Robotics Project: MicroMouse Documentation

Course: IL305 Robotics Fall 2024

By Allen, Ankit, Lulu

Goal: Design and construct a robotic “mouse” capable of solving a maze autonomously in the shortest time possible.

Parts List (only listed for the final prototype) :

- 2 mini stepper motors and 2 Mecanum wheels
- 1 motor driver shield
- 1 Arduino board
- 1 L298N motor driver
- 3 Ultrasonic Rangefinder
- Wires
- Fully Customized Chasis for the robot
- Batteries
- Small BreadBoard

Photos of the robot:

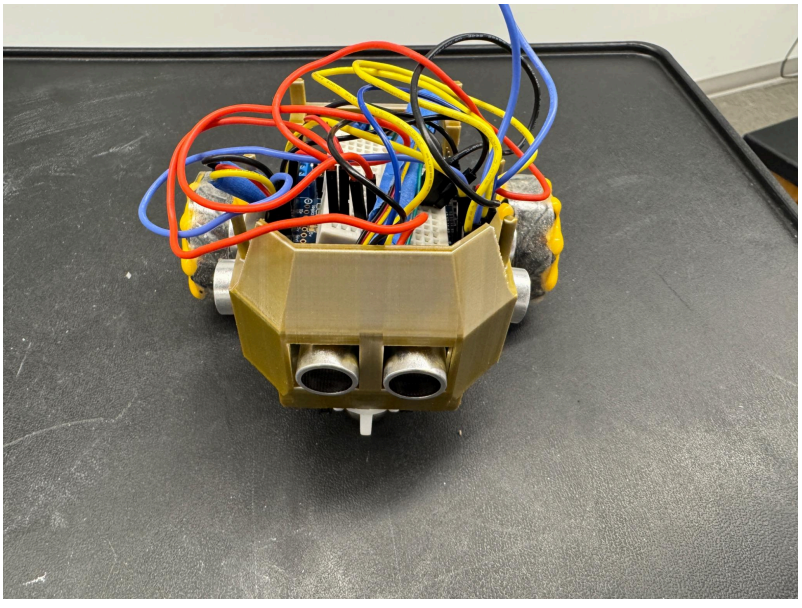


Figure 1: Picture of the robot

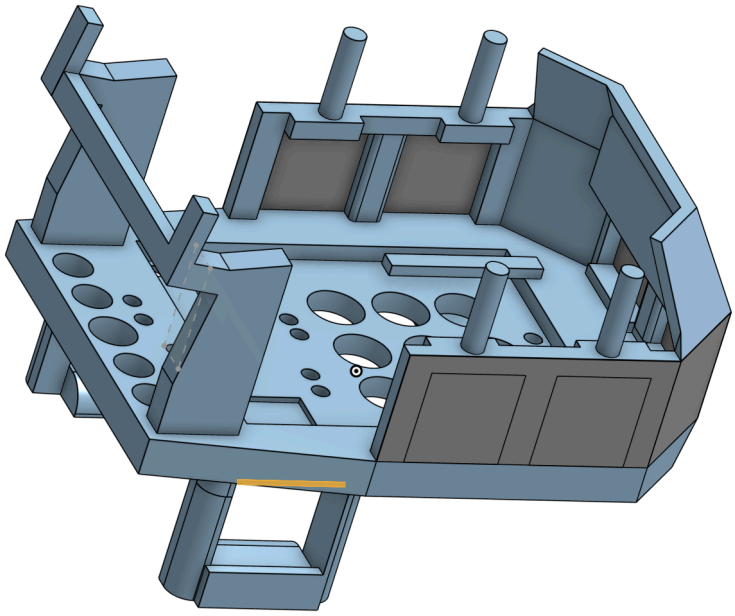


Figure 2: Onshape 3d model of the chasis

Design Procedures & Decisions

1. The customized 3D-printed chassis has been finalized after multiple trials and fine-tunings. It's very challenging to find the perfect fit, but the current design is space-efficient, perfectly accommodating the sensors, motors, and ensuring stable placement for the Arduino, motor driver, and batteries on top. (Took about 4 design changes to find this setup along with multiple changes to dimensions).
2. For the structure, a 2-wheel configuration with stepper motors was chosen after learning from the challenges of a 4-wheel design. 4-wheel design needed two motor shields and separate configuration which made it very difficult to work with. This 2-wheel structure not only provides stable movement but also simplifies coding implementation. Additionally, compared to the 4-wheel prototype, the current design is significantly more space-efficient.

