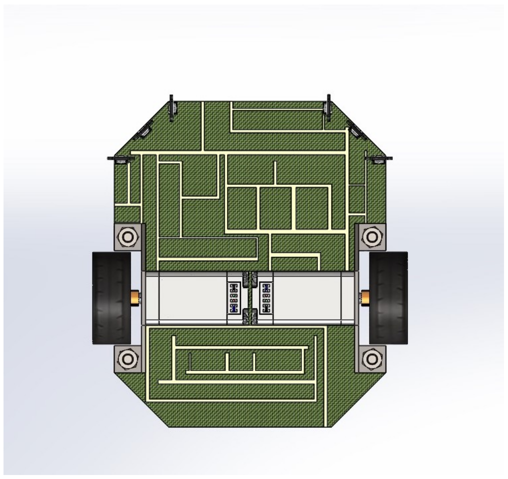
Design Details Document

Sardar Vallabhbhai National Institute of Technology, Surat

The **Maze Solving Robot** also known as Micromouse, which can get to the centre of a maze in the shortest possible time. A micromouse is an autonomous robot which can decode the path on its own to solve the maze successfully. Various algorithms can be used to decode the path and reach the end goal. The **flood fill algorithm** is one of those algorithms and it has been used for better performance and accuracy.

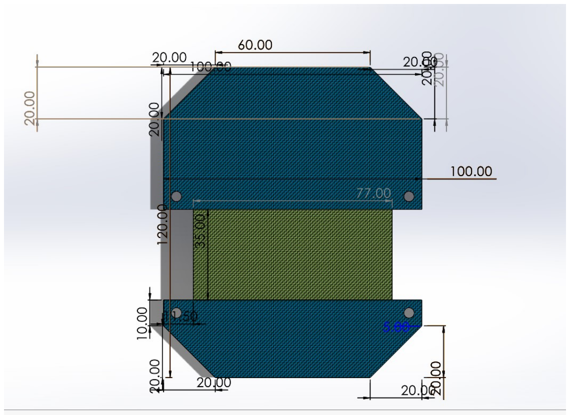
Design of Robot



CAD model of Robot

- **Overall dimension : 12 cm×10 cm×7 cm**
- **Weight : 500 g**

The chassis of our bot is made of **two PCB**, the upper PCB consists of a sensor microcontroller, lower PCB contains two stepper motor drivers and a voltage regulator.



Chassis

1 Actuators and sensors integrated

- **NEMA8 2-Phase Hybrid Stepper Motor:** The motors used are small NEMA 8 size hybrid bipolar stepping motor with 1.8° step angle (200 steps/revolution). Each phase draws 600 mA at 3.6 V, allowing for a holding torque of 180 g-m. These stepper motors only weight 60 g each allowing them to be a good alternative to DC motors.

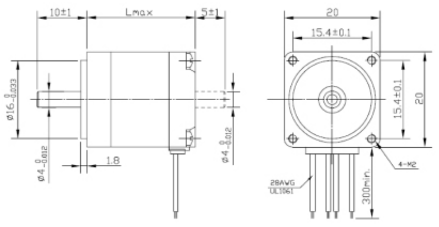
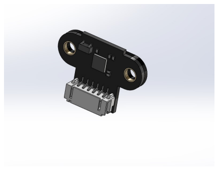


Fig 1: Dimension of motor

- **TOF10120 Lidar:** TOF10120's time-of-flight sensing technology is realized by Sharp's original SPAD (Single Photon Avalanche Diodes) using low-cost standard CMOS process. It enables accurate ranging result, higher immunity to ambient light and better robustness to cover-glass optical cross-talk by special optical package design.



tof10120 lidar

- **ESP-32 WROOM-DA Microcontroller:** Xtensa dual-core 32-bit LX6 microprocessor, operating at 80 MHz to 240 MHz and performing at up to 600 DMIPS and 8 MB flash.

ESP-32 Wroom da

- **DRV8834:** 11-V, 1.5-A, dual H-bridge or stepper motor driver with 1/32 microstep indexer.
- **Lipo Battery:** 3 x 3.7V 180mAH.

