L293D DC Motor Control using Raspberry Pi

Introduction:

The L293D motor driver is an integrated circuit (IC) used to control DC motors by providing bidirectional current. It acts as a bridge between the Raspberry Pi and the motor, allowing the low-power signals from the Raspberry Pi to drive motors that require higher power.

A DC motor is an electrical device that converts direct current (DC) into mechanical rotation. Raspberry Pi alone cannot provide enough current to drive a DC motor directly, so we use the L293D IC as an interface.

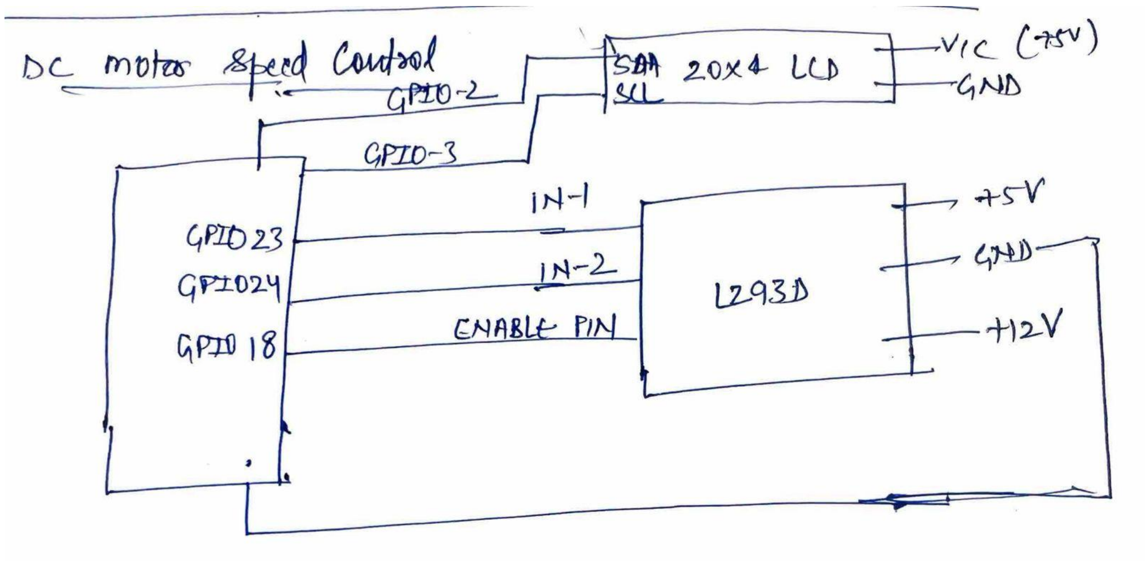
The L293D motor driver works on the H-Bridge principle, which allows the current to flow in either direction, enabling forward and reverse rotation of the motor. It has four input pins (to receive signals from Raspberry Pi) and four output pins (to control the motor). The Enable pins help in regulating motor speed.

Components Required:

- Raspberry Pi
- L293D Motor Driver IC
- DC Motor
- Jumper Wires
- Breadboard
- Power Supply

Circuit Connections:

- Pin 1 (Enable 1-2) → Connect to 3.3V/5V from Raspberry Pi
- Pin 2 (Input 1) → Connect to GPIO Pin X on Raspberry Pi
- Pin 3 (Output 1) → Connect to one terminal of DC Motor
- Pin 4, 5 (GND) → Connect to GND of Raspberry Pi and Power Supply
- Pin 6 (Output 2) → Connect to another terminal of DC Motor
- Pin 7 (Input 2) → Connect to GPIO Pin Y on Raspberry Pi
- Pin 8 (VCC2) → Connect to 9V Battery or External Power Supply
- Pin 16 (VCC1) → Connect to 5V of Raspberry Pi



Applications:

- Robotics – Used in autonomous robots, line-following robots, and obstacle-avoiding robots.
- Smart Home Automation – Controls motorized doors, windows, and appliances.
- Conveyor Belt Systems – Used in industries for automated material movement.
- Electric Vehicles (EVs) Prototyping – Helps in small-scale EV model testing.

