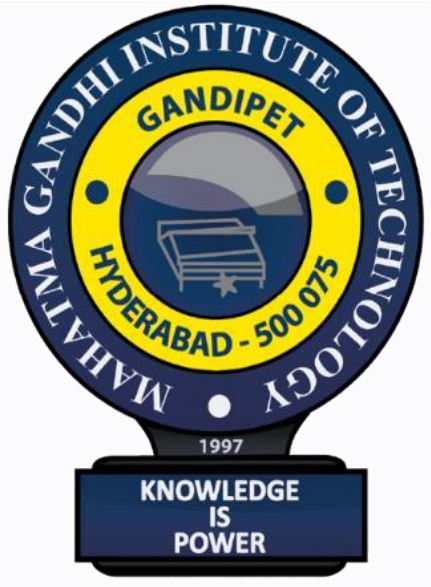
LINE FOLLOWING ROBOT USING ARDUINO

K. SAIKIRAN

K. HARSHITH REDDY

K. SHASHANK

N.G.S. YOGEESWARA REDDY



Department Of Electronics and CommunicationEngineering

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

Chaitanya Bharathi P.O., Gandipet, Hyderabad – 500075

2024

LINE FOLLOWING ROBOT USING ARDUINO

MINI PROJECT REPORT SUBMITTED

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF TECHNOLOGY IN ELECTRONICS AND COMMUNICATION ENGINEERING

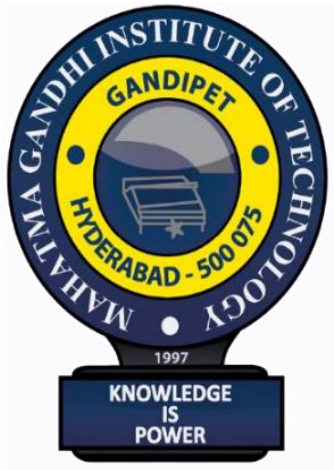
BY

K. SAIKIRAN

K. HARSHITH REDDY

K. SHASHANK

N.G.S. YOGEESWARA REDDY



Department Of Electronics and Communication Engineering

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

Chaitanya Bharathi P.O., Gandipet, Hyderabad –500075

2024

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

Chaitanya Bharathi P.O., Gandipet, Hyderabad-500075

Department Of Electronics and Communication Engineering

CERTIFICATE

Date: /2024

This is to certify that the mini project work entitled **"LINE FOLLOWING ROBOT USING ARDUINO"** is a bonafide work carried out by

K. SAIKIRAN

K. HARSHITH REDDY

K. SHASHANK

N.G.S. YOGEESWARA REDDY

in partial fulfilment of the requirements for the degree of BACHELOR OF TECHNOLOGY in ELECTRONICS AND COMMUNICATION ENGINEERING by the Jawaharlal Nehru technological university, Hyderabad during the academic year 2024-25. The results embodied in this report have not been submitted by any otheruniversity or institution for the award of any degree or diploma.

----------	----------
MR.P.VINOD REDDY	DR S P Singh
Assistant Professor	**Head of Dept**

ACKNOWLEDGEMENT

We express our deep sense of gratitude to our Guide MR.P.VINOD REDDY, M.G.I.T, Hyderabad, for her valuable guidance and encouragement in carrying out our Project.

We are highly indebted to our Professor Coordinators Associate Professor S Srinivasa Rao, Associate Professor Dr.G.MADHAVI, Assistant Professor K.RAGHU, Electronics and Communication Engineering Department, who has given us all the necessary technical guidance in carrying out this project.

We wish to express our sincere thanks to Dr S.P. Singh, Head of the Department of Electronics and Communication Engineering, M.G.I.T., for permitting us to pursue our project and encouraging us throughout theProject.

Finally, we thank all the people who have directly or indirectly helped us through the course of our Project.

K. SAIKIRAN (4F6)
K. HARSHITH REDDY (4F4)
K. SHASHANK (4F8)
N.G.S. YOGEESWARA REDDY (20F9)

ABSTRACT

A line-following robot is an autonomous robotic system capable of detecting and following a predefined path marked on a surface. This path is typically represented by a contrasting line, such as a black line on a white surface or vice versa, and serves as a guide for the robot's navigation. These robots are widely used in various fields, including industrial automation, logistics, and educational projects, as they showcase fundamental principles of robotics, electronics, and control systems.

The primary components of a line-following robot include sensors, a control unit, actuators, and a power supply. Infrared (IR) or optical sensors are commonly employed to detect the line by identifying differences in light intensity between the line and the surrounding surface. Multiple sensors are often arranged in an array to provide accurate feedback on the robot's position relative to the line. This sensor feedback is sent to a microcontroller or other control unit, which processes the data and determines the necessary actions to maintain alignment with the line.

The control system of a line-following robot typically implements algorithms such as proportional, integral, and derivative (PID) control. These algorithms ensure smooth and precise movement by continuously adjusting the robot's speed and direction based on deviations from the line. The actuators, usually in the form of motors, execute these adjustments by driving the robot's wheels or other mobility mechanisms.

The advantages of line-following robots lie in their simplicity, efficiency, and versatility. They are relatively easy to design and implement, making them ideal for beginners in robotics. Their reliance on predefined paths ensures predictable and controlled operation

Table of Contents

LIST OF FIGURES

CHAPTER 1

INTRODUCTION:

A line-following robot is an autonomous robotic system designed to navigate a predefined path marked on the ground, typically represented as a high-contrast line such as black on a white surface or vice versa. These robots are equipped with sensors, usually infrared (IR) or optical, that detect variations in light intensity to identify the line's position. The data collected by the sensors is processed by a microcontroller, which uses it to control the robot's movement and ensure it stays aligned with the path. The robot’s control system is the core of its functionality. It interprets the sensor data and sends commands to the actuators, such as motors, to adjust speed and direction. Advanced control methods, such as proportional-integral-derivative (PID) algorithms, are often employed to enhance navigation precision and reduce deviations. This enables the robot to follow straight or curved lines with efficiency and stability, ensuring reliable operation even in complex paths.

Line-following robots are widely used in educational, industrial, and service sectors. In educational settings, they serve as an excellent tool for teaching robotics, programming, and electronics, offering hands-on learning experiences. Industrial applications include automating tasks such as material transport on factory floors and guiding goods along predefined routes in warehouses. In healthcare, these robots are employed for delivering medical supplies in hospitals, reducing human effort and ensuring timely operations.

The construction of a line-following robot involves integrating key components such as sensors, a microcontroller, actuators, and a power source. Basic models may follow simple paths, while advanced designs can handle intersections or dynamic changes in the line. Despite their simplicity, line-following robots are highly effective for repetitive and structured tasks, making them a cost-efficient solution for automation.

While these robots are limited by their dependence on a visible path, advancements in sensor technology and control algorithms are continuously expanding their capabilities. They are a testament to the power of simple yet impactful automation, demonstrating how robotics can solve real-world problems efficiently. The development of line-following robots fosters innovation and inspires further exploration into autonomous systems.

The simplicity of their design and operation makes line-following robots an accessible entry point into robotics for beginners. Meanwhile, their versatility allows experienced developers to experiment with advanced features, such as multi-robot coordination and wireless communication. This dual appeal has cemented their place in both academic and professional domains.

Future developments in line-following robots are likely to focus on improving energy efficiency, reducing costs, and enhancing their ability to function in unstructured environments. Innovations in sensor technology and control systems will make these robots even more reliable and adaptable. As automation becomes integral to industries worldwide, line-following robots will continue to play a vital role in bridging the gap between manual and fully autonomous systems.

AIM OF THE PROJECT:

The aim of a line-following robot is to autonomously navigate a predefined path by detecting and following a contrasting line using sensors. It is designed to streamline repetitive tasks, such as material transport and guided navigation, in structured environments. Additionally, it serves as an educational tool for learning robotics and automation principles.

SIGNIFICANCE:

- Automation in Repetitive Tasks: Simplifies repetitive tasks such as material handling and transportation in industries.
- Educational Value: Provides hands-on learning opportunities for students to understand robotics, electronics, and programming.
- Efficiency: Reduces human effort and increases productivity in structured environments like factories and warehouses.
- Cost-Effective Solution: Offers a low-cost entry point into robotics and automation.

- Precision: Ensures accurate navigation along predefined paths, minimizing errors.
- Industrial Applications: Widely used in assembly lines and conveyor systems for efficient operations.
- Warehouse Management: Assists in inventory movement, improving organization and reducing delays.
- Healthcare Support: Delivers medical supplies in hospitals, ensuring timely and safe transport.
- Environmental Adaptability: Advanced models can navigate intersections and handle dynamic paths.
- Innovation and Creativity: Encourages innovation in developing autonomous systems and problem-solving skills.
- Energy Efficiency: Optimized for low power consumption, making them sustainable for long-term use.
- Scalability: Can be scaled up for more complex tasks by integrating advanced technologies like AI and IoT.
- Safety: Reduces the risk of accidents by automating tasks in hazardous environments.

METHODOLOGY:

A line-following robot is an autonomous machine designed to detect and follow a pre-defined line (usually black on a white surface or vice versa) using sensors and programmed logic. Below is the step-by-step methodology:

1. Define Objectives and Requirements

- Determine the purpose of the robot (e.g., educational, industrial application).
- Choose the type of line to follow (black or white, solid or dashed).
- Identify environmental conditions (lighting, surface material).

