Obstacle Avoiding Car

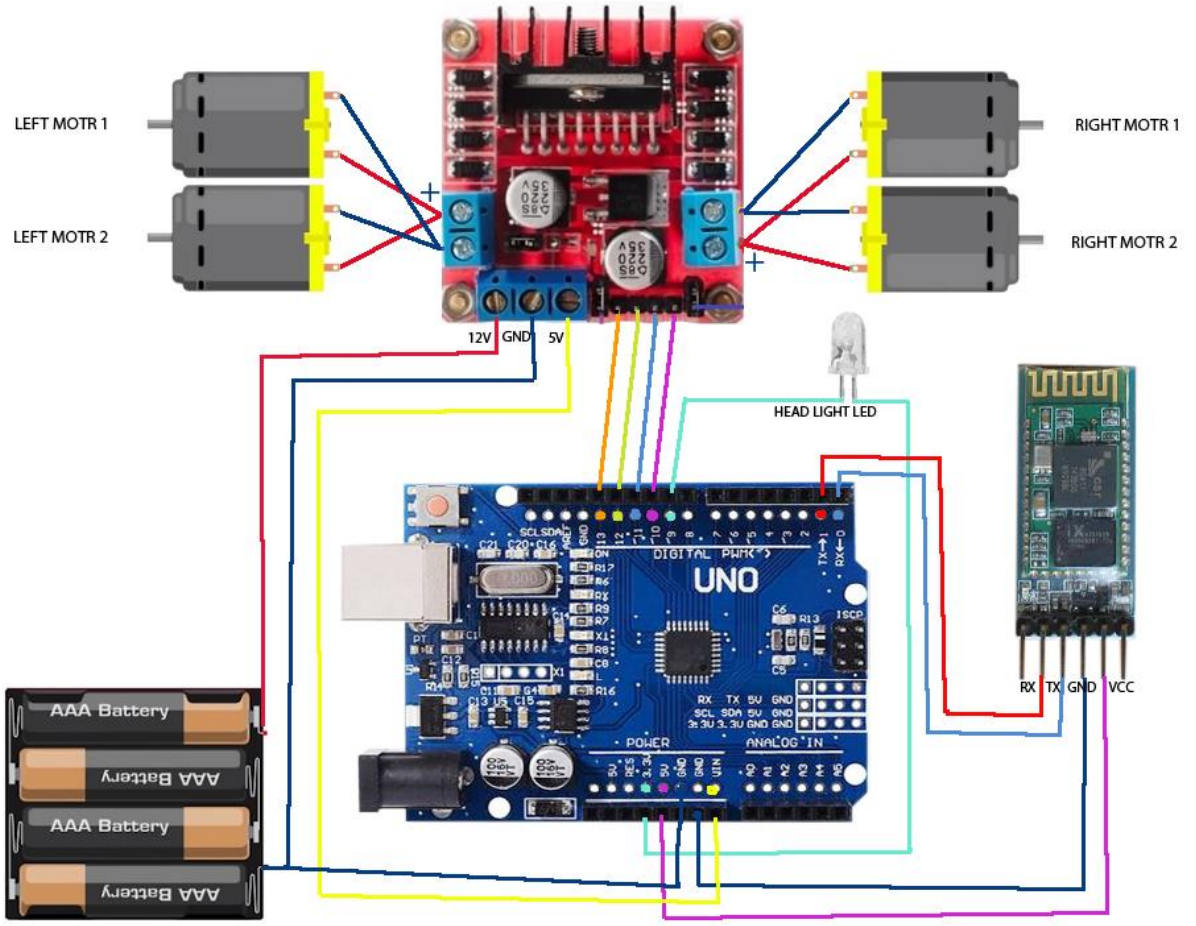
Code:

```
#include <Servo.h>
Servo Myservo;
#define trigPin 9 // Trig Pin Of HC-SR04
#define echoPin 8 // Echo Pin Of HC-SR04
#define MLa 4 //left motor 1st pin
#define MLb 5 //left motor 2nd pin
#define MRa 6 //right motor 1st pin
#define MRb 7 //right motor 2nd pin
long duration, distance;

void setup() {
Serial.begin(9600);
pinMode(MLa, OUTPUT); // Set Motor Pins As O/P
pinMode(MLb, OUTPUT);
pinMode(MRa, OUTPUT);
pinMode(MRb, OUTPUT);
pinMode(trigPin, OUTPUT); // Set Trig Pin As O/P
To Transmit Waves
pinMode(echoPin, INPUT); //Set Echo Pin As I/P To
Receive Reflected Waves
Myservo.attach(10);
}
void loop()
{
```

```
Serial.begin(9600);
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH); // Transmit Waves For
10us
delayMicroseconds(10);
duration = pulseIn(echoPin, HIGH); // Receive
Reflected Waves
distance = duration / 58.2; // Get Distance
Serial.println(distance);
delay(10);
if (distance > 15) // Condition For Absence Of
Obstacle
{
Myservo.write(90);
digitalWrite(MRb, HIGH); // Move Forward
digitalWrite(MRa, LOW);
digitalWrite(MLb, HIGH);
digitalWrite(MLa, LOW);
}
else if ((distance < 10)&&(distance > 0)) //
Condition For Presence Of Obstacle
{
digitalWrite(MRb, LOW); //Stop
digitalWrite(MRa, LOW);
digitalWrite(MLb, LOW);
```

```
digitalWrite(MLa, LOW);
delay(100);
Myservo.write(0);
delay(500);
Myservo.write(180);
delay(500);
Myservo.write(90);
delay(500);
digitalWrite(MRb, LOW); // Move Backward
digitalWrite(MRa, HIGH);
digitalWrite(MLb, LOW);
digitalWrite(MLa, HIGH);
delay(500);
digitalWrite(MRb, LOW); //Stop
digitalWrite(MRa, LOW);
digitalWrite(MLb, LOW);
digitalWrite(MLa, LOW);
delay(100);
digitalWrite(MRb, HIGH); // Move Left
digitalWrite(MRa, LOW);
digitalWrite(MLa, LOW);
digitalWrite(MLb, LOW);
delay(500);
}

}
```

Diagram:



RC Bluetooth Car

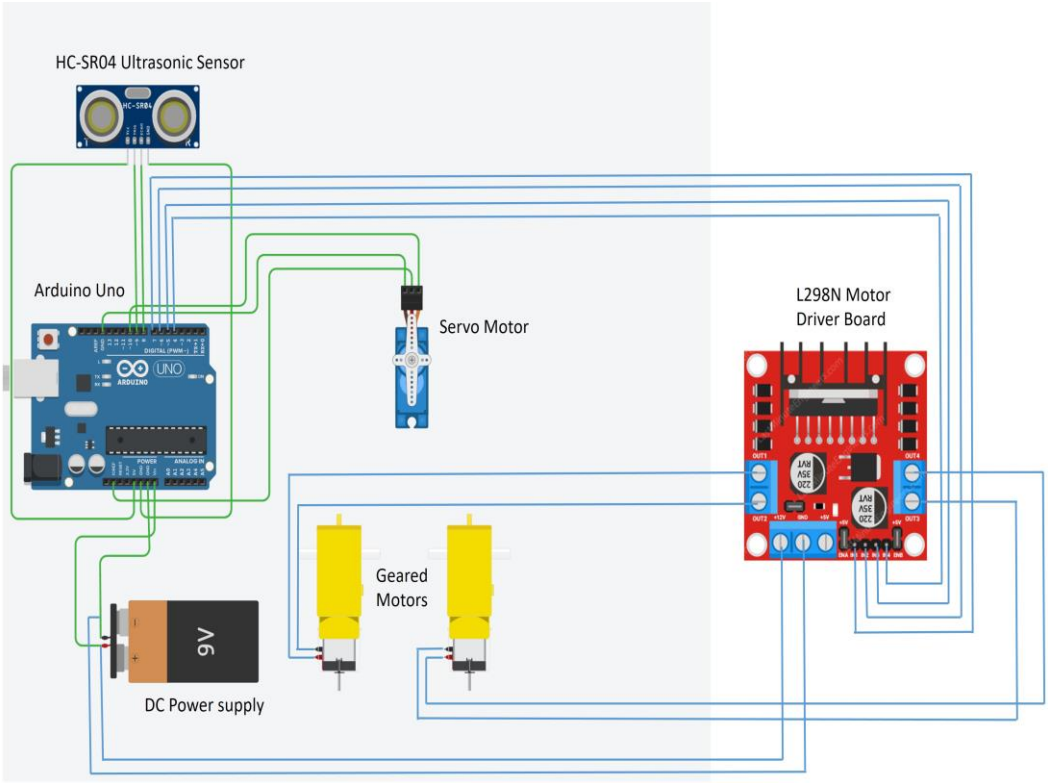
Code:

```
char t;
void setup() {
pinMode(13,OUTPUT); //left motors forward
pinMode(12,OUTPUT); //left motors reverse
pinMode(11,OUTPUT); //right motors forward
pinMode(10,OUTPUT); //right motors reverse
pinMode(9,OUTPUT); //Led
```

```
Serial.begin(9600);
}
void loop() {
if(Serial.available()){
t = Serial.read();
Serial.println(t);
}
if(t == 'F'){ //move forward(all motors rotate in forward direction)
digitalWrite(13,HIGH);
digitalWrite(11,HIGH);
}
else if(t == 'B'){ //move reverse (all motors rotate in reverse direction)
digitalWrite(12,HIGH);
digitalWrite(10,HIGH);
}
else if(t == 'L'){ //turn right (left side motors rotate in forward direction, right side motors doesn't rotate)
digitalWrite(11,HIGH);
}
else if(t == 'R'){ //turn left (right side motors rotate in forward direction, left side motors doesn't rotate)
digitalWrite(13,HIGH);
}
```

```
else if(t == 'W'){ //turn led on or off)
digitalWrite(9,HIGH);
}
else if(t == 'w'){
digitalWrite(9,LOW);
}
else if(t == 'S'){ //STOP (all motors stop)
digitalWrite(13,LOW);
digitalWrite(12,LOW);
digitalWrite(11,LOW);
digitalWrite(10,LOW);
}
delay(100);
}
```

