.
- Path making: perfect cells that we need are cells that has entrance and exit so it has 2 neighbor cells one with value less than its value and the other with value more than its value. The idea is a loop that take each cell and see its neighbors, if there is no cell with less value or large value, the value of the cell will be set to -10. Recruiting the method many times will eliminate the cells that leads to a dead end and leave the cells that have a clear path

between the start cell and destination. Start cell and destination are excepted from this method to avoid eliminating the whole array. After getting the array with nearly no cells leading to a dead end, the robot just sees the minimum cell available and take the run to the destination. Saving the maze after run to try again with higher speed.

Discussion

A. Share your AEMC 2021 journey and experience

Throughout our journey working to build up our robot, the main challenge for us that was such a great motive making us participate in this competition was the robot's required scale and dimension. The dimension was very challenging as it is relatively small compared to other competitions which makes us eager to work on it thinking about how the needed components will be fit in our robot. This was very useful for us as we got introduced to new smaller motors and encoders that we didn't use before. We also did lots of searching trying to find some new kind of small batteries that is efficient enough to be used during testing our robot without any issues. Also, in this competition, the speed factor is so critical compared to other competitions and this made us search and learn more about different types of sensors (as the ToF) with higher efficiency and less error than the traditional ultrasonic sensor we were always using.

B. Results & Findings

Here, the results of the ToF sensor is discussed as its walk error was somehow different from the known error range. First, we searched and knew that the walk error of ToF value is generated by change of the received signal power depending on distance between the LIDAR sensor and object and that it has a constant error of ~0.05 m obtained after the walk error correction while an increasing error up to ~1 m obtained with conventional method. When calculating our error range, it was from about ~0.04 m to ~1.3 m. Although it wasn't that better, but it worked out doing its mission effectively.

C. Bugs & Errors, and overcoming them

Building up our source code, we faced some bugs and error that needed to be fixed. One of them was the stuck of our robot in some infinity loops in specific cells of the maze. To avoid such a bug, the more visited cell were multiplied by certain values to become larger more and more and then avoid visiting such loops while running the code. Doing so, we faced another critical bug as we got so large values from multiplying the most visited cells to avoid them. Therefore, we calculated the maximum value for these cells to a limit and it was 225 to be not exceed while running the code. This way, the robot will be faster as the code consumes less time and memory which was efficient enough for our mission.

Conclusion

In conclusion, the flood-fill algorithm used in our solution proved its efficiency while testing it several times. All of the required components were successfully fit in a way that efficient enough to achieve our mission. Besides, working on such a competition with its rules adds for us a lot and was such a great experience.

Acknowledgements

We would like to thank our mentor and Eng. Ahmed Essam, IEEE Chairman, for their effort and time helping us a lot during this journey.

References

- 1- Mackorone. (n.d.). *Mackorone/MMS: A micromouse simulator: Write and Test Maze-solving code without a physical robot*. GitHub. Retrieved October 3, 2021, from <https://github.com/mackorone/mms>.
- 2- Administrator. (2018, February 12). *Raspberry Pi L298n interface tutorial: Control a DC motor with l298n and Raspberry Pi*. Electronics Hub. Retrieved October 3, 2021, from <https://www.electronicshub.org/raspberry-pi-l298n-interface-tutorial-control-dc-motor-l298n-raspberry-pi/>.
- 3- McMerkin, M., V, J., Felix, C., J., Paul, 42601, J. V.--, Derek, M, C., Kaan, John, Mysticovl, KonstantinosH, Fernandes, R., Mother, Y., Karina, Kvist, D., Street, O., Emiliano, Neto, P., ... Jay. (2015, November 10). *Using a raspberry pi distance sensor (Ultrasonic Sensor HC-SR04)*. Raspberry Pi Tutorials. Retrieved October 3, 2021, from <https://tutorials-raspberrypi.com/raspberry-pi-ultrasonic-sensor-hc-sr04/>.
- 4- Abdul RehmanAbdul Rehman 25111 gold badge33 silver badges99 bronze badges, NareshNaresh 57911 gold badge55 silver badges1313 bronze badges, IanIan 10.7k11 gold badge2020 silver badges6363 bronze badges, Ben ♦ Ben 4, & Josh Vander HookJosh Vander Hook 5. (1961, July 1). *Self learning maze solving robot using 8bit microcontroller?* Robotics Stack Exchange. Retrieved October 3, 2021, from <https://robotics.stackexchange.com/questions/1060/self-learning-maze-solving-robot-using-8bit-microcontroller/1077#1077>.
- 5- Mertmkarakas, & Instructables. (2017, October 1). *Maze solver with Arduino and ArduMoto(Micromouse) v:1.2*. Instructables. Retrieved October 3, 2021, from <https://www.instructables.com/Maze-Solver-With-Arduino-and-ArduMotoMicromouse/>.