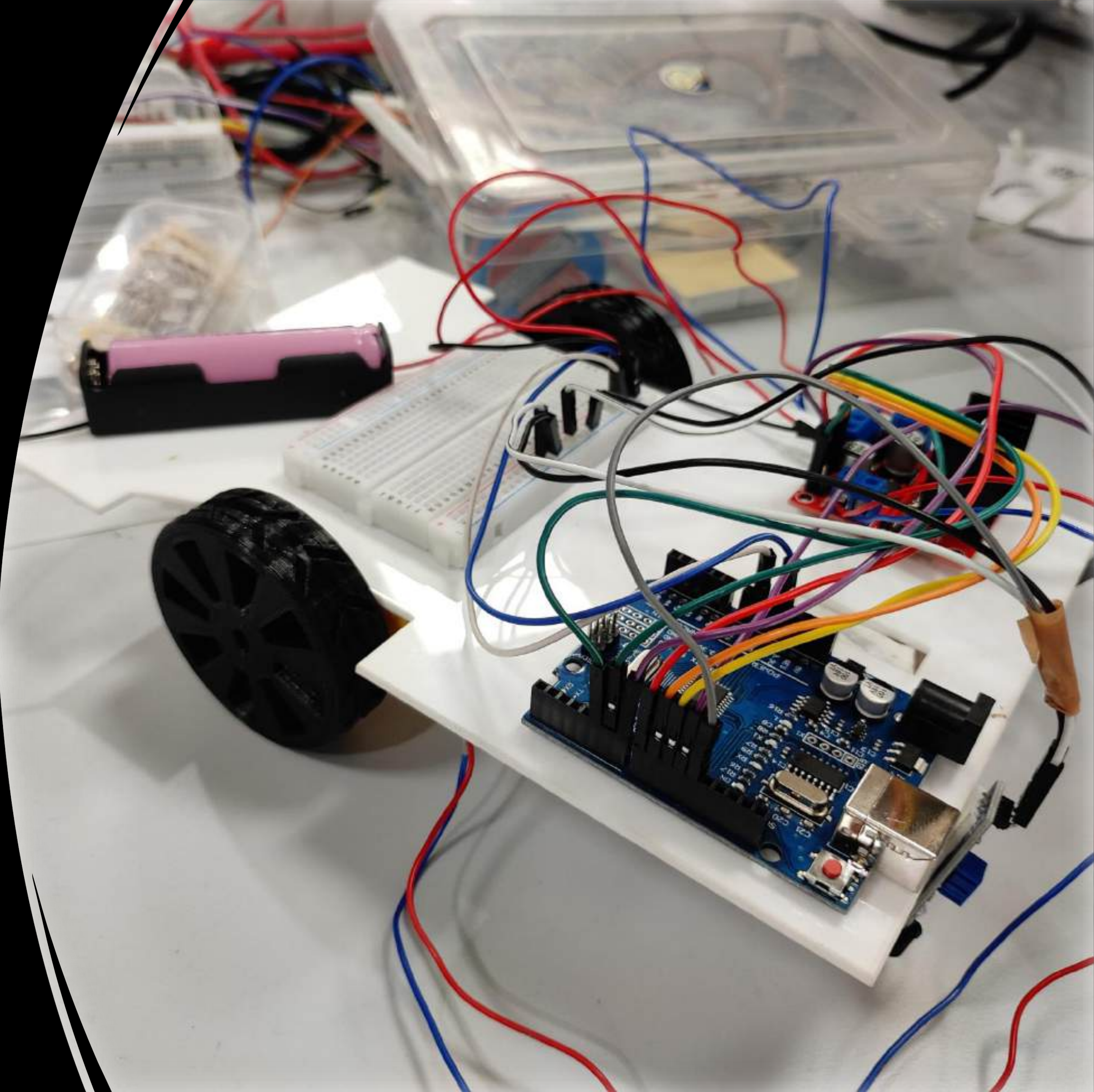
IC 152 MAKERSPACE LAB

LINE FOLLOWER PROJECT

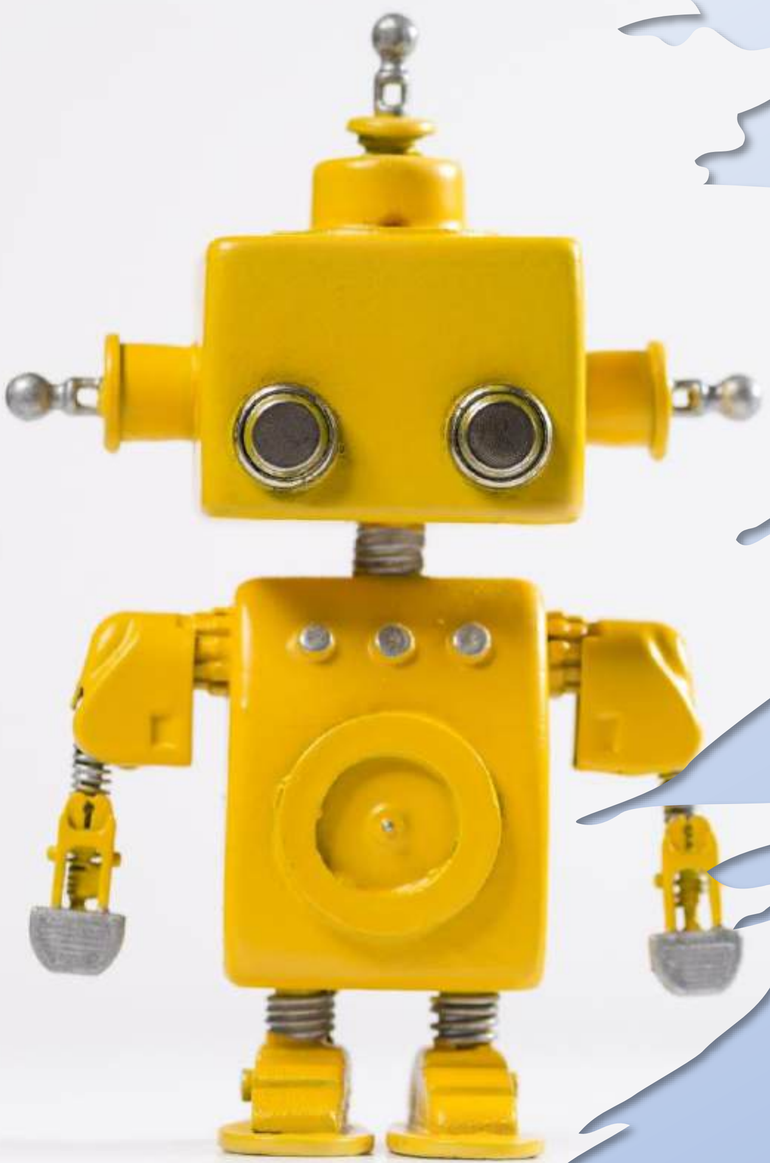
BATCH-B1 GROUP-A

GROUP MEMBERS:-

- Abhijeet Parmar(230002001)
- Abhitulya Mishra(230002002)
- Aditi Luniya(230002003)
- Aditya Gandhra(230002004)
- Aditya Naskar(230002005)
- Advait Varma(230002006)



INTRODUCTION



A line follower robot is a self-guided vehicle programmed to track and follow a designated visual path, typically marked by a contrasting black line on a white surface. We can also adjust it as a white line on a black surface as per our need. This is done by using infrared sensors.

WHY LINE FOLLOWER ROBOT?