Coding Implement (See Code A for the complete sketch):

Setup

- *The robot initializes the Adafruit Motor Shield and sets up two stepper motors (left and right).*
- *Ultrasonic sensors (front, left, and right) are configured to measure distances, with test readings displayed on the serial monitor for debugging.*
- *Motors are set to a specific speed, and any connection issues are flagged.*

Movement

- *Forward: Both motors move in the same direction to propel the robot forward.*
- *Backward: Motors reverse to move backward.*
- *Turn Left/Right: One motor moves forward while the other moves backward to pivot the robot.*
- *Stop: Motors are released to halt movement immediately.*

Challenges: one challenge we encountered here is that althohugh we can get the distance from walls to the robot to tell its location in each cell, the reading can not tell the angle at a single moment because the angle to the right and left has the same reading. This is solved by letting the adjustment to be extremely frequent (every 5 step for stepper motor), and since the sencer is infront of the wheel, we can avoid the block just before it hits the wall.

Maze Solving

- *The robot reads distances from sensors to evaluate its surroundings:*
    - *Moves forward if the front is clear.*
    - *Turns left or right when obstacles are detected in front.*
    - *Adjusts slightly if walls are too close on the left or right.*
- *Decisions are based on real-time sensor data to navigate and avoid collisions.*

Code A: Complete Micromouse Implement:

```
#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_MS_PWMServoDriver.h"

// ---------------------------------------------------------------------
// Motor Shield and Stepper Motor Setup
// ---------------------------------------------------------------------
Adafruit_MotorShield AFMS = Adafruit_MotorShield();
Adafruit_StepperMotor *leftStepper = AFMS.getStepper(200, 1);
Adafruit_StepperMotor *rightStepper = AFMS.getStepper(200, 2);

// Movement speed (in steps per second)
int stepSpeed = 100;

// ---------------------------------------------------------------------
// Ultrasonic Sensor Pins (Left, Front, Right Configuration)
// ---------------------------------------------------------------------
const int trigPinLeft = 2;
const int echoPinLeft = 3;
const int trigPinFront = 4;
const int echoPinFront = 5;
const int trigPinRight = 6;
const int echoPinRight = 7;

int tempNumberL = 0; // Initialize the temporary counter variable
int tempNumberR = 0;

// ---------------------------------------------------------------------
// Setup routine
// ---------------------------------------------------------------------
void setup() {
Serial.begin(9600);

if (!AFMS.begin()) {
Serial.println("Motor Shield not found. Check your connections.");
while (1);
}

leftStepper->setSpeed(stepSpeed);
rightStepper->setSpeed(stepSpeed);

pinMode(trigPinLeft, OUTPUT);
pinMode(echoPinLeft, INPUT);
pinMode(trigPinFront, OUTPUT);
```

```
pinMode(echoPinFront, INPUT);
pinMode(trigPinRight, OUTPUT);
pinMode(echoPinRight, INPUT);

// Test sensors during setup
Serial.println("Testing sensors...");
for (int i = 0; i < 5; i++) {
long leftDistance = readUltrasonicDistance(trigPinLeft, echoPinLeft);
long frontDistance = readUltrasonicDistance(trigPinFront, echoPinFront);
long rightDistance = readUltrasonicDistance(trigPinRight, echoPinRight);

Serial.print("Left: ");
Serial.print(leftDistance);
Serial.print(" cm, Front: ");
Serial.print(frontDistance);
Serial.print(" cm, Right: ");
Serial.println(rightDistance);
delay(500);
}
Serial.println("Sensor test complete. Starting maze navigation.");

// moveForward(240);
// testWheels();
}