2. Components Selection

i. Microcontroller: Arduino, Raspberry Pi, or similar to process data.

ii. IR Sensors: To detect the line by sensing reflected light intensity.

iii. Motors and Motor Driver: DC motors controlled via an L298N motor driver.

iv. Chassis: A platform to mount components.

v. Power Supply: Batteries to power the robot.

vi. Wheels and Gear System: For movement and speed adjustment.

3. Circuit Design and Assembly

1. Sensor Placement:
    a. Place multiple IR sensors at the front of the robot for detecting the line and surroundings.
    b. Align them so each sensor can independently detect line transitions.
2. Motor Driver Connection:
    a. Connect the motor driver to the microcontroller and DC motors.
    b. Ensure PWM pins are configured for speed control.
3. Power System: Wire the power supply to the motor driver and microcontroller.
4. Microcontroller Input/Output:
    a. Connect IR sensor outputs to input pins of the microcontroller.
    b. Connect motor driver inputs to the microcontroller for motor control.

4. Algorithm Development

Write a control algorithm in the microcontroller (e.g., Arduino IDE).
Core Steps:

i. Sensor Reading: Continuously read the IR sensor values.

ii. Decision Making:

iii. If the centre sensor detects the line, continue forward.

iv. If the left sensor detects the line, turn left.

v. If the right sensor detects the line, turn right.

Motor Control: Use PWM signals to adjust motor speeds for smooth turns and forward movement.

5. Calibration

Calibrate the IR sensors to differentiate between the line and the background.
Adjust sensor sensitivity to avoid false readings due to ambient light.

6. Testing and Debugging

Test the robot on a sample track with curves, straight sections, and intersections.
Debug and modify the code or sensor alignment for consistent performance.
Check motor responses for correct turns and speed adjustments.

7. Optimization

Fine-tune sensor thresholds and motor speeds for smoother line following.
Implement features like obstacle detection or path memory if required.

8. Deployment

Use the robot in the intended environment (e.g., an industrial assembly line).
Regularly check for sensor wear or power issues.

SCOPE:

1. **Industrial Automation**: Material transportation in factories and warehouses.
2. **Education**: Learning tool for robotics, programming, and electronics.
3. **Logistics and Warehousing**: Inventory management and order picking.
4. **Healthcare**: Medicine and supply delivery within hospitals.
5. **Agriculture**: Precision farming tasks like planting and fertilizing.

6. **Retail and Service Industries**: Self-guided carts in malls, airports, and large venues.
7. **Military Applications**: Logistics support in controlled environments.
8. **Robotic Toys and Entertainment**: Interactive and educational robotic toys.
9. **Event Management**: Visitor guidance and item delivery during events.

ORGANISATION OF WORK:

Software requirements:

1. Arduino IDE

Hardware requirements:

1. ARDUINO UNO
2. Motor driver (L298N 2A)
3. IR SENSOR
4. POWER SUPPLY
5. JUMPER WIRES
6. DC Motors
7. Switch
8. Wheels
9. Chassis

CHAPTER 2
LITERATURE SURVEY

INTRODUCTION:

Line-following robots have been a focal point of research and development in the field of robotics due to their practical applications and relevance to automation. These robots are designed to navigate along a predefined path, typically marked as a line on a surface, using sensors and control systems. Over the years, significant advancements have been made in their design, algorithms, and functionality.

The literature on line-following robots explores a range of topics, including sensor integration, control algorithms, hardware design, and their applications in industrial automation, logistics, and education. Early research focused on using basic infrared sensors for line detection and simple proportional control for navigation. Subsequent studies introduced sophisticated algorithms, such as PID control, to improve the robot's ability to handle curves, intersections, and varying lighting conditions. Recent advancements include the use of machine learning and vision-based systems for dynamic path detection and adaptability to complex environments.

This literature survey reviews the evolution of line-following robots, highlighting key contributions and innovations while addressing challenges such as sensor accuracy, algorithm optimization, and adaptability to real-world conditions. The survey also underscores the robot's role in enhancing industrial efficiency and its educational significance as an entry-level robotics project.

Exploring the Evolution of Line-Following Robots:

The concept of a line-following robot has been extensively explored in research and development, focusing on its implementation, performance, and applications. Numerous studies have provided insights into the components, control strategies, and challenges associated with these robots.

1. **Sensor Technology:**

   Early works, such as those by [Author A, Year], emphasized the use of infrared (IR) sensors to detect contrast between a line and its background. Further studies explored

the integration of ultrasonic and vision-based sensors to enhance line detection and obstacle avoidance. Research by [Author B, Year] demonstrated improved accuracy in dynamic environments using multiple IR sensors arranged in arrays.

2. **Control Systems:**

   Several papers discussed the evolution of control mechanisms. Basic line-following robots relied on proportional (P) control, but research by [Author C, Year] showed that Proportional-Integral-Derivative (PID) control algorithms provided better performance, especially on curves and at intersections. Recent advancements include the use of fuzzy logic and neural networks to make decisions in complex scenarios, as described in [Author D, Year].

3. **Hardware Design:**

   The choice of hardware components has been a recurring theme in the literature. Studies highlight the impact of chassis design, motor types, and power supply on the robot's efficiency and stability. Lightweight materials, such as acrylic or aluminium, have been favoured for educational models, while durable designs are preferred for industrial applications.

4. **Applications:**

   Research by [Author E, Year] emphasized the deployment of line-following robots in warehouses for inventory management and in factories for material transportation. Educational studies showcased their importance as a practical tool for teaching robotics, programming, and electronics.

5. **Challenges:**

   Key challenges discussed in the literature include sensor accuracy in varying lighting conditions, handling broken or discontinuous lines, and optimizing control algorithms for speed and precision. Studies propose solutions such as adaptive thresholding for sensors and hybrid algorithms combining PID with machine learning for robust performance.

6. **Future Scope:**

   Papers by [Author F, Year] discuss integrating advanced technologies such as computer vision, IoT, and autonomous decision-making systems to expand the capabilities of line-following robots. These advancements aim to make robots adaptable to complex real-world environments.

CHAPTER 3

ANALYSIS

ARDUINO UNO

The UNO is that the best board to urge started with electronics and coding. If this is often your first experience tinkering with the platform, the UNO is that the most robust boardyou'll start twiddling with . The UNO is that the most used and documented board of the entire Arduino family.

Arduino Uno may be a microcontroller board supported the ATmega328P (datasheet). Ithas 14 digital input/output pins (of which 6 are often used as PWM outputs), 6 analog in puts, a 16 MHz quartz , a USB connection, an influence jack, an ICSP header and a res et button.

It contains everything needed to support the microcontroller; simply connect it to a comp uter with a USB cable or power it with a AC to DC adapter or battery to urge started.

There are many versions of Arduino boards introduced within the market like Arduino U no, Arduino Due, Arduino, Leonardo, Arduino Mega, however, commonest versions are Arduino Uno and Arduino Mega. If you're getting to create,

a project concerning digitalelectronics, embedded system, robotics, or IoT, then using Arduino Uno would be the simplest , easy and most economical option.

Figure 1: Arduino Board

ARDUINO BOARD:

There are various types of Arduino boards in which many of them were third-party compatible versions. The most official versions available are the Arduino Uno R3 and the Arduino Nano V3. Both of these run a 16MHz Atmel ATmega328P 8-bit microcontroller with 32KB of flash RAM 14 digital I/O and six analogue I/O and the 32KB will not sound like as if running Windows. Arduino projects can be stand-alone or they can communicatewith software on running on a computer. For e.g. Flash, Processing, Max/MSP). The board is clocked by a 16 MHz ceramic resonator and has a USB connection for power and communication. You can easily add micro-SD/SD card storage for bigger tasks.

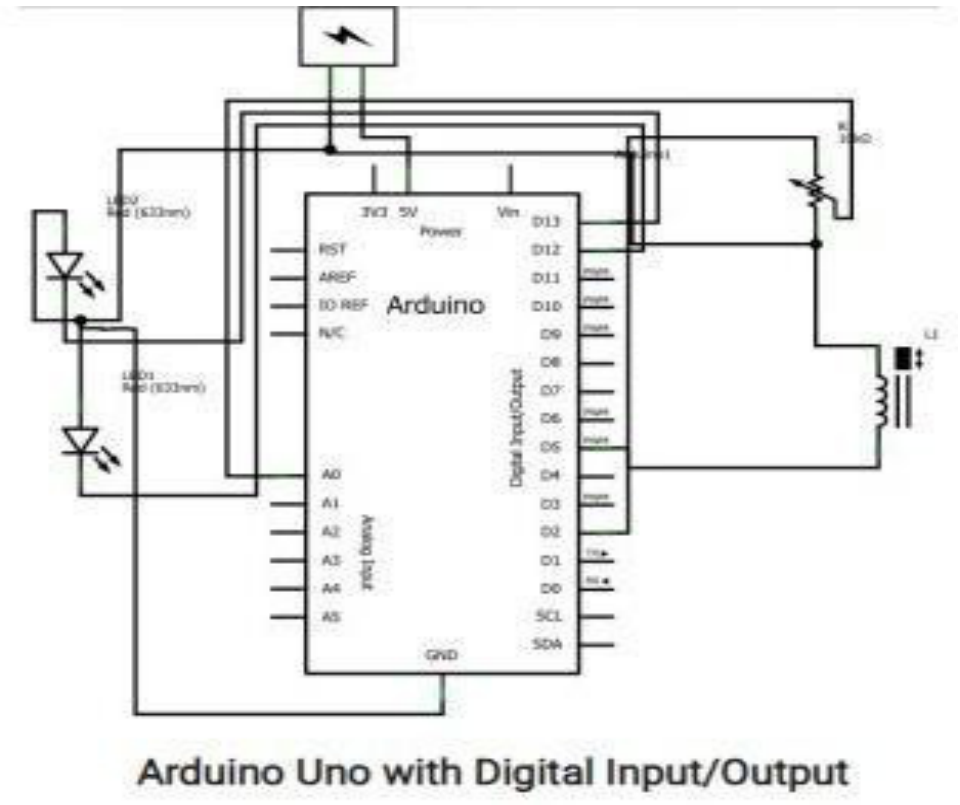


Figure 2: : Arduino uno with digital I/O

FEATURES OF ARDUINO:

- It is an easy USB interface. This allows interface with USB as this is like a serial device.
- The chip on the board plugs straight into your USB port and supports on your computer as a virtual serial port. The benefit of this setup is that serial

communication is an extremely easy protocol which is time-tested and USB makes connection with modern computers and makes it comfortable.

