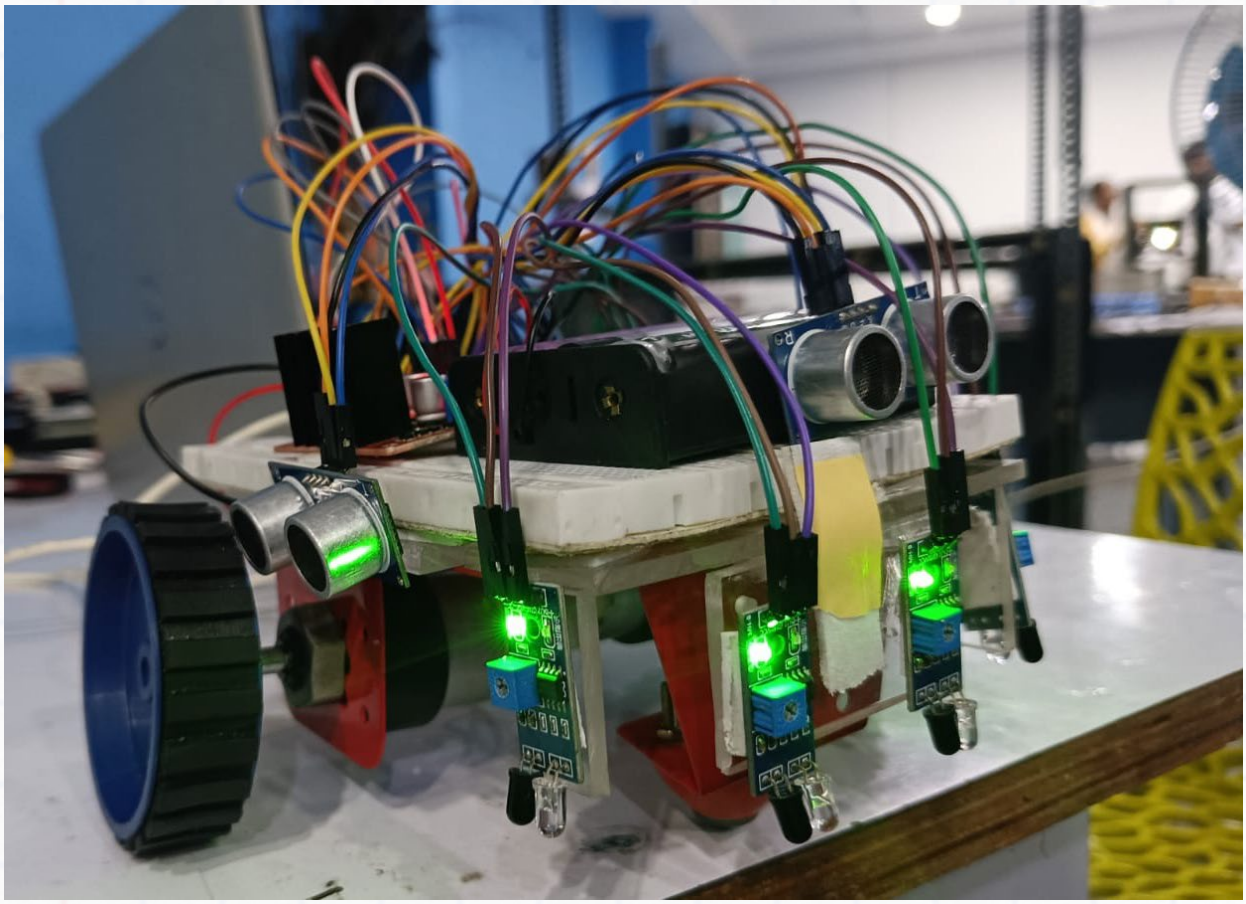
Build an autonomous robot that can navigate a maze



Maze Robot

Index

Prerequisites

Aim

Components

Part 1: Experimenting with the Sensors & Motors

1) IR Sensors

2) Ultrasonic Sensors

3) DC Motors

Part 2: Build a Maze Solving Robot

- Code

- Explanation

Challenges

Prerequisites

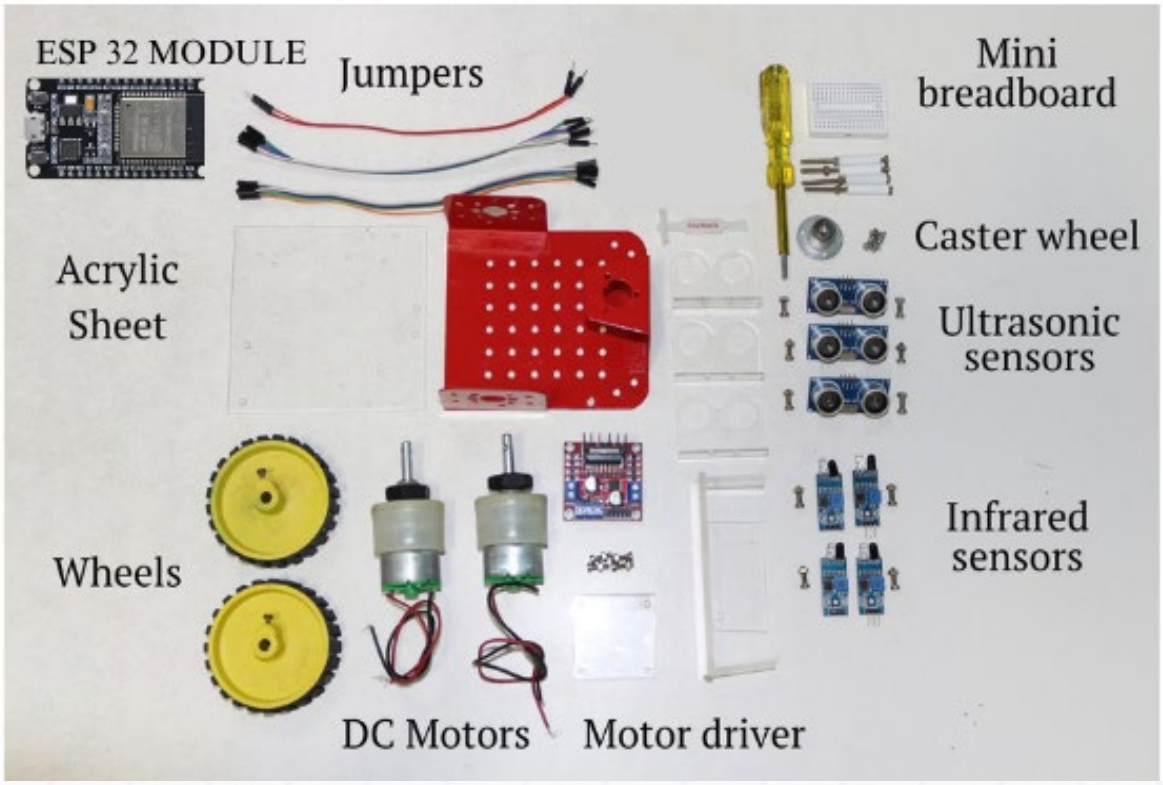
Topic	Resources
Rolling Display Project	Build Club website
Understanding Sensors & Actuators	Project Videos

Aim

Build a Micro-mouse robot equipped with sensors and wheels which can autonomously navigate and solve a maze.