Line follower robots offer a straightforward introduction to robotics, making them accessible to beginners.

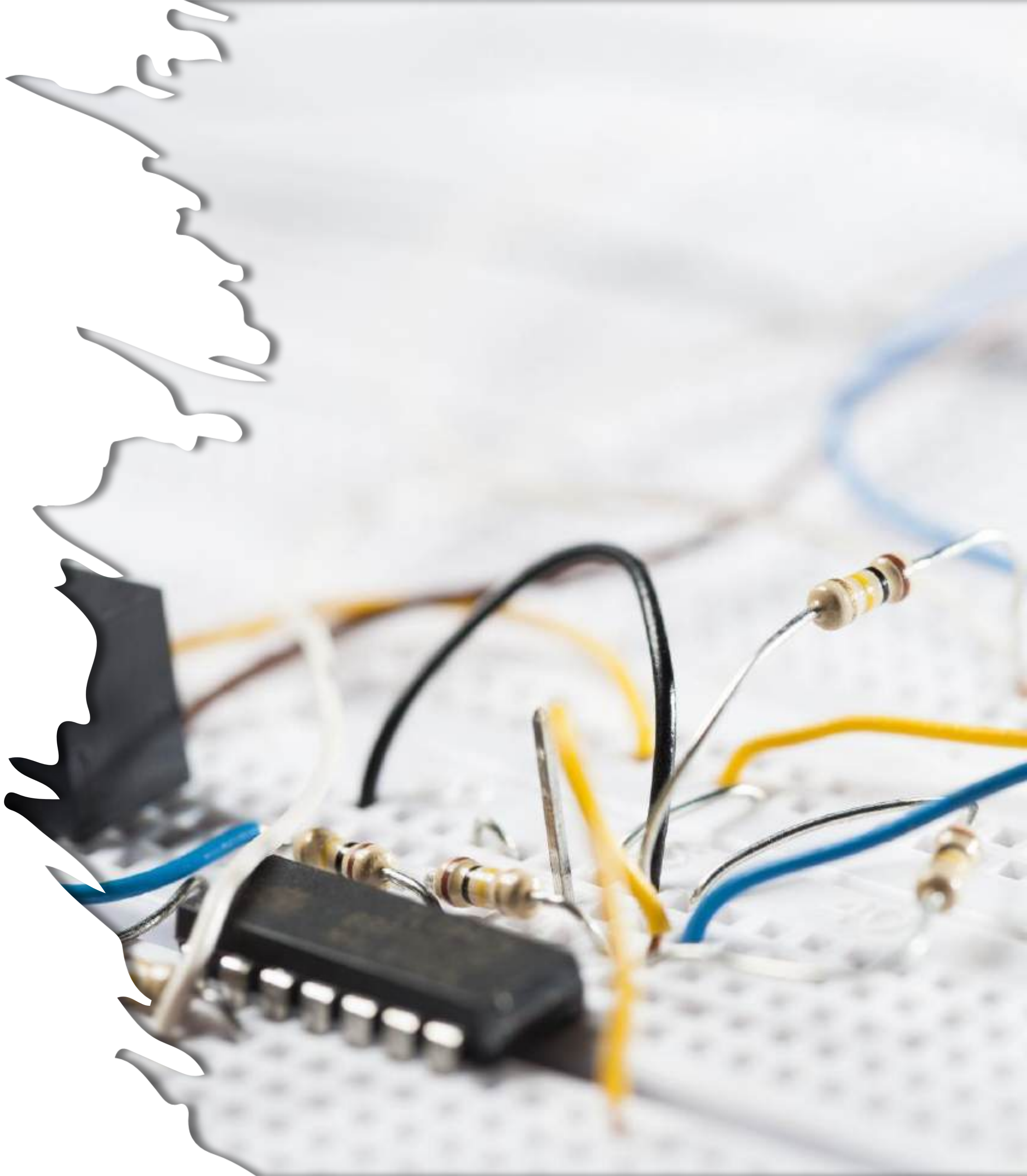
They can be adapted for various tasks and environments, from navigating mazes to warehouse logistics.

Working on line follower robots provides practical experience in STEM fields, fostering problem-solving skills.

They incorporate various technologies like sensors and microcontrollers, preparing individuals for complex robotics projects.

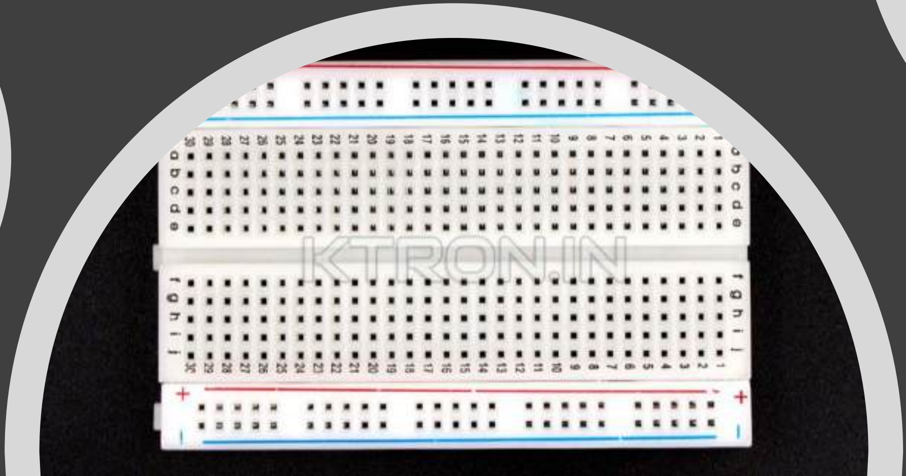
WORKING PRINCIPLE

In this line follower robot, we have used IR transmitters and receivers. When IR light falls on a white surface, it gets reflected back towards the IR receiver, generating some voltage changes that are analyzed by the Arduino. When IR light falls on a black surface, it gets absorbed by the black surface, and no rays are reflected back thus, the IR receiver doesn’t receive any rays. In this project, when the IR sensor senses a white surface, an Arduino gets HIGH as input, and when it senses a black line, an Arduino gets LOW as input. Based on these inputs, an Arduino Uno provides the proper output to control the line follower.