Diagram:



CONBINE:

```
#include <Servo.h>

// Pin definitions for motors
#define MLa 13 // Left motor forward
#define MLb 12 // Left motor reverse
#define MRa 11 // Right motor forward
#define MRb 10 // Right motor reverse

// Pin definitions for other components
#define trigPin 9 // Trig pin of HC-SR04
#define echoPin 8 // Echo pin of HC-SR04
```

```
#define servoPin 7 // Servo motor control pin
#define ledPin 9 // LED pin

Servo Myservo;
char t;
long duration, distance;
bool mode = true; // true for RC mode, false for
Obstacle Avoidance mode

void setup() {
Serial.begin(9600);
// Motor pins setup
pinMode(MLa, OUTPUT);
pinMode(MLb, OUTPUT);
pinMode(MRa, OUTPUT);
pinMode(MRb, OUTPUT);
// Ultrasonic sensor setup
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
// LED and Servo setup
pinMode(ledPin, OUTPUT);
Myservo.attach(servoPin);
}

void loop() {
if (Serial.available()) {
t = Serial.read();
```

```
if (t == 'M') { // Switch mode when 'M' is received
mode = !mode;
if (mode) {
Serial.println("Switched to RC Mode");
} else {
Serial.println("Switched to Obstacle Avoidance
Mode");
}
}
}

if (mode) {
// Remote Control Mode
remoteControl();
} else {
// Obstacle Avoidance Mode
obstacleAvoidance();
}
}

void remoteControl() {
if (t == 'F') { // Move forward (all motors rotate in
forward direction)
digitalWrite(MLa, HIGH);
digitalWrite(MRa, HIGH);
digitalWrite(MLb, LOW);
digitalWrite(MRb, LOW);
```

```
} else if (t == 'B') { // Move reverse (all motors rotate in reverse direction)
digitalWrite(MLb, HIGH);
digitalWrite(MRb, HIGH);
digitalWrite(MLa, LOW);
digitalWrite(MRa, LOW);
} else if (t == 'L') { // Turn left (right side motors rotate in forward direction, left side motors don't rotate)
digitalWrite(MRa, HIGH);
digitalWrite(MLa, LOW);
digitalWrite(MLb, LOW);
digitalWrite(MRb, LOW);
} else if (t == 'R') { // Turn right (left side motors rotate in forward direction, right side motors don't rotate)
digitalWrite(MLa, HIGH);
digitalWrite(MRa, LOW);
digitalWrite(MLb, LOW);
digitalWrite(MRb, LOW);
} else if (t == 'W') { // Turn LED on
digitalWrite(ledPin, HIGH);
} else if (t == 'w') { // Turn LED off
digitalWrite(ledPin, LOW);
} else if (t == 'S') { // STOP (all motors stop)
digitalWrite(MLa, LOW);
```

```
digitalWrite(MLb, LOW);
digitalWrite(MRa, LOW);
digitalWrite(MRb, LOW);
}
}

void obstacleAvoidance() {
// Ultrasonic sensor
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH); // Transmit waves for
10us
delayMicroseconds(10);
duration = pulseIn(echoPin, HIGH); // Receive
reflected waves
distance = duration / 58.2; // Get distance
Serial.println(distance);
delay(10);

if (distance > 15) { // Condition for absence of
obstacle
Myservo.write(90);
digitalWrite(MRb, LOW); // Move forward
digitalWrite(MRa, HIGH);
digitalWrite(MLb, LOW);
digitalWrite(MLa, HIGH);
```

```
} else if ((distance < 10) && (distance > 0)) { // Condition for presence of obstacle
digitalWrite(MRb, LOW); // Stop
digitalWrite(MRa, LOW);
digitalWrite(MLb, LOW);
digitalWrite(MLa, LOW);
delay(100);
Myservo.write(0);
delay(500);
Myservo.write(180);
delay(500);
Myservo.write(90);
delay(500);
digitalWrite(MRb, HIGH); // Move backward
digitalWrite(MRa, LOW);
digitalWrite(MLb, HIGH);
digitalWrite(MLa, LOW);
delay(500);
digitalWrite(MRb, LOW); // Stop
digitalWrite(MRa, LOW);
digitalWrite(MLb, LOW);
digitalWrite(MLa, LOW);
delay(100);
digitalWrite(MRb, HIGH); // Move left
digitalWrite(MRa, LOW);
digitalWrite(MLa, LOW);
```

```
digitalWrite(MLb, LOW);
delay(500);
}
}
```

DIAGRAM:

Components and Pin Connections

1. **Motors:**
   - **Left Motor (Forward):** Connect to **Pin 13** on the Arduino.
   - **Left Motor (Reverse):** Connect to **Pin 12** on the Arduino.
   - **Right Motor (Forward):** Connect to **Pin 11** on the Arduino.
   - **Right Motor (Reverse):** Connect to **Pin 10** on the Arduino.
2. **Ultrasonic Sensor (HC-SR04):**
   - **Trig Pin:** Connect to **Pin 9** on the Arduino.
   - **Echo Pin:** Connect to **Pin 8** on the Arduino.
   - **VCC:** Connect to the 5V pin on the Arduino.
   - **GND:** Connect to the GND pin on the Arduino.
3. **Servo Motor:**

- **Control Signal:** Connect to **Pin 7**
- **VCC:** Connect to the 5V pin on the Arduino.
- **GND:** Connect to the GND pin on the Arduino.

4. **LED:**
    - **Positive Lead (Anode):** Connect to **Pin 9** on the Arduino.
    - **Negative Lead (Cathode):** Connect to GND through a 220-ohm resistor.
5. **HC-05 Bluetooth Module:**
    - **TXD (Transmitter):** Connect to the **RX** pin of a SoftwareSerial instance (you can assign custom pins if needed).
    - **RXD (Receiver):** Connect to the **TX** pin of a SoftwareSerial instance (you can assign custom pins if needed).
    - **VCC:** Connect to the 5V pin on the Arduino.
    - **GND:** Connect to the GND pin on the Arduino.
    - **EN/KEY (if present):** Leave unconnected or connect to 3.3V if needed for certain configurations.