2 Motion Control

The driver controls the motor current, the direction of rotation, and rotates the motor according to programmed running modes and input pulses. The time interval between these consecutive input pulses determine the speed of step motors, being the acceleration the rate of change of the velocity in the time interval. Each rising edge of the pulse will rotate the motor one step, according to the direction set by the driver DIR pin. The number of steps required for one full rotation depends on the selected mode (select modes can be from full-step to 32 microsteps/step). Considering Ks a parameter regarding the number of steps required for one full rotation, K_s = MODE × 200 pulse/round, MODE = 1, 2, 4, 8, 16, 32, and d_w the wheel diameter, the velocity and acceleration can be calculated by:

$$v_i = \frac{d_w \pi}{K_s T_i}$$

$$a_i = \frac{v_{i+1} - v_i}{T_i}$$

Combining above equation we can determine the next pulse time interval, T_{i+1}, given the present time interval T_i and the desired acceleration a_i.

$$T_{i+1} = \frac{d_w \pi T_i}{K_s T_i^2 a_i + d_w \pi}$$

3 Type of drive and Odometry

Drive used in robot is diffrentail drive. Signal applied to stepper motors (left and right) will also be used to calculate the position of the micromouse in the maze. Considering , if the robot starts from a position $p(x, y, \theta)$, and the right and left wheels move respectively the linear distances ΔS_R and ΔS_L, the new position $p'(x', y', \theta')$ is given by

$$p' = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{bmatrix}$$

where,

$$\Delta x = \Delta S \cos\left(\theta + \frac{\Delta\theta}{2}\right)$$

$$\Delta y = \Delta S \sin\left(\theta + \frac{\Delta\theta}{2}\right)$$

$$\Delta\theta = \frac{S_R - S_L}{L}$$

and

$$\Delta S = \frac{S_R - S_L}{2}$$

Combined equations will give the odometry equation modelling the micromouse motion,

$$p' = f(x, y, \theta, \Delta S_R, \Delta S_L) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{S_R - S_L}{2} \cos\left(\theta + \frac{S_R - S_L}{2L}\right) \\ \frac{S_R - S_L}{2} \sin\left(\theta + \frac{S_R - S_L}{2L}\right) \\ \frac{S_R - S_L}{L} \end{bmatrix}$$

Considering N_L and N_R the number of pulses used to move the respective left and right stepper motors since last move, ΔS_L and ΔS_R can be calculate from

$$\Delta S_L = N_L \frac{d_w \pi}{K_s}$$

$$\Delta S_R = N_R \frac{d_w \pi}{K_s}$$

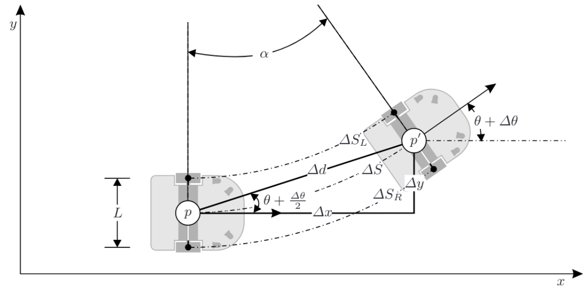


Fig 2: Position calculation of the micromouse in a maze

4 Maze Wall Detection and Algorithm

For wall detection, the micromouse kit uses four tof10120 Lidar. Searching all paths using *LSRB*.
Four lidar sensor are mounted on chassis, two pointing in forward direction and two at an angle of 45°. Lidar will give the distance between the bot and the wall near to it. It will detect weather the wall is present or not.
Two lidar sensor will be arrange at an angle of 90° to control the motion of the bot and be at the centre of the path using control schemes such **PID** control algorithm .
LSRB Algorithm This LSRB algorithm can be simplified into these simple conditions:

- If you can turn left then go ahead and turn left,
- else if you can continue driving straight then drive straight,
- else if you can turn right then turn right.
- If you are at a dead end then turn around.