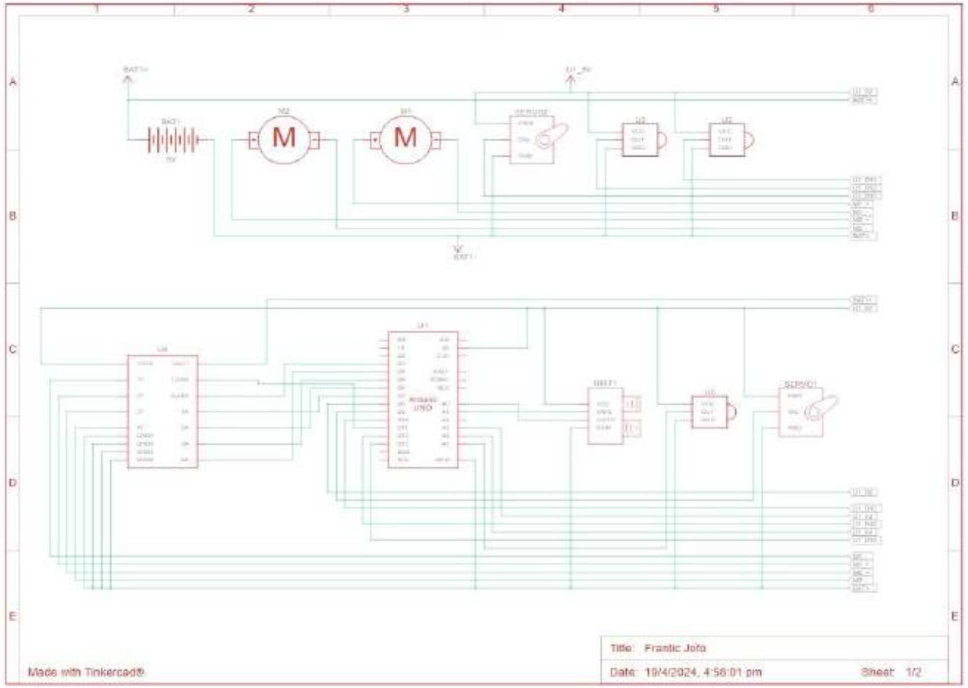


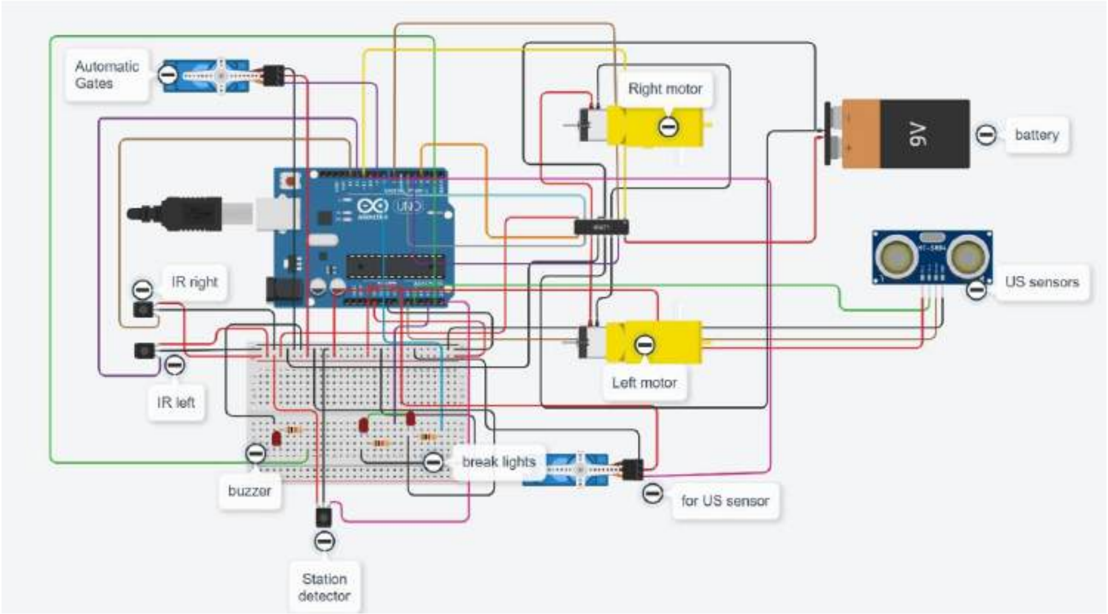
COMPONENTS USED

- Arduino board
- Jumper wire
- Laser cut chassis
- 3d printed wheels
- Motor driver module
- Servo motor
- Ultrasonic sensor
- Lithium ion battery
- Connecting wires
- Soldering wires
- Soldering iron
- Breadboard
- Infrared sensors
- LED lights



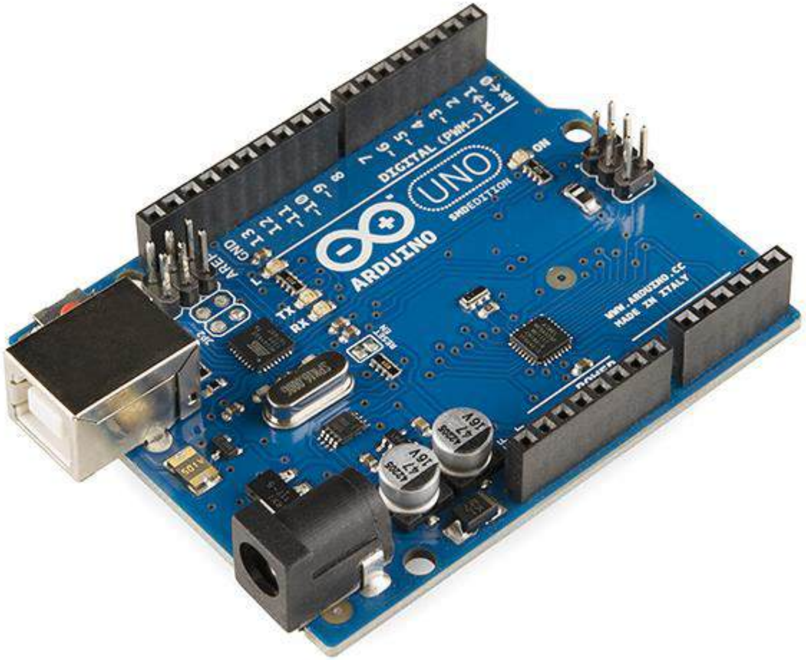
CIRCUIT DIAGRAM





ABOUT THE COMPONENTS

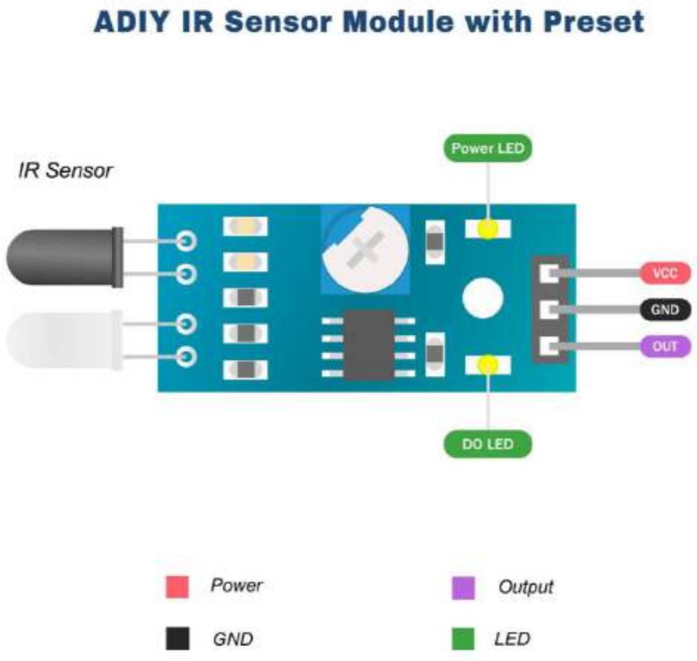
1. <u>ARDUINO UNO:</u>
   - The Arduino Uno is built around the ATmega328P microcontroller, offering 8-bit processing power. It's equipped with essential components like a crystal oscillator, serial communication interface, and voltage regulator. Featuring 14 digital I/O pins, including 6 PWM-capable pins, it also boasts six analog input channels. Connectivity options include USB, a power barrel jack, and an ICSP header, complemented by a reset button for convenience.
   - Programmable using the Arduino IDE, facilitated by a type B USB cable, the Uno offers flexibility in both power and programming. It can be powered via USB or an external voltage source within the range of 7 to 20 volts.