Components

1. Esp32 Module
2. IR Sensors - 4
3. Ultrasonic sensors - 3
4. L298 Motor driver
5. 2 DC Motors
6. 2 Wheels
7. Caster wheel
8. Metal Chassis
9. Acrylic sheet cutouts - 8 pieces
10. Screws & nuts
11. Screwdriver
12. Superglue
13. Double sided tape
14. Mini Breadboard (170 point)
15. Jumper wires (male to female - 9 nos.)
16. Two 3.6V NMC cells
17. 1 USB cables - 1 for powering the esp32
18. 3 X 1.5V AA CELL BATTERY CASE HOLDER

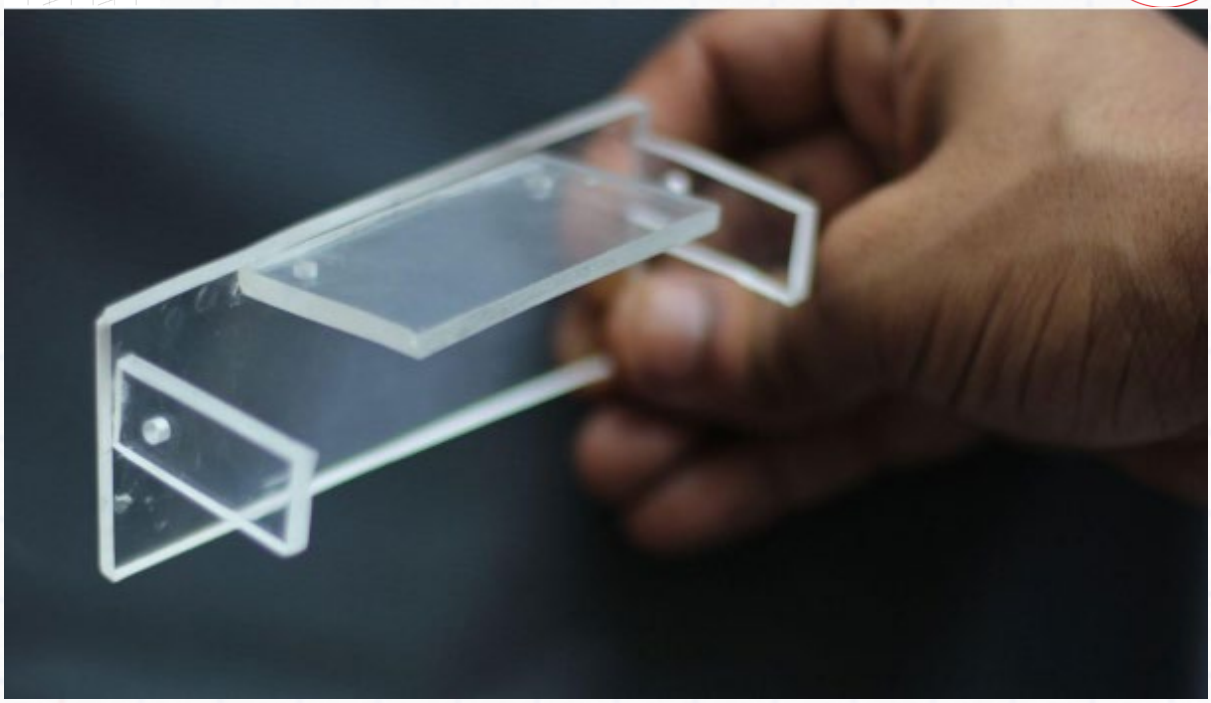


Part 1: Experimenting with the Sensors & Motors

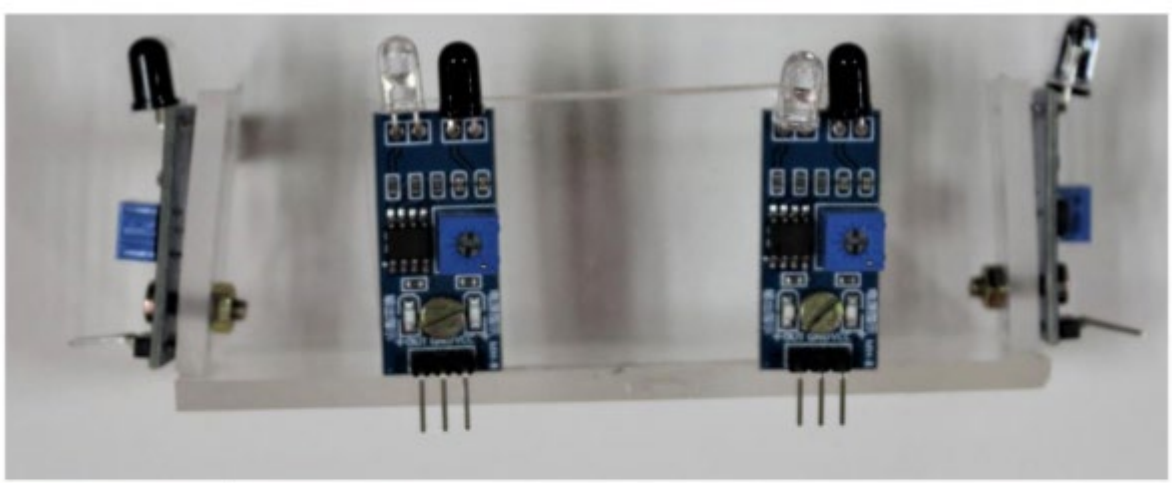
IR Sensors

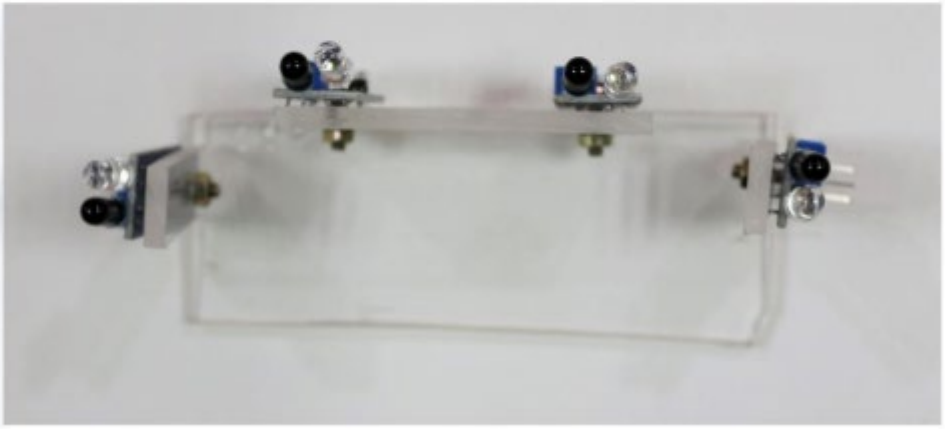
Assembly:

Take the acrylic cutouts for the IR sensors and superglue them to make the following shape:



Next, using 4 sets of screws & nuts, fix the 4 IR sensors on the setup as shown:





IR Sensor

IR Sensors connected on the acrylic structure

Part 1: Experimenting with the Sensors & Motors

Connections:

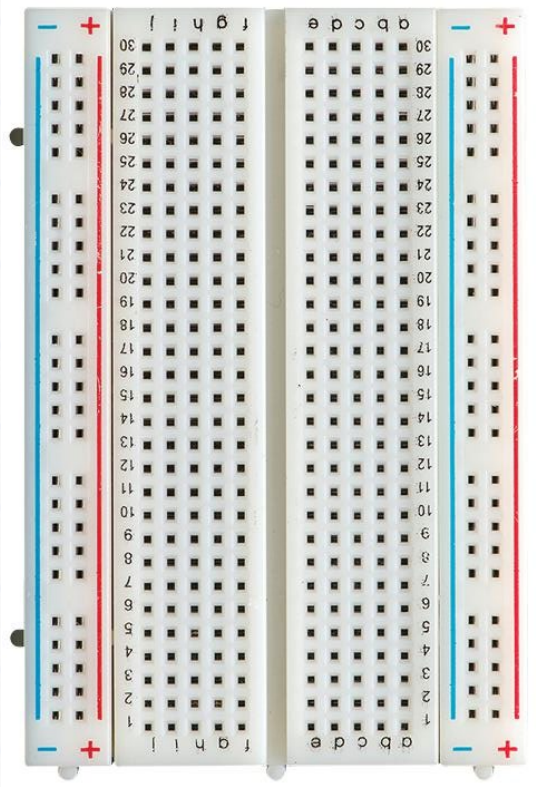
NOTE: Before starting the connections, verify using a multimeter that all the jumper wires are working. Also ensure that the connections are strong, else the setup may not work. Make sure the connections are connected at respective pins.

Take 12 female-to-male jumper wires and connect them to the pins (OUT, GND, VCC) of the 4 IR sensors. Connect the other ends as per the below connections:

Important

Sensor's Pin	Connect to...			
	Left sensor	Left_centre sensor	Right_centre sensor	Right sensor
OUT	25	26	33	27
GND	Breadboard Row R2	Breadboard Row R2	Breadboard Row R2	Breadboard Row R2
VCC	Breadboard Row R1	Breadboard Row R1	Breadboard Row R1	Breadboard Row R1

Take 2 male-to-male jumper wires and connect them as follows:



This documentation provides detailed information on the robot control system code, which includes checking IR sensors, ultrasonic sensors, and controlling DC motors for various movements. The pin connections for each component and the code structure are explained.