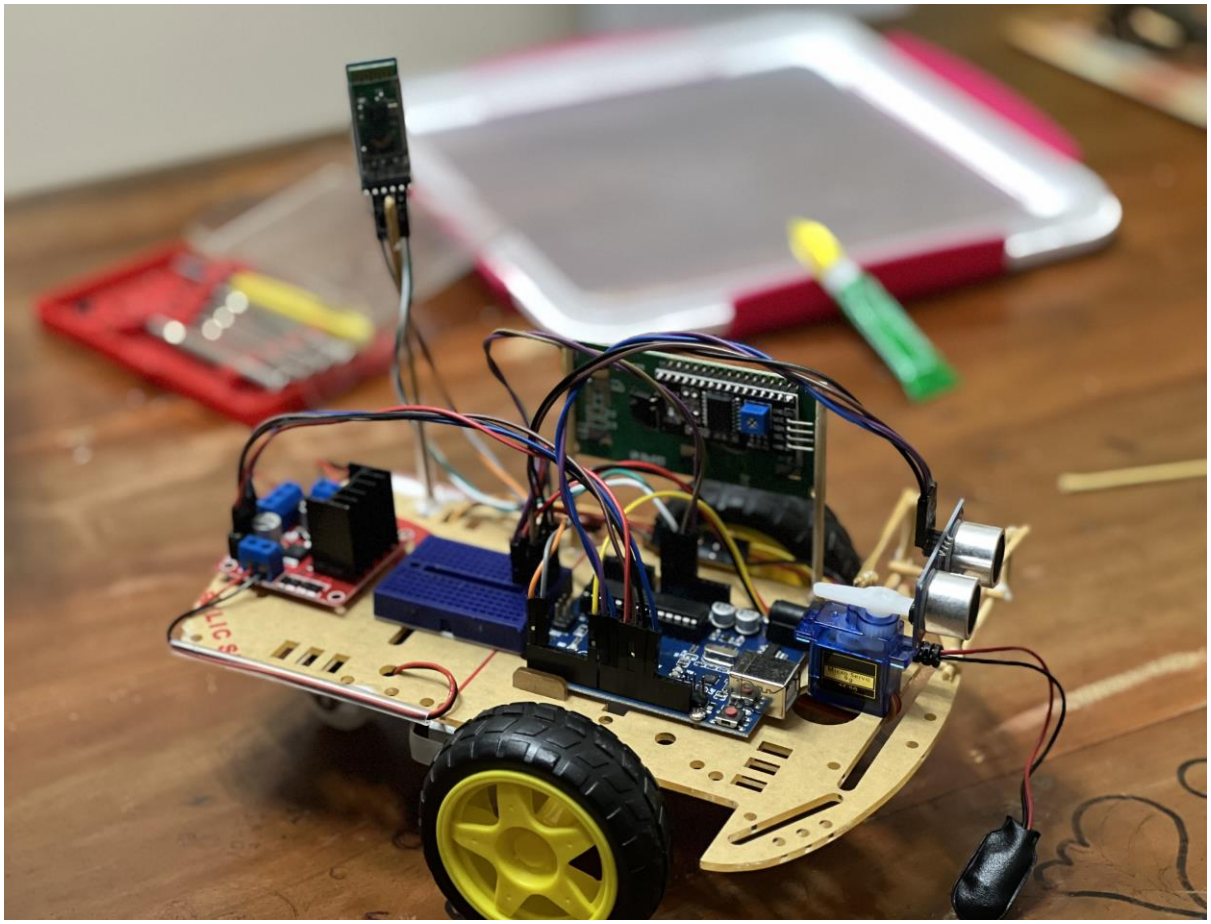
Power Supply:

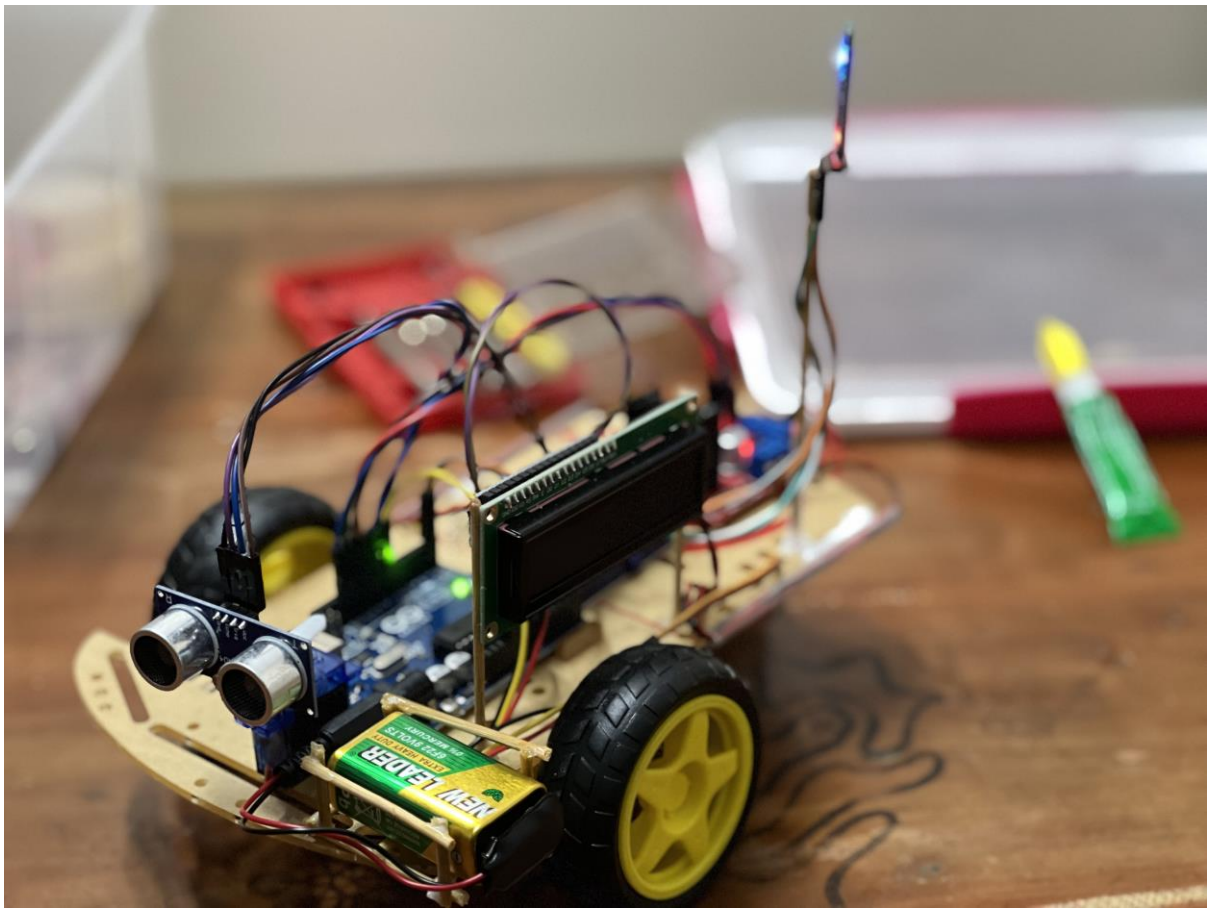
- **Battery Pack (for Motors):** Connect the positive terminal to the VIN or external power supply terminal of the motor driver (if using one), and the negative terminal to GND.
- **Arduino Power:** If you’re using the onboard power regulator, you can connect your battery pack directly to the Arduino's VIN and GND pins.

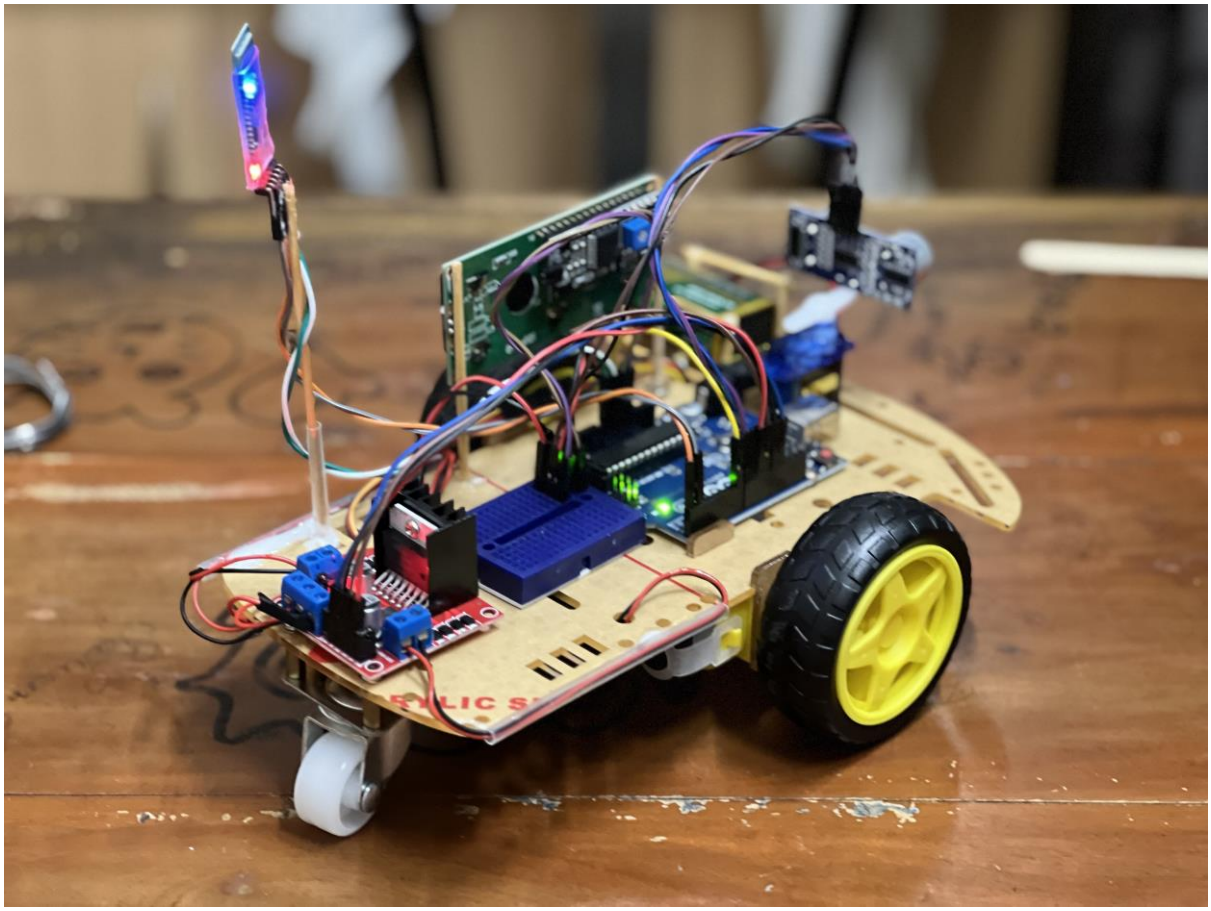
Summary:

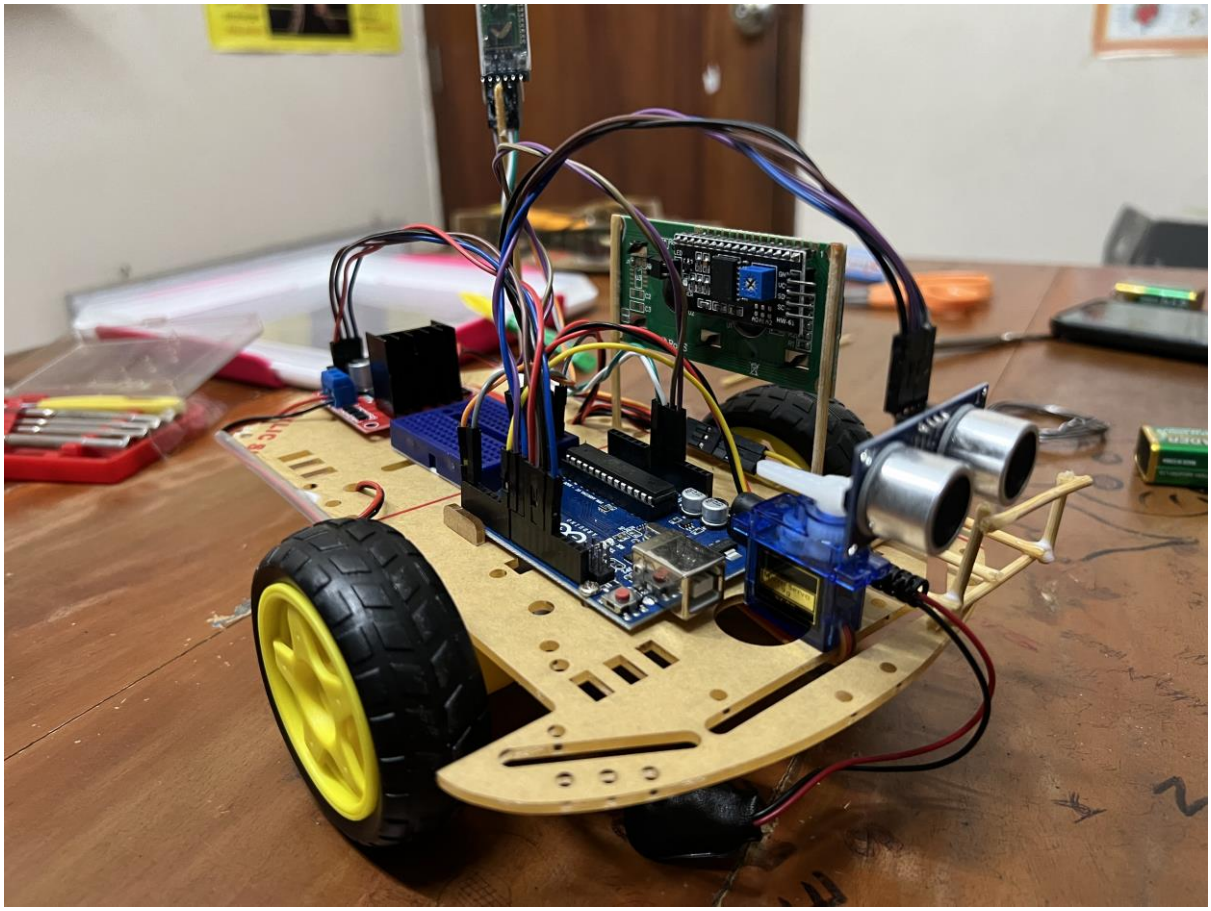
- **Motor Pins:** Pins 13, 12, 11, and 10 control the left and right motors in both directions.
- **Ultrasonic Sensor:** Pins 9 and 8 handle the distance measurement for obstacle avoidance.
- **Servo Motor:** Pin 7
- **Bluetooth Module:** Custom SoftwareSerial pins manage communication with the Bluetooth module.

Ready project:




NEW LEADER





Combine with led lights

Code:

```
#include <Servo.h>

// Pin definitions for motors
#define MLa 13 // Left motor forward
#define MLb 12 // Left motor reverse
#define MRa 11 // Right motor forward
#define MRb 10 // Right motor reverse

// Pin definitions for other components
#define trigPin 9 // Trig pin of HC-SR04
#define echoPin 8 // Echo pin of HC-SR04
#define servoPin 7 // Servo motor control pin
#define ledPin 6 // LED pin

Servo Myservo;
char t;
long duration, distance;
bool mode = true; // true for RC mode, false for Obstacle Avoidance mode
bool ledsOn = false; // LED control state

void setup() {
Serial.begin(9600);
// Motor pins setup
pinMode(MLa, OUTPUT);
pinMode(MLb, OUTPUT);
pinMode(MRa, OUTPUT);
pinMode(MRb, OUTPUT);
// Ultrasonic sensor setup
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
// LED and Servo setup
pinMode(ledPin, OUTPUT);
Myservo.attach(servoPin);
}

void loop() {
if (Serial.available()) {
t = Serial.read();
if (t == 'M') { // Switch mode when 'M' is received
```

```
mode = !mode;
if (mode) {
Serial.println("Switched to RC Mode");
} else {
Serial.println("Switched to Obstacle Avoidance Mode");
}
} else if (t == 'N') { // Turn LEDs on/off when 'N' is received
ledsOn = !ledsOn;
digitalWrite(ledPin, ledsOn ? HIGH : LOW);
}
}

if (mode) {
// Remote Control Mode
remoteControl();
} else {
// Obstacle Avoidance Mode
obstacleAvoidance();
}
}
void remoteControl() {
if (t == 'F') { // Move forward (all motors rotate in forward direction)
digitalWrite(MLa, HIGH);
digitalWrite(MRa, HIGH);
digitalWrite(MLb, LOW);
digitalWrite(MRb, LOW);
} else if (t == 'B') { // Move reverse (all motors rotate in reverse direction)
digitalWrite(MLb, HIGH);
digitalWrite(MRb, HIGH);
digitalWrite(MLa, LOW);
digitalWrite(MRa, LOW);
} else if (t == 'L') { // Turn left (right side motors rotate in forward direction, left side
motors don't rotate)
digitalWrite(MRa, HIGH);
digitalWrite(MLa, LOW);
digitalWrite(MLb, LOW);
digitalWrite(MRb, LOW);
} else if (t == 'R') { // Turn right (left side motors rotate in forward direction, right side
motors don't rotate)
digitalWrite(MLa, HIGH);
digitalWrite(MRa, LOW);
digitalWrite(MLb, LOW);
digitalWrite(MRb, LOW);
} else if (t == 'W') { // Turn LED on
```