ABOUT THE COMPONENTS

2.<u>INFRARED SENSORS:</u>

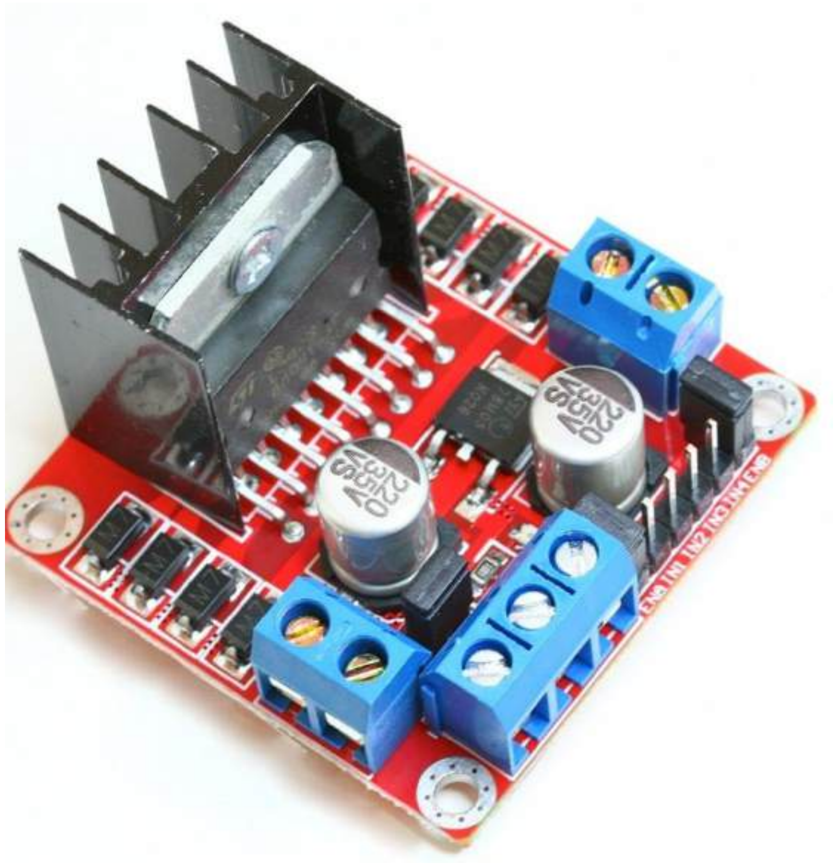
- An infrared sensor emits light to detect certain surroundings. In the infrared spectrum, all the objects radiate some form of thermal radiation that is invisible to our eyes, but an IR sensor can detect these radiations. Here, the IR LED is an emitter, and the IR photodiode is a detector.
- An IR LED emits the IR light, and the photodiode is sensitive to this IR light. When IR light falls on the photodiode, the output voltages and the resistances will change in proportion to the magnitude of the received IR light.



ABOUT THE COMPONENTS

3.<u>MOTOR DRIVER:</u>

- L298N is one of the easiest and best ways to control DC motors. It is the two-channel motor driver that can control the speed and spinning direction of DC motors. This L298N motor driver is a high-power motor driver module. It is used for driving DC and stepper motors. This motor driver consists of an L298N motor driver IC and a 78M05 5V voltage regulator, resistors, capacitor, power LED, and 5V jumper in an integrated circuit. When the jumper is placed, it enables the 78M05 voltage regulator.
- When the power supply is less than or equal to 12 volts, the voltage regulator will power the internal circuitry. When the power supply is more than 12 volts, then the jumper should not be placed and should give a separate 5 volts to power the internal circuitry. Here, the ENA and ENB pins are speed control pins for Motor A and Motor B. IN1 and IN2 and IN3 and IN4 are direction control pins for Motor A and Motor B.



ABOUT THE COMPONENTS

4.BATTERY OPERATED MOTOR:

- A BO motor is known as a battery operated motor. These motors are commonly used in hobby-grade projects where the user requires a small DC motor as a simple actuator.
- BO series linear motors provide good torque and rpm at lower operating voltages.
- The BO motors are available in single-shaft, dual-shaft, and DC plastic gear BO. These motors consume a low current. In this project, we have used four single-shaft BO motors.

ABOUT THE COMPONENTS

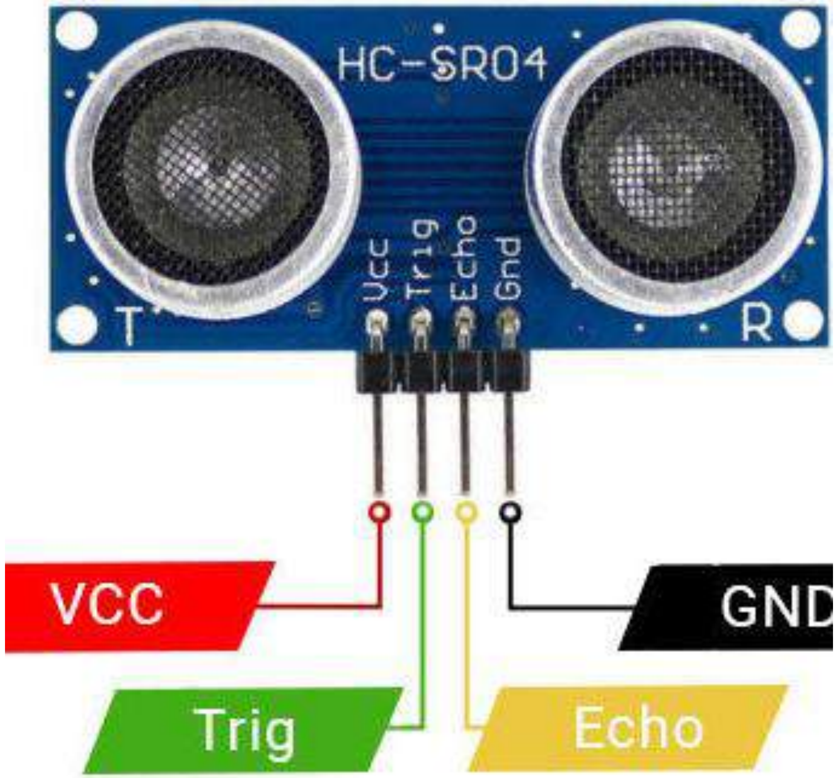
5.<u>LITHIUM ION BATTERY</u>:

- Lithium-ion batteries are rechargeable power sources that we often see in everyday gadgets like smartphones, laptops, and other electronics, as well as in electric cars.
- They're also becoming more popular in military equipment and aerospace technology.

ABOUT THE COMPONENTS

6.ULTRASONIC SENSOR:

- An ultrasonic sensor, commonly used in obstacle avoiders, works by emitting high-frequency sound waves and then detecting the echoes that bounce back off objects in its path. These sensors are crucial components in robotics and automated systems designed to navigate their environment safely.
- In an obstacle avoider application, the ultrasonic sensor serves as the “eyes” of the device, constantly sending out pulses of ultrasonic waves and measuring the time it takes for them to return after bouncing off nearby objects.
- By calculating the distance to obstacles based on the time it takes for the sound waves to travel and return, the sensor can provide real-time feedback to the system, enabling it to adjust its course or take evasive action to avoid collisions.