// ---------------------------------------------------------------------------
// Movement Functions
// ---------------------------------------------------------------------------
void moveForward(int steps) {
for (int i = 0; i < steps; i++) {
leftStepper->step(1, FORWARD, SINGLE);
rightStepper->step(1, FORWARD, SINGLE);
}
}

void moveBackward(int steps) {
for (int i = 0; i < steps; i++) {
leftStepper->step(1, BACKWARD, SINGLE);
rightStepper->step(1, BACKWARD, SINGLE);
}
}

void turnLeft(int steps) {
for (int i = 0; i < steps; i++) {
leftStepper->step(1, FORWARD, SINGLE);
```

```
rightStepper->step(1, BACKWARD, SINGLE);
}
}

void turnRight(int steps) {
for (int i = 0; i < steps; i++) {
leftStepper->step(1, BACKWARD, SINGLE);
rightStepper->step(1, FORWARD, SINGLE);
}
}

void stopMotors() {
leftStepper->release();
rightStepper->release();
}

// ---------------------------------------------------------------------
// Ultrasonic Sensor Functions
// ---------------------------------------------------------------------
long readUltrasonicDistance(int trigPin, int echoPin) {
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

long duration = pulseIn(echoPin, HIGH);
return duration * 0.034 / 2; // Convert to cm
}

// ---------------------------------------------------------------------
// Main Navigation Logic
// ---------------------------------------------------------------------
void loop() {
// Read distances from all sensors
long leftDistance = readUltrasonicDistance(trigPinLeft, echoPinLeft);
long frontDistance = readUltrasonicDistance(trigPinFront, echoPinFront);
long rightDistance = readUltrasonicDistance(trigPinRight, echoPinRight);

// Debugging distances
Serial.print("Left: ");
Serial.print(leftDistance);
Serial.print(" cm, Front: ");
Serial.print(frontDistance);
Serial.print(" cm, Right: ");
```

```
Serial.println(rightDistance);

if (frontDistance > 14 && rightDistance >7 && rightDistance <20){
moveForward(115);
turnRight(110);
moveForward(50);

}

// Obstacle detected in front
if (frontDistance <= 1) {
Serial.println("Obstacle detected! Deciding turn...");
// Check for open path to the left
if (leftDistance > 7) {
Serial.println("Turning left...");
turnLeft(110);
// Perform a 90-degree left turn
} else if (rightDistance > 7) {
Serial.println("Turning right...");
turnRight(110);
// Perform a 90-degree right turn
} else {
Serial.println("Turning right...");
turnLeft(220);
// Perform a 90-degree right turn
}
delay(600);
// Short delay to stabilize
return;
}


moveForward(5);


if (leftDistance <= 2) {
turnRight(7);
// Perform a 90-degree right turn
}


if (rightDistance <= 2) {

turnLeft(7);
// Perform a 90-degree right turn
```

```
}

// Short delay to stabilize sensor readings
}

// ---------------------------------------------------------------------
// Test Motors Function
// ---------------------------------------------------------------------
void testWheels() {
Serial.println("Testing forward motion...");
moveForward(100);

delay(1000); // Pause for 1 second

Serial.println("Testing backward motion...");
moveBackward(100);

delay(1000); // Pause for 1 second

Serial.println("Testing left turn...");
turnLeft(250);
delay(1000); // Pause for 1 second

turnRight(250);

delay(1000); // Pause for 1 second

Serial.println("Testing right turn...");
turnRight(250);
delay(1000); // Pause for 1 second

turnLeft(250);

delay(1000); // Pause for 1 second

Serial.println("Wheel test complete.");
}
```

Performance Assessment:

1. Overall integrity and size of the model was good.
2. Was able to Move.
3. Was able to Turn when it detects walls and opening
4. Adjustment to keep the robot in the middle of the way.
5. We will know more as we go about it.