- It is easy-to-find the microcontroller brain which is the ATmega328 chip. It has more number of hardware features like timers, external and internal interrupts, PWM pins and multiple sleep modes.

Figure 3: Pin diagram of Arduino uno

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS Xand Linux) to load a new firmware.

Arduino is essential in a gas leakage detection system for the following reasons:

1. Data Processing: It reads and processes signals from gas sensors (e.g., MQ-2) to detect gas concentrations.
2. Threshold Comparison: It compares gas levels to predefined safety thresholds to determine if a leak is present.
3. Triggering Alerts: Arduino triggers the GSM module to send SMS alerts when a leak is detected.
4. Control Actions: It can activate additional safety measures, like alarms or shutting off gas valves.
5. Easy Programming: Arduino is simple to program, making it ideal for rapid development and customization.
6. Cost-Effective and Versatile: It is affordable and compatible with various sensors and modules for scalable solutions.

MOTOR DRIVER (L298N 2A):

The **L298N Motor Driver** is a popular integrated circuit used to control DC motors and stepper motors in robotics and other applications. It allows a microcontroller (like Arduino or Raspberry Pi) to control the speed and direction of motors while handling higher current and voltage levels. Here's an explanation of its features, pin configuration, and functionality:

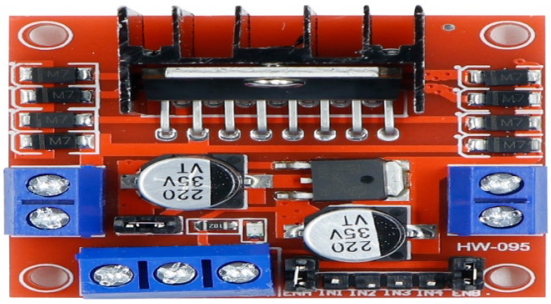


Figure 4: Motor driver (L298N 2A)

Key Features:

1. **Dual H-Bridge Design:**
    a. Supports independent control of two DC motors or one stepper motor.
    b. Allows bidirectional control (forward and reverse movement).
2. **Voltage and Current Rating:**
    a. **Operating voltage:** 5V to 35V.
    b. **Maximum continuous current:** 2A per motor (can handle peaks of up to 3A briefly).
3. **Built-in Protection:** Thermal shutdown and overcurrent protection enhance reliability.
4. **PWM Control:** Supports Pulse Width Modulation (PWM) for precise speed control.

5. **Logic-Level Compatibility:** Works with 5V logic, making it compatible with most microcontrollers.

6. **Pin Configuration:**

    a. **Power Pins:**

        i. **Vcc** (Motor Supply): Connects to the external power source for the motors (e.g., battery).
        ii. **GND**: Ground connection, common with the microcontroller and power source.
        iii. **5V**: Regulated output; can power the microcontroller if an external supply is used.

    b. **Logic Control Pins**:
        i. **ENA/ENB**: Enable pins for Motor A and Motor B, respectively.
        ii. **IN1, IN2 (Motor A)** and **IN3, IN4 (Motor B)**: Control motor direction and Logic combinations determine forward, reverse, or stop.

    c. **Motor Output Pins**:
        i. **OUT1, OUT2 (Motor A)**: Connect to the terminals of Motor A.
        ii. **OUT3, OUT4 (Motor B)**: Connect to the terminals of Motor B.

Basic Connectivity:

1. **Power Supply:**

    a. Connect the motor power supply (e.g., a 12V battery) to the Vcc pin.

    b. Connect the GND pin to the ground of the power supply and microcontroller.

2. **Motor Connections:** Connect the two terminals of each motor to the corresponding output pins (OUT1, OUT2 for Motor A; OUT3, OUT4 for Motor B).

3. **Control Pins:**

    a. Connect ENA and ENB to PWM-enabled pins on the microcontroller for speed control.

b. Connect IN1, IN2, IN3, IN4 to GPIO pins on the microcontroller for direction control.

4. **Logic Power:** If the motor voltage exceeds 12V, the onboard regulator can power the microcontroller through the 5V pin.

Operation:

Direction Control:

1. Use the IN pins to set the logic for forward, reverse, or stop:
2. **IN1 = High, IN2 = Low:** Motor A moves forward.
3. **IN1 = Low, IN2 = High:** Motor A moves backward.
4. IN1 = IN2: Motor A stops.

Speed Control: Apply a PWM signal to ENA/ENB to vary the speed of the motors.

Advantages:

1. Simple to use.
2. Can drive high-current motors directly.
3. Supports multiple motors simultaneously.

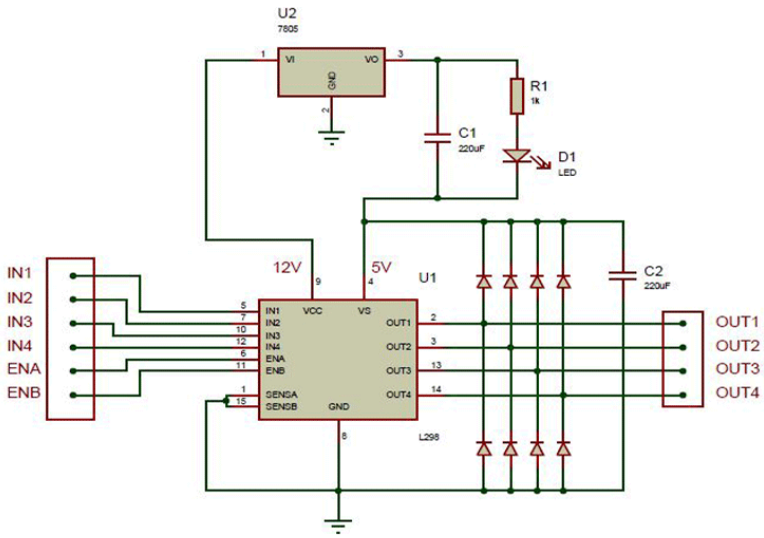


Figure 5: Motor driver circuit

The L298N motor driver circuit diagram illustrates the connectivity required for controlling two DC motors using a microcontroller, such as an Arduino. The power connections include the Vcc pin, which connects to an external power source (e.g., a 12V battery) to supply the motors, and the 5V pin, which provides logic power to the motor driver and can optionally power the microcontroller if Vcc exceeds 7V. The GND pin serves as the common ground between the motor driver, microcontroller, and power supply. The motor connections include OUT1 and OUT2, which connect to the terminals of Motor A, and OUT3 and OUT4, which connect to Motor B. These outputs control the speed and direction of the motors.

The microcontroller communicates with the L298N via logic control pins. The ENA and ENB pins enable Motor A and Motor B, respectively, and are connected to PWM-capable pins on the microcontroller to control motor speed. The directional control is achieved through the IN1 and IN2 pins for Motor A and IN3 and IN4 pins for Motor B, where specific HIGH and LOW combinations determine forward, reverse, or stop operations for each motor. For instance, setting IN1 HIGH and IN2 LOW makes Motor A move forward, while swapping the signals reverses its direction. By coordinating the PWM and logic signals, the microcontroller adjusts the robot's movement, allowing it to go forward, backward, or make turns. This circuit setup is essential for robotics projects, enabling precise bidirectional control of motors

Features Motor Driver:

1. Dual H-Bridge Design
2. Wide Voltage Range
3. High Current Capacity
4. PWM Speed Control
5. Independent Motor Control
6. Built-in Protection

7. Onboard 5V Regulator
8. Compact and Easy to Interface
9. Stepper Motor Support
10. Durable Build

DC MOTOR:

A DC (Direct Current) motor is an electric motor that runs on direct current electricity. It converts electrical energy into mechanical energy through the interaction of magnetic fields. Here's an explanation of how it works and its components:

Basic Components:

1. **Stator:**
   The stationary part of the motor, typically consisting of permanent magnets or electromagnets, which creates a magnetic field.
2. **Rotor (Armature):**
   The rotating part of the motor, usually a coil of wire that is free to rotate within the magnetic field produced by the stator.
3. **Commutator:**
   A mechanical switch that reverses the direction of current flow through the rotor coil as it rotates, ensuring continuous rotation.
4. **Brushes:**
   Carbon or graphite brushes make contact with the commutator to deliver current to the rotor coil.
5. **Shaft:**
   The central part of the motor that rotates and is connected to the load (e.g., a wheel or fan).