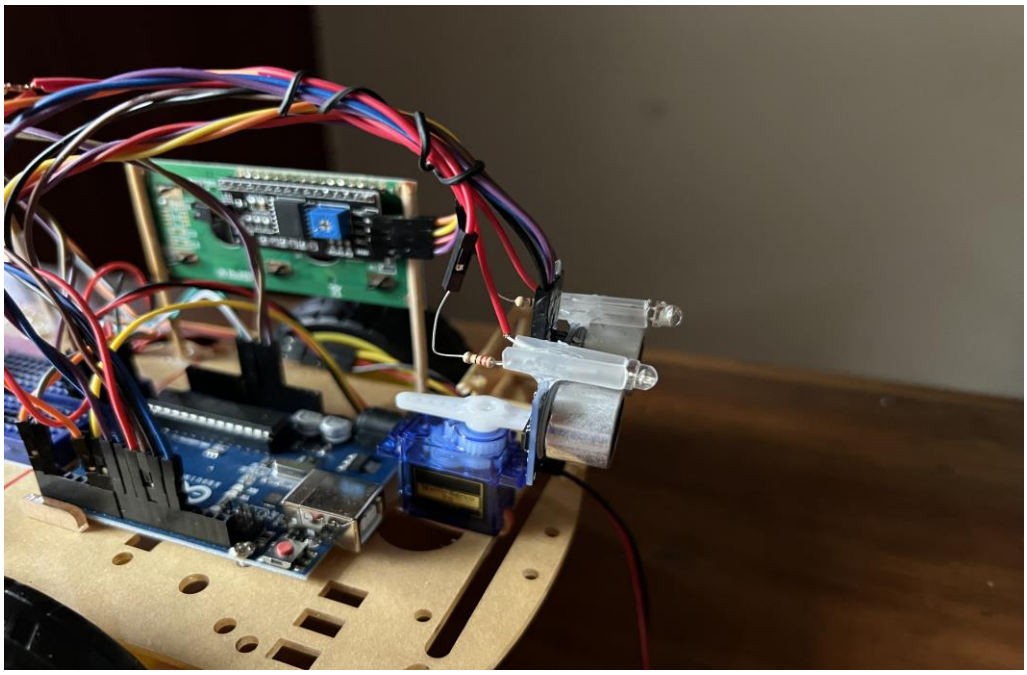
```
digitalWrite(ledPin, HIGH);
} else if (t == 'w') { // Turn LED off
digitalWrite(ledPin, LOW);
} else if (t == 'S') { // STOP (all motors stop)
digitalWrite(MLa, LOW);
digitalWrite(MLb, LOW);
digitalWrite(MRa, LOW);
digitalWrite(MRb, LOW);
}
}

void obstacleAvoidance() {
// Ultrasonic sensor
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH); // Transmit waves for 10us
delayMicroseconds(10);
duration = pulseIn(echoPin, HIGH); // Receive reflected waves
distance = duration / 58.2; // Get distance
Serial.println(distance);
delay(10);

if (distance > 15) { // Condition for absence of obstacle
Myservo.write(90);
digitalWrite(MRb, LOW); // Move forward
digitalWrite(MRa, HIGH);
digitalWrite(MLb, LOW);
digitalWrite(MLa, HIGH);
} else if ((distance < 10) && (distance > 0)) { // Condition for presence of obstacle
digitalWrite(MRb, LOW); // Stop
digitalWrite(MRa, LOW);
digitalWrite(MLb, LOW);
digitalWrite(MLa, LOW);
delay(100);
Myservo.write(0);
delay(500);
Myservo.write(180);
delay(500);
Myservo.write(90);
delay(500);
digitalWrite(MRb, HIGH); // Move backward
digitalWrite(MRa, LOW);
digitalWrite(MLb, HIGH);
digitalWrite(MLa, LOW);
```

```
delay(500);
digitalWrite(MRb, LOW); // Stop
digitalWrite(MRa, LOW);
digitalWrite(MLb, LOW);
digitalWrite(MLa, LOW);
delay(100);
digitalWrite(MRb, HIGH); // Move left
digitalWrite(MRa, LOW);
digitalWrite(MLa, LOW);
digitalWrite(MLb, LOW);
delay(500);
}
}
```

Added Diagram:

LEDs:

- **Anodes (longer legs) of both LEDs** are connected together and to **digital pin 6** on the Arduino.
- **Cathodes (shorter legs) of both LEDs** are connected to ground via individual resistors.
- **Resistors** (220Ω or 330Ω) are connected between the cathodes of the LEDs and the GND rail.



Combine mode with led and display

Code:

```
#include <Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Pin definitions for motors
#define MLa 13 // Left motor forward
#define MLb 12 // Left motor reverse
#define MRa 11 // Right motor forward
#define MRb 10 // Right motor reverse

// Pin definitions for other components
#define trigPin 9 // Trig pin of HC-SR04
#define echoPin 8 // Echo pin of HC-SR04
#define servoPin 7 // Servo motor control pin
#define ledPin 6 // LED pin

Servo Myservo;
LiquidCrystal_I2C lcd(0x27, 16, 2); // Adjust 0x27 to your LCD's I2C address

char t;
long duration, distance;
bool mode = true; // true for RC mode, false for Obstacle Avoidance mode
bool ledsOn = false; // LED control state

void setup() {
Serial.begin(9600);
// Motor pins setup
pinMode(MLa, OUTPUT);
pinMode(MLb, OUTPUT);
pinMode(MRa, OUTPUT);
pinMode(MRb, OUTPUT);
// Ultrasonic sensor setup
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
// LED and Servo setup
pinMode(ledPin, OUTPUT);
```

```
Myservo.attach(servoPin);
// Initialize the LCD with the correct number of columns and rows
lcd.begin(16, 2);
lcd.print("RC Mode");
}

void loop() {
if (Serial.available()) {
t = Serial.read();
if (t == 'M') { // Switch mode when 'M' is received
mode = !mode;
lcd.clear();
if (mode) {
lcd.print("RC Mode");
Serial.println("Switched to RC Mode");
} else {
lcd.print("Obstacle Avoidance");
Serial.println("Switched to Obstacle Avoidance Mode");
}
} else if (t == 'N') { // Turn LEDs on/off when 'N' is received
ledsOn = !ledsOn;
digitalWrite(ledPin, ledsOn ? HIGH : LOW);
}
}

if (mode) {
// Remote Control Mode
remoteControl();
} else {
// Obstacle Avoidance Mode
obstacleAvoidance();
}
}

void remoteControl() {
if (t == 'F') { // Move forward (all motors rotate in forward direction)
digitalWrite(MLa, HIGH);
digitalWrite(MRa, HIGH);
digitalWrite(MLb, LOW);
digitalWrite(MRb, LOW);
} else if (t == 'B') { // Move reverse (all motors rotate in reverse direction)
digitalWrite(MLb, HIGH);
digitalWrite(MRb, HIGH);
digitalWrite(MLa, LOW);
```

```
digitalWrite(MRa, LOW);
} else if (t == 'L') { // Turn left (right side motors rotate in forward direction, left side motors don't rotate)
digitalWrite(MRa, HIGH);
digitalWrite(MLa, LOW);
digitalWrite(MLb, LOW);
digitalWrite(MRb, LOW);
} else if (t == 'R') { // Turn right (left side motors rotate in forward direction, right side motors don't rotate)
digitalWrite(MLa, HIGH);
digitalWrite(MRa, LOW);
digitalWrite(MLb, LOW);
digitalWrite(MRb, LOW);
} else if (t == 'W') { // Turn LED on
digitalWrite(ledPin, HIGH);
} else if (t == 'w') { // Turn LED off
digitalWrite(ledPin, LOW);
} else if (t == 'S') { // STOP (all motors stop)
digitalWrite(MLa, LOW);
digitalWrite(MLb, LOW);
digitalWrite(MRa, LOW);
digitalWrite(MRb, LOW);
}
}

void obstacleAvoidance() {
// Ultrasonic sensor
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH); // Transmit waves for 10us
delayMicroseconds(10);
duration = pulseIn(echoPin, HIGH); // Receive reflected waves
distance = duration / 58.2; // Get distance
Serial.println(distance);
delay(10);

if (distance > 15) { // Condition for absence of obstacle
Myservo.write(90);
digitalWrite(MRb, LOW); // Move forward
digitalWrite(MRa, HIGH);
digitalWrite(MLb, LOW);
digitalWrite(MLa, HIGH);
} else if ((distance < 10) && (distance > 0)) { // Condition for presence of obstacle
digitalWrite(MRb, LOW); // Stop
```

```
digitalWrite(MRa, LOW);
digitalWrite(MLb, LOW);
digitalWrite(MLa, LOW);
delay(100);
Myservo.write(0);
delay(500);
Myservo.write(180);
delay(500);
Myservo.write(90);
delay(500);
digitalWrite(MRb, HIGH); // Move backward
digitalWrite(MRa, LOW);
digitalWrite(MLb, HIGH);
digitalWrite(MLa, LOW);
delay(500);
digitalWrite(MRb, LOW); // Stop
digitalWrite(MRa, LOW);
digitalWrite(MLb, LOW);
digitalWrite(MLa, LOW);
delay(100);
digitalWrite(MRb, HIGH); // Move left
digitalWrite(MRa, LOW);
digitalWrite(MLa, LOW);
digitalWrite(MLb, LOW);
delay(500);
}
}
```

Added diagram:

Wiring for the LCD

Assuming you're using an I2C 16x2 LCD display, connect the LCD as follows:

- **SDA** (Serial Data) to **A4** on the Arduino
- **SCL** (Serial Clock) to **A5** on the Arduino
- **VCC** to **5V**
- **GND** to **GND**

Triple main (rc, oa, lf)

