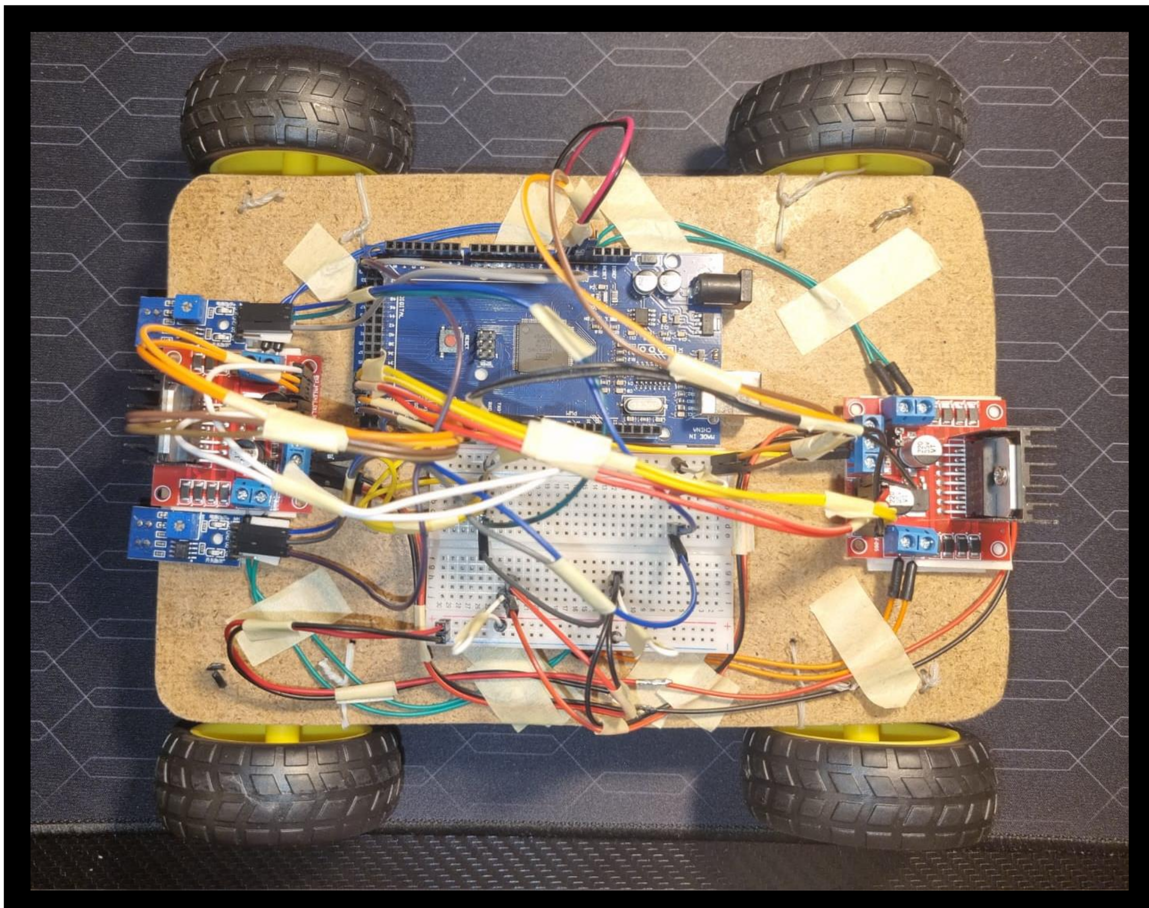
MY FIRST LINE FOLLOWER

ZAMFIREL V1.0



Contents

1. Introduction

A line follower robot is a mobile machine specifically designed to detect and follow a line drawn on the floor. This line serves as a predefined path that guides the robot's movement. The path can be visible, such as a black line on a white surface or a white line on a darker surface, where the high contrast makes it easier for sensors to detect the line. Alternatively, the path can be invisible, created using technologies like a magnetic field or embedded wires, which require specialized sensors to recognize and track the invisible cues.

This project appealed to me because I thought it would be a good starting point for someone with no background in robotics. The idea of a line follower robot is quite basic rather, I consider it a good platform to start understanding electronics, programming and mechanical concepts. It allowed me to explore how different components, such as motor drivers, sensors and micro controllers, work together to create a functional system. The intention behind taking up this project was to provide myself with a good foundation for working on more complex robotics challenges later.

2. Components

2.1 Arduino Mega 2560

The Arduino Mega is the microcontroller board responsible for processing the input from the sensors and controlling the motors through the motor driver modules. It is chosen for its multiple input/output pins, which are essential for handling multiple motors and sensors simultaneously.