Future Improvements:

1. Floodfill Algorithm: Add Floodfill algorithm to memorize the places it passed, improving maze-solving efficiency
2. Increase Movement Speed: Enhance movement speed by fine-tuning the stepper motor parameters and optimizing the motor control code

References

Robotics_Assignments_Micromouse.pdf (n.d.)

https://thespring.skidmore.edu/d2l/common/viewFile.d2lfile/Database/MTc1ODM2OA/Robotics_Assignments_PIDBalance.pdf?ou=55798

Industries, A. (n.d.). Mini Stepper Motor - 200 steps - 20x30mm NEMA-8 size. adafruit industries blog RSS.

https://www.adafruit.com/product/4411

Industries, A. (n.d.-a). Mini Stepper Motor - 200 steps - 20x30mm NEMA-8 size. adafruit industries blog RSS.

https://www.adafruit.com/product/4411

Failed attempt:

Parts List (only listed for the final prototype) :

- 4 mini stepper motors and 4 mecanum wheels
- 2 motor driver
- 1 arduino board
- 1 L298N motor driver
- screws of various lengths and nuts
- 1 chassis

What we achieved with this attempt

Movement for individual wheels, independent testing for sensors, poor movement result.

Why we stopped:

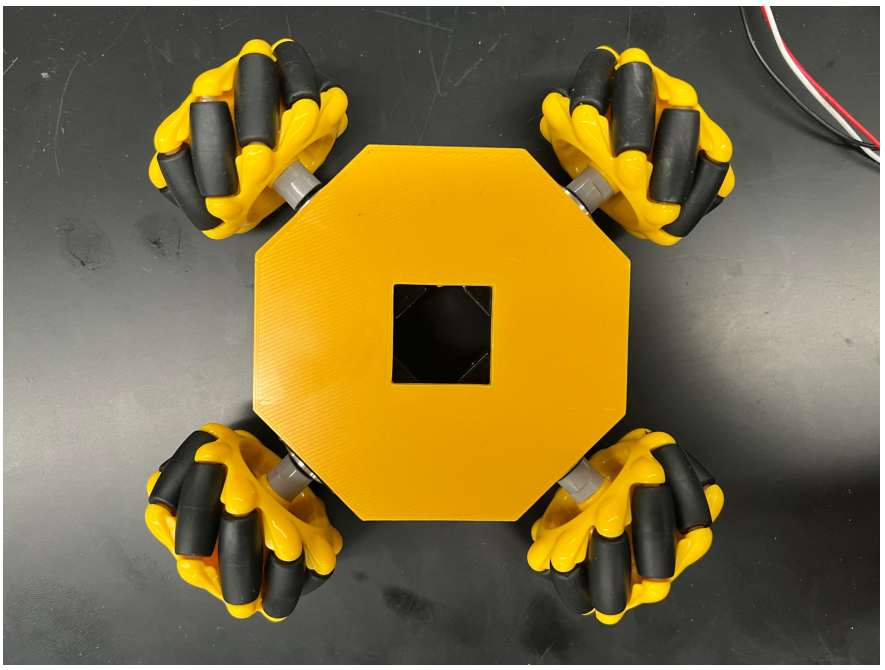
Uncertainty if we can solve the following problems in time:

Unwanted shifting movement: We believe it's caused by the unconventional placement of wheels, and this issue would likely requires more traction and random distribution of wheel traction during movement due to its slight shift.

Part constraints: limited availability of motor shield slows down the testing cycle.

Design Procedures & Decisions

4-wheel Design:



The goal of this design is to be able to use Mecanum Wheels while having a small body to fit inside the maze. We chose the mini stepper motors and the Mecanum Wheels for the flexibility to turn around freely in all directions within the required size limits. We then used Shaper3D to design the wheel holder for stepper motors, leaving gaps for cooling the motor(see Image A). This design is not strong enough and breaks very soon during testing. The second design has more support to prevent breaking.