Code:

```
#include <Servo.h>

// Pin definitions for motors
#define MLa 13 // Left motor forward
#define MLb 12 // Left motor reverse
#define MRa 11 // Right motor forward
#define MRb 10 // Right motor reverse

// Pin definitions for other components
#define trigPin 9 // Trig pin of HC-SR04
#define echoPin 8 // Echo pin of HC-SR04
#define servoPin 7 // Servo motor control pin
#define ledPin 6 // LED pin
#define IR_SENSOR_RIGHT 4 // Right IR sensor
#define IR_SENSOR_LEFT 5 // Left IR sensor

Servo Myservo;
char t;
long duration, distance;
int mode = 1; // 1 for RC mode, 2 for Obstacle Avoidance mode, 3 for Line Following mode
bool ledsOn = false; // LED control state

void setup() {
  Serial.begin(9600);

  // Motor pins setup
  pinMode(MLa, OUTPUT);
  pinMode(MLb, OUTPUT);
  pinMode(MRa, OUTPUT);
  pinMode(MRb, OUTPUT);

  // Ultrasonic sensor setup
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
```

```
 // LED and Servo setup
 pinMode(ledPin, OUTPUT);
 Myservo.attach(servoPin);
 Myservo.write(90); // Initialize the servo to the neutral position

 // IR sensor setup
 pinMode(IR_SENSOR_RIGHT, INPUT);
 pinMode(IR_SENSOR_LEFT, INPUT);
}

void loop() {
 if (Serial.available()) {
  t = Serial.read();
  if (t == 'M') { // Switch between RC and Obstacle Avoidance modes
   mode = (mode == 1) ? 2 : 1;
   switch (mode) {
    case 1:
     Serial.println("Switched to RC Mode");
     break;
    case 2:
     Serial.println("Switched to Obstacle Avoidance Mode");
     break;
   }
  } else if (t == 'X') { // Switch to Line Following mode when 'X' is received
   mode = 3;
   Serial.println("Switched to Line Following Mode");
  } else if (t == 'N') { // Turn LEDs on/off when 'N' is received
   ledsOn = !ledsOn;
   digitalWrite(ledPin, ledsOn ? HIGH : LOW);
  }
 }

 switch (mode) {
  case 1:
   remoteControl();
   break;
  case 2:
   obstacleAvoidance();
   break;
  case 3:
```

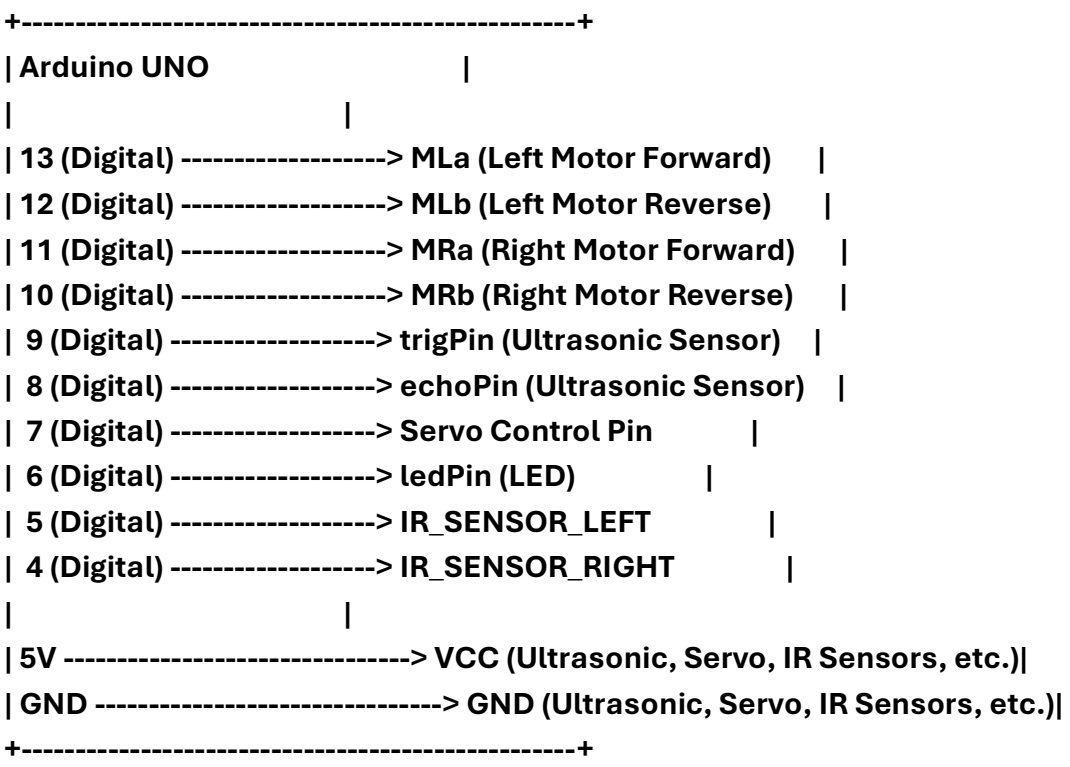
```
    lineFollowing();
    break;
  }
}

void remoteControl() {
  if (t == 'F') { // Move forward (all motors rotate in forward direction)
    digitalWrite(MLa, HIGH);
    digitalWrite(MRa, HIGH);
    digitalWrite(MLb, LOW);
    digitalWrite(MRb, LOW);
  } else if (t == 'B') { // Move reverse (all motors rotate in reverse direction)
    digitalWrite(MLb, HIGH);
    digitalWrite(MRb, HIGH);
    digitalWrite(MLa, LOW);
    digitalWrite(MRa, LOW);
  } else if (t == 'L') { // Turn left (right side motors rotate in forward direction, left side
motors don't rotate)
    digitalWrite(MRa, HIGH);
    digitalWrite(MLa, LOW);
    digitalWrite(MLb, LOW);
    digitalWrite(MRb, LOW);
  } else if (t == 'R') { // Turn right (left side motors rotate in forward direction, right
side motors don't rotate)
    digitalWrite(MLa, HIGH);
    digitalWrite(MRa, LOW);
    digitalWrite(MLb, LOW);
    digitalWrite(MRb, LOW);
  } else if (t == 'W') { // Turn LED on
    digitalWrite(ledPin, HIGH);
  } else if (t == 'w') { // Turn LED off
    digitalWrite(ledPin, LOW);
  } else if (t == 'S') { // STOP (all motors stop)
    digitalWrite(MLa, LOW);
    digitalWrite(MLb, LOW);
    digitalWrite(MRa, LOW);
    digitalWrite(MRb, LOW);
  }
}

void obstacleAvoidance() {
```