Pin Connections

IR Sensors:

- **IR_SENSOR_1:** GPIO 27
- **IR_SENSOR_2:** GPIO 26
- **IR_SENSOR_3:** GPIO 25
- **IR_SENSOR_4:** GPIO 33

Ultrasonic Sensors:

Ultrasonic Sensor 1:

- **TRIG_PIN_1:** GPIO 14
- **ECHO_PIN_1:** GPIO 12

Ultrasonic Sensor 2:
- **TRIG_PIN_2:** GPIO 21
- **ECHO_PIN_2:** GPIO 22

Ultrasonic Sensor 3:
- **TRIG_PIN_3:** GPIO 15
- **ECHO_PIN_3:** GPIO 23

DC Motors:
- **Motor A:**
  - **enA:** GPIO 32
  - **in1:** GPIO 16 (RX2)
  - **in2:** GPIO 17 (TX2)
- **Motor B:**
  - **enB:** GPIO 13
  - **in3:** GPIO 18
  - **in4:** GPIO 19

Breadboard Connections

****IR Sensors:****
1. Connect IR_SENSOR_1 (GPIO 27) to a digital input pin on the breadboard.
2. Connect IR_SENSOR_2 (GPIO 26) to a digital input pin on the breadboard.
3. Connect IR_SENSOR_3 (GPIO 25) to a digital input pin on the breadboard.
4. Connect IR_SENSOR_4 (GPIO 33) to a digital input pin on the breadboard.
5. Ensure each sensor is connected to VCC (3.3V or 5V depending on the sensor specification) and GND.

****Ultrasonic Sensors:****
1. Connect TRIG_PIN_1 (GPIO 14) to the trigger pin of the first ultrasonic sensor.
2. Connect ECHO_PIN_1 (GPIO 12) to the echo pin of the first ultrasonic sensor.
3. Connect TRIG_PIN_2 (GPIO 21) to the trigger pin of the second ultrasonic sensor.
4. Connect ECHO_PIN_2 (GPIO 22) to the echo pin of the second ultrasonic sensor.
5. Connect TRIG_PIN_3 (GPIO 15) to the trigger pin of the third ultrasonic sensor.
6. Connect ECHO_PIN_3 (GPIO 23) to the echo pin of the third ultrasonic sensor.

7. Ensure each sensor is connected to VCC (5V) and GND.

DC Motors:
1. Connect enA (GPIO 32) to the enable pin of Motor A on the motor driver.
2. Connect in1 (GPIO 16) and in2 (GPIO 17) to the input pins of Motor A on the motor driver.
3. Connect enB (GPIO 13) to the enable pin of Motor B on the motor driver.
4. Connect in3 (GPIO 18) and in4 (GPIO 19) to the input pins of Motor B on the motor driver.
5. Connect the motor driver to an appropriate power supply and GND.

Code Structure

IR Sensor Check:

The IR sensor check code initialises the IR sensors as inputs and continuously reads their values, printing them to the Serial Monitor.

Code Explanation

Pin Definitions:

- **const int IR_SENSOR_1 = 27;** defines the GPIO pin number for the first IR sensor.
- **const int IR_SENSOR_2 = 26;** defines the GPIO pin number for the second IR sensor.
- **const int IR_SENSOR_3 = 25;** defines the GPIO pin number for the third IR sensor.
- **const int IR_SENSOR_4 = 33**; defines the GPIO pin number for the fourth IR sensor.

Setup Function:

- **Serial.begin(115200);** initializes the serial communication at 115200 baud rate, which is used for debugging.
- **pinMode(IR_SENSOR_1, INPUT);** sets the pin connected to the first IR sensor as an input.
- **pinMode(IR_SENSOR_2, INPUT);** sets the pin connected to the second IR sensor as an input.
- **pinMode(IR_SENSOR_3, INPUT);** sets the pin connected to the third IR sensor as an input.
- **pinMode(IR_SENSOR_4, INPUT);** sets the pin connected to the fourth IR sensor as an input.

Loop Function:

- **int sensorValue1 = digitalRead(IR_SENSOR_1);** reads the digital value from the first IR sensor (HIGH or LOW).
- **int sensorValue2 = digitalRead(IR_SENSOR_2);** reads the digital value from the second IR sensor (HIGH or LOW).
- **int sensorValue3 = digitalRead(IR_SENSOR_3);** reads the digital value from the third IR sensor (HIGH or LOW).
- **int sensorValue4 = digitalRead(IR_SENSOR_4);** reads the digital value from the fourth IR sensor (HIGH or LOW).

Serial Printing:

- **Serial.print("Sensor 1: ")**; prints the label "Sensor 1: " to the Serial Monitor.
- **Serial.print(sensorValue1);** prints the value of the first IR sensor.
- **Serial.print("\t");** prints a tab character for formatting.
- **Serial.print("Sensor 2: ");** prints the label "Sensor 2: " to the Serial Monitor.
- **Serial.print(sensorValue2)**; prints the value of the second IR sensor.
- **Serial.print("\t");** prints a tab character for formatting.
- **Serial.print("Sensor 3: ");** prints the label "Sensor 3: " to the Serial Monitor.
- **Serial.print(sensorValue3);** prints the value of the third IR sensor.
- **Serial.print("\t");** prints a tab character for formatting.
- **Serial.print("Sensor 4: ");** prints the label "Sensor 4: " to the Serial Monitor.
- **Serial.println(sensorValue4);** prints the value of the fourth IR sensor and moves to the next line.

Delay:

- **delay(500);** introduces a delay of 500 milliseconds before the next iteration of the loop.

Create a checkIRsensor.ino file and copy the below code and paste it on the created file.

```
// Define the pin numbers for the IR sensors
const int IR_SENSOR_1 = 27;  // Pin connected to IR sensor 1
const int IR_SENSOR_2 = 26;  // Pin connected to IR sensor 2
const int IR_SENSOR_3 = 25;  // Pin connected to IR sensor 3
const int IR_SENSOR_4 = 33;  // Pin connected to IR sensor 4

void setup() {
  // Initialize serial communication at 115200 baud
  Serial.begin(115200);  // This allows for communication between the ESP32 and the computer for debugging

  // Set the IR sensor pins as input
  pinMode(IR_SENSOR_1, INPUT);  // Configure pin 27 as an input
//write the pinMode for other IR_SENSOR pins accordingly
}

void loop() {
  // Read the values from the IR sensors
  int sensorValue1 = digitalRead(IR_SENSOR_1);  // Read the digital value from sensor 1 (either HIGH or LOW)
  int sensorValue2 = digitalRead(IR_SENSOR_2);  // Read the digital value from sensor 2 (either HIGH or LOW)
  int sensorValue3 = digitalRead(IR_SENSOR_3);  // Read the digital value from sensor 3 (either HIGH or LOW)
  int sensorValue4 = digitalRead(IR_SENSOR_4);  // Read the digital value from sensor 4 (either HIGH or LOW)

  // Print the sensor values to the Serial Monitor
```

```
  Serial.print("Sensor 1: ");  // Print the label "Sensor 1: " to the Serial Monitor
  Serial.print(sensorValue1);  // Print the value of sensor 1
  Serial.print("\t");        // Print a tab character for formatting

  Serial.print("Sensor 2: ");  // Print the label "Sensor 2: " to the Serial Monitor
  Serial.print(sensorValue2);  // Print the value of sensor 2
  Serial.print("\t");        // Print a tab character for formatting

  Serial.print("Sensor 3: ");  // Print the label "Sensor 3: " to the Serial Monitor
  Serial.print(sensorValue3);  // Print the value of sensor 3
  Serial.print("\t");        // Print a tab character for formatting

  Serial.print("Sensor 4: ");  // Print the label "Sensor 4: " to the Serial Monitor
  Serial.println(sensorValue4); // Print the value of sensor 4 and move to the next
line

  // Delay for a short period of time before reading the sensors again
  delay(500);  // Wait for 500 milliseconds before the next loop iteration
}
```

Ultrasonic Sensor Check:

The ultrasonic sensor check code initialises the sensors, measures distances, and prints the values to the Serial Monitor.

Code Explanation

Pin Definitions:

- **const int TRIG_PIN_1 = 14;** defines the GPIO pin number for the trig pin of the first ultrasonic sensor.
- **const int ECHO_PIN_1 = 12;** defines the GPIO pin number for the echo pin of the first ultrasonic sensor.
- Similar definitions are provided for the second and third ultrasonic sensors.

Maximum Distance Definition:

- **const long MAX_DISTANCE = 10;** sets the maximum distance (in centimeters) for the sensors to be considered "in range".