The travelling is simplifying the register value using the following equation and shorted and follow the value of the register the equations are shown below.
LBR = B
LBS = R
RBL = B
SBL = R
SBS = B
LBL = S
Shortest paths(using Flood Fill algorithm): Based on the phenomenon of pouring the liquid on the plane which spreads evenly in all directions if there are no obstacles. In subsequent steps, increasing values of flooded blocks are increased iteratively until the maze is flooded. Block diagram of the flooding algorithm for the entire maze is shown in Figure 3.

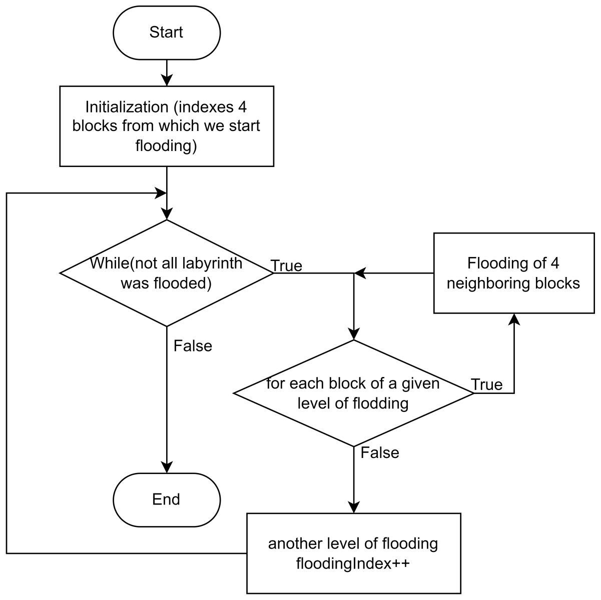

Start
Initialization (indexes 4 blocks from which we start flooding)
While(not all labyrinth was flooded)
True
False
Flooding of 4 neighboring blocks
for each block of a given level of flodding
True
False
End
another level of flooding floodingIndex++

Fig 3: Block Diagram of flooding algorithm for the entire maze

Flood level is reduced for a single block (current step) in order to see if we can pour four neighbouring blocks horizontally and vertically as shown in Figure 4.

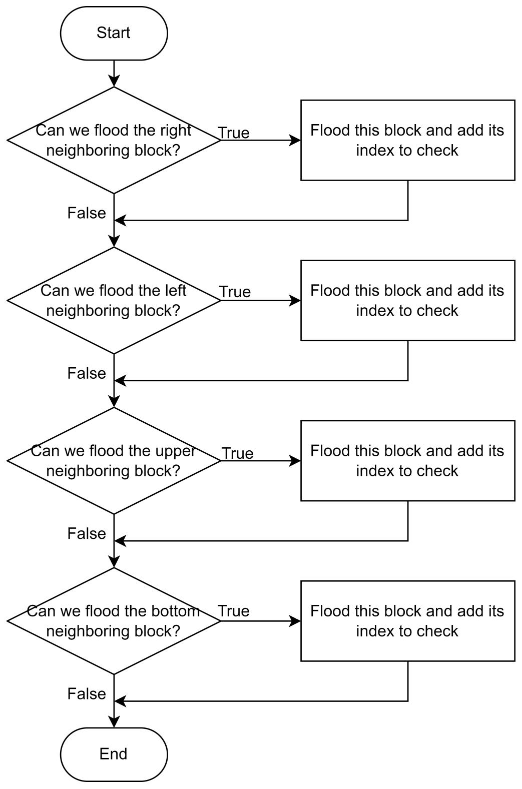

Start
Can we flood the right neighboring block?
True
Flood this block and add its index to check
False
Can we flood the left neighboring block?
True
Flood this block and add its index to check
False
Can we flood the upper neighboring block?
True
Flood this block and add its index to check
False
Can we flood the bottom neighboring block?
True
Flood this block and add its index to check
False
End

Fig 4: Block diagram of flooding function of neighbouring cells

For all maze blocks, if the cells are indexed from 0 to 255, we can easily find neighbouring blocks. The only blocks that are flooded are those that haven't been flooded yet (with values 0) and don't contain walls that prevent liquid from flowing (Figure 4). If a neighbouring block meets these requirements then it is flooded.