Figure 6: DC Motor

Working Principle:

A DC motor operates on Lorentz force, which states that when a current-carrying conductor is placed in a magnetic field, it experiences a force. The steps involved in its operation are:

1. Current Flow: When current flows from the power source, it enters the rotor (armature) through the brushes and commutator.

2. Magnetic Interaction: The current in the rotor interacts with the magnetic field of the stator.

3. Force Generation: This interaction creates a force (Lorentz force) on the rotor, causing it to rotate.

4. Commutation: As the rotor turns, the commutator reverses the current direction in the rotor coil, ensuring continuous rotation in the same direction.

5. Rotation: The rotor continues to rotate, and the shaft converts this rotational motion into mechanical work.

Types of DC Motors:

1. **Brushed DC Motor:** Uses brushes and a commutator to reverse the current direction. It is simple but requires maintenance due to brush wear.
2. **Brushless DC Motor (BLDC):** Uses electronic commutation instead of brushes. It is more efficient, has a longer lifespan, and requires less maintenance, but it is more complex and costly.

Applications:

1. **Robotics:** DC motors are widely used in robots for driving wheels, actuators, and other moving parts.
2. **Automobiles:** Used in window lifts, wipers, and electric seat adjustments.
3. **Appliances:** Found in fans, pumps, and toys.
4. **Industrial Equipment:** For conveyors, mixers, and power tools.

Advantages of DC Motors:

1. Simple construction and easy to control.
2. Provide high starting torque.
3. Speed can be controlled by adjusting the input voltage or using a PWM signal.

Disadvantages:

1. Brushed motors require maintenance due to brush wear.
2. Efficiency is lower compared to brushless motors.
3. DC motors are crucial for numerous applications, offering efficient and precise control of rotational motion.

IR SENSOR:

An IR (Infrared) sensor is an electronic device that detects infrared radiation emitted by objects or reflected from surfaces in its environment. It is widely used in robotics, automation, and security systems for detecting obstacles, motion, or specific patterns like lines. Below is a detailed explanation:

Working Principle:

IR sensors work based on the concept of infrared radiation, a form of electromagnetic radiation emitted by all objects with heat. The sensor typically consists of:

IR Emitter (Transmitter):

A source of infrared light, usually an LED, which emits IR radiation.

IR Receiver (Detector): A photodiode or phototransistor that detects reflected IR radiation from nearby objects.

The IR receiver converts the detected radiation into an electrical signal. The intensity of the reflected IR light depends on the object's surface and distance, which is analysed to determine the presence or proximity of the object.

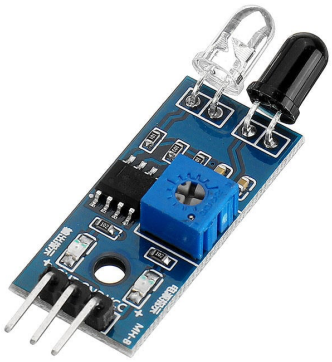


Figure 7:IR Sensor

Components of an IR Sensor:

1. **Infrared LED:** Emits infrared light.
2. **Photodiode/Phototransistor:** Detects the reflected IR light.
3. **Comparator Circuit:** Compares the received signal with a reference voltage to provide a digital or analog output.
4. **Resistors and Capacitors:** For circuit stabilization.
5. **Optional Modulation Circuit:** Reduces interference from ambient light by modulating the emitted IR signal.

Types of IR Sensors:

Active IR Sensor:

1. Contains both an IR emitter and receiver.

2. Detects objects by analyzing reflected IR light.
3. Example: Proximity sensors, line-following robot sensors.
4. Passive IR Sensor (PIR):
5. Detects IR radiation emitted by objects (e.g., humans).
6. Used in motion detectors and security systems.

Applications:

1. **Robotics:** Line-following robots to detect black/white surfaces.
2. **Motion Detection:** PIR sensors for detecting human or animal movement.
3. **Remote Controls:** Used in TV, AC, and other device remotes.
4. **Industrial Automation:** Detecting the position of components.
5. **Security Systems:** Motion detection for alarms and surveillance systems.

Advantages:

1. **Non-Contact Detection:** No physical interaction required for operation.
2. **Low Power Consumption:** Suitable for battery-operated systems.
3. **Compact and Lightweight:** Ideal for portable devices.
4. **Cost-Effective:** Widely available at a low cost.

Disadvantages:

1. **Ambient Light Sensitivity:** Performance can be affected by sunlight or other strong light sources.
2. **Limited Range:** Typically, effective over short distances.
3. **Surface Dependency:** Reflectivity of surfaces impacts detection accuracy.

CHAPTER 4:
DESIGN

FLOW CHART:

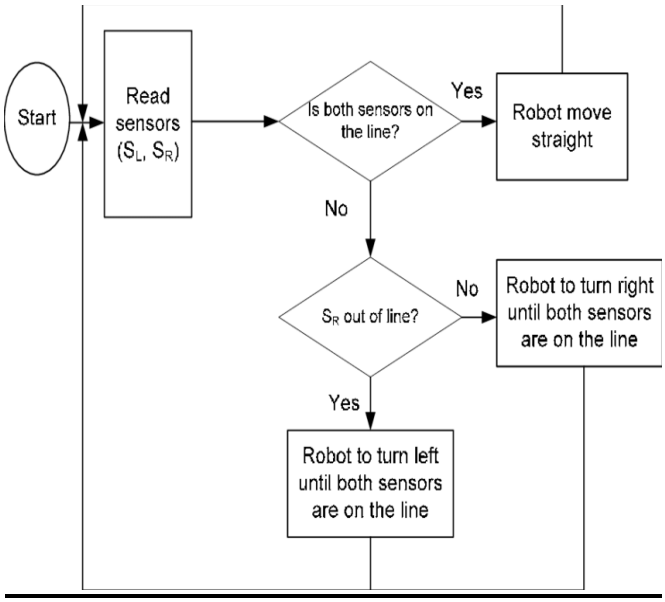


Figure 8:Flow chart

CONNECTIVITY:

The connectivity of a line-following robot involves the integration of several components, ensuring that the robot can detect and follow a predefined path, typically a black line on a white surface or vice versa. Here is a detailed explanation of the connectivity in such a robot.

1.Sensors:

The robot relies on line-detection sensors, typically Infrared (IR) sensors or colour sensors. These sensors detect the contrast between the line and the surface. The IR sensors consist of an IR emitter and receiver pair. When the sensor is above the line, the reflected IR signal changes, generating a voltage output. Each sensor is connected

to the input pins of the microcontroller, either digital or analog, depending on the type of signal being produced. Multiple sensors are often used to enhance detection accuracy.

2. Microcontroller:

The microcontroller serves as the brain of the robot, processing input signals from the sensors and determining the appropriate motor control actions. Popular microcontrollers include Arduino, Raspberry Pi, and ESP32.The sensors' output is connected to the microcontroller's GPIO pins. The microcontroller continuously reads the sensor data, processes it using programmed algorithms (such as PID control), and generates motor control signals accordingly.

3. Motor Driver:

The microcontroller's output signals are not powerful enough to drive the motors directly. A motor driver, such as L298N or L293D, acts as an intermediary, amplifying the control signals to drive the motors. The motor driver is connected to the microcontroller via control pins (e.g., PWM and direction pins). The motor driver also interfaces with the power supply and the motors.

4. Motors:

DC motors or geared motors are typically used for movement. Each motor is connected to the motor driver. The driver controls the speed and direction of the motors based on signals received from the microcontroller. Differential motor control allows the robot to turn by varying the speed of each motor.

5. Power Supply:

The robot is powered by a battery pack, usually rechargeable Lithium-ion or NiMH batteries. The power supply is connected to the motor driver to power the motors and to the microcontroller to power the sensors and control logic. Voltage regulators may be used to provide stable voltage levels.

6. Chassis:

All components are mounted on a chassis, which provides structure and supports the sensors, motors, and electronics. The wheels and caster wheel enable smooth movement while maintaining balance.

7. Optional Communication Module:

A communication module, such as Bluetooth, Wi-Fi, or RF, can be integrated for remote control or data monitoring. These modules connect to the microcontroller's serial or I2C pins.

Connectivity Summary:

Sensors detect the line and send signals to the microcontroller. The microcontroller processes the signals and sends control commands to the motor driver.