Distance Measurement Function:

- **long measureDistance(int trigPin, int echoPin)** is a function that takes the trig and echo pin numbers as arguments and returns the measured distance.
- Inside this function:
    1. The trig pin is cleared by setting it to LOW for 2 microseconds.
    2. The trig pin is set to HIGH for 10 microseconds to trigger the ultrasonic pulse.
    3. The echo pin is read to get the duration of the returned pulse in microseconds.
    4. The duration is converted to distance using the formula duration * 0.034 / 2.

Setup Function:

- **Serial.begin(115200);** initializes serial communication at a baud rate of 115200 for debugging purposes.
- pinMode is used to set the trig pins as outputs and the echo pins as inputs for all three sensors.

Loop Function:

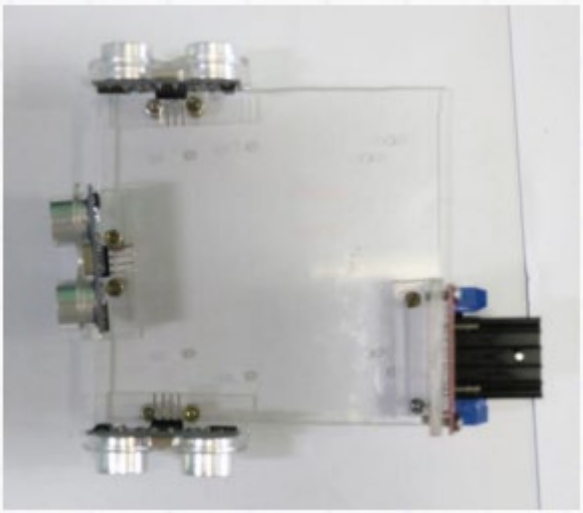
- measureDistance is called for each sensor to get their respective distances.
- The distances are printed to the Serial Monitor.
- If the distance is greater than the maximum distance, "Out of range" is printed.
- There is a 1-second delay before the loop repeats to take the next measurement.
- This code reads the distances from three ultrasonic sensors and prints their values to the Serial Monitor every second.

Create a checkUltraSonicsensor.ino file and copy the below code and paste it on the created file.

```
// Define the pin numbers for the ultrasonic sensors
const int TRIG_PIN_1 = 14;  // Pin connected to the trig pin of the first ultrasonic sensor
const int ECHO_PIN_1 = 12;  // Pin connected to the echo pin of the first ultrasonic sensor

const int TRIG_PIN_2 = 21;  // Pin connected to the trig pin of the second ultrasonic sensor
const int ECHO_PIN_2 = 22;  // Pin connected to the echo pin of the second ultrasonic sensor

const int TRIG_PIN_3 = 15;  // Pin connected to the trig pin of the third ultrasonic sensor
const int ECHO_PIN_3 = 23;  // Pin connected to the echo pin of the third ultrasonic sensor

// Define the maximum distance (in centimeters)
const long MAX_DISTANCE = 10;  // You can set this to the desired maximum distance

// Function to measure distance using an ultrasonic sensor
long measureDistance(int trigPin, int echoPin) {
  // Clear the trigPin
  digitalWrite(trigPin, LOW);  // Set the trig pin to LOW to clear any previous signals
  delayMicroseconds(2);       // Wait for 2 microseconds

  // Trigger the sensor by setting the trigPin high for 10 microseconds
  digitalWrite(trigPin, HIGH);  // Set the trig pin to HIGH to start the ultrasonic pulse
  delayMicroseconds(10);       // Wait for 10 microseconds to send out the pulse
  digitalWrite(trigPin, LOW);   // Set the trig pin to LOW to end the pulse

  // Read the echoPin, which returns the time in microseconds
  long duration = pulseIn(echoPin, HIGH);  // Measure the time the echo pin stays HIGH

  // Calculate the distance in centimeters
  long distance = duration * 0.034 / 2;  // Convert the time into distance

  return distance;  // Return the measured distance
}
```

```
void setup() {
  // Initialize serial communication at 115200 baud
  Serial.begin(115200);  // Start serial communication for debugging

  // Set the ultrasonic sensor pins as output/input
  pinMode(TRIG_PIN_1, OUTPUT);  // Set the trig pin of the first sensor as an output
  pinMode(ECHO_PIN_1, INPUT);   // Set the echo pin of the first sensor as an input

//write your pin mode function for remaining ultra sonic sensors(2 and 3)

}

void loop() {
  // Measure the distance for each ultrasonic sensor
  long distance1 = measureDistance(TRIG_PIN_1, ECHO_PIN_1);  // Get the distance
from the first sensor
  long distance2 = measureDistance(TRIG_PIN_2, ECHO_PIN_2);  // Get the distance
from the second sensor
  long distance3 = measureDistance(TRIG_PIN_3, ECHO_PIN_3);  // Get the distance
from the third sensor

  // Print the distances to the Serial Monitor
  Serial.print("Distance 1: ");  // Print label for the first sensor distance
  if (distance1 > MAX_DISTANCE) {  // Check if the distance is out of range
    Serial.print("Out of range");  // Print "Out of range" if distance exceeds
MAX_DISTANCE
  } else {
    Serial.print(distance1);      // Print the measured distance
    Serial.print(" cm");          // Print the unit "cm"
  }
  Serial.print("\t");  // Print a tab character for formatting

  Serial.print("Distance 2: ");  // Print label for the second sensor distance
  if (distance2 > MAX_DISTANCE) {  // Check if the distance is out of range
    Serial.print("Out of range");  // Print "Out of range" if distance exceeds
MAX_DISTANCE
  } else {
    Serial.print(distance2);      // Print the measured distance
    Serial.print(" cm");          // Print the unit "cm"
  }
```

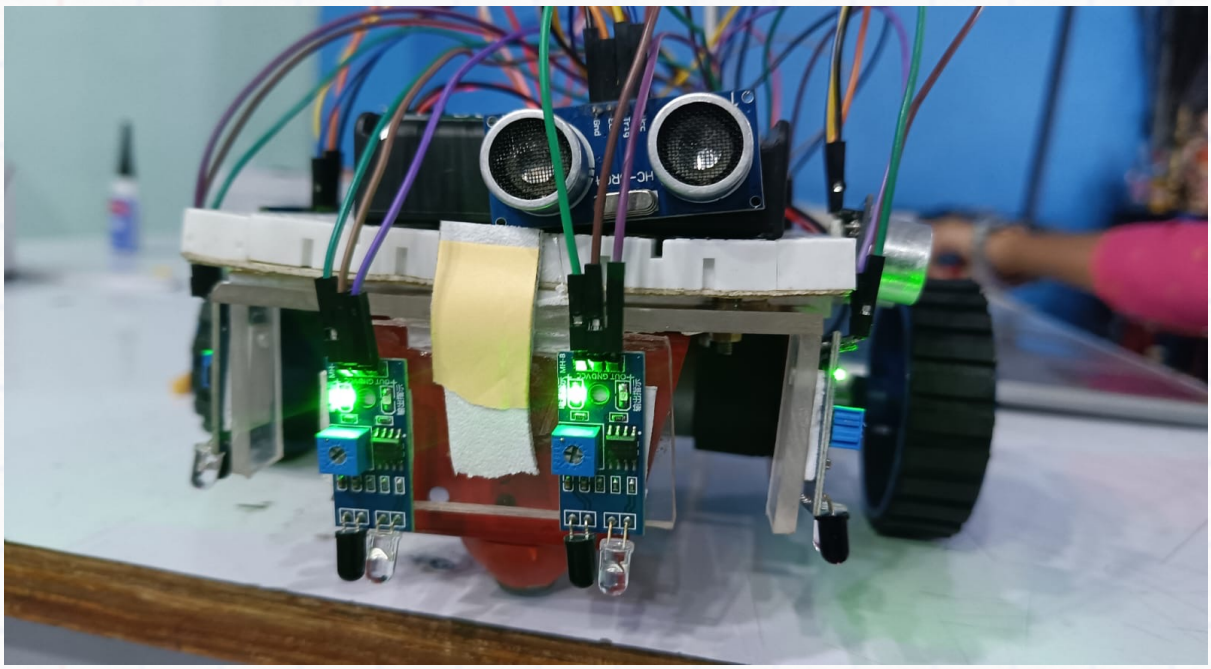
```
  Serial.print("\t");  // Print a tab character for formatting

  Serial.print("Distance 3: ");  // Print label for the third sensor distance
  if (distance3 > MAX_DISTANCE) {  // Check if the distance is out of range
    Serial.print("Out of range");  // Print "Out of range" if distance exceeds
MAX_DISTANCE
  } else {
    Serial.print(distance3);      // Print the measured distance
    Serial.print(" cm");          // Print the unit "cm"
  }

  Serial.println();  // Print a new line

  // Delay before taking the next measurement
  delay(1000);  // Wait for 1000 milliseconds before the next loop iteration
}
```