2.2 L298N Motor Driver Modules

The L298N is a dual H-bridge motor driver capable of controlling two DC motors independently. Two modules are used in this project to control the four motors of the car. Each module:

- Allows forward and reverse motion for two motors.
- Can handle a maximum current of 2A per channel, ensuring sufficient power delivery.

2.3 Breadboard (400 Points)

The breadboard acts as a central hub for power distribution. It allows for multiple connections of power (VCC) and ground (GND) from the batteries, motor drivers, sensors, and Arduino, reducing the complexity of wiring.

2.4 Battery Holders and Batteries

6 AA Battery Holder with Batteries: Provides a total voltage of 9V (1.5V × 6), sufficient to power the motors via the motor drivers.

9V Battery Holder with Battery: Dedicated to powering the Arduino to isolate it from the motor power supply. This prevents voltage drops caused by the motors from affecting Arduino's performance.

2.5 TCRT5000 Infrared Sensor Modules

These modules detect the line by using infrared light.

- **Working Principle**: The TCRT5000 sensor emits infrared light and measures the intensity of the reflected light.
    - Dark surfaces (like a black line) absorb most of the infrared light, reflecting very little.
    - Light surfaces (like a white background) reflect more infrared light. The sensor outputs a digital signal (HIGH or LOW) based on the reflection detected, which the Arduino processes to determine the car's position relative to the line.

2.6 Dual Axis TT DC Gear Motors

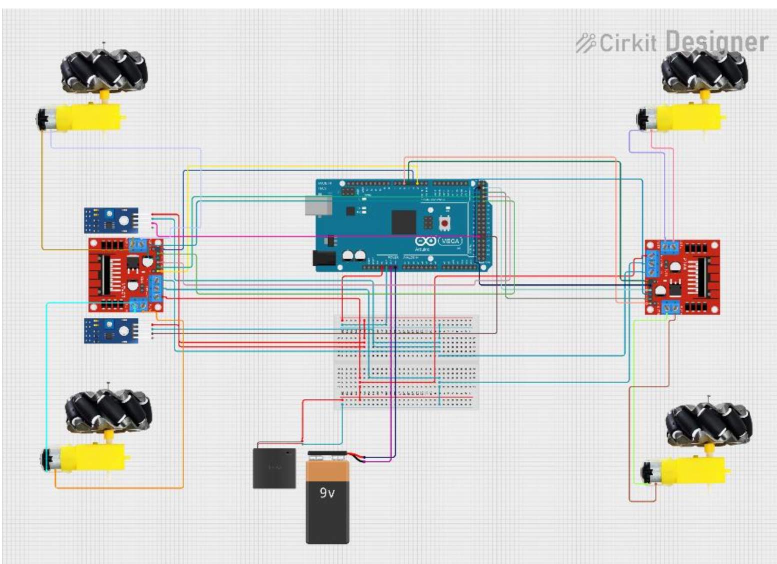
Four TT motors drive the wheels of the car. These motors are equipped with gears that provide the torque needed for smooth motion.

- Two motors are mounted in the rear, and two are in the front.
- Each pair is controlled by one motor driver.

2.7 Jumper Wires

These wires provide electrical connections between components such as the Arduino, motor drivers, sensors, and breadboard. Different lengths and types (male-to-male, male-to-female) are used depending on the connection points.

3. Construction and Connections



Four TT DC motors are mounted at each corner of the base, with wheels attached to enable motion. These motors are connected to the motor drivers via their output pins: OUT1**,** OUT2**,** OUT3**,** and OUT4. This setup allows the motor drivers to control both the speed and direction of each motor, providing the necessary mobility for the car.

3.1 L298N motor driver modules connections

Each motor driver module plays a crucial role in powering and controlling the motors.

- Power Supply:
  - The VCC and GND pins of the motor drivers are connected to the breadboard, which is powered by the 6 AA battery pack. This 9V supply ensures that the motors receive sufficient power to operate effectively.
  - The GND pins of the motor drivers are also connected to the GND pin on the Arduino, creating a shared ground for all components to ensure stable operation.
- Control Pins:
  - The pins IN1, IN2, IN3, and IN4 are linked to digital output pins on Arduino. These pins are responsible for controlling the direction of the motors by switching the current flow through the motor coils.
  - The ENA and ENB pins are connected to PWM (Pulse Width Modulation) pins on the Arduino.

What is PWM and Why Is It Used?

PWM, or Pulse Width Modulation, is a technique used to vary the average voltage supplied to a device by rapidly switching the signal on and off. By adjusting the duty cycle—the percentage of time the signal remains HIGH—PWM allows precise control over the motor speed. This capability is essential for fine-tuning the car's movement, especially when it needs to make turns or adjust its speed based on sensor inputs. With PWM, the car achieves smoother and more accurate motion control, critical for following the line effectively.

