Project 1:

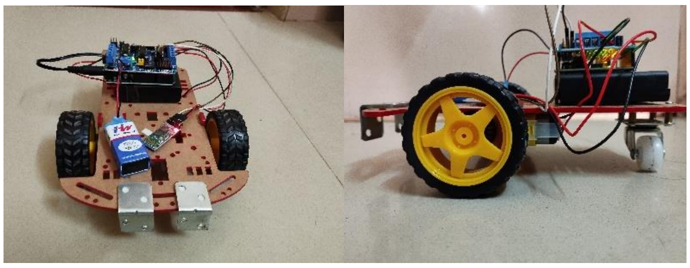
Title: <u>BLCK LINE FOLLOWING ROBOT USING</u>

Methodology, Software used, Tools: A black line follower robot uses sensors to detect a black line on a contrasting surface and move automatically along it.

- Arduino Uno
- IR sensor
- L293D motor driver
- BO motor
- Wheels
- Lithium-ion battery
- Jumper cables

Industrial application / Societal application: Warehouse Automation, Assembly Line Operations, Quality Inspection, Inventory Management, Cleanroom Operations.

Project proof in image form:



A **black line-following and Bluetooth-controlled robot** is a combination of two functionalities: autonomous line following and manual control via Bluetooth. Here's an overview of its main components and functions:

1. **Line Following Mechanism**:
   - The robot detects and follows a black line on a contrasting surface using **infrared (IR) sensors**. These sensors differentiate between the black line and the surface, sending signals to the microcontroller, which adjusts the motors to keep the robot on track.
   - The robot adjusts its movement based on sensor feedback. If the left sensor detects the black line, the robot steers left, and if the right sensor detects the line, it steers right.
2. **Bluetooth Control**:
   - The robot can be remotely controlled using a **Bluetooth module** (e.g., HC-05 or HC-06) connected to a microcontroller.
   - A smartphone or Bluetooth-enabled device acts as a controller, sending commands to the robot (e.g., move forward, backward, left, or right). These commands override the line-following mode when activated.
3. **Microcontroller**:

- Typically, an **Arduino** or **Raspberry Pi** is used as the brain of the robot, handling input from both the IR sensors and Bluetooth module, then controlling the motors.
-

```
// Pin Definitions
#define leftSensor A0
#define rightSensor A1

// Motor Pins
#define leftMotor1 5
#define leftMotor2 6
#define rightMotor1 9
#define rightMotor2 10

// Enable Pins
#define enableLeft 3
#define enableRight 4

// Bluetooth Pins (RX, TX)
#include <SoftwareSerial.h>
SoftwareSerial bluetooth(10, 11);  // RX, TX

void setup() {
  // Initialize motor pins as outputs
  pinMode(leftMotor1, OUTPUT);
  pinMode(leftMotor2, OUTPUT);
  pinMode(rightMotor1, OUTPUT);
  pinMode(rightMotor2, OUTPUT);
  pinMode(enableLeft, OUTPUT);
  pinMode(enableRight, OUTPUT);

  // Set initial motor speed (can adjust)
  analogWrite(enableLeft, 200);
  analogWrite(enableRight, 200);

  // Initialize sensor pins as inputs
  pinMode(leftSensor, INPUT);
  pinMode(rightSensor, INPUT);

  // Initialize Bluetooth communication at 9600 baud rate
  bluetooth.begin(9600);
  Serial.begin(9600);  // For debugging
}

void loop() {
  // Read sensor values
  int leftValue = analogRead(leftSensor);
  int rightValue = analogRead(rightSensor);

  // Bluetooth control
  if (bluetooth.available()) {
    char command = bluetooth.read();
    bluetoothControl(command);
  } else {
    // Autonomous Line-Following Mode
    lineFollowing(leftValue, rightValue);
  }
}

void bluetoothControl(char command) {
  switch (command) {
    case 'F':  // Move forward
      moveForward();
      break;
    case 'B':  // Move backward
      moveBackward();
      break;
    case 'L':  // Turn left
      turnLeft();
      break;
    case 'R':  // Turn right
      turnRight();
      break;
    case 'S':  // Stop
      stopMotors();
      break;
    default:
      stopMotors();
      break;
  }
}

void lineFollowing(int leftValue, int rightValue) {
  // Line following logic
  if (leftValue < 500 && rightValue < 500) {
    moveForward();  // Move straight if both sensors detect the line
  } else if (leftValue < 500) {
    turnLeft();  // Turn left if the left sensor detects the line
  } else if (rightValue < 500) {
    turnRight();  // Turn right if the right sensor detects the line
  } else {
    stopMotors();  // Stop if no line is detected
  }
}

// Motor Control Functions
```

```
void moveForward() {
  digitalWrite(leftMotor1, HIGH);
  digitalWrite(leftMotor2, LOW);
  digitalWrite(rightMotor1, HIGH);
  digitalWrite(rightMotor2, LOW);
}

void moveBackward() {
  digitalWrite(leftMotor1, LOW);
  digitalWrite(leftMotor2, HIGH);
  digitalWrite(rightMotor1, LOW);
  digitalWrite(rightMotor2, HIGH);
}

void turnLeft() {
  digitalWrite(leftMotor1, LOW);
  digitalWrite(leftMotor2, LOW);
  digitalWrite(rightMotor1, HIGH);
  digitalWrite(rightMotor2, LOW);
}

void turnRight() {
  digitalWrite(leftMotor1, HIGH);
  digitalWrite(leftMotor2, LOW);
  digitalWrite(rightMotor1, LOW);
  digitalWrite(rightMotor2, LOW);
}

void stopMotors() {
  digitalWrite(leftMotor1, LOW);
  digitalWrite(leftMotor2, LOW);
  digitalWrite(rightMotor1, LOW);
  digitalWrite(rightMotor2, LOW);
}

```