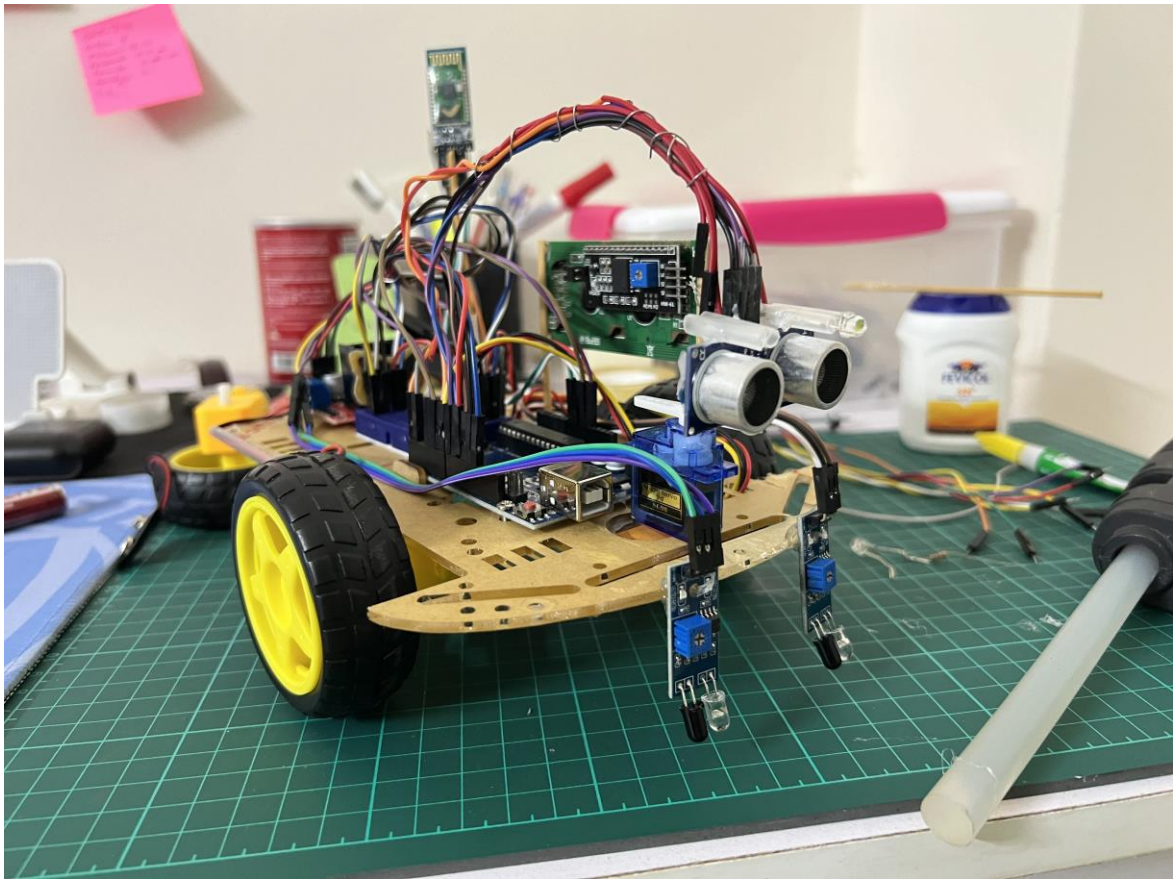
```
// Ultrasonic sensor
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH); // Transmit waves for 10us
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH); // Receive reflected waves
distance = duration / 58.2; // Calculate distance
Serial.println(distance);
delay(10);

if (distance > 15) { // Condition for absence of obstacle
  Myservo.write(90); // Keep the servo at a neutral position
  digitalWrite(MRb, LOW); // Move forward
  digitalWrite(MRa, HIGH);
  digitalWrite(MLb, LOW);
  digitalWrite(MLa, HIGH);
} else if (distance <= 15 && distance > 0) { // Condition for presence of obstacle
  digitalWrite(MRb, LOW); // Stop
  digitalWrite(MRa, LOW);
  digitalWrite(MLb, LOW);
  digitalWrite(MLa, LOW);
  delay(100);

  Myservo.write(0); // Rotate servo left
  delay(500);

  Myservo.write(180); // Rotate servo right
  delay(500);

  Myservo.write(90); // Return to neutral position
  delay(500);

  digitalWrite(MRb, HIGH); // Move backward
  digitalWrite(MRa, LOW);
  digitalWrite(MLb, HIGH);
  digitalWrite(MLa, LOW);
  delay(500);

  digitalWrite(MRb, LOW); // Stop
```

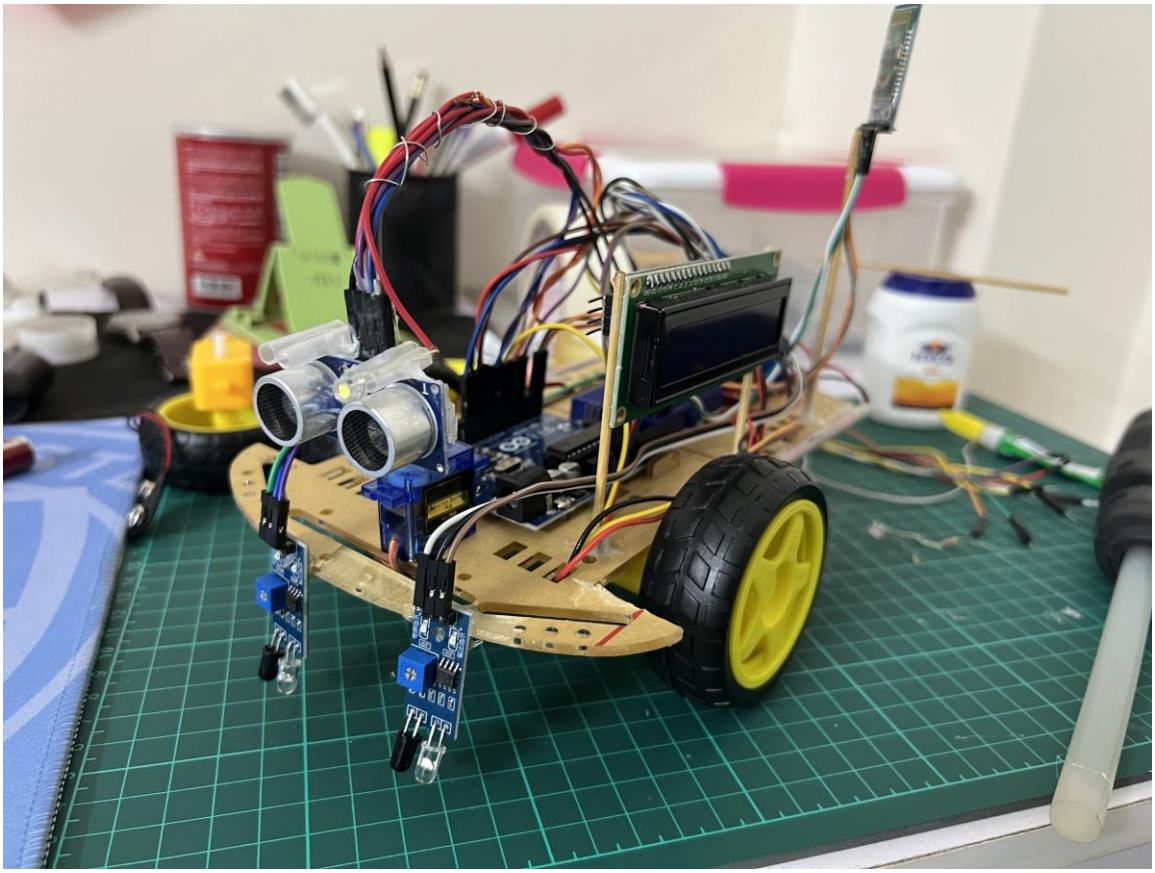
```
  digitalWrite(MRa, LOW);
  digitalWrite(MLb, LOW);
  digitalWrite(MLa, LOW);
  delay(100);

  digitalWrite(MRb, HIGH); // Move left
  digitalWrite(MRa, LOW);
  digitalWrite(MLa, LOW);
  digitalWrite(MLb, LOW);
  delay(500);
 }
}

void lineFollowing() {
 int rightIRSensorValue = digitalRead(IR_SENSOR_RIGHT);
 int leftIRSensorValue = digitalRead(IR_SENSOR_LEFT);

 // If none of the sensors detect the black line, go straight
 if (rightIRSensorValue == LOW && leftIRSensorValue == LOW) {
  digitalWrite(MLa, HIGH);
  digitalWrite(MRa, HIGH);
  digitalWrite(MLb, LOW);
  digitalWrite(MRb, LOW);
 }
 // If the right sensor detects the black line, turn right
 else if (rightIRSensorValue == HIGH && leftIRSensorValue == LOW) {
  digitalWrite(MLa, HIGH);
  digitalWrite(MRa, LOW);
  digitalWrite(MLb, LOW);
  digitalWrite(MRb, LOW);
 }
 // If the left sensor detects the black line, turn left
 else if (rightIRSensorValue == LOW && leftIRSensorValue == HIGH) {
  digitalWrite(MLa, LOW);
  digitalWrite(MRa, HIGH);
  digitalWrite(MLb, LOW);
  digitalWrite(MRb, LOW);
 }
 // If both sensors detect the black line, stop
 else {
  digitalWrite(MLa, LOW);
```

