SENSOR & DEVICES LAB

1. Connect an LED to GPIO pin 25 and control it Through command line

Hardware:

1. Connect the anode (longer leg) of the LED to a current-limiting resistor (220-470 ohms is typical) and then connect the other end of the resistor to GPIO pin 25 (physical pin 22).
2. Connect the cathode (shorter leg) of the LED to the ground (GND) pin (physical pin 6) on the Raspberry Pi.

Software:

1. First, ensure you have the necessary libraries installed. Open a terminal on your Raspberry Pi and run the following commands:

```
sudo apt-get update
sudo apt-get install python3-gpiozero
```

Controlling the LED via Command Line:

- To turn on the LED from the command line, run:

```
gpiozero-output 25 1
```

- To turn off the LED from the command line, run:

```
gpiozero-output 25 0
```

Controlling the LED via Python Program:

1. Create a Python script (e.g., **led_control.py**) to control the LED:

```
from gpiozero import LED
from time import sleep

led = LED(25)

while True:
    led.on()
    sleep(1)
    led.off()
    sleep(1)
```

- Run the Python script:

```
python3 led_control.py
```

This script will turn the LED on for one second, then off for one second, creating a blinking effect. You can stop the script by pressing **Ctrl + C**.

2. Connect an LED to GPIO pin 24 and a Switch to GPIO 25 and control the LED with the switch.

1. Connect the LED's longer leg (anode) to GPIO pin 24 and the shorter leg (cathode) to a current-limiting resistor (e.g., 220-470 ohms), and connect the other end of the resistor to a ground (GND) pin on the Raspberry Pi.
2. Connect one end of a push-button switch to GPIO pin 25 and the other end to a ground (GND) pin on the Raspberry Pi.
3. Save the following Python script as **led_switch_control.py** on your Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

# GPIO pin numbers for the LED and switch
LED_PIN = 24
SWITCH_PIN = 25

# Setup GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)
GPIO.setup(SWITCH_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Function to control the LED state
def set_led(state):
    GPIO.output(LED_PIN, state)

try:
    while True:
        # Read the state of the switch
        switch_state = GPIO.input(SWITCH_PIN)

        if switch_state == GPIO.LOW:
            set_led(True)  # Turn on the LED if the switch is pressed
        else:
            set_led(False) # Turn off the LED if the switch is not pressed

        time.sleep(0.1) # Small delay to debounce the switch

except KeyboardInterrupt:
    pass

# Clean up GPIO on program exit
GPIO.cleanup()
```

1. Open a terminal on your Raspberry Pi and navigate to the folder where you saved **led_switch_control.py**.

2. Run the script using the command:

```
python led_switch_control.py
```

3. When you press the push-button switch connected to GPIO pin 25, the LED connected to GPIO pin 24 will turn on. Releasing the switch will turn off the LED.

3 THE STATE OF LED SHOULD TOGGLE WITH EVERY PRESS OF THE SWITCH USE DHT11 TEMPERATURE SENSOR AND PRINT THE TEMPERATURE AND HUMIDITY OF THE ROOM WITH AN INTERVAL OF 15 SECONDS

1. Connect the DHT11 sensor to a free GPIO pin (e.g., GPIO 18) on your Raspberry Pi. Connect the sensor's VCC pin to 3.3V, the data pin to the chosen GPIO pin, and the sensor's GND pin to ground.
2. Save the following Python script as **led_switch_dht11.py** on your Raspberry Pi:

```
import RPi.GPIO as GPIO
import dht11
import time

# GPIO pin numbers for the LED, switch, and DHT11 sensor
LED_PIN = 24
SWITCH_PIN = 25
DHT11_PIN = 18

# Setup GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)
GPIO.setup(SWITCH_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Initialize DHT11 sensor instance
dht_sensor = dht11.DHT11(pin=DHT11_PIN)