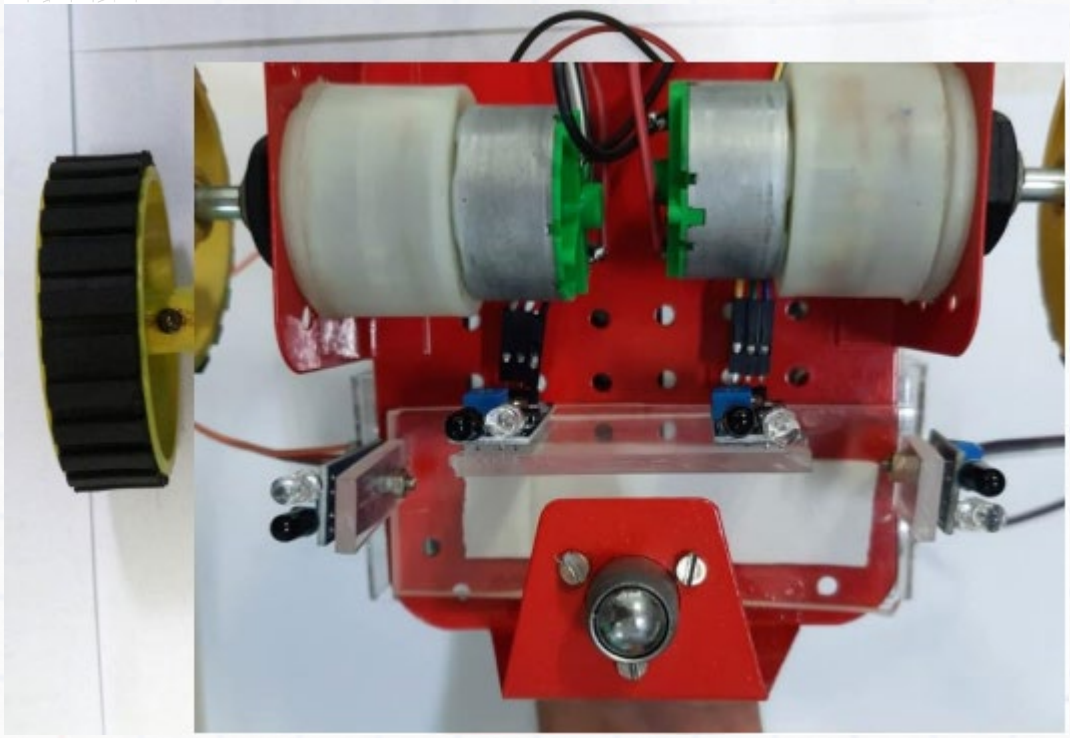
Tasks

Place a 30cm ruler on the floor perpendicular to a wall so that the 0cm mark touches the wall. Keep any one Ultrasonic sensor at the 10cm mark and verify if the reading in Live Expressions is also approximately 10c. Take 3 readings for each sensor (say at 20cm, 30cm, 17cm) to verify that they are working properly. If the distance readings for the sensors are not accurate, go to the top of the app.c file. In the

Calibration Values section, you will find a variable called ultrasonic_calibration set to the value 53. Adjust this value until the readings are accurate (Note: You must test with different distances to ensure accuracy). 2) Keep one of the Ultrasonic sensors at the 20cm mark. Now slowly keep moving the sensor backwards. Note at what distance the readings lose their accuracy. This is the sensor's maximum range of functioning.



Motor Control:



The motor control code provides functions to control the motion of the robot, including moving forward, backward, and turning left or right.

Code Explanation

Pin Definitions:

- **int enA = 32;** defines the PWM pin for Motor A.
- **int in1 = 16;** and int in2 = 17; define the control pins for Motor A.
- Similar definitions are provided for Motor B.

PWM Configuration:

- **int pwmChannelA = 1;** and **int pwmChannelB = 1**; define the PWM channels for Motors A and B (Note: using the same channel for both motors will not work; each motor should have a unique channel).
- **int pwmFrequency = 5000;** sets the PWM frequency to 5000 Hz.
- **int pwmResolution = 8;** sets the PWM resolution to 8 bits.

Setup Function:

- **pinMode** sets the motor control pins as outputs.
- **ledcSetup** configures the PWM channels with the specified frequency and resolution.
- **ledcAttachPin** attaches the PWM channels to the GPIO pins.
- **Serial.begin(9600);** initializes serial communication at 9600 baud.
- **digitalWrite** ensures that the motors are initially turned off.

Loop Function:

The loop function calls the **forwardMotion, backwardMotion, leftMotion,** and **rightMotion** functions in sequence with delays in between.

Motor Control Functions:

forwardMotion, backwardMotion, leftMotion, and rightMotion control the direction and speed of the motors by setting the appropriate control pins and PWM values.

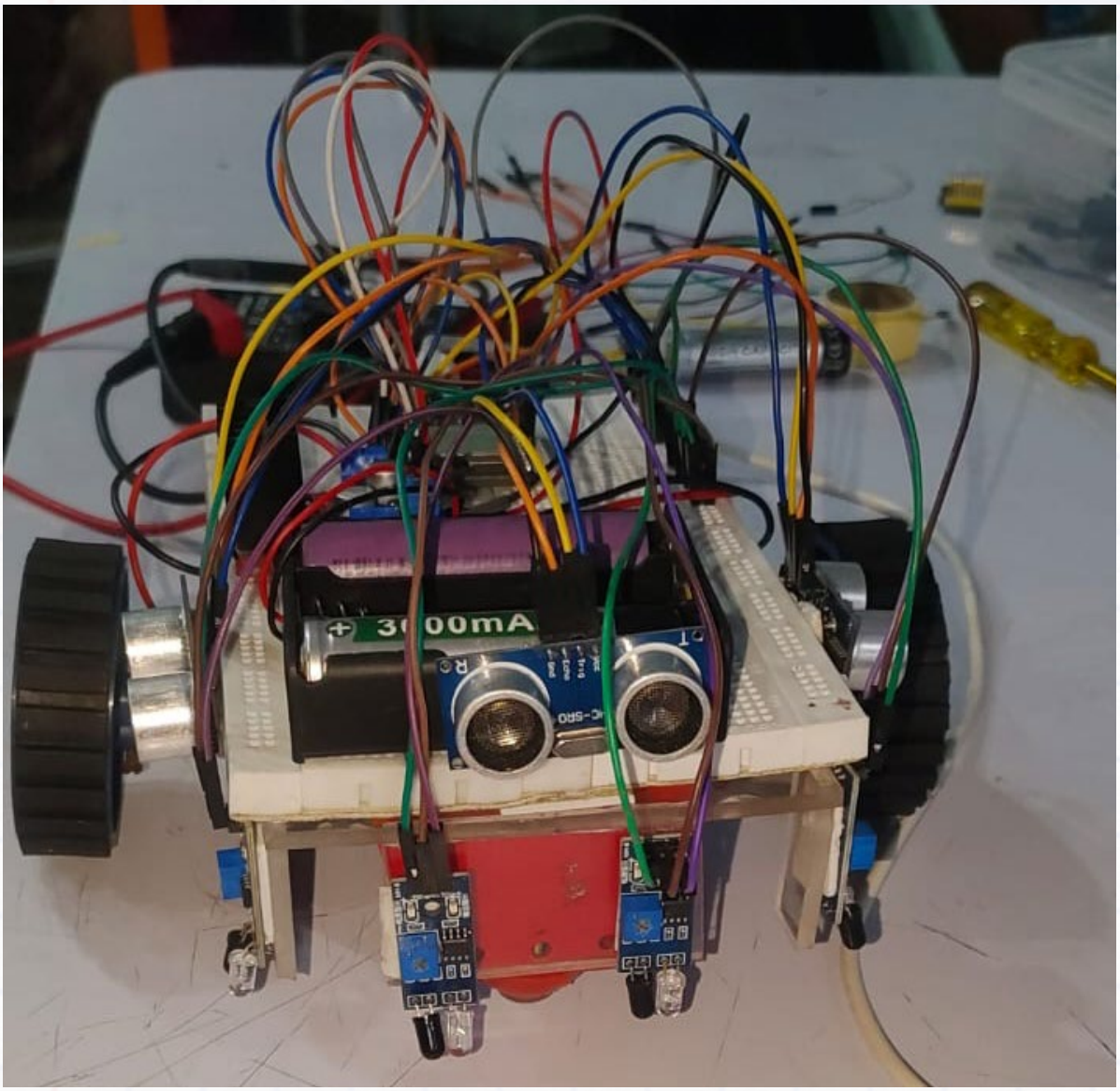
Create a motorCheck.ino file and copy the below code and paste it on your file.

```
// Motor A connections
int enA = 32;  // PWM pin for Motor A
int in1 = 16;  // Control pin 1 for Motor A
int in2 = 17;  // Control pin 2 for Motor A

// Motor B connections
int enB = 13;  // PWM pin for Motor B
int in3 = 18;  // Control pin 1 for Motor B
int in4 = 19;  // Control pin 2 for Motor B

// Define PWM channels
int pwmChannelA = 1;  // PWM channel for Motor A
int pwmChannelB = 1;  // PWM channel for Motor B (Note: using the same channel for both motors will not work)
int pwmFrequency = 5000;  // PWM frequency in Hz
```

```
int pwmResolution = 8;    // PWM resolution in bits

void setup() {
  // Set all the motor control pins to outputs
  pinMode(in1, OUTPUT);  // Set Motor A control pin 1 as output
  pinMode(in2, OUTPUT);  // Set Motor A control pin 2 as output
  pinMode(in3, OUTPUT);  // Set Motor B control pin 1 as output
  pinMode(in4, OUTPUT);  // Set Motor B control pin 2 as output

  // Configure PWM channels
  ledcSetup(pwmChannelA, pwmFrequency, pwmResolution);  // Setup PWM for
Motor A
  ledcSetup(pwmChannelB, pwmFrequency, pwmResolution);  // Setup PWM for
Motor B (Note: using the same channel for both motors will not work)

  // Attach PWM channels to GPIO pins
  ledcAttachPin(enA, pwmChannelA);  // Attach Motor A PWM pin to its PWM
channel
  ledcAttachPin(enB, pwmChannelB);  // Attach Motor B PWM pin to its PWM
channel

  // Start serial communication for debugging
  Serial.begin(9600);  // Initialize serial communication at 9600 baud

  // Turn off motors - Initial state
  digitalWrite(in1, LOW);  // Ensure Motor A is off
  digitalWrite(in2, LOW);  // Ensure Motor A is off
  digitalWrite(in3, LOW);  // Ensure Motor B is off
  digitalWrite(in4, LOW);  // Ensure Motor B is off
}

void loop() {
  // Move forward
  forwardMotion();  // Call function to move forward
  delay(2000);  // Adjust delay as needed


 //call backward function here and add 2 seconds delay
//call left function here and add 2 seconds delay
//call right function here and add 2 seconds delay
```