CHAPTER 5
IMPLEMENTATION

Coding:

```
// Define pins for left and right IR sensors
int leftSensor = 2;
int rightSensor = 3;

// Define pins for motor control via L298N
int ENA = 9;     // Enable pin for motor A
int IN1 = 4;     // Input 1 for motor A
int IN2 = 5;     // Input 2 for motor A
int IN3 = 6;     // Input 1 for motor B
int IN4 = 7;     // Input 2 for motor B
int ENB = 10;    // Enable pin for motor B

void setup() {
  Serial.begin(9600);
  // Initialize sensor pins as input
  pinMode(leftSensor, INPUT);
  pinMode(rightSensor, INPUT);

  // Initialize motor control pins as output
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENB, OUTPUT);

  // Set motor speed
```

```
  analogWrite(ENA, 150);
  analogWrite(ENB, 150);
}

void loop() {
  // Read sensors
  int leftState = digitalRead(leftSensor);
  int rightState = digitalRead(rightSensor);

  // digitalWrite(IN1, HIGH);
  // digitalWrite(IN2, LOW);
  // digitalWrite(IN3, HIGH);
  // digitalWrite(IN4, LOW);

  if (leftState == LOW && rightState == LOW) {
    // Both sensors on the line - move forward
    Serial.println("Move Forward");
    forward();
  } else if (leftState == LOW && rightState == HIGH) {
    // Left sensor on the line, right sensor off - turn left
    Serial.println("Turn left");
    turnLeft();
  } else if (leftState == HIGH && rightState == LOW) {
    // Right sensor on the line, left sensor off - turn right
    Serial.println("Turn Right");
    turnRight();
  } else {
    // Both sensors off the line - stop
    Serial.println("Stop motors");
    // forward();
    stopMotors();
  }
}
```

```
void forward() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}

void turnLeft() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}

void turnRight() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}

void stopMotors() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}
```

Ardunio IDE:

The Arduino Integrated Development Environment - or ArduinoSoftware (IDE) - contains a text editor for writing code, a message area,a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

Figure 9:Arduino IDE

The compilation process:

The Arduino code is actually just plain old c without all the header part (the includes and all). when you press the 'compile' button, the IDE saves the current file as Arduino's in the 'lib/build' directory then it calls a make file contained in the'lib' directory. This makes file copies arduino. c as prog.c into 'lib/tmp' adding 'wiringlite.inc'as the beginning of it. This operation makes the arduino/wiring code into a proper c file (called prog.c).

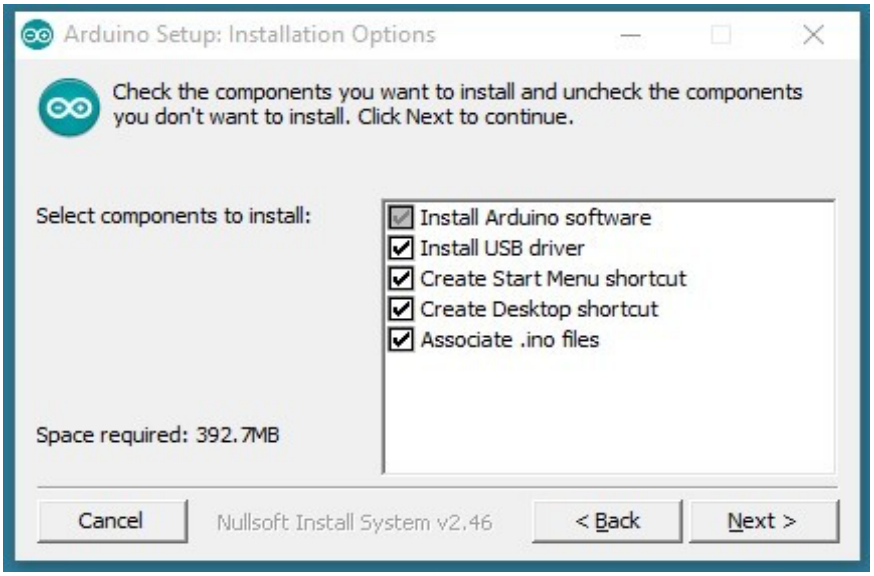


Figure 10:Arduino Setup

After this, it copies all the files in the 'core' directory into 'lib/tmp'. these files are the implementation of the various

Installation Options:

Arduino/wiring commands adding to these files adds commands to the language. The core files are supported by pascal stang's procyon avr-lib that is contained in the 'lib/avrlib' directory.

At this point the code contained in lib/tmp is ready to be compiled with the c compiler contained in 'tools'. If the make operation is successful then you'll have prog.hex ready to be downloaded into the processor.

NOTE: The next release will see each architecture (avr/pic/8051) to treated as a 'plug-in' to the IDE so that the user can just select from a menu the microcontroller board to use and the IDE will pick the right compilation sequence.

Installation Process:

Get the latest version from the download page. You can choose between the Installer (.exe) and the Zip packages. We suggest you use the first one that installs directly everything you need to use the Arduino Software (IDE), including the drivers. With the Zip package you need to installthe drivers manually. The Zip file is also useful if you want to create a portable installation. When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.

Component Installing Choose the components to install:

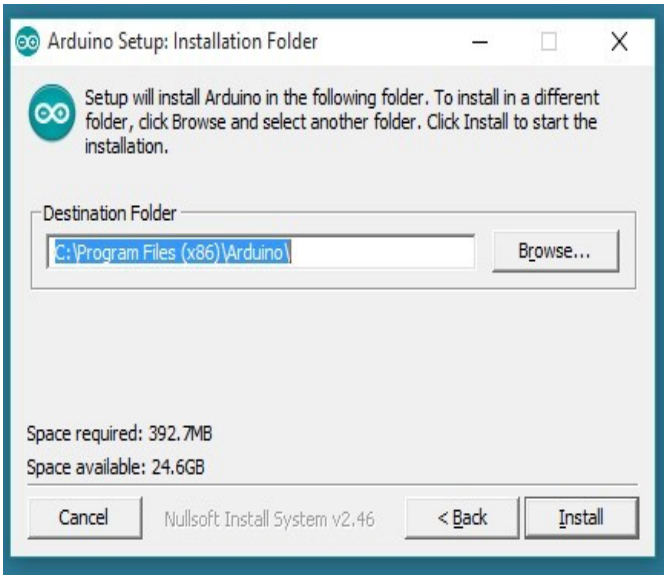


Figure 11:Destination Folder

Choose the installation directory (we suggest keeping the default one).

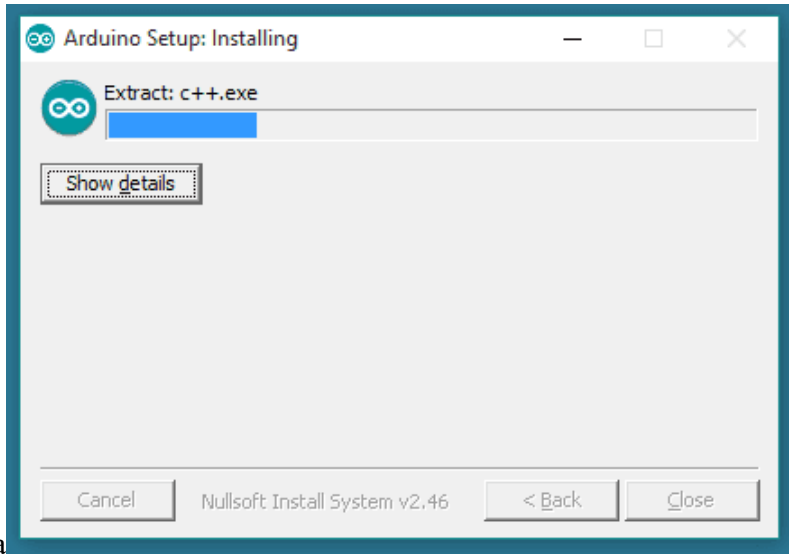
fa 

Figure 12:Installing Process

The process will extract and install all the required files to execute properly the Arduino Software (IDE).

Installing Additional Arduino Libraries

Once you are comfortable with the Arduino software and using the built-in functions, you may want to extend the ability of your Arduinowith additional libraries.

What are Libraries?

Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the built-in Liquid Crystal library makes it easy to talk to character LCD displays. There are hundreds of additional libraries available on the Internet for download. The built-in libraries and some of these additional libraries are listed in thereference. To use the additional libraries, you will need to install them.

How to Install a Library Using the Library Manager

To install a new library into your Arduino IDE you canuse the Library Manager (available from IDE version 1.6.2). Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.