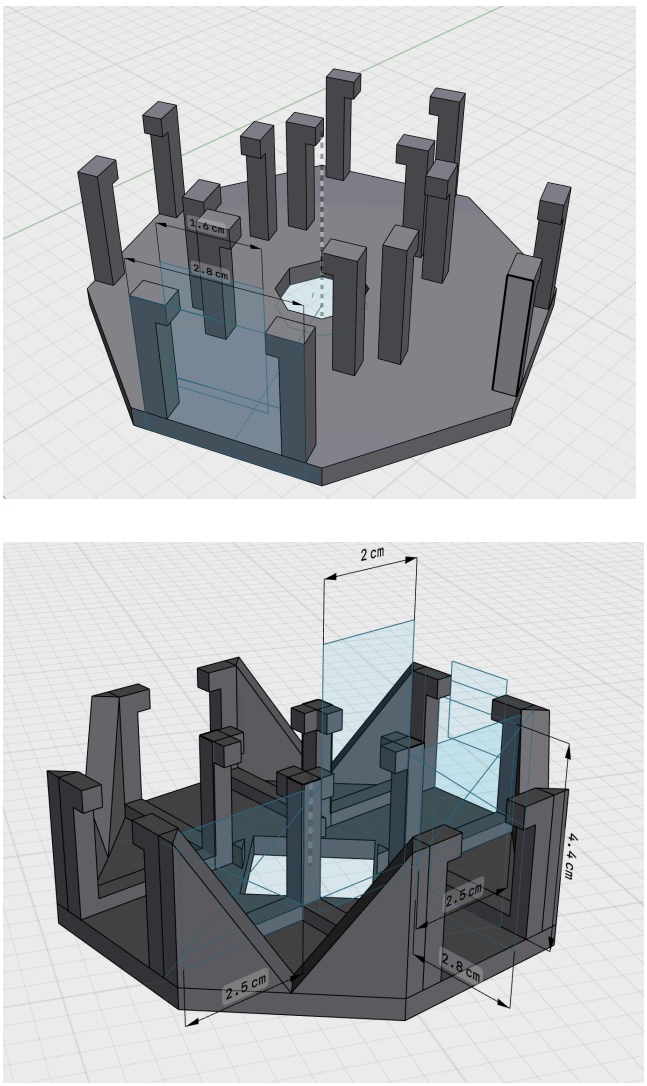


Image A: 3D-printing files of wheel supporter

For the control system, we start by writing codes to control individual stepper motors and ensure the functionality of the sensor (see Code 1, Code 2). We encountered a challenge when trying to control four wheels separately. We ended up changing the I2C address of the second Arduino board from the default 0x60 to 0x61 by soldering the jumpers to represent a new I2C address, then wrote corresponding codes to be able to control each motor separately successfully (see Code 3).

Code Sketches:

Code 1 - Stepper Motor Test:

```
#include <Adafruit_MotorShield.h>
#include <math.h>
#include <Servo.h>

#include <Wire.h>
#include <Adafruit_MotorShield.h>

//Stepper Motor Control

Adafruit_MotorShield AFMS = Adafruit_MotorShield();

Adafruit_StepperMotor *myStepper = AFMS.getStepper(1000, 1); // Port 1
(M1 & M2)

// Setup - Stepper Motors
void setup() {
  Serial.begin(9600);
  Serial.println("Stepper Motor Test!");

  // Initialize the Motor Shield
  if (!AFMS.begin()) {
    Serial.println("Could not find Motor Shield. Check connections.");
    while (1);
  }
  Serial.println("Motor Shield found.");

  // Set the speed in RPM
  myStepper->setSpeed(60); // 60 RPM
}

// Main loop
void loop() {
  Serial.println("Step forward");
  myStepper->step(30000, FORWARD, SINGLE); // steps each circle
  delay(100); // Wait for 1 second

  Serial.println("Step backward");
  myStepper->step(30000, BACKWARD, SINGLE); // Move backward 400 steps

  delay(100); // Wait for 1 second

}
```