Learnings:

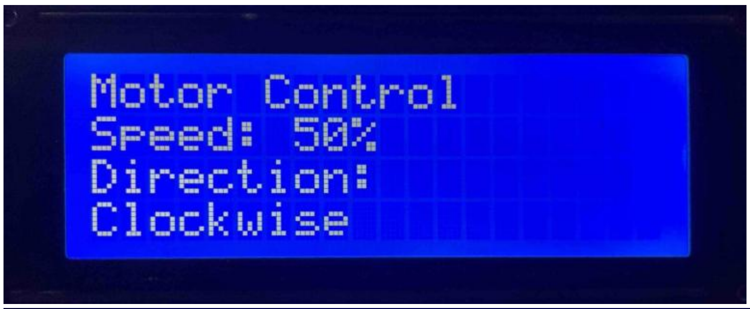
By completing this project, you will learn:

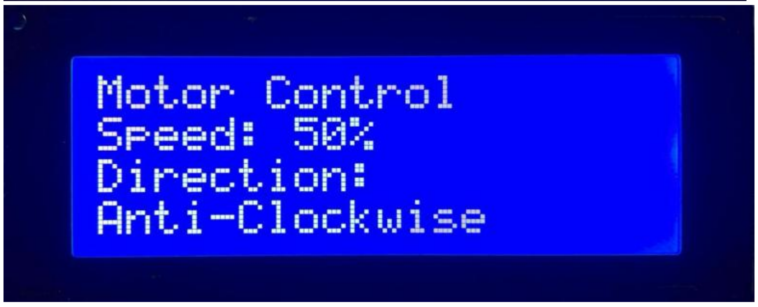
- How to use the L293D motor driver to control a DC motor.
- How to use GPIO pins of the Raspberry Pi for motor control.
- How an H-Bridge circuit works in motor drivers.
- The importance of separate power sources for the motor and Raspberry Pi.
- How to connect and interface hardware components with Raspberry Pi.

Conclusion:

The L293D motor driver is an essential component when working with DC motors and Raspberry Pi. This setup allows you to control a motor’s direction and speed, which is useful in robotics and automation projects. Understanding how to interface a motor driver with Raspberry Pi opens doors to more advanced projects like line-following robots, obstacle-avoiding robots, and smart home automation.

Outcome:


```
Motor Control
Speed: 50%
Direction:
Clockwise
```


```
Motor Control
Speed: 50%
Direction:
Anti-Clockwise
```

Program:

```
import RPi.GPIO as GPIO

import time

from RPLCD.i2c import CharLCD


GPIO.setwarnings(False)


# Pin Definitions

ENABLE_PIN = 18   # PWM pin

IN1 = 23        # Motor Direction Pin 1

IN2 = 24        # Motor Direction Pin 2

BUTTON_PIN = 17  # Button Pin


# Initialize GPIO

GPIO.setmode(GPIO.BCM)

GPIO.setup(ENABLE_PIN, GPIO.OUT)

GPIO.setup(IN1, GPIO.OUT)

GPIO.setup(IN2, GPIO.OUT)
GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)  # Pull-up resistor

# Initialize LCD
lcd = CharLCD('PCF8574', 0x27)

# PWM setup for speed control (range 0-100)
pwm = GPIO.PWM(ENABLE_PIN, 1000)  # 1 kHz frequency
pwm.start(50)  # Start at 50% speed

def motor_control(direction):
    if direction == 0:  # Clockwise
        GPIO.output(IN1, GPIO.HIGH)
        GPIO.output(IN2, GPIO.LOW)
        direction_text = "Clockwise"
    else:  # Anticlockwise
        GPIO.output(IN1, GPIO.LOW)
        GPIO.output(IN2, GPIO.HIGH)
        direction_text = "Anti-Clockwise"

    return direction_text

try:
    while True:
        # Read button state (0 = Clockwise, 1 = Anticlockwise)
        direction_state = GPIO.input(BUTTON_PIN)

        # Get motor direction
        direction_text = motor_control(direction_state)

        # Display Speed and Direction on LCD
        lcd.clear()
        lcd.cursor_pos = (0, 0)

        lcd.write_string("Motor Control\n")

        lcd.cursor_pos = (1, 0)

        lcd.write_string(f"Speed: 50%\n")  # Static speed for now

        lcd.cursor_pos = (2, 0)

        lcd.write_string(f"Direction:")

        lcd.cursor_pos = (3, 0)

        lcd.write_string(f"{direction_text}")


        time.sleep(0.5)  # Small delay


except KeyboardInterrupt:

    pwm.stop()

    GPIO.cleanup()

    lcd.clear()

    print("Program Stopped")
```