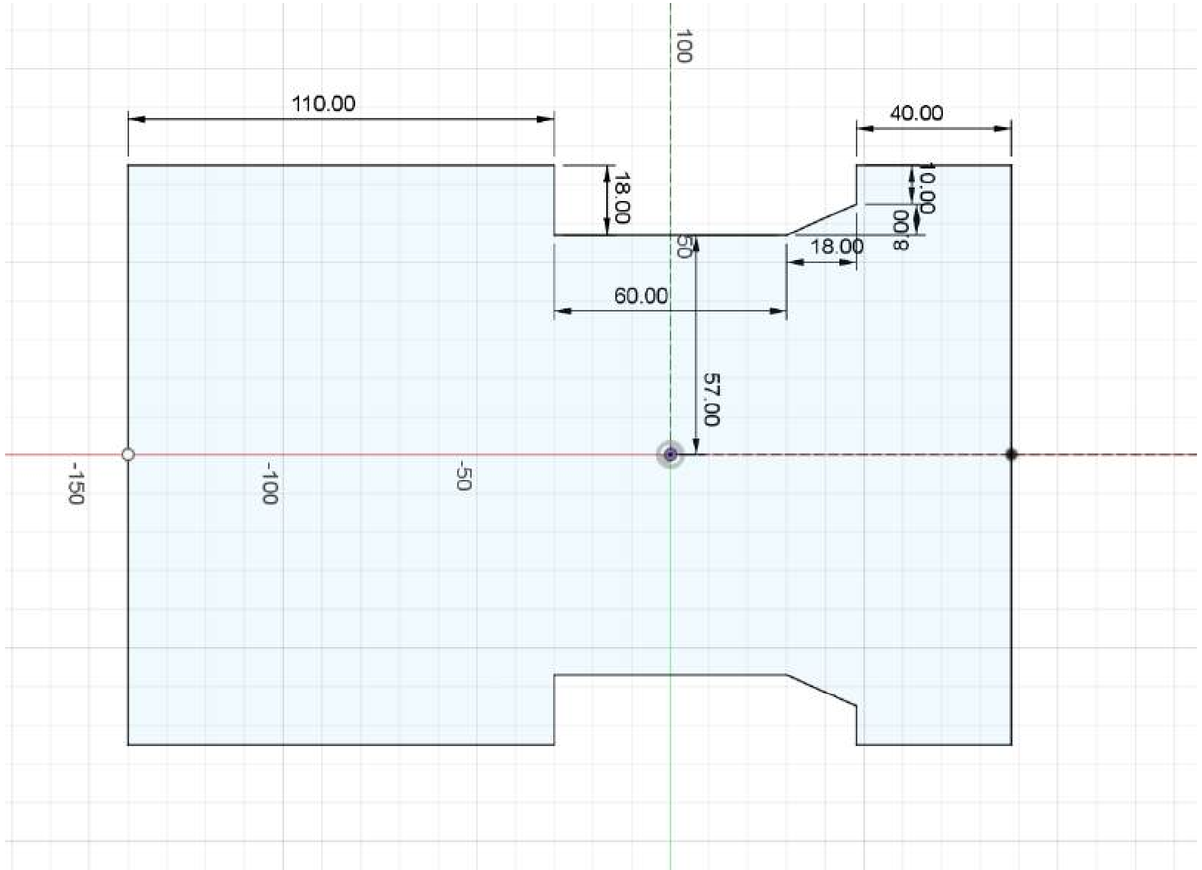
PROCEDURE

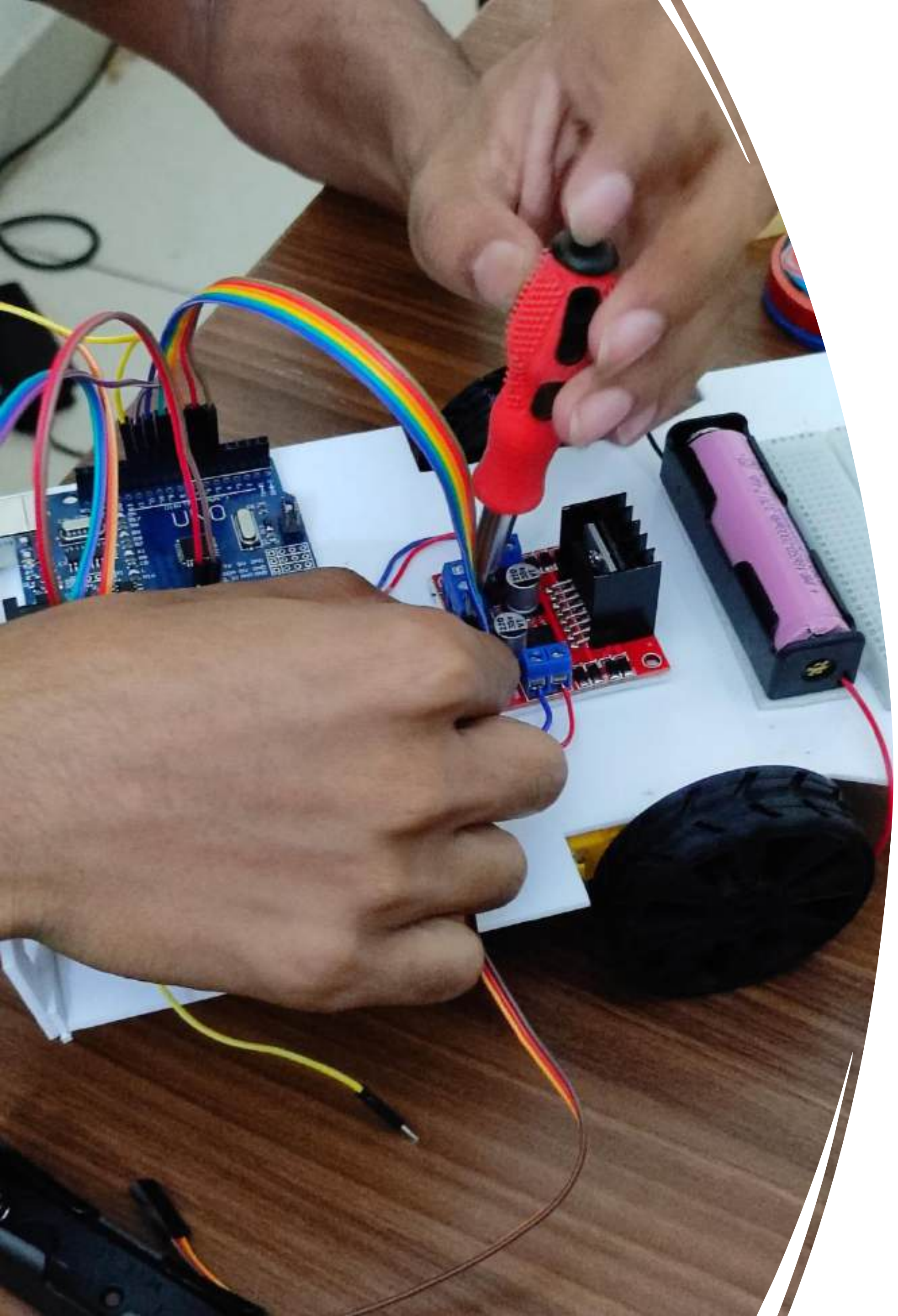
1) <u>3D PRINTING THE WHEELS:</u>
   a. Designing of wheels in fusion 360
   b. Exporting the design in stl file type
   c. Printing the wheels from 3d printer
   d. Mounting and testing

PROCEDURE

2) <u>LASER PRINTING OF CHASSIS:</u>

a. Create a sketch for your chassis in fusion 360
b. Export the file in dxf file type
c. Choose a suitable material for laser cutting, like acrylic or plywood, and set up your laser cutter according to the material specifications.
d. Load the material onto the cutter bed, import the design into the laser cutter software, and adjust cutting settings.
e. Perform a test cut to ensure accuracy before initiating the full cutting process.