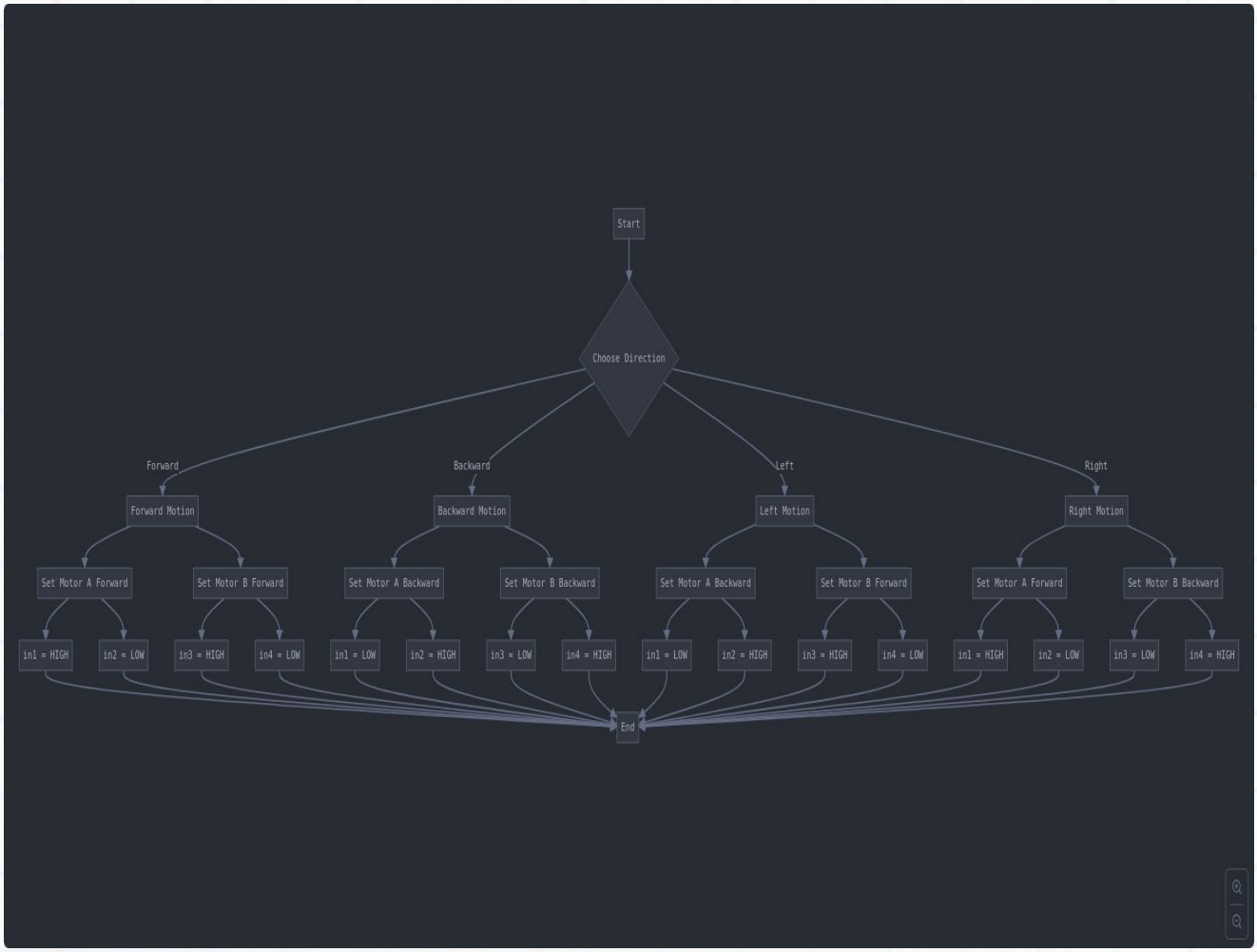
```
}

// Function to move forward
void forwardMotion() {
  // Set Motor A forward
  digitalWrite(in1, HIGH);  // Set Motor A control pin 1 to HIGH
  digitalWrite(in2, LOW);   // Set Motor A control pin 2 to LOW

  // Set Motor B forward
  digitalWrite(in3, HIGH);  // Set Motor B control pin 1 to HIGH
  digitalWrite(in4, LOW);   // Set Motor B control pin 2 to LOW

  // Set speed (0-255)
  ledcWrite(pwmChannelA, 200);  // Set Motor A speed
  ledcWrite(pwmChannelB, 200);  // Set Motor B speed
}

// Function to move backward
void backwardMotion() {
//set Backward action for motor A and B here

// Set speed (0-255)
  ledcWrite(pwmChannelA, 200);  // Set Motor A speed
  ledcWrite(pwmChannelB, 200);  // Set Motor B speed

}

// Function to turn left
void leftMotion() {
 //set Left action for motor A and B here

// Set speed (0-255)
  ledcWrite(pwmChannelA, 200);  // Set Motor A speed
  ledcWrite(pwmChannelB, 200);  // Set Motor B speed

}

// Function to turn right
void rightMotion() {
//set Backward action for motor A and B here
```