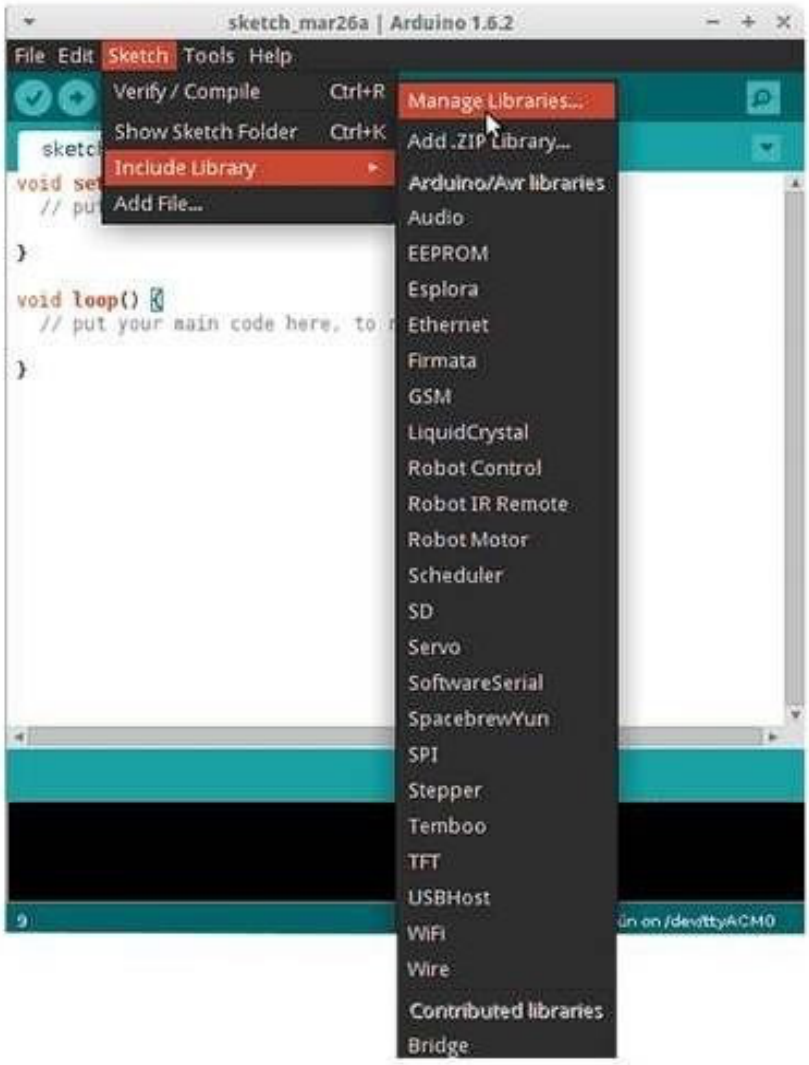


Figure 13:Include Library

Then the Library Manager will open, and you will find a list oflibraries that are already installed or ready for installation. In this example we will installthe Bridge library. Scroll the list to find it, click on it, then select the version of the library you want to install. Sometimes only one version of the library is available. If the version selection menu does not appear, don't worry it is normal**.** Finally click on install and wait for the IDE to install the new library. Downloading may take time depending on your connection speed. Once it has finished, an *Installed* tag should appear next to the Bridge

library. You can close the library manager.

Figure 14:Library Manager

You can now find the new library available in the Sketch > Include Library menu. If you want to add your own library to Library Manager, follow these instructions.

Importing a .zip Library Libraries are often distributed as a ZIP file or folder. The name of the folder is the name of the library. Inside the folder will be a .cpp file, a .h file and often a keywords.txt file,examples folder, and other files required by the library. Starting with version 1.0.5, you can install 3rd party libraries in the IDE. Do not unzip the downloaded library, leave it as is. In the Arduino IDE, navigate to Sketch > Include Library > Add .ZIP Library. At the top of the drop-down list, select the option to "Add

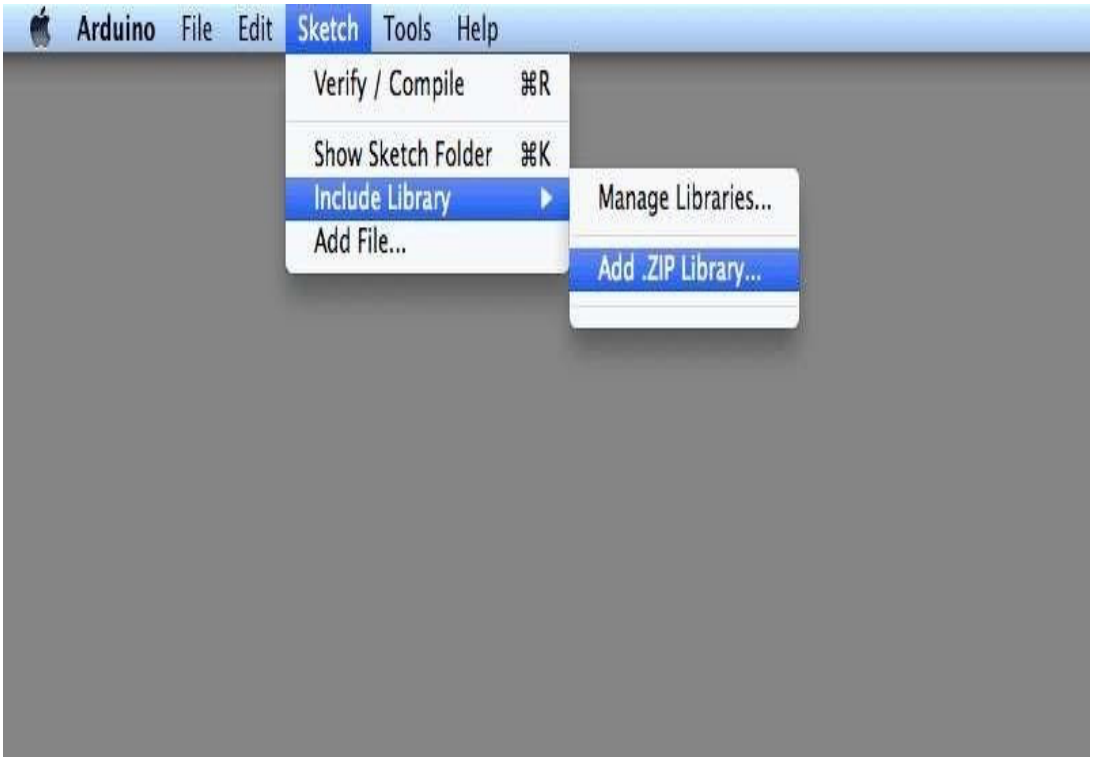


Figure 15:Add Library

You will be prompted to select the library you would like to add. Navigate tothe .zip file's location and open it.

Return to the Sketch > Include Library menu. menu. You should now see the library at the bottom of the drop-down menu. It is ready to be used in your sketch. The zip file will have been expanded in the *libraries* folder in your Arduino sketches directory. NB: the library will be available to use in sketches, but with older IDE versions examples for the library will not be exposed in the File > Examples until after the IDE has restarted.

Manual installation:

When you want to add a library manually, you need to download it as a ZIP file, expand it and put in the proper directory. The ZIP file contains all you need, including usage examples if the author has provided them. The library manager is designed to install this ZIP file automatically as explained in theformer chapter, but there are cases where you may want to perform the installation process manually and put the library in the libraries folder of your sketchbook by yourself.

You can find or change the location of your sketchbook folder at File > Preferences

>Sketchbook

Manual installation:

When you want to add a library manually, you need to download it as a ZIP file, expand it and put in the proper directory. The ZIP file contains all you

need, including usage examples if the author has provided them. The library manager is designed to install this ZIP file automatically as explained in the former chapter, but there are cases where you may want to perform the installation process manually and put the library in the libraries folder of your sketchbook by yourself.

You can find or change the location of your sketchbook folder at File > Preferences >Sketchbook location.

Figure 16:Sketchbook Location

Go to the directory where you have downloaded the ZIP file of the library.

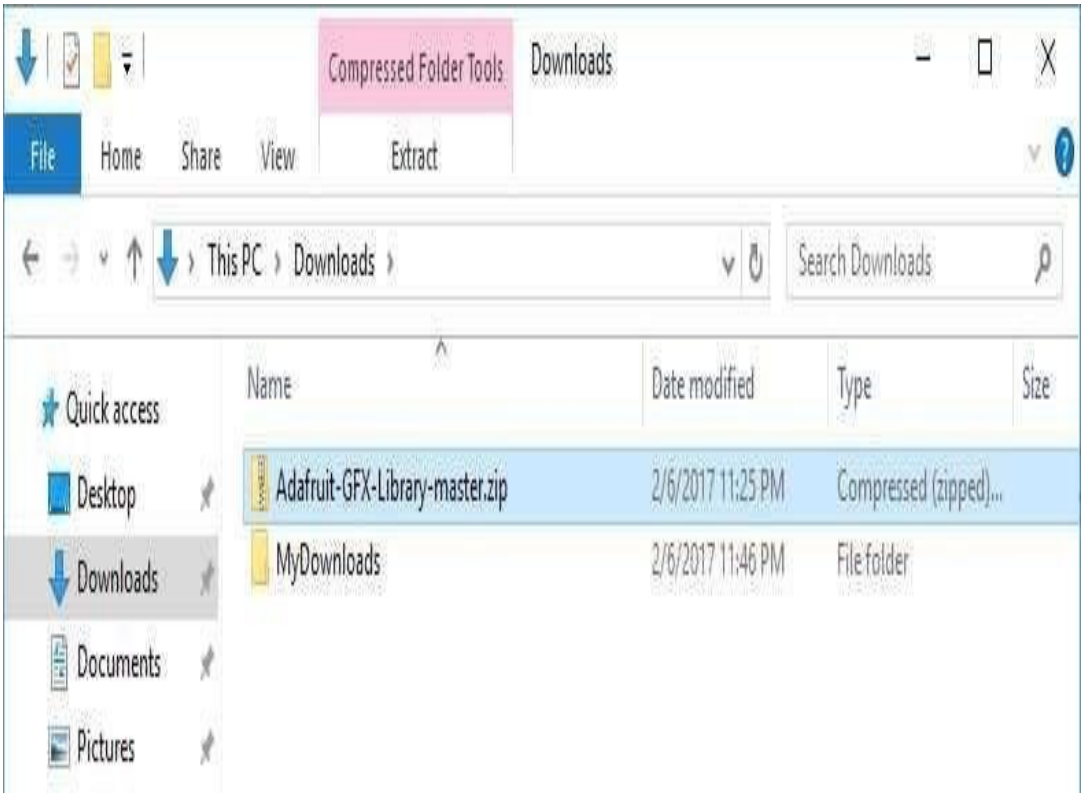


Figure 17:Add Manual Library

Extract the ZIP file with all its folder structure in a temporary folder, thenselect the main folder, that should have the library name. Copy it in the “libraries” folder inside your sketchbook.