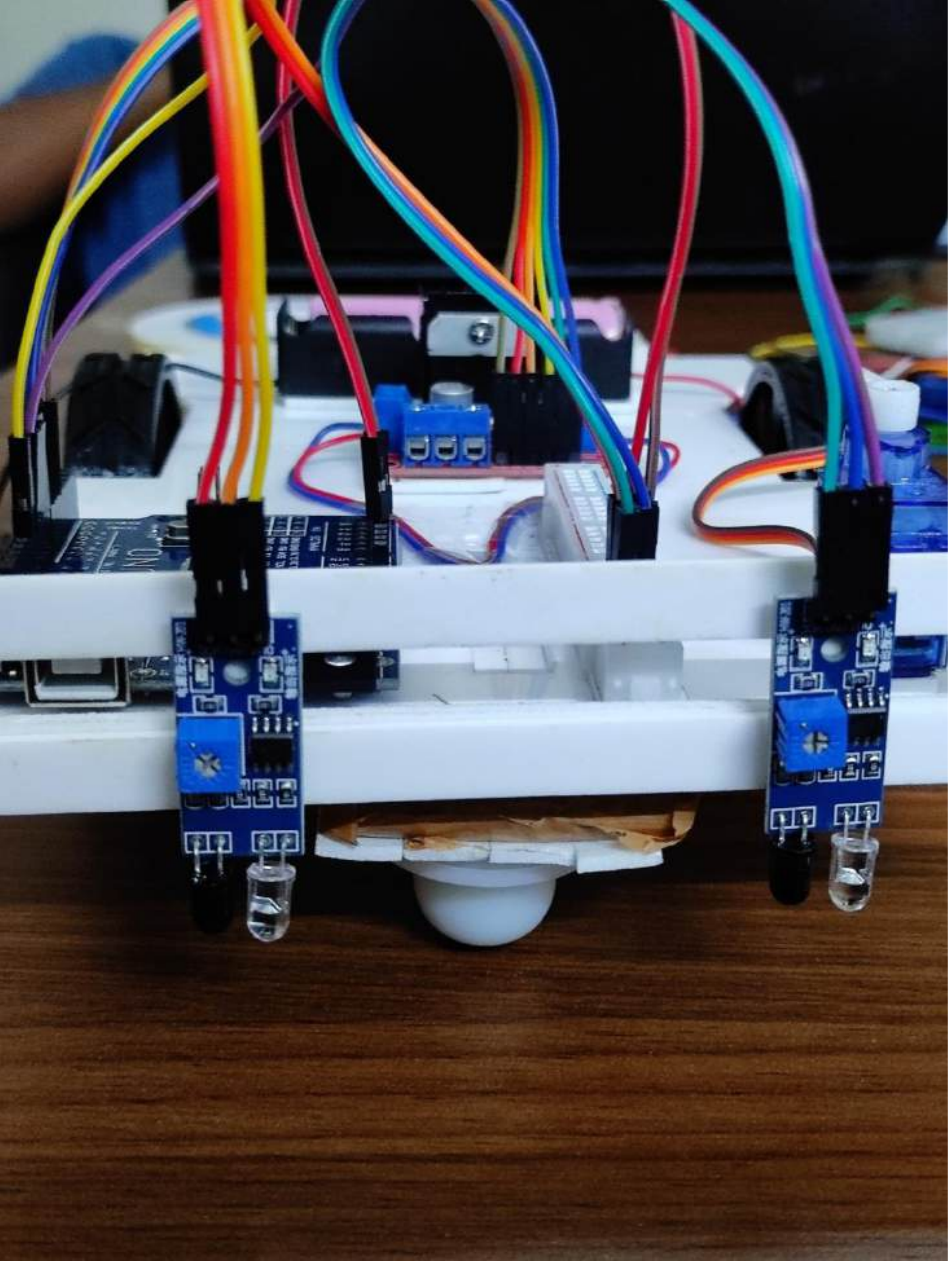




PROCEDURE

3) <u>CONNECTIONS AND WIRING</u>:

- All the material is loaded onto the chassis board and is placed such that the wiring is as clean as possible.
- The connections are made in accordance with the circuit diagram shown in the beginning.
- Motors are attached below the chassis board to maximize the space available on the top of the board for the other components.
- IR sensors are placed at the front of the line follower robot.
- Ultrasonic sensor is also mounted at the front of the robot for obstacle avoiding.