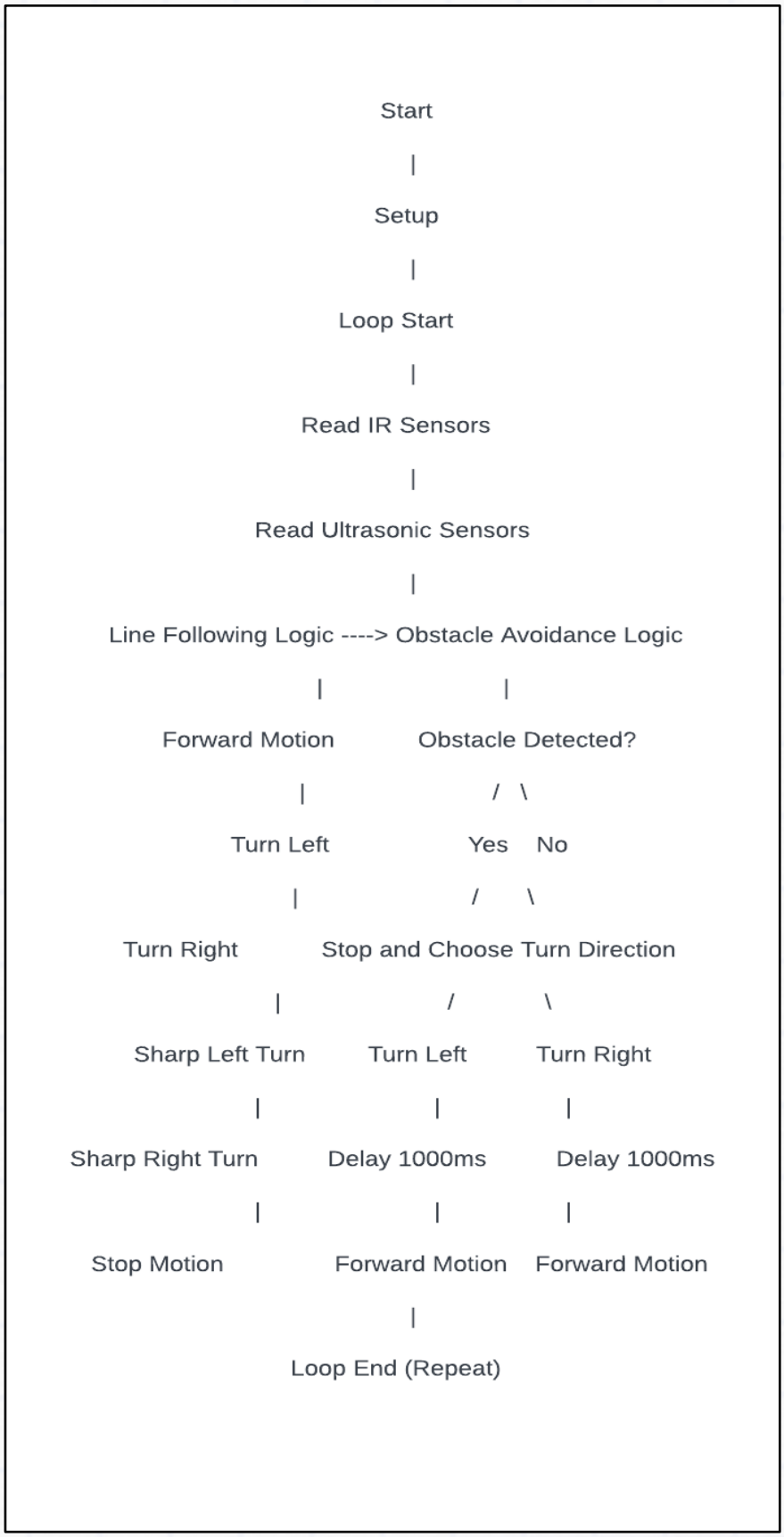
```
// Set speed (0-255)
  ledcWrite(pwmChannelA, 200);  // Set Motor A speed
  ledcWrite(pwmChannelB, 200);  // Set Motor B speed

}
```



Maze Robot

FLOW DIAGRAM




Start
Setup
Loop Start
Read IR Sensors
Read Ultrasonic Sensors
Line Following Logic ----> Obstacle Avoidance Logic
Forward Motion
Obstacle Detected?
Turn Left
Yes
No
Turn Right
Stop and Choose Turn Direction
Sharp Left Turn
Turn Left
Turn Right
Sharp Right Turn
Delay 1000ms
Delay 1000ms
Stop Motion
Forward Motion
Forward Motion
Loop End (Repeat)

After Checking the sensor use below code for maze solver:

Part 2: Build a Maze Solving Robot

Below Tabular column represent the condition to be check on line following logic. Make sure edit the below given code accordingly

Sensor 1	**Sensor 2**	**Sensor 3**	**Sensor 4**	**Action**
LOW	HIGH	HIGH	LOW	Move forward
HIGH	LOW	-	-	Turn left
-	-	LOW	HIGH	Turn right
LOW	LOW	HIGH	HIGH	Sharp left turn
HIGH	HIGH	LOW	LOW	Sharp right turn
-	-	-	-	Stop

TAB-1

Below Tabular column represent the actions of the motor A and B. Use it as a reference and write it the motor control function accordingly.

Function	in1	in2	in3	in4	pwmChannelA	pwmChannelB
forwardMotion	HIGH	LOW	HIGH	LOW	200	200
backwardMotion	LOW	HIGH	LOW	HIGH	200	200
leftMotion	LOW	HIGH	HIGH	LOW	200	200
rightMotion	HIGH	LOW	LOW	HIGH	200	200
stopMotion	LOW	LOW	LOW	LOW	0	0
sharpLeftMotion	LOW	HIGH	HIGH	LOW	255	255

sharpRightMotion	HIGH	LOW	LOW	HIGH	255	255

TAB-2

Create a new sketch in Arduino IDE named ***mazeSolver.ino, Copy*** the below code and paste it on created file, edit with given instructions.

```
// Define the pin numbers for the IR sensors
const int IR_SENSOR_1 = 27;
const int IR_SENSOR_2 = 26;
const int IR_SENSOR_3 = 25;
const int IR_SENSOR_4 = 33;

// Define the pin numbers for the ultrasonic sensors
const int TRIG_PIN_1 = 14;
const int ECHO_PIN_1 = 12;

const int TRIG_PIN_2 = 21;
const int ECHO_PIN_2 = 22;

const int TRIG_PIN_3 = 15;
const int ECHO_PIN_3 = 23;

// Motor A connections
int enA = 32;
int in1 = 16;
int in2 = 17;

// Motor B connections
int enB = 13;
int in3 = 18;
int in4 = 19;

// Define PWM channels
int pwmChannelA = 1;
int pwmChannelB = 2;
int pwmFrequency = 5000;
int pwmResolution = 8;

// Define the maximum distance (in centimeters) for obstacle detection
```

```
const long MAX_DISTANCE = 30;  // Adjust as needed

void setup() {
  // Initialize serial communication
  Serial.begin(115200);

  // Set the IR sensor pins as input
  pinMode(IR_SENSOR_1, INPUT);
  //write the pinMode for other IR_SENSOR pins accordingly

  // Set the ultrasonic sensor pins as output/input
  pinMode(TRIG_PIN_1, OUTPUT);
  pinMode(ECHO_PIN_1, INPUT);

  pinMode(TRIG_PIN_2, OUTPUT);
  pinMode(ECHO_PIN_2, INPUT);

  pinMode(TRIG_PIN_3, OUTPUT);
  pinMode(ECHO_PIN_3, INPUT);

  // Set all the motor control pins to outputs
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

  // Configure PWM channels
  ledcSetup(pwmChannelA, pwmFrequency, pwmResolution);
  ledcSetup(pwmChannelB, pwmFrequency, pwmResolution);

  // Attach PWM channels to GPIO pins
  ledcAttachPin(enA, pwmChannelA);
  ledcAttachPin(enB, pwmChannelB);

  // Start with motors stopped
  stopMotion();
}

void loop() {
  // Read IR sensors
  int sensor1 = digitalRead(IR_SENSOR_1);
```

```
  int sensor2 = digitalRead(IR_SENSOR_2);
  int sensor3 = digitalRead(IR_SENSOR_3);
  int sensor4 = digitalRead(IR_SENSOR_4);

  // Read ultrasonic sensors
  long distanceFront = measureDistance(TRIG_PIN_1, ECHO_PIN_1);
  long distanceLeft = measureDistance(TRIG_PIN_2, ECHO_PIN_2);
  long distanceRight = measureDistance(TRIG_PIN_3, ECHO_PIN_3);


  // Line following logic
  //write a logic for the line follower here with reference of the above tabular
column-1.
  if () {
     } else if () {
     } else if () {
     } else if () {
     } else if () {
     } else {
     }


  // Obstacle avoidance logic
  if (distanceFront < MAX_DISTANCE) {
    stopMotion();
    delay(500);
    if (distanceLeft > distanceRight) {
      leftMotion();
      delay(1000);
    } else {
      rightMotion();
      delay(1000);
    }
  }

  delay(100);
}

// Function to measure distance using an ultrasonic sensor
long measureDistance(int trigPin, int echoPin) {
  // Clear the trigPin
```

```
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Trigger the sensor by setting the trigPin high for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the echoPin, which returns the time in microseconds
  long duration = pulseIn(echoPin, HIGH);

  // Calculate the distance in centimeters
  long distance = duration * 0.034 / 2;

  return distance;
}

// Motor control functions
void forwardMotion() {
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  ledcWrite(pwmChannelA, 200);
  ledcWrite(pwmChannelB, 200);
}

void backwardMotion() {
//write a logic here with above tabular column-2
}

void leftMotion() {
//write a logic here with above tabular column-2
}

void rightMotion() {
//write a logic here with above tabular column-2
}

void stopMotion() {
//write a logic here with above tabular column-2
```

```
}

void sharpLeftMotion() {
//write a logic here with above tabular column-2
}

void sharpRightMotion() {
//write a logic here with above tabular column-2
}
```

Here's an explanation of the code:

Constants and Variables

1. **IR Sensors:**
   - `const int IR_SENSOR_1 = 27;`
   - `const int IR_SENSOR_2 = 26;`
   - `const int IR_SENSOR_3 = 25;`
   - `const int IR_SENSOR_4 = 33;`
   - These constants define the GPIO pins connected to the four IR sensors.