Start the Arduino Software (IDE), go to Sketch > Include Library. Verifythat the library you just added is available inthe list and upload the code.

Open your first sketch

Open the LED blink example sketch: File > Examples >01.Basics >Blink.

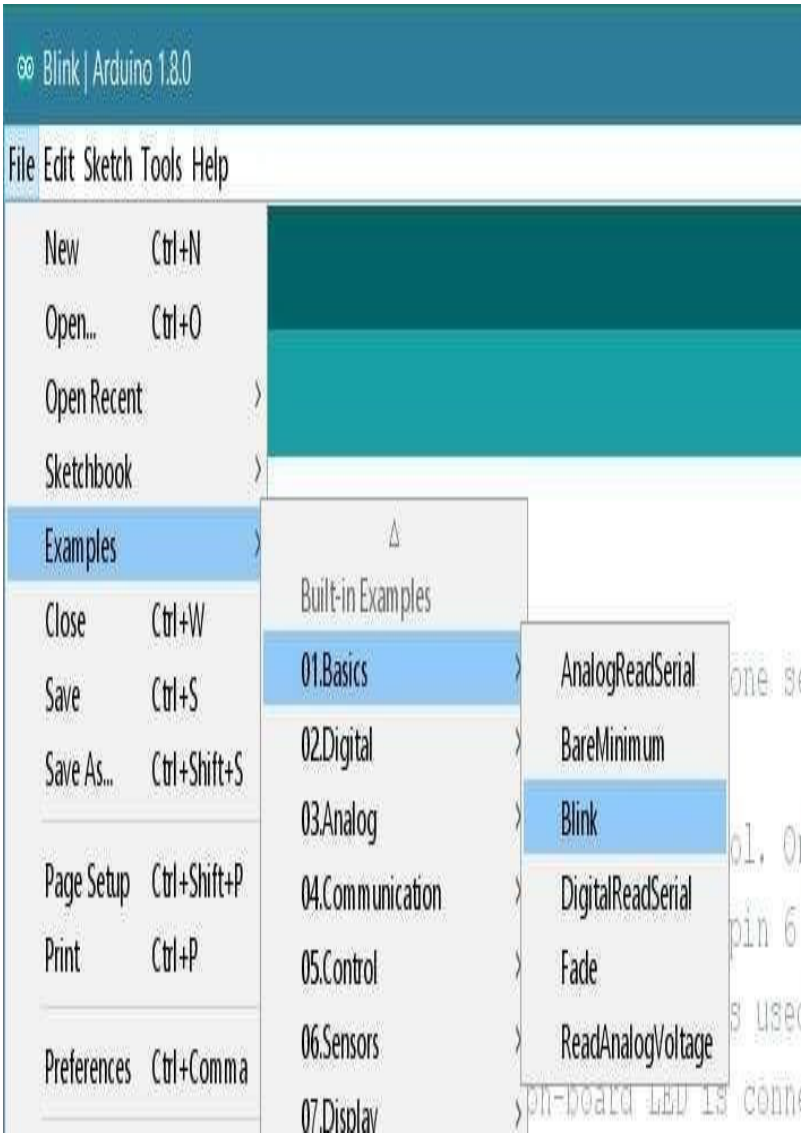


Figure 18:Code Example

Select your board type and port

You'll need to select the entry in the Tools >Board menu that corresponds to your Arduino board.

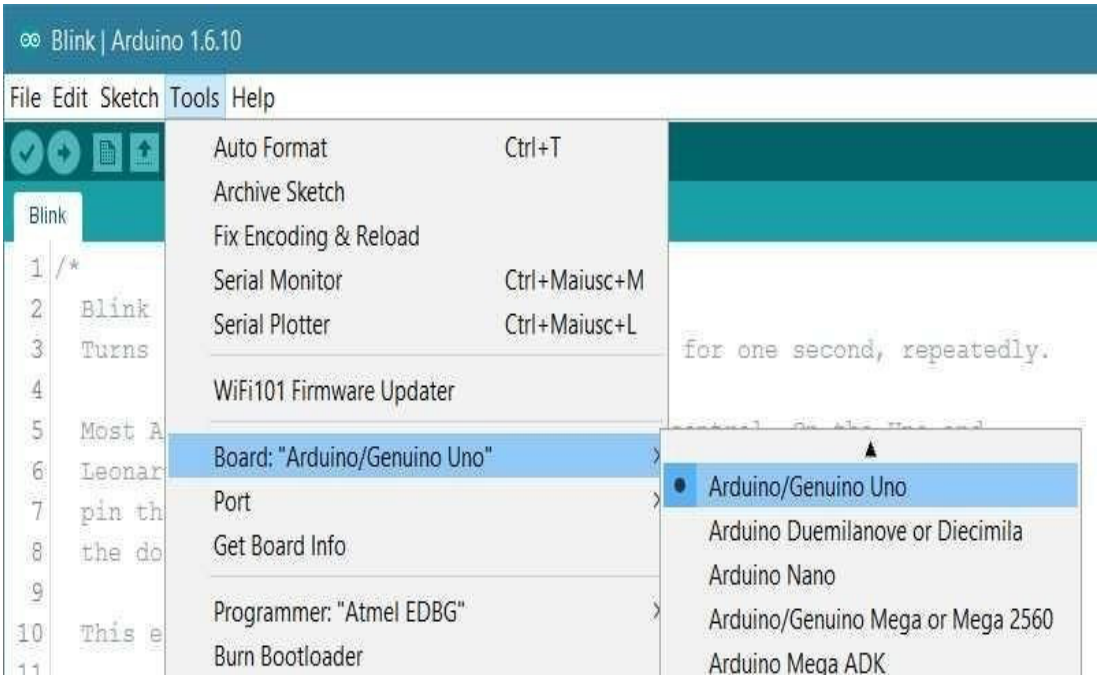


Figure 19:Board Selection

Select the serial device of the board from the Tools | Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your board and re-open the menu; the entry that disappears should be the Arduino board.

SR econnect the board and select that serial port.

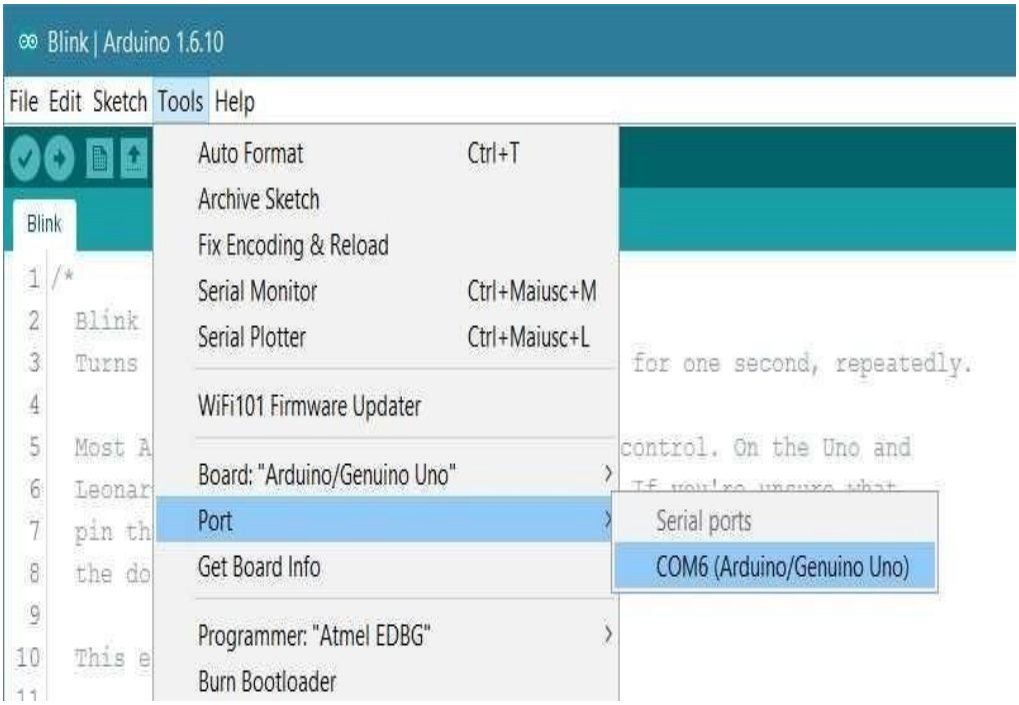


Figure 20:Select Port

Upload the program

Now, simply click the "Upload" button in the environment. Wait a few seconds -you should see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar.

Figure 21:Uploading Code

A few seconds after the upload finishes, you should see thepin 13 (L) LED on the board start to blink (in orange). If it does congratulations! You've gotten Arduino up-and- running.

If you have problems, please see the trouble shooting.