Code 2 - Sensor Test:

```
// Pin Definitions
#define TRIG_PIN 10  // Trigger pin connected to pin 10 (yellow wire)
#define ECHO_PIN 9   // Echo pin connected to pin 9 (blue wire)

void setup() {
  Serial.begin(9600);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  Serial.println("Ultrasonic Sensor Initialized");
}

long measureDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // Measure the time for the echo
  long duration = pulseIn(ECHO_PIN, HIGH, 30000); // Timeout at 30ms
(400cm max range)

  if (duration == 0) {
    Serial.println("No echo received. Object may be out of range or
misaligned.");
    return -1; // Indicate an error
  }

  long distance = duration * 0.034 / 2; // Convert time to distance in
cm
  return distance;
}

void loop() {
  long distance = measureDistance();
```

```
  if (distance >= 0) {
    Serial.print("Distance: ");
    Serial.print(distance);
    Serial.println(" cm");
  } else {
    Serial.println("Distance: Out of range or no object detected");
  }

  delay(500);
}
```

Code 3 - Stepper Motor Test:

```
#include <Wire.h>
#include <Adafruit_MotorShield.h>

// Initialize two motor shields
Adafruit_MotorShield AFMS1 = Adafruit_MotorShield(0x61); // Middle shield (0x61)
Adafruit_MotorShield AFMS2 = Adafruit_MotorShield(0x60); // Top shield (0x60)

// Define stepper motors
Adafruit_StepperMotor *motor1 = AFMS1.getStepper(200, 1); // Motor 1 on middle shield (0x61), M1&M2
Adafruit_StepperMotor *motor2 = AFMS1.getStepper(200, 2); // Motor 2 on middle shield (0x61), M3&M4
Adafruit_StepperMotor *motor3 = AFMS2.getStepper(200, 1); // Motor 3 on top shield (0x60), M1&M2
Adafruit_StepperMotor *motor4 = AFMS2.getStepper(200, 2); // Motor 4 on top shield (0x60), M3&M4
```

```
void setup() {
  // Initialize motor shields
  AFMS1.begin(); // Start middle shield (0x61)
  AFMS2.begin(); // Start top shield (0x60)

  // Set stepper motor speed (unit: RPM)
  motor1->setSpeed(60); // Set speed for motor 1
  motor2->setSpeed(60); // Set speed for motor 2
  motor3->setSpeed(60); // Set speed for motor 3
  motor4->setSpeed(60); // Set speed for motor 4
}

void loop() {
  // Move motors forward 100 steps
  motor1->step(500, FORWARD, SINGLE);  // Motor 1 moves forward 100 steps
  motor2->step(500, BACKWARD, SINGLE); // Motor 2 moves backward 100 steps
  motor3->step(500, FORWARD, SINGLE);  // Motor 3 moves forward 100 steps
  motor4->step(500, BACKWARD, SINGLE); // Motor 4 moves backward 100 steps

  delay(100); // Wait for 0.1 second

  // Move motors backward 100 steps
  motor1->step(500, BACKWARD, SINGLE); // Motor 1 moves backward 100 steps
  motor2->step(500, FORWARD, SINGLE);  // Motor 2 moves forward 100 steps
  motor3->step(500, BACKWARD, SINGLE); // Motor 3 moves backward 100 steps
  motor4->step(500, FORWARD, SINGLE);  // Motor 4 moves forward 100 steps

  delay(100); // Wait for 0.1 second
```

}

References

Robotics_Assignments_Micromouse.pdf (n.d.)

https://thespring.skidmore.edu/d2l/common/viewFile.d2lfile/Database/MTc1ODM2OA/Robotics_Assignments_PIDBalance.pdf?ou=55798