This combination of carefully routed power, efficient motor connections, and the integration of PWM provides the foundation for the car's ability to move dynamically and respond to its environment.

3.2 IR Sensor Connections

The IR sensors are essential for detecting the line and ensuring the robot follows its path accurately. Their connections are simple yet crucial for reliable performance.

- **Power Supply**:
  - The VCC and GND pins of each IR sensor are connected to the corresponding VCC and GND lines on the breadboard.
  - The breadboard is powered by the Arduino, providing a stable power supply to both sensors.
- **Signal Output**:
  - The digital output pins of the sensors are connected to specific digital input pins on the Arduino.

- These connections allow the sensors to send positional data—indicating whether the line is detected—directly to the Arduino.

The sensors are mounted at the front of the car, spaced slightly apart. This placement helps provide a broader detection area, allowing the robot to identify the line's position more accurately and adjust its movement as needed. By continuously feeding information to the Arduino, the sensors ensure precise and consistent path tracking.

4. Powering the car

Ensuring the correct power distribution is essential for the stable and efficient operation of my line follower car, ZAMFIREL V1.0. By using separate supplies for the Arduino and the motors, I reduce noise interference and provide each system with adequate current. Below is a more detailed explanation of the power strategy and voltage considerations:

4.1 Arduino Mega (9V Battery)

- **Dedicated Supply:**

  The Arduino is powered by a separate 9V battery. This choice isolates the microcontroller from potential fluctuations caused by the motors, which can draw significant current and introduce electrical noise.

- **Voltage Regulation:**

  When connecting a 9V battery to Arduino's VIN or barrel jack, the onboard regulator ensures the board receives a stable 5V and 3.3V where needed. This helps protect the sensitive logic components from voltage dips.

- **Reduced Interference:**

  By segregating the Arduino's power from the motor supply, I minimize the risk of random resets or erratic sensor readings due to momentary drops in voltage or electromagnetic interference (EMI).

4.2 Motors (6 AA Batteries)

- **Series Configuration:**
  - Each AA battery provides 1.5V and placing six of them in series yields a total of 9V (1.5V × 6 = 9V). This arrangement is sufficient to power the L298N motor drivers and the TT DC motors.
- **Why 9V for Motors?**
  - The L298N modules are rated for motor voltages around 7–35V, and 9V falls comfortably within that range.
  - TT motors typically operate well around 6–12V, making 9V a balanced choice for performance and longevity.

- **Battery Considerations:**
  - Fresh Batteries: Using fresh or rechargeable high-capacity AA cells ensures I have enough current for optimal performance.
  - Voltage Drop: As the batteries discharge, the voltage will gradually drop, potentially reducing the torque motors’ speed.

4.3 Total Current (Intensity)

- **L298N Current Capability:**
  - Each channel on L298N can supply up to 2A. This means each motor driver can handle up to 4A total (2A per channel × 2 channels).
- **Motor Current Draw:**
  - TT DC motors under normal load conditions generally draw less than 1A each, but the exact current depends on factors such as load, friction, and speed.
- **Battery Capacity:**
  - To avoid excessive voltage sag, the AA batteries must collectively supply enough current.

5. Arduino Code

The following sections explain the most important parts of the code.

Pin Definitions

The pins for the IR sensors and motor drivers are defined at the start of the code for clarity and ease of modification:

- **IR Sensors**: IR_R and IR_L are connected to pins 52 and 53, respectively. These sensors detect the line, with HIGH indicating a black surface and LOW indicating a white surface.
- **Motor Driver Pins**: Each motor (front-left, front-right, back-left, back-right) has a dedicated enable pin (e.g., enA, enB) for speed control and two direction control pins (IN1, IN2, etc.). This setup allows independent speed and direction control for all four motors.

Variables

- **motorSpeed**: Controls the speed of the motors using PWM, with a default value of 200 for moderate, stable movement.
- **sensorRight and sensorLeft**: Store the digital readings from the IR sensors to determine the robot’s position relative to the line.

Motor Control Functions

The code uses modular functions to handle movement, ensuring clear and reusable logic:

- **Forward()**: All motors are driven forward by setting the direction pins and applying the PWM signal using analogWrite().
- **Backward()**: Reverses all motors by swapping the direction pins while keeping the same speed control logic.

- **Left() and Right()**: These functions implement differential steering. For example, during a left turn, the left motors reverse while the right motors move forward. A small delay ensures precise turning adjustments.
- **Parking()**: Stops all motors by setting their PWM values to 0, effectively halting the robot.

6. Conclusions

I am new to electronics, and this project was perfect for me. It gave me a practical way to test and apply my limited knowledge in a real-world scenario. Perhaps soon, there will even be a Zamfirel V2.0!