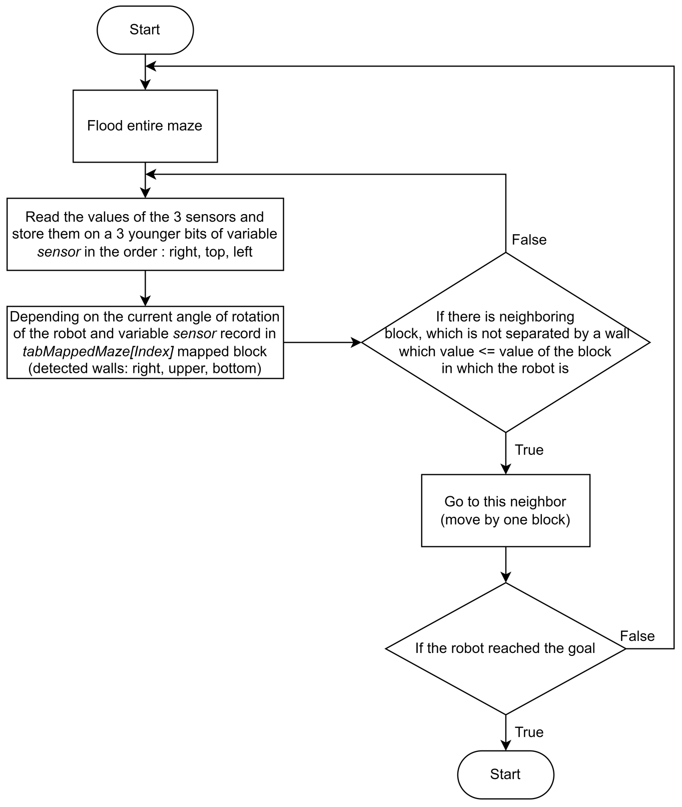

Start
Flood entire maze
Read the values of the 3 sensors and store them on a 3 younger bits of variable *sensor* in the order : right, top, left
Depending on the current angle of rotation of the robot and variable *sensor* record in *tabMappedMaze[Index]* mapped block (detected walls: right, upper, bottom)
If there is neighboring block, which is not separated by a wall which value <= value of the block in which the robot is
False
True
Go to this neighbor (move by one block)
If the robot reached the goal
False
True
Start

Fig 5: Mapping the maze with flood fill algorithm

When the value of a block is greater by one than the value of the block from which it was flooded, its index is stored in an array and checked in the next step of flooding. Using flooding during the mapping and goal-searching algorithm makes the robot quickly heading for the center of the maze. It was assumed that the robot would map the maze until it reached its goal to reduce simulation time. As shown in Figure 5, the robot detects the current cell wall and then checks whether there is a neighbouring block with a lower or equal value where it can go. When all neighbouring blocks have higher values than the block in which the robot is located, a flooding maze begins. With this operation, the current values of the maze blocks are changed, and a new route to the goal is set in the direction of decreasing values.

5 References

1. Grigora S: A Low-Cost, High Performance Micromouse Kit Antonio Valente, Paulo Salgado, and Jos'e BoaventuraCunha

2. A micromouse kit for teaching autonomous mobile robots Juing-Huei Su, Chyi-Shyong Lee, Hsin-Hsiung Huang and Jheng-Yu Huang Department of Electronic Engineering, Lunghwa University of Science and Technology, Taiwan

3. Fault-tolerant algorithm for a mobile robot solving a maze

4. A Study of Point-to-Point Routing Problem in Mazes Der-Cherng Liaw, Chien-Chih Kuo, Hung-Tse Lee

5. International Conference on Innovative Technologies, IN-TECH 2014, Leiria, 10. - 13.09.2014