Namal University, Mianwali

Department of Electrical Engineering

EEN-325L – Microprocessor-Based Embedded Systems (Lab)

Lab-13

Raspberry Pi 4 GPIO and Sensor Interfacing

Student's Name	**Hassan Ali Ahmad**
Student's ID	**NUM-BSEE-2023-03**
Date Performed	**14 January 2025**
Marks Obtained	

Course Instructor:
Dr. Naureen Shaukat

Lab Instructor:
Engr. Majid Ali

Document History:

Version 1.0 Engr. Majid Ali 15/09/2025

1. LAB OBJECTIVE

The object of this lab is

- Understand the Raspberry Pi 4 GPIO pinout and pin functions
- Interface basic sensors and actuators with Raspberry Pi 4
- Develop simple embedded applications using Python on Linux

2. LEARNING OUTCOMES

This lab addresses the following Course Learning Outcomes (CLOs):

CLO-3: Constructs embedded applications microcontrollers and single-board computers through hardware-software integration.

CLO-4: Report performed tasks effectively in oral and written form

3. REQUIRED COMPONENTS & TOOLS

- System with Raspberry Pi 4 set up
- Raspberry Pi 4 Model B board
- Breadboard
- Jumper wires
- LEDs
- Current limiting resistors
- Push button switch
- Sensors
- Actuators

4. BACKGROUND

The Raspberry Pi 4 is a low-cost single board computer designed for learning programming and embedded systems. It runs a Linux-based operating system called Raspberry Pi OS and supports languages such as Python and C. The board includes GPIO pins that allow direct interaction with external hardware like LEDs and sensors. Due to its small size and low power consumption, Raspberry Pi is widely used in education, IoT, and prototyping applications.

4.1. **Raspberry Pi 4 - GPIO**

GPIO (General Purpose Input/Output) pins on the Raspberry Pi 4 allow the board to interact with external hardware such as LEDs, push buttons, sensors, and actuators. These pins can be configured as either input or output through software, enabling control and monitoring of connected devices. The Raspberry Pi 4 provides a 40-pin GPIO header, where each pin has a specific function. Some pins are dedicated to power supply (3.3V, 5V) and ground (GND), while others are programmable GPIO pins used for digital input and output. Certain GPIO pins also support special communication protocols such as I²C, SPI, and UART.

All GPIO pins operate at 3.3V logic level, and applying higher voltage may permanently damage the board. GPIO pins can be accessed and controlled using programming languages such as Python with libraries like RPi.GPIO or GPIO Zero.

In this lab, GPIO pins are used to:

- Control LEDs as digital outputs
- Read push button states as digital inputs
- Interface basic sensors and actuators

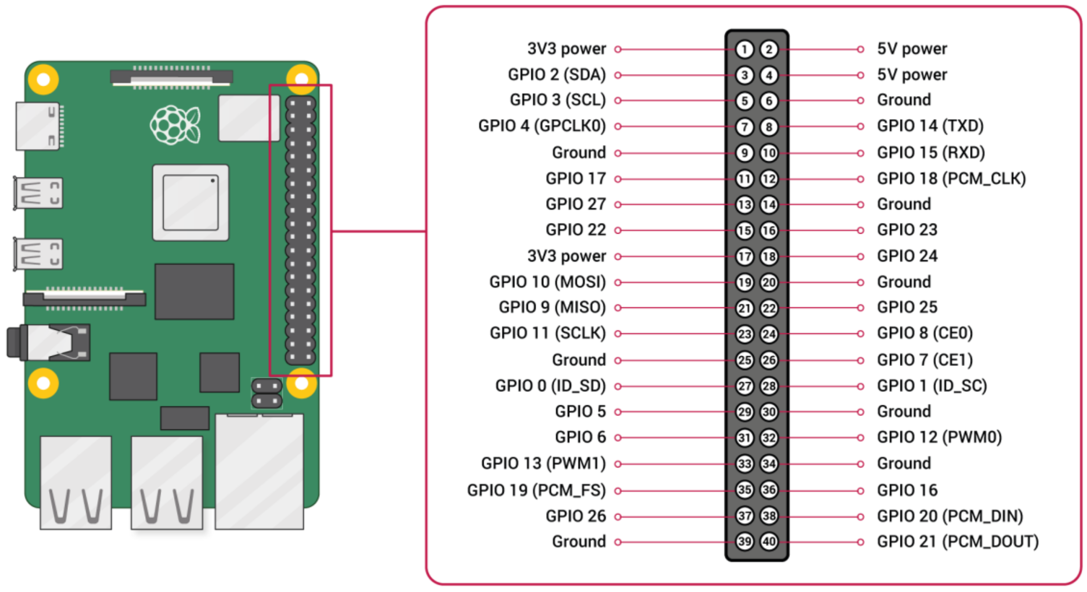


Fig.1: Raspberry Pi 4 GPIO pinout

4.2. Power and Ground Pins in Raspberry Pi 4

The Raspberry Pi 4 provides dedicated power and ground pins on its 40-pin GPIO header to supply power to external devices such as sensors and modules. The board includes three types of power pins:

- **5V Pins:** Provide 5-volt power directly from the power supply
- **3.3V (3V3) Pins:** Provide regulated 3.3-volt power for low-voltage devices
- **Ground (GND) Pins**: Provide a common reference voltage (0V)

These power pins can be used to power external components such as PIR sensors, humidity sensors, temperature sensors, and other low-power peripherals. Care must be taken not to exceed the current limits of the Raspberry Pi power rails.

4.3. GPIO Pins in Raspberry Pi 4

GPIO (General Purpose Input/Output) pins on the Raspberry Pi 4 are used to interface with external hardware devices. These pins can be configured through software as:

- **Input pins:** to read digital signals from devices like push buttons and sensors
- **Output pins:** to control devices such as LEDs, buzzers, and relays

In addition to general-purpose input/output functionality, certain GPIO pins support special functions, including communication protocols. The availability of these functions depends on the specific pin.

All GPIO pins operate at a 3.3V logic level, and applying higher voltages may permanently damage the Raspberry Pi.

4.4. GPIO Pin Numbering Conventions

The Raspberry Pi 4 supports two different GPIO pin numbering conventions:

BCM (Broadcom) Numbering

GPIO pins are labeled from GPIO2 to GPIO27 using the Broadcom (BCM) naming scheme. This convention is commonly used when programming GPIO pins with Python libraries such as *RPi.GPIO* and *GPIO Zero*.

Physical (Board) Numbering

Pins are numbered from 1 to 40 based on their physical location on the GPIO header. This convention is useful when BCM numbering is not supported and is used in some programming libraries.

Selecting the correct numbering scheme is essential to avoid incorrect pin connections and software errors.

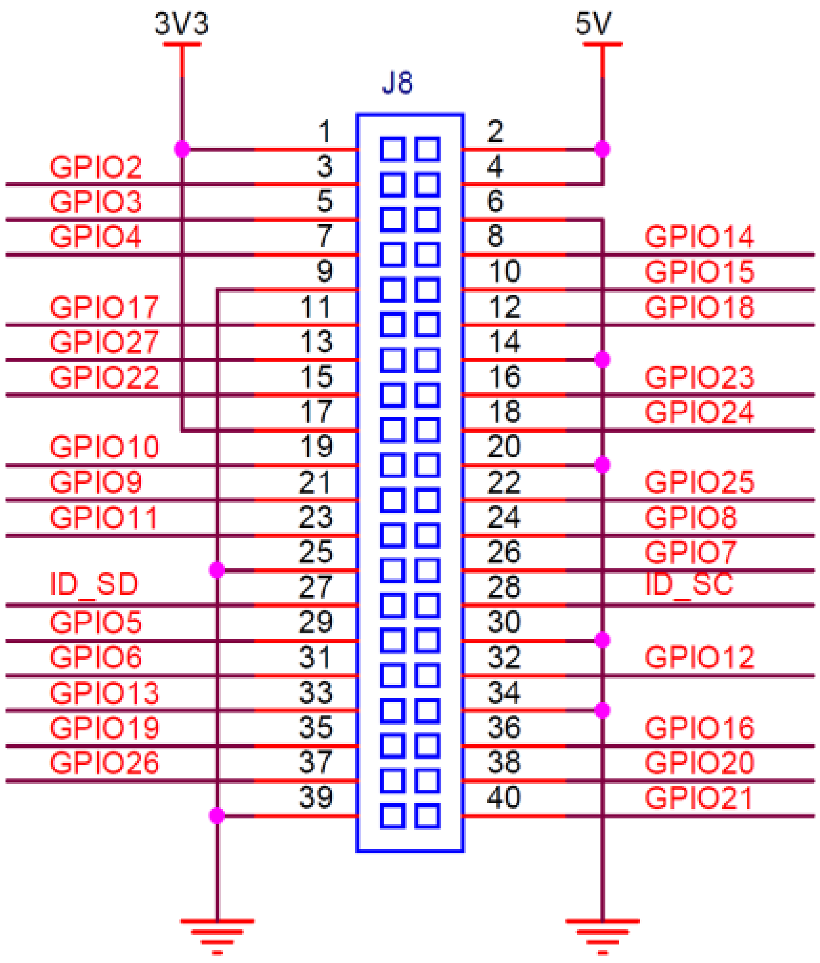


Fig.2: GPIO Connector Pinout

5. LAB TASKS

5.1. **Interface a DC motor with the Raspberry Pi 4 using an L298N motor driver module, connecting the motor driver input pins to GPIO outputs and the motor power and ground appropriately. Write a program to control the speed of the motor using PWM and to change the direction of rotation. Observe the motor speed and direction changes to verify correct actuator control.**

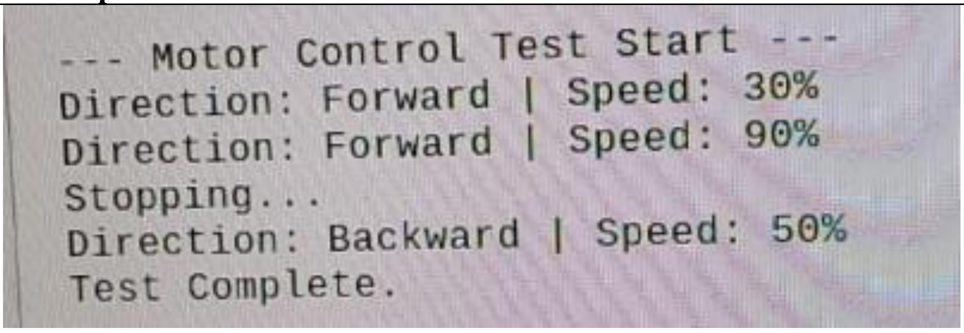
Code

```
import RPi.GPIO as GPIO
import time
# GPIO mode
GPIO.setmode(GPIO.BCM)
# Pin definitions
IN1 = 23
IN2 = 24
ENA = 18  # PWM pin
# Setup GPIO pins
GPIO.setup(IN1, GPIO.OUT)
GPIO.setup(IN2, GPIO.OUT)
GPIO.setup(ENA, GPIO.OUT)
# Setup PWM
pwm = GPIO.PWM(ENA, 1000)  # 1kHz frequency
pwm.start(0)
def motor_forward(speed):
   GPIO.output(IN1, GPIO.HIGH)
   GPIO.output(IN2, GPIO.LOW)
   pwm.ChangeDutyCycle(speed)
def motor_backward(speed):
   GPIO.output(IN1, GPIO.LOW)
   GPIO.output(IN2, GPIO.HIGH)
   pwm.ChangeDutyCycle(speed)
def motor_stop():
   GPIO.output(IN1, GPIO.LOW)
   GPIO.output(IN2, GPIO.LOW)
   pwm.ChangeDutyCycle(0)
try:
   while True:
      print("Motor Forward - 30% Speed")
      motor_forward(30)
      time.sleep(3)

      print("Motor Forward - 70% Speed")
      motor_forward(70)
      time.sleep(3)
      print("Motor Backward - 50% Speed")
      motor_backward(50)
      time.sleep(3)
      print("Motor Stopped")
      motor_stop()
      time.sleep(2)
```

```
        except KeyboardInterrupt:
            print("Program terminated")
        finally:
            pwm.stop()
    GPIO.cleanup()
```

Console Output



```
--- Motor Control Test Start ---
Direction: Forward | Speed: 30%
Direction: Forward | Speed: 90%
Stopping...
Direction: Backward | Speed: 50%
Test Complete.
```

Figure 1: Console Output

Hardware Implementation

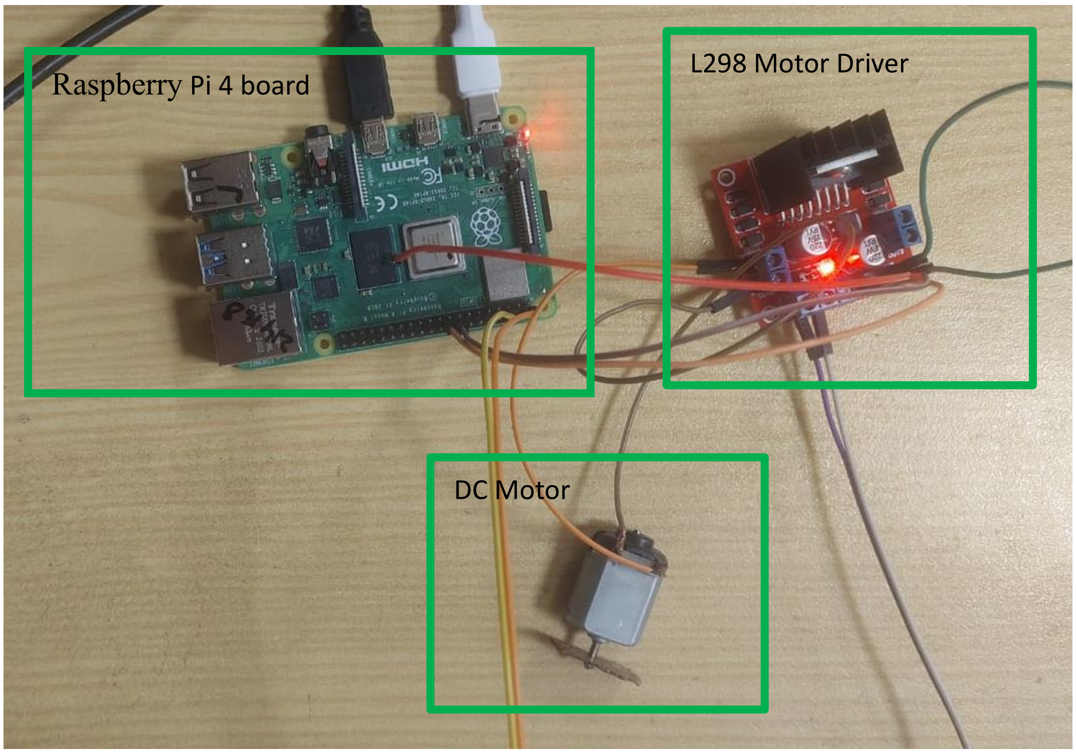


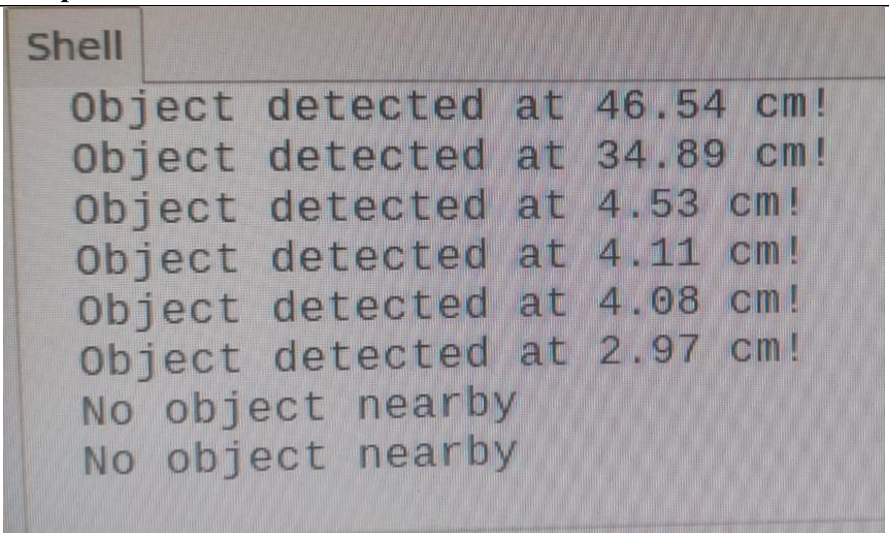
Figure 2: Hardware Implementation

5.2. **Interface an HC-SR04 ultrasonic sensor with the Raspberry Pi 4 by connecting VCC to 5V, GND to ground, Trig to a GPIO output pin, and Echo to a GPIO input pin. Write a program to send a trigger pulse, read the echo pulse, and calculate the distance to an object. Display the measured distance on the terminal.**

Code:

```
import RPi.GPIO as GPIO
import time
# GPIO mode
GPIO.setmode(GPIO.BCM)
# Pin definitions
TRIG = 23
ECHO = 24
# Setup pins
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)
# Initialize trigger
GPIO.output(TRIG, False)
time.sleep(2)
# Detection distance threshold in cm
DETECTION_DISTANCE = 50
try:
    while True:
        # Send 10µs trigger pulse
        GPIO.output(TRIG, True)
        time.sleep(0.00001)
        GPIO.output(TRIG, False)
        # Initialize times
        pulse_start = time.time()
        pulse_end = time.time()
        # Wait for echo to go HIGH (start)
        timeout = time.time() + 0.04
        while GPIO.input(ECHO) == 0 and time.time() < timeout:
            pulse_start = time.time()
        # Wait for echo to go LOW (end)
        timeout = time.time() + 0.04
        while GPIO.input(ECHO) == 1 and time.time() < timeout:
            pulse_end = time.time()
        # Calculate pulse duration
        pulse_duration = pulse_end - pulse_start
        # Distance calculation
        distance = pulse_duration * 17150
        distance = round(distance, 2)
        # Check detection
        if distance > 2 and distance <= DETECTION_DISTANCE:
            print(f"Object detected at {distance} cm!")
        else:
            print("No object nearby")
            time.sleep(1)
        except KeyboardInterrupt:
            print("Measurement stopped")
        finally:
    GPIO.cleanup()
```

Console Output



```
Shell
Object detected at 46.54 cm!
Object detected at 34.89 cm!
Object detected at 4.53 cm!
Object detected at 4.11 cm!
Object detected at 4.08 cm!
Object detected at 2.97 cm!
No object nearby
No object nearby
```

Figure 3: Console Output

Hardware Implementation

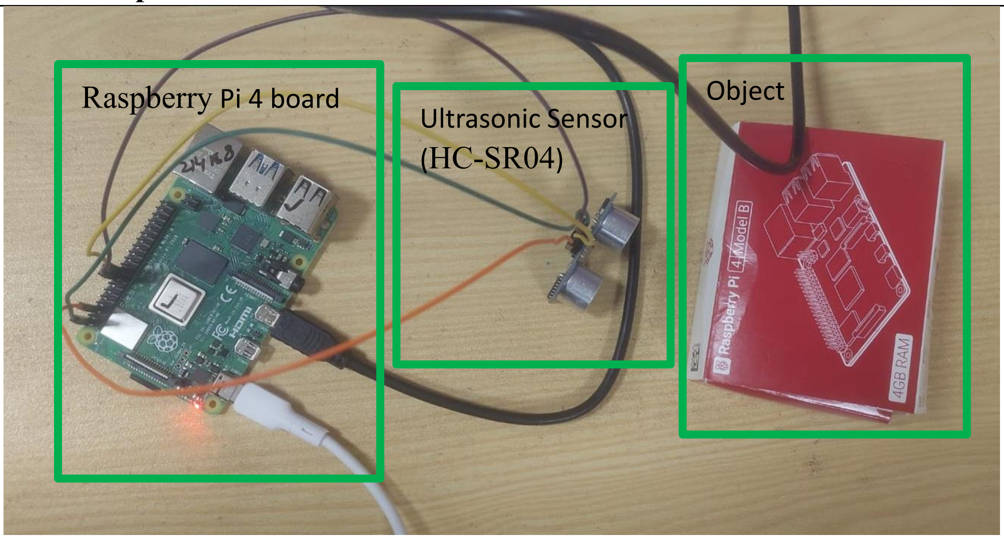


Figure 4: Hardware Implementation

6. CONCLUSION

In this lab, I have explored the GPIO functionality of Raspberry Pi 4. We interfaced a DC motor using the L298N driver and controlled its speed and direction through PWM. We also connected an HC-SR04 ultrasonic sensor and measured distance accurately using Python. The experiments helped us understand GPIO pin configuration, BCM numbering. Hardware and software integration was achieved using Python libraries and Linux environment.

Marking Rubrics							
Domain	**Criteria**	**Excellent (10-9)**	**Good (8 -7)**	**Satisfactory (6-5)**	**Unsatisfactory (4-3)**	**Poor (2-0)**	**Marks**
Psychomotor (P4)	**Programming & Simulation (CLO-3)**	Program runs correctly with accurate logic, comments, and circuit simulation; no errors	Program mostly correct with minor logical/syntax issues	Program compiles and runs with limited accuracy; basic circuit simulated	Program has major errors or incomplete simulation	No program or irrelevant submission	
	Hardware Implementation & Testing (CLO-3)	Circuit built correctly, neat wiring, hardware works exactly as intended	Circuit built with minor issues but works mostly correct	Circuit partially working with noticeable errors or unstable behavior	Circuit built incorrectly, does not function as required	No circuit built or completely wrong setup	
Affective (A3)	**Lab Viva (CLO-4)**	Answers confidently with clear and correct explanation of concepts	Answers mostly correct with only minor hesitation in response	Answers basic questions but shows gaps in deeper concepts	Answers weakly with major mistakes and unclear responses	Gives no answer or provides irrelevant responses	
	Report Writing (CLO-4)	Report well-structured with complete details and accurate results	Report mostly corrects with minor issues in format or details	Report basic with missing parts and limited explanation	Report weak with major errors or AI/plagiarized content	Report missing, incomplete, or not submitted at all	