PROCEDURE

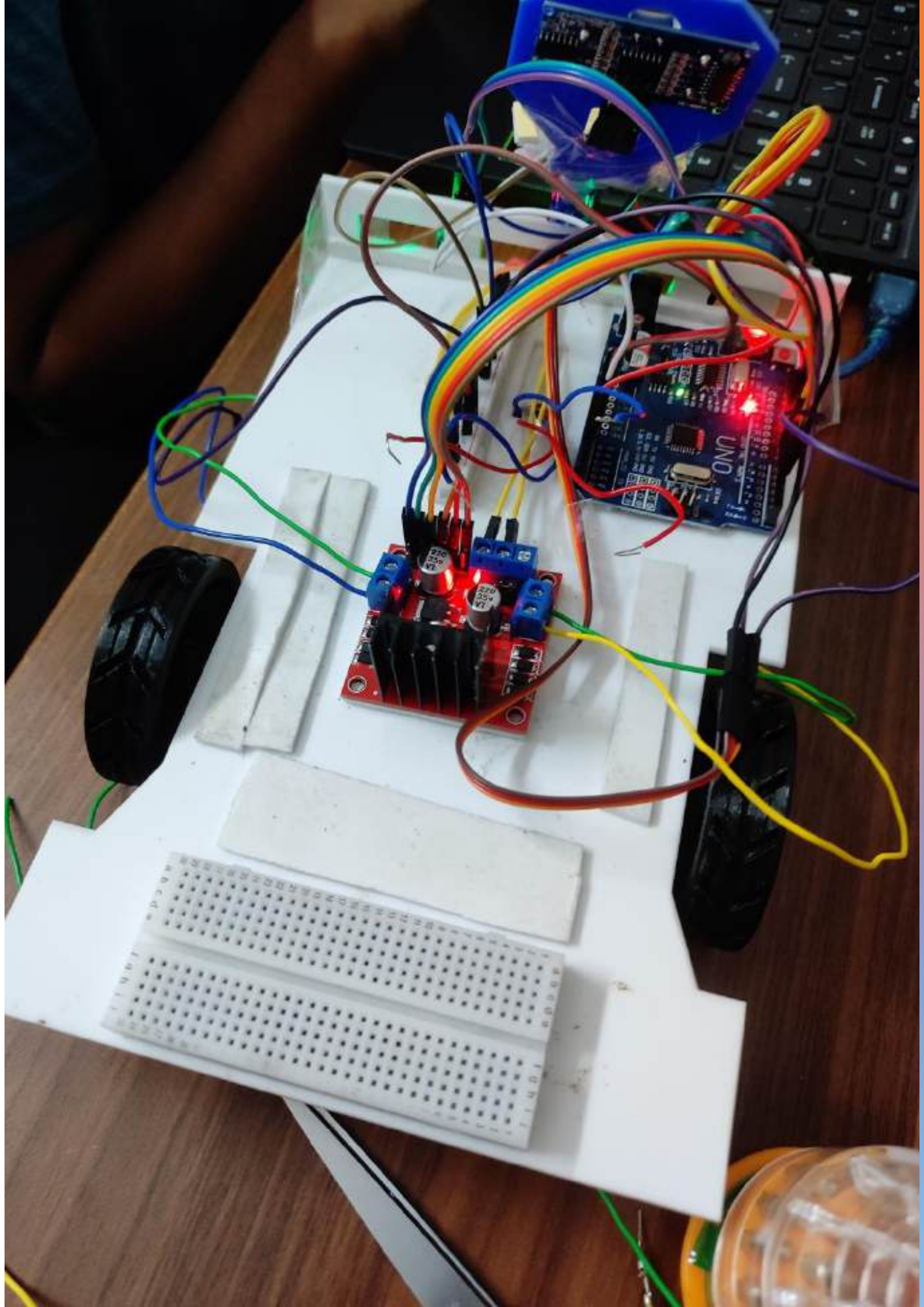
4) <u>PLACEMENT OF SENSORS:</u>

- The placement of sensors is a crucial consideration in optimizing the performance of a line follower robot. In a typical two-wheeled configuration, Infrared sensors are often strategically positioned beneath the chassis.
- Placing one sensor on the left and another on the right side, oriented to straddle the line, allows the robot to detect deviations from the path.
- The precise alignment and spacing of these sensors are essential for the robot's ability to make accurate and timely adjustments. Factors such as the robot's width, the thickness of the line, and the desired responsiveness to turns all influence the optimal sensor placement.
- Additionally, ensuring the sensors are at an appropriate height above the surface contributes to the reliability of the line-following behavior.
- Experimentation and testing play a key role in fine-tuning the sensor placement for optimal performance in various environments.

PROCEDURE

5) CALIBRATION AND TESTING:

Calibration and testing are crucial phases in the development of a line follower robot, ensuring optimal performance and accuracy in following predefined paths. Calibration involves fine-tuning the robot's sensors, such as infrared sensors or cameras, to accurately detect and respond to the line on the surface. This process may include adjusting sensor sensitivity, threshold values, and ensuring proper alignment.



CODE

```
#include <NewPing.h>
#include <Servo.h>
#define TRIGGER_PIN A1
#define ECHO_PIN A0
#define max_distance 50
#define in1 7  //Motor1  L293 Pin in1
#define in2 6  //Motor1  L293 Pin in1
#define in3 5  //Motor2  L293 Pin in1
#define in4 4  //Motor2  L293 Pin in1
int breakled = 8;
#define ir3 A5
#define buzzer A4
```

```
#define R_S 13  //ir sensor Righ
t#define L_S 12  //ir sensor Left
#define enb 3
#define ena 11
Servo servo;
Servo gate;
NewPing sonar(TRIGGER_PIN, ECHO_PIN,
max_distance);
int distance = 0
;int leftDistance;
int rightDistance;
boolean object;
```

CODE

```
void setup() {
pinMode(L_S, INPUT);
servo.attach(10);
servo.write(90);
gate.attach(9);
gate.write(0);
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
pinMode(in3, OUTPUT);
pinMode(in4, OUTPUT);
pinMode(ena, OUTPUT);
pinMode(enb, OUTPUT);
pinMode(ir3, INPUT);
pinMode(buzzer, OUTPUT);
pinMode(R_S, INPUT);
analogWrite(ena, 50);
analogWrite(enb, 50);
pinMode(breakled, OUTPUT);  d
elay(1000);
}
```

CODE

```
void loop() {
if (digitalRead(ir3) == LOW) {
Stop();
open_gates();
forward();
delay(20);
 }  if ((digitalRead(R_S) == 0) &&
 (digitalRead(L_S) == 0)) {
objectAvoid();
}
else if ((digitalRead(R_S) == 1) &&
(digitalRead(L_S) == 0)) {
objectAvoid();
turnRight();
}
 //if Right Sensor is White and Left Sensor
is Black then it will call turn Left function
else if ((digitalRead(R_S) == 0) &&
(digitalRead(L_S) == 1)) {
objectAvoid();
turnLeft();
}
```

CODE

```
//if Right Sensor is Black and Left Sensor is Black then it will call turn Stop function

else if ((digitalRead(R_S) == 1) && (digitalRead(L_S) == 1)) {

Stop(); }

delay(10);

}

void open_gates() {

for (int pos = 0;

pos <= 90;

pos += 1) {

gate.write(pos);

delay(10);}

delay(5000);
```

```
for (int i = 1; i < 100 ; i++) {

if (i % 2 == 1) {

digitalWrite(buzzer, HIGH);

 } else {

digitalWrite(buzzer, LOW);

}

delay(50);

}

digitalWrite(buzzer,LOW);

for (int pos = 90; pos >= 0; pos -= 1) {
gate.write(pos);

delay(10);

}}
```

CODE

```
void forward() {            //forward

digitalWrite(in1, LOW);   //Left Motor backword Pin  digitalWrite(in2, HIGH);  //Left Motor forward Pin  digitalWrite(in3, HIGH);  //Right Motor forward Pin  digitalWrite(in4, LOW);  //Right Motor backword Pin

  digitalWrite(breakled, LOW);}

void turnLeft() {           //turnRight

digitalWrite(in1, LOW);   //Left Motor backword Pin  digitalWrite(in2, HIGH);  //Left Motor forward Pin  digitalWrite(in3, LOW);   //Right Motor forward Pin  digitalWrite(in4, HIGH);  //Right Motor backword Pin

  digitalWrite(breakled, LOW);

delay(350);}

void turnRight() {          //turnLeft

Pin  digitalWrite(in2, LOW);   //Left Motor forward Pin  digitalWrite(in3, HIGH);  //Right Motor forward Pin  digitalWrite(in4, LOW);  //Right Motor backword Pin

  digitalWrite(breakled, LOW);

delay(350);}


void Stop() {               //stop

digitalWrite(in1, LOW);  //Left Motor backword Pin  digitalWrite(in2, LOW);  //Left Motor forward Pin  digitalWrite(in3, LOW);  //Right Motor forward Pin  digitalWrite(in4, LOW);  //Right Motor backword Pin
digitalWrite(breakled, HIGH);

}
```

CODE

```
void objectAvoid() {
distance = getDistance();
if (distance <= 20) {    //stop
Stop();
lookRight();
lookLeft();
delay(100);
 if (rightDistance <= leftDistance) {
//left
object = true;
 turn();}
else {
//right
object = false;
turn();}
 delay(100);  }
else {
//forward
 forward();  }}
int getDistance() {
delay(50);
int cm = sonar.ping_cm();
if (cm == 0) {
cm = 100;  }
return cm;}
```