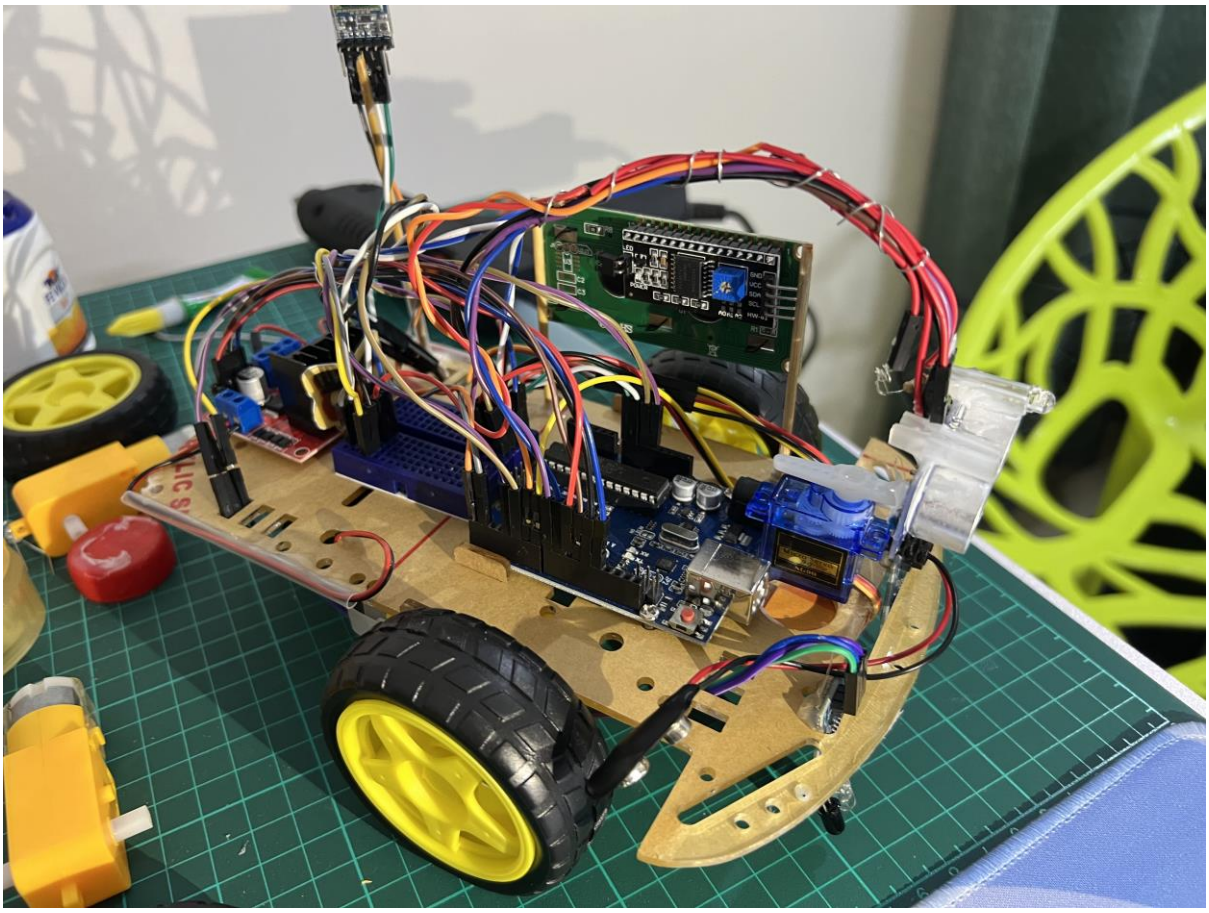
```
   digitalWrite(MRa, LOW);
:
   digitalWrite(MLb, LOW);
   digitalWrite(MRb, LOW);
 }
}
```

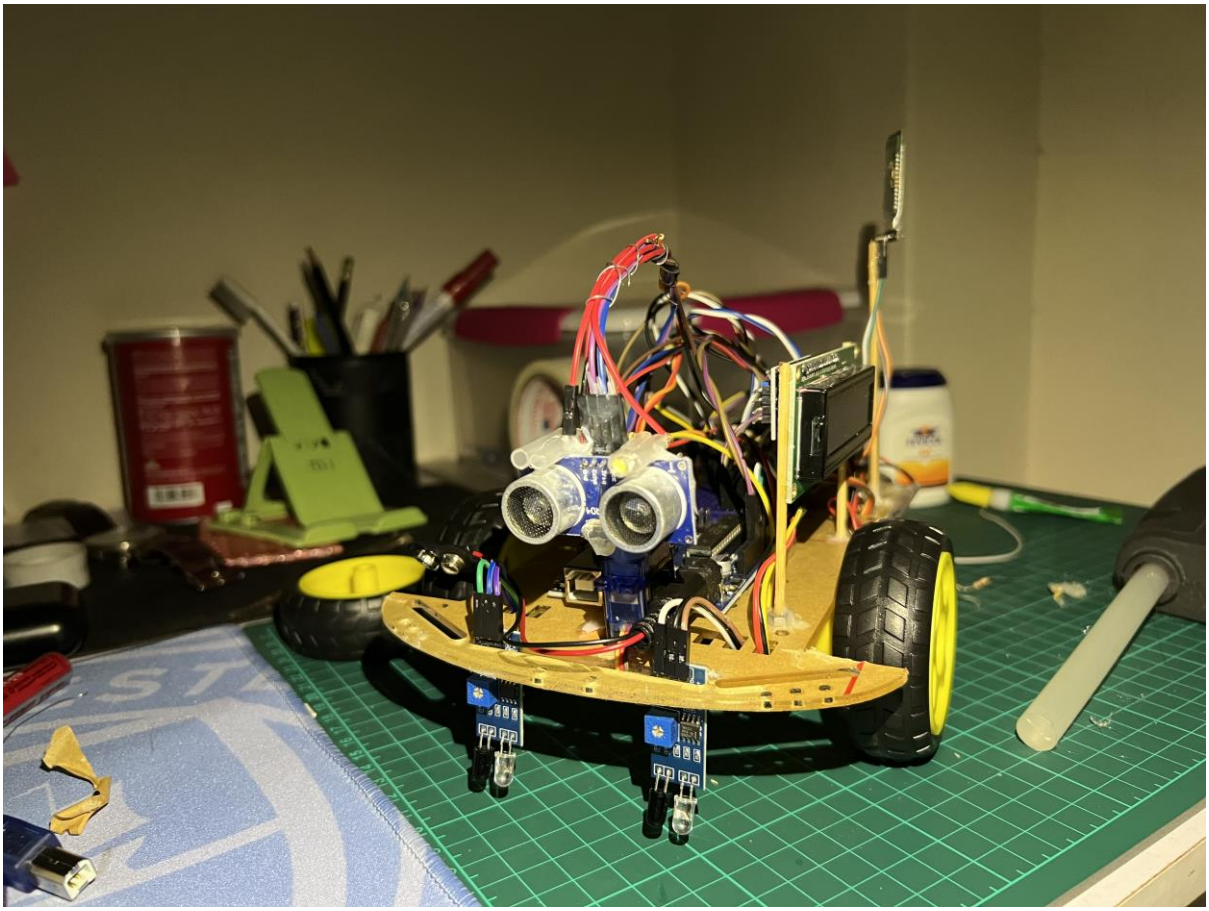
Diagram:

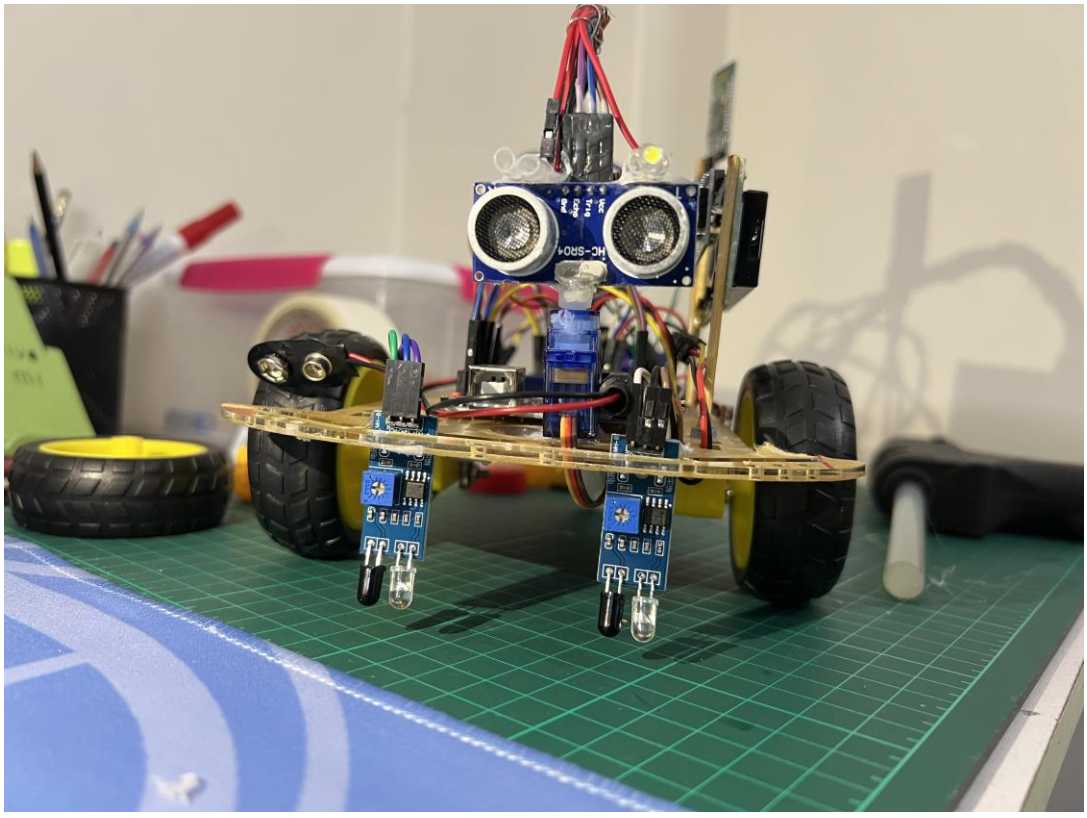











HC-SR04