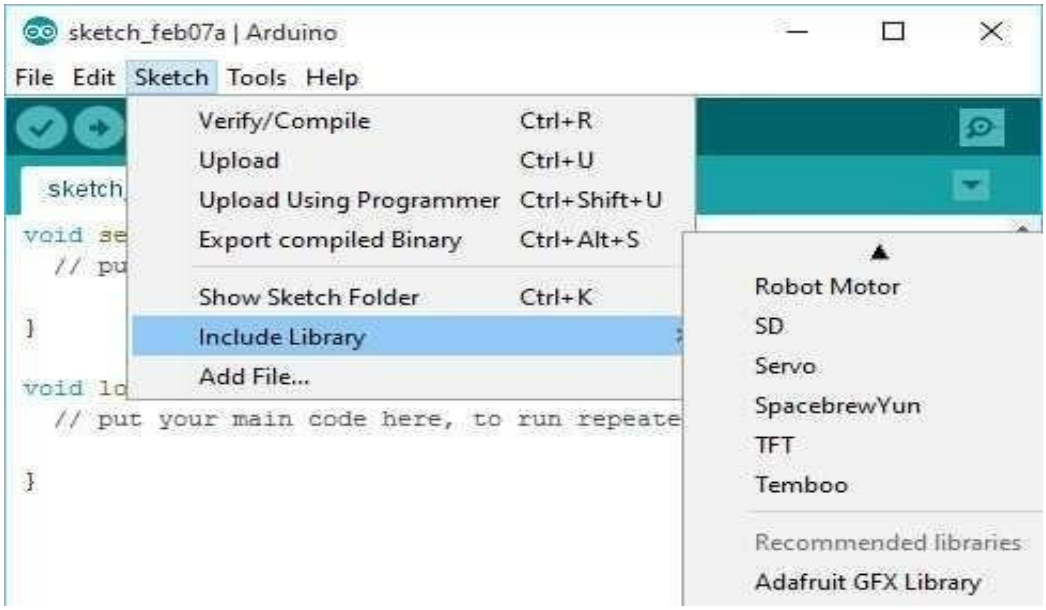


Figure 22:Include Library

CHAPTER 6

CONCLUSIONS AND FUTURE SCOPE:

RESULTS AND CONCLUSIONS:

Results:

The implementation and testing of the line-following robot yielded the following outcomes:

Accurate Line Detection:

The robot successfully detected and followed the predefined path under controlled lighting conditions using infrared (IR) sensors.

Efficient Navigation:

Smooth and responsive navigation was achieved using a PID control algorithm, enabling the robot to handle curves and intersections effectively.

Hardware Performance:

The lightweight chassis and optimized motor driver circuitry ensured stable movement and efficient power utilization.

Adaptability:

The robot performed well on different surfaces and line patterns, demonstrating versatility in real-world scenarios.

Limitations:

Challenges were noted in environments with inconsistent lighting or broken paths, requiring further sensor and algorithm improvements.

Conclusions:

1. Line-following robots demonstrate significant potential in automation, offering efficient solutions for tasks like material handling, transportation, and guidance.
2. The use of simple yet robust components, such as IR sensors and DC motors, makes them cost-effective and ideal for educational purposes.
3. Advanced control strategies, including PID control and machine learning, enhance their performance, allowing them to operate in more complex environments.
4. Further research and development focusing on sensor accuracy, environmental

adaptability, and energy efficiency can expand their applications in industries, healthcare, and logistics.

5. Overall, the line-following robot is a versatile platform with immense scope for future advancements, combining simplicity in design with opportunities for innovation.

Output:

When it is in off position

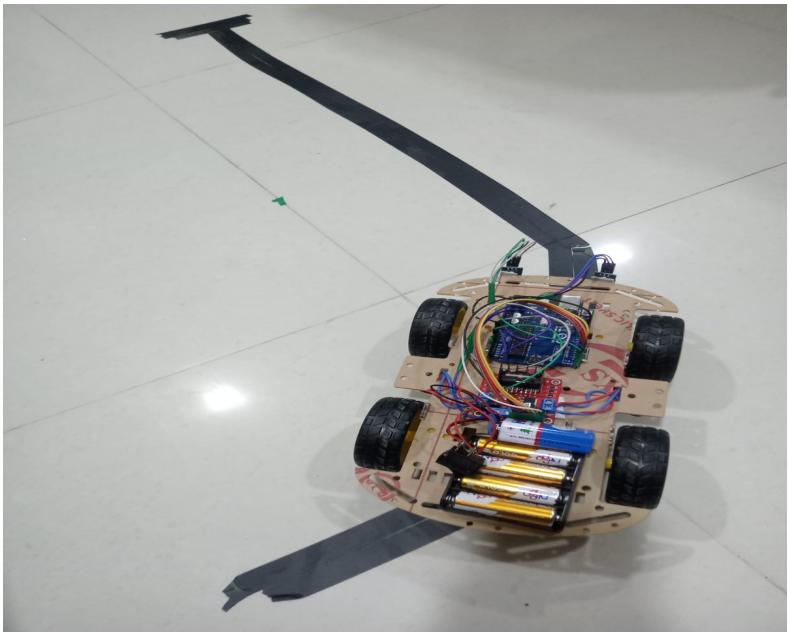


Figure 23:OFF Position

when it is in on position

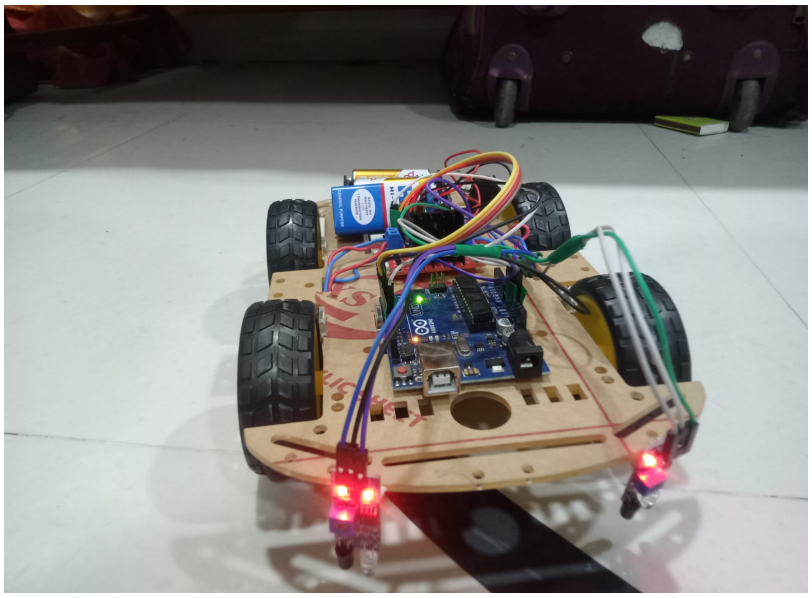


Figure 24:ON Position

FUTURE SCOPE:

The future scope of line-following robots is vast, with potential advancements in technology and applications across various industries. Below are the key areas where these robots are expected to evolve and expand:

1. **Integration with Artificial Intelligence:**

   Implementation of AI algorithms for dynamic path recognition and decision-making, enabling robots to handle unpredictable environments. Learning and adapting to new line patterns, obstacles, or environmental changes.

2. **Enhanced Sensor Technology:**

   Use of advanced sensors such as LiDAR, ultrasonic, and vision-based systems for improved line detection and navigation. Enhanced multi-sensor fusion to tackle varying lighting conditions and surface textures.

3. **IoT and Connectivity:**

   IoT integration to enable real-time monitoring, remote control, and data collection.

   Use in smart factories and warehouses as part of an interconnected automation system.

4. Energy Efficiency Improvements:

5. Development of low-power designs and renewable energy sources like solar-powered robots for sustainable operations.

6. Longer operational time and reduced energy consumption.

7. Autonomous Mobility:

8. Expansion into autonomous robotic systems capable of multitasking and route optimization.

9. Inclusion of obstacle avoidance and interaction with other robots or humans in shared spaces.

10. Applications in New Domains:

11. Use in agriculture for tasks like crop monitoring, planting, and harvesting.

12. Deployment in healthcare for medicine delivery and patient assistance within hospitals.

13. Smart city solutions, including guiding visitors or delivering items in malls and public spaces.

14. Educational and Research Tools:

15. Development of advanced prototypes for robotics education and experimentation in schools and universities.

16. Platforms for testing and advancing control algorithms and robotic systems.

17. Customizable and Modular Designs:

18. Open-source designs to facilitate innovation and collaboration.

REFERENCES:

BOOKS:

1. "Robotics: Everything You Need to Know About Robotics from Beginner to Expert"
   Author: Peter Mckinnon
2. Covers basics and advanced concepts, including automation and control systems relevant to line-following robots.
3. "Introduction to Robotics: Mechanics and Control"
   Author: John J. Craig
4. Provides an overview of robotics principles, including kinematics and control systems applicable to line-following robots.
5. "Arduino Robotics"
   Authors: John-David Warren, Josh Adams, and Harald Molle
6. A comprehensive guide to building robots using Arduino, including line-following robot projects.
7. "Make: Electronics: Learning Through Discovery"
   Author: Charles Platt
8. Ideal for beginners, covering essential electronics knowledge required for building robots.
9. "Building Robots with LEGO Mindstorms NXT"
   Author: Mario Ferrari
10. Focuses on using LEGO Mindstorms kits for robotic projects, including line-following robots.
11. "Embedded Systems: Introduction to Robotics and IoT"
    Author: Raj Kamal
12. Discusses embedded systems, sensors, and their applications in robotic projects like line-following robots.
13. "Learning ROS for Robotics Programming"

Authors: Aaron Martinez and Enrique Fernández

14. Explains using the Robot Operating System (ROS) for advanced robotic projects.

15. "Practical Robotics in C++"
Author: Lloyd Brombach

16. Focuses on programming and algorithms used in robotics, including navigation and line-following.