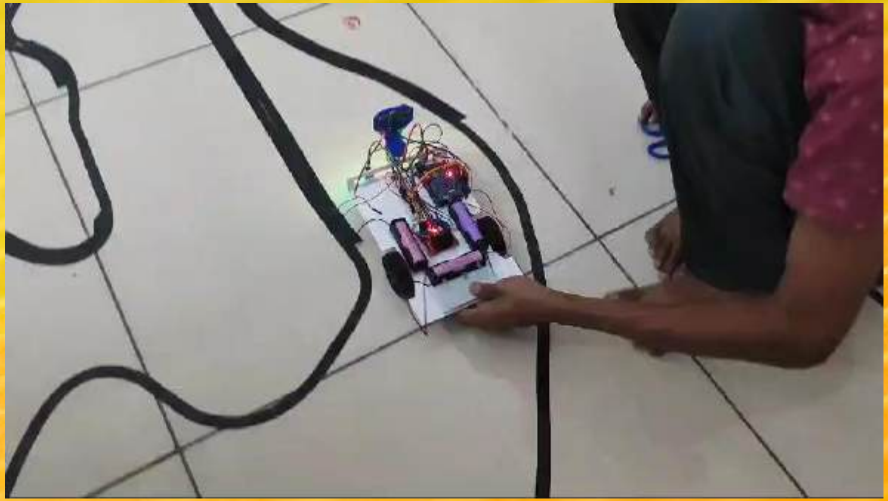
CODE

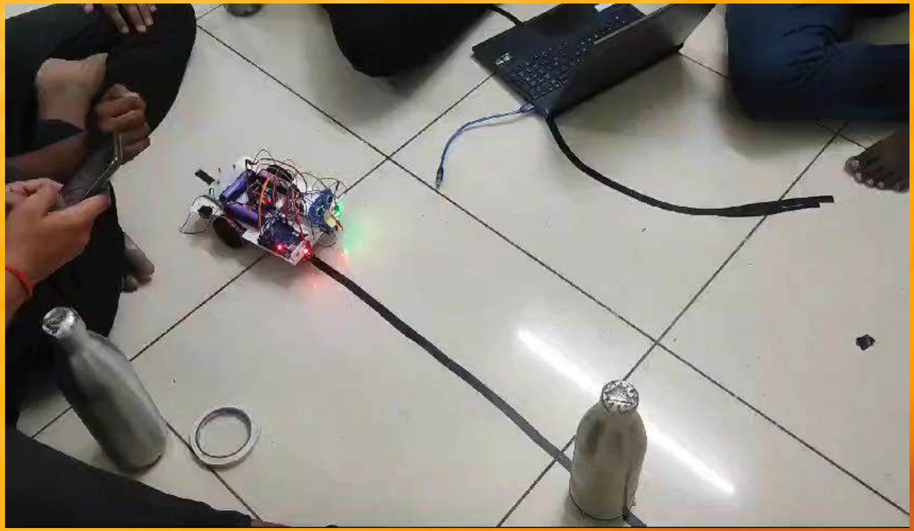
```
int lookLeft() {
//lock left
servo.write(150);
delay(500);
leftDistance = getDistance();
delay(100);
servo.write(90);
return leftDistance;
delay(100);
}
int lookRight() {
//lock right
servo.write(30);
delay(500);
rightDistance = getDistance();
delay(100);
servo.write(90);
return rightDistance;
delay(100);}
```

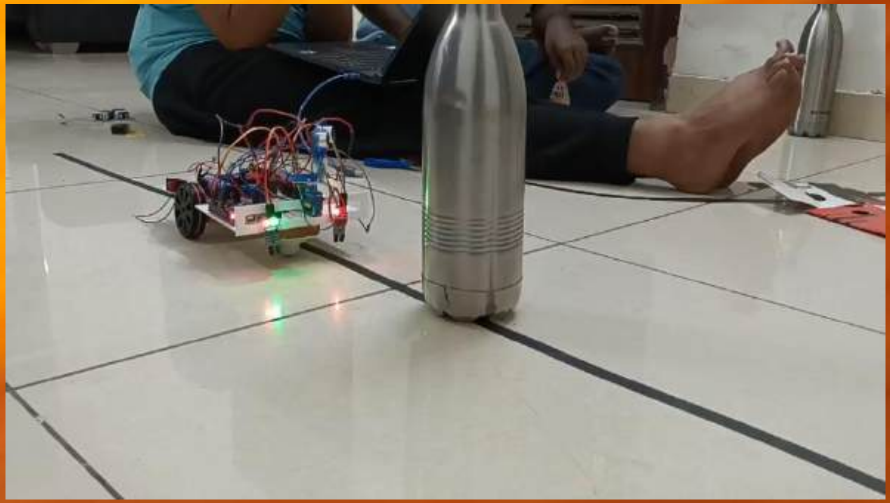
CODE

```
void turn() {
if (object == false) {
turnRight();
delay(600);
forward();
delay(6000);
turnLeft();
delay(600);
if (digitalRead(R_S) == 1) {
loop();
} else {
forward();   }
} else {
turnLeft();
delay(600);
forward();
delay(6000);
turnRight();
delay(1400);
 if (digitalRead(L_S) == 1) {
loop();
} else {     f
orward();   }
  }
}
```

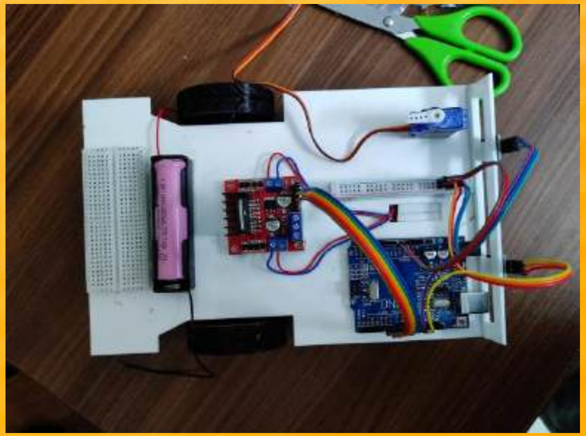
VIDEOS

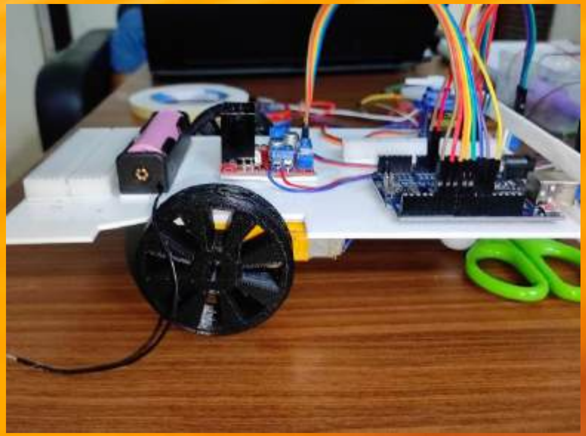


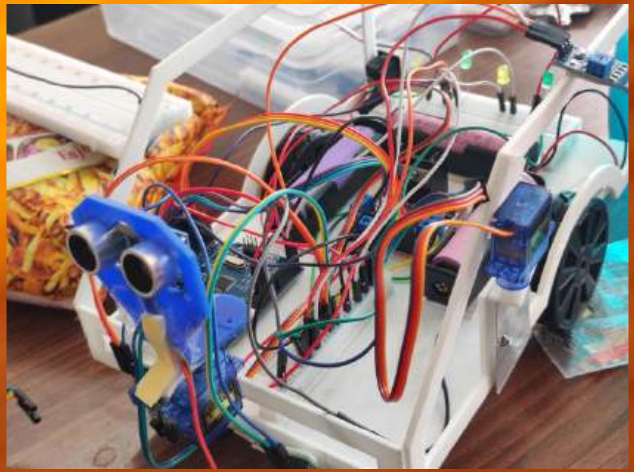




PHOTOS







CONCLUSION

In summary, the line follower robot is an impressive blend of technology and usefulness. It can follow paths all by itself using sensors and smart algorithms, showing how far robotics has come. Plus, it has tons of practical uses, from making factories more efficient to helping students learn about robotics. As technology gets better, these robots will only improve, becoming even more precise and versatile. So, the future looks bright for line follower robots, as they keep pushing the boundaries of what's possible with technology.


THANK YOU