2. **Ultrasonic Sensors:**
   - `const int TRIG_PIN_1 = 14;`
   - `const int ECHO_PIN_1 = 12;`
   - `const int TRIG_PIN_2 = 21;`
   - `const int ECHO_PIN_2 = 22;`
   - `const int TRIG_PIN_3 = 15;`
   - `const int ECHO_PIN_3 = 23;`
   - These constants define the GPIO pins connected to the trigger and echo pins of three ultrasonic sensors.

3. **Motor Connections:**
   - `int enA = 32;`
   - `int in1 = 16;`

- `int in2 = 17;`
- `int enB = 13;`
- `int in3 = 18;`
- `int in4 = 19;`
- These integers define the GPIO pins connected to the motor driver for controlling two motors.

4. **PWM Configuration:**
- `int pwmChannelA = 1;`
- `int pwmChannelB = 2;`
- `int pwmFrequency = 5000;`
- `int pwmResolution = 8;`
- These integers configure the PWM channels, frequency, and resolution for controlling motor speed.

5. **Maximum Distance:**
- `const long MAX_DISTANCE = 30;`
- This constant defines the maximum distance (in centimeters) for obstacle detection using ultrasonic sensors.

`setup()` Function

The `setup()` function runs once when the microcontroller starts.

1. **Serial Communication:**
- `Serial.begin(115200);`
- Initializes serial communication at a baud rate of 115200 for debugging.

2. **IR Sensor Pins:**
- Sets the IR sensor pins as inputs using `pinMode()`.

3. **Ultrasonic Sensor Pins:**
- Sets the ultrasonic sensor trigger pins as outputs and echo pins as inputs using `pinMode()`.

4. **Motor Control Pins:**

- Sets the motor control pins as outputs using `pinMode()`.

5. **PWM Configuration:**
   - `ledcSetup()` configures the PWM channels.
   - `ledcAttachPin()` attaches the PWM channels to the motor enable pins.

6. **Initial Motor State:**
   - Calls `stopMotion()` to ensure motors are stopped at the start.

`loop()` Function

The `loop()` function runs continuously after the `setup()` function.

1. **Read IR Sensors:**
   - Reads the values from the IR sensors using `digitalRead()`.

2. **Read Ultrasonic Sensors:**
   - Calls `measureDistance()` to get the distance from each ultrasonic sensor.

3. **Line Following Logic:**
   - Depending on the IR sensor values, calls functions to move forward, turn left/right, or make sharp turns.

4. **Obstacle Avoidance Logic:**
   - If an obstacle is detected within `MAX_DISTANCE` in front, the robot stops and then turns left or right based on the distances measured by the side ultrasonic sensors.

5. **Delay:**
   - A short delay before the next loop iteration to prevent excessive processing.

`measureDistance()` Function

The `measureDistance()` function measures the distance using an ultrasonic sensor.

1. **Clear `trigPin`:**

- Sets the `trigPin` to LOW for 2 microseconds to ensure a clean signal.

2. **Trigger the Sensor:**
   - Sets the `trigPin` to HIGH for 10 microseconds to send the ultrasonic pulse.

3. **Read Echo:**
   - Reads the duration of the echo pulse on the `echoPin` using `pulseIn()`.

4. **Calculate Distance:**
   - Converts the duration to distance in centimeters and returns it.

Motor Control Functions

These functions control the motor directions and speeds.

1. **`forwardMotion()`:**
   - Moves the robot forward by setting motor A and B to move forward with a PWM value of 200.

2. **`backwardMotion()`:**
   - Moves the robot backward by setting motor A and B to move backward with a PWM value of 200.

3. **`leftMotion()`:**
   - Turns the robot left by setting motor A backward and motor B forward with a PWM value of 200.

4. **`rightMotion()`:**
   - Turns the robot right by setting motor A forward and motor B backward with a PWM value of 200.

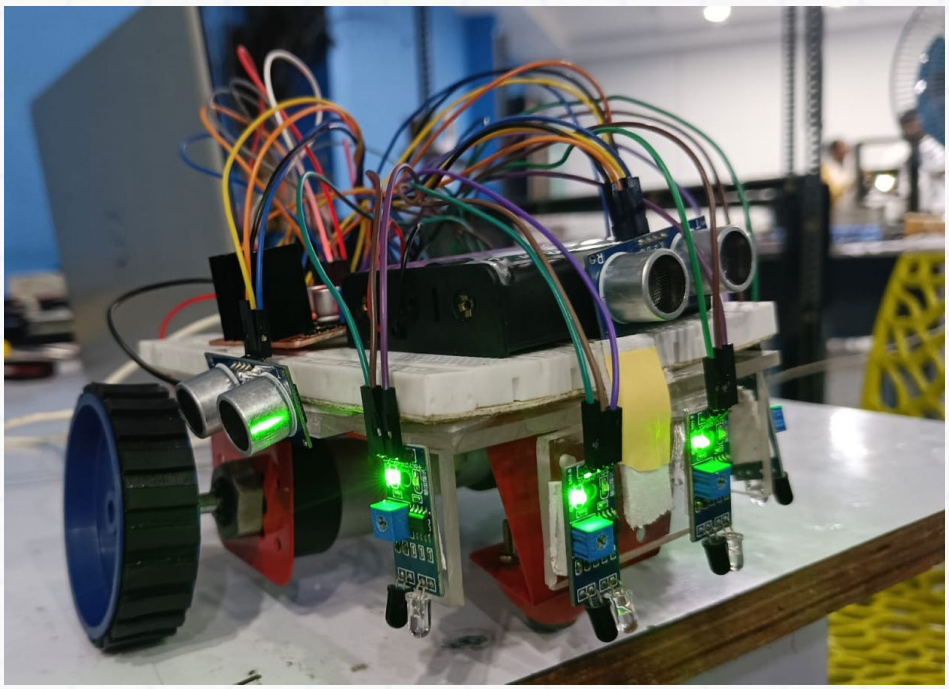
5. **`stopMotion()`:**
   - Stops the robot by setting all motor control pins to LOW and PWM values to 0.

6. **`sharpLeftMotion()`:**

- Makes a sharp left turn by setting motor A backward and motor B forward with a maximum PWM value of 255.

7. **`sharpRightMotion()`:**
- Makes a sharp right turn by setting motor A forward and motor B backward with a maximum PWM value of 255.

This code integrates line following and obstacle avoidance using IR and ultrasonic sensors, respectively, and controls the robot's movements accordingly.



Tasks

1) Build the fastest and most efficient Maze Solver! Write programs for implementing various popular maze solving algorithms such as: a) Left-hand following b) Right-hand following Explore more such algorithms on the internet. Eg. Wikipedia Note down how many attempts it takes to solve the maze for each algorithm. Which one was the best?

2) Build a Maze Learning Robot! Devise a program such that the Micro-mouse is made to repeatedly attempt the maze until it is solved. In the process of traversing the maze, the robot should learn the maze and map it in its memory. Therefore, once fully mapped, the Micro-mouse should be able to solve the maze without making any mistake in a single attempt.

3) Build a Line Following Robot. On a white surface, Draw a path using black tape. Using only the centre 2 IR sensors to detect the black line, the Micro-mouse should be able to follow it.

Note: The black tape width must be less than the gap between the centre 2 IR sensors. Don't create abrupt turns in the black line, else the robot may not detect it.