# Function to control the LED state
def set_led(state):
    GPIO.output(LED_PIN, state)

# Function to read temperature and humidity
def read_dht11():
    result = dht_sensor.read()
    if result.is_valid():
        return result.temperature, result.humidity
    else:
        return None, None

led_state = False

try:
    while True:
        # Read the state of the switch
        switch_state = GPIO.input(SWITCH_PIN)

        if switch_state == GPIO.LOW:
            led_state = not led_state  # Toggle the LED state on switch press
            set_led(led_state)

        temperature, humidity = read_dht11()

        if temperature is not None and humidity is not None:
            print(f"Temperature: {temperature}°C, Humidity: {humidity}%")

        time.sleep(0.1)  # Small delay to debounce the switch

except KeyboardInterrupt:
    pass

# Clean up GPIO on program exit
GPIO.cleanup()
```

3. Open a terminal on your Raspberry Pi and navigate to the folder where you saved **led_switch_dht11.py**.
4. Run the script using the command:

python led_switch_dht11.py

Now, with every press of the push-button switch connected to GPIO pin 25, the LED connected to GPIO pin 24 will toggle its state (on/off). Additionally, the script will read the temperature and humidity from the DHT11 sensor connected to GPIO pin 18 and print the values every 15 seconds.

Please note that the DHT11 sensor may not be the most accurate temperature and humidity sensor available. If precision is important, you might consider using a more accurate sensor like the DHT22 or the BME280. Also, make sure you have the **dht11** library installed on your Raspberry Pi. If not, you can install it using the following command:

pip install dht11

4.CREATE A TRAFFIC LIGHT SIGNAL WITH THREE COLORED LIGHTS (RED, ORANGE AND GREEN) WITH A DUTY CYCLE OF 5-2-10 SECONDS.

Traffic light signal with three colored lights (Red, Orange, and Green) using a Raspberry Pi, you can utilize the GPIO pins to control the LEDs. You can achieve the desired duty cycle by controlling the timings for each LED state. Below is an example of how you can do this:

1. Connect three LEDs to GPIO pins on the Raspberry Pi. For this example, let's assume:
   - Red LED: GPIO 17
   - Orange LED: GPIO 18
   - Green LED: GPIO 27
2. Here's a Python script to control the traffic light with the specified duty cycles:

```
import RPi.GPIO as GPIO
import time

# GPIO pin numbers for the LEDs
RED_LED_PIN = 17
ORANGE_LED_PIN = 18
GREEN_LED_PIN = 27

# Setup GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(RED_LED_PIN, GPIO.OUT)
GPIO.setup(ORANGE_LED_PIN, GPIO.OUT)
GPIO.setup(GREEN_LED_PIN, GPIO.OUT)

def set_traffic_light_state(red_state, orange_state, green_state):
    GPIO.output(RED_LED_PIN, red_state)
    GPIO.output(ORANGE_LED_PIN, orange_state)
    GPIO.output(GREEN_LED_PIN, green_state)

try:
    while True:
        # Red light
        set_traffic_light_state(True, False, False)
        time.sleep(5)

        # Orange light
        set_traffic_light_state(False, True, False)
        time.sleep(2)

        # Green light
        set_traffic_light_state(False, False, True)
        time.sleep(10)

except KeyboardInterrupt:
    pass

# Clean up GPIO on program exit
GPIO.cleanup()
```

3. Open a terminal on your Raspberry Pi and navigate to the folder where you saved the script.
4. Run the script using the command:

```
python traffic_light.py
```

This script will simulate a traffic light with the specified duty cycles. The LEDs will turn on and off based on the timing sequence: 5 seconds for the red light, 2 seconds for the orange light, and 10 seconds for the green light. The cycle will then repeat.

5.USE LIGHT DEPENDENT RESISTOR (LDR) AND CONTROL AN LED THAT SHOULD SWITCH-ON/OFF DEPENDING ON THE LIGHT.

To control an LED using a Light Dependent Resistor (LDR) to switch it on/off depending on the light intensity, follow these steps:

Circuit Connection:

1. Connect one leg of the LDR to a GPIO pin (e.g., GPIO 18) on the Raspberry Pi.
2. Connect the other leg of the LDR to a voltage divider setup. Connect a resistor (e.g., 10k ohms) from the LDR's other leg to the ground (GND) pin of the Raspberry Pi.
3. Connect an LED to another GPIO pin (e.g., GPIO 17) on the Raspberry Pi. Connect its anode (longer leg) to the GPIO pin and the cathode (shorter leg) to a current-limiting resistor (e.g., 220 ohms), then connect the other end of the resistor to the ground (GND) pin on the Raspberry Pi.

Python Program:

1. Make sure you have the RPi.GPIO library installed. If not, you can install it using the command: **pip install RPi.GPIO**

2. Save the following Python script as **ldr_led_control.py** on your Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

# GPIO pin numbers for the LDR and LED
LDR_PIN = 18
LED_PIN = 17

# Setup GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(LDR_PIN, GPIO.IN)
GPIO.setup(LED_PIN, GPIO.OUT)

def read_light_intensity():
    return GPIO.input(LDR_PIN)

try:
    while True:
        light_intensity = read_light_intensity()

        if light_intensity == GPIO.LOW:
            GPIO.output(LED_PIN, GPIO.HIGH)  # Turn on the LED in low light
        else:
            GPIO.output(LED_PIN, GPIO.LOW)   # Turn off the LED in bright light

        time.sleep(0.5)  # Add a small delay for stability

except KeyboardInterrupt:
    pass

# Clean up GPIO on program exit
GPIO.cleanup()
```

3. Open a terminal on your Raspberry Pi and navigate to the folder where you saved the script.
4. Run the script using the command:

```
python ldr_led_control.py
```

The LED connected to GPIO 17 will turn on when the LDR's light intensity is low (dark) and turn off when the LDR detects high light intensity (bright).

6.CONVERT AN ANALOG VOLTAGE TO DIGITAL VALUE AND SHOW IT ON THE SCREEN

To convert an analog voltage to a digital value using a Raspberry Pi, you'll typically use an Analog to Digital Converter (ADC) chip. One commonly used ADC chip is the MCP3008, which is a 10-bit ADC that can read up to 8 analog inputs.

Here's how you can set up the MCP3008 ADC and display the converted digital value on the screen using a Python program:

Circuit Connection:

1. Connect the VDD and VREF pins of the MCP3008 to 3.3V.
2. Connect the VREF of the MCP3008 to a voltage source that represents the maximum voltage you want to measure (e.g., 3.3V for the full range).
3. Connect the AGND and DGND pins of the MCP3008 to the ground (GND).
4. Connect the CLK, DOUT, DIN, and CS pins of the MCP3008 to GPIO pins on the Raspberry Pi (e.g., CLK to GPIO 11, DOUT to GPIO 9, DIN to GPIO 10, and CS to GPIO 8).
5. Connect an analog voltage source (e.g., a potentiometer) to one of the analog inputs (e.g., CH0) of the MCP3008.

Python Program:

1. Install the **spidev** library by running the command: **pip install spidev**
2. Save the following Python script as **adc_read.py** on your Raspberry Pi:

```
import spidev
import time

# Create an instance of the SPI device
spi = spidev.SpiDev()
spi.open(0, 0)  # (bus, device)

def read_adc(channel):
    # Read analog data from the specified channel
    adc_data = spi.xfer2([1, (8 + channel) << 4, 0])
    digital_value = ((adc_data[1] & 3) << 8) + adc_data[2]
    return digital_value

try:
    while True:
        # Read analog data from channel 0
        adc_value = read_adc(0)
        # Print the digital value
        print(f"Analog value: {adc_value}")

        time.sleep(1)

except KeyboardInterrupt:
    pass

spi.close()
```

3. Open a terminal on your Raspberry Pi and navigate to the folder where you saved the script.
4. Run the script using the command:

```
python adc_read.py
```

This script will read the analog value from the specified channel of the MCP3008 ADC and print the corresponding digital value on the screen.

7.CREATE AN APPLICATION THAT HAS THREE LEDS (RED, GREEN AND WHITE). THE LEDS SHOULD FOLLOW THE CYCLE (ALL OFF, RED ON, GREEN ON, AND WHITE ON) FOR EACH CLAP (USE SOUND SENSOR).

Hardware Components:

1. Raspberry Pi (any model with GPIO pins)
2. Breadboard
3. Sound sensor module (e.g., KY-038)
4. Red, Green, and White LEDs
5. Resistors (220Ω recommended for each LED)
6. Jumper wires

Software Components:

1. Raspbian OS installed on your Raspberry Pi
2. Python programming language
3. GPIO library (usually pre-installed with Raspbian)

Steps:

1. **Wiring:**
   - Connect the power (VCC) and ground (GND) pins of the sound sensor to the 3.3V and GND pins on the Raspberry Pi.
   - Connect the signal (OUT) pin of the sound sensor to a GPIO pin (e.g., GPIO23).
   - Connect each LED anode (longer leg) to a separate GPIO pin with a resistor in series.
   - Connect all LED cathodes (shorter legs) to a common ground (GND) pin.

2. **Code:** Create a Python script to control the LEDs based on the clap detected by the sound sensor.

```
import RPi.GPIO as GPIO
import time

# GPIO pin numbers for LEDs and sound sensor
RED_LED_PIN = 17
GREEN_LED_PIN = 18
WHITE_LED_PIN = 27
SOUND_SENSOR_PIN = 23

# Setup GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(RED_LED_PIN, GPIO.OUT)
GPIO.setup(GREEN_LED_PIN, GPIO.OUT)
GPIO.setup(WHITE_LED_PIN, GPIO.OUT)
GPIO.setup(SOUND_SENSOR_PIN, GPIO.IN)

# Function to control LED states
def set_leds(red, green, white):
    GPIO.output(RED_LED_PIN, red)
    GPIO.output(GREEN_LED_PIN, green)
    GPIO.output(WHITE_LED_PIN, white)

# Initial state: All LEDs off
set_leds(False, False, False)

try:
    while True:
        # Wait for the sound sensor to detect a clap
        GPIO.wait_for_edge(SOUND_SENSOR_PIN, GPIO.RISING, timeout=5000)

        # Cycle through LED states
        set_leds(False, False, False)  # All Off
        time.sleep(1)

        set_leds(True, False, False)   # Red On
        time.sleep(1)

        set_leds(False, True, False)   # Green On
        time.sleep(1)

        set_leds(False, False, True)   # White On
        time.sleep(1)

except KeyboardInterrupt:
    pass

# Clean up GPIO on program exit
GPIO.cleanup()
```

Run the Script: Save the Python script on your Raspberry Pi (e.g., **led_clap_app.py**), and run it using the command:

```
python3 led_clap_app.py
```

This script waits for a clap sound detected by the sensor, and upon each clap, it cycles through turning on the red, green, and white LEDs with a brief delay between each state. The cycle starts again when another clap is detected. Remember to adjust GPIO pin numbers according to your wiring.

8.CONTROL A 230V DEVICE (BULB) WITH RASPBERRY PI USING A RELAY

Controlling a high-voltage device like a 230V bulb using a Raspberry Pi requires a relay module, as the Raspberry Pi's GPIO pins cannot directly handle such high voltages. Here's how to set up the connection and write a Python program to control the bulb using a relay module:

Circuit Connection:

1. Connect the VCC and GND pins of the relay module to the 5V and GND pins on the Raspberry Pi, respectively.
2. Connect the IN pin of the relay module to a GPIO pin on the Raspberry Pi (e.g., GPIO 17).
3. Connect the COM (Common) and NO (Normally Open) terminals of the relay to the two terminals of the bulb.
4. Connect the two terminals of the bulb to the AC power source (230V).

Python Program:

1. Save the following Python script as **relay_control.py** on your Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

# GPIO pin number for the relay control
RELAY_PIN = 17

# Setup GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(RELAY_PIN, GPIO.OUT)

# Function to turn the relay on or off
def set_relay(state):
    GPIO.output(RELAY_PIN, state)

try:
    while True:
        print("1. Turn on the bulb")
        print("2. Turn off the bulb")
        print("3. Quit")

        choice = input("Enter your choice (1/2/3): ")

        if choice == '1':
            set_relay(True)  # Turn on the bulb
        elif choice == '2':
            set_relay(False)  # Turn off the bulb
        elif choice == '3':
            break
        else:
            print("Invalid choice. Please enter 1, 2, or 3.")

except KeyboardInterrupt:
    pass

# Turn off the relay and clean up GPIO on program exit
set_relay(False)
GPIO.cleanup()
```

2. Open a terminal on your Raspberry Pi and navigate to the folder where you saved the script.
3. Run the script using the command:

   python relay_control.py

   The script will display a menu that allows you to turn the bulb on or off, and you can quit the script by selecting option 3.

9. SWITCH ON AND SWITCH OF A DC MOTOR BASED ON THE POSITION OF A SWITCH.

Controlling a DC motor based on the position of a switch using a Raspberry Pi involves using a relay module or an H-bridge motor driver. In this example, I'll demonstrate using a relay module to control the motor. Here's how to set up the connection and write a Python program:

Circuit Connection:

1. Connect the VCC and GND pins of the relay module to the 5V and GND pins on the Raspberry Pi, respectively.
2. Connect the IN pin of the relay module to a GPIO pin on the Raspberry Pi (e.g., GPIO 17).
3. Connect the COM (Common) and NO (Normally Open) terminals of the relay module to the terminals of the DC motor.
4. Connect the terminals of the DC motor to a separate power source suitable for the motor's voltage and current requirements (e.g., a battery).

Python Program:

1. Save the following Python script as **motor_control.py** on your Raspberry Pi:

```
import RPi.GPIO as GPIO
import time

# GPIO pin number for the relay control
RELAY_PIN = 17

# Setup GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(RELAY_PIN, GPIO.OUT)

# Function to turn the relay on or off
def set_relay(state):
    GPIO.output(RELAY_PIN, state)

try:
    while True:
        print("1. Turn on the motor")
        print("2. Turn off the motor")
        print("3. Quit")

        choice = input("Enter your choice (1/2/3): ")
        if choice == '1':
            set_relay(True)  # Turn on the motor
        elif choice == '2':
            set_relay(False)  # Turn off the motor
        elif choice == '3':
            break
        else:
            print("Invalid choice. Please enter 1, 2, or 3.")

except KeyboardInterrupt:
    pass

# Turn off the relay and clean up GPIO on program exit
set_relay(False)
GPIO.cleanup()
```

2. Open a terminal on your Raspberry Pi and navigate to the folder where you saved the script.
3. Run the script using the command:

python motor_control.py

The script will display a menu that allows you to turn the motor on or off, and you can quit the script by selecting option 3

10. CONTROL A 230V DEVICE USING A THRESHOLD TEMPERATURE, USING TEMPERATURE SENSOR.

To control a 230V device based on a threshold temperature using a temperature sensor and a Raspberry Pi, you'll need a temperature sensor (such as the DS18B20) and a relay module. Here's how to set up the connection and write a Python program:

Circuit Connection:

1. Connect the data pin of the temperature sensor to GPIO 4 (BCM numbering) on the Raspberry Pi.
2. Connect the VCC and GND pins of the temperature sensor to 3.3V and GND on the Raspberry Pi, respectively.
3. Connect the VCC and GND pins of the relay module to 5V and GND on the Raspberry Pi, respectively.
4. Connect the IN pin of the relay module to a GPIO pin on the Raspberry Pi (e.g., GPIO 17).
5. Connect the COM (Common) and NO (Normally Open) terminals of the relay module to the device you want to control (e.g., a 230V device).

Python Program:

1. Enable the 1-Wire interface for the DS18B20 temperature sensor:
   Open a terminal and enter the following commands:

```
. sudo raspi-config
```

1. Navigate to Interface Options > 1-Wire and enable it. Reboot if prompted.
2. Save the following Python script as temperature_control.py on your Raspberry Pi:

```
import RPi.GPIO as GPIO
import time
import os

# GPIO pin number for the relay control
RELAY_PIN = 17

# Path to the DS18B20 sensor data
SENSOR_PATH = "/sys/bus/w1/devices/28-*/w1_slave"

# Threshold temperature in Celsius
THRESHOLD_TEMPERATURE = 25.0

# Setup GPIO
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(RELAY_PIN, GPIO.OUT)

# Function to turn the relay on or off
def set_relay(state):
    GPIO.output(RELAY_PIN, state)

# Function to read temperature from the DS18B20 sensor
def read_temperature():
    try:
        sensor_file = open(SENSOR_PATH, "r")
        lines = sensor_file.readlines()
        sensor_file.close()

        # Extract temperature from the second line
        temperature_str = lines[1].strip().split(" ")[-1]
        temperature = float(temperature_str[2:]) / 1000.0

        return temperature
    except:
        return None

try:
    while True:
        temperature = read_temperature()

        if temperature is not None:
            print(f"Current temperature: {temperature:.2f}°C")

            if temperature >= THRESHOLD_TEMPERATURE:
                set_relay(True)  # Turn on the device
                print("Device turned on")
            else:
                set_relay(False)  # Turn off the device
                print("Device turned off")
        else:
            print("Error reading temperature")

        time.sleep(5)

except KeyboardInterrupt:
    pass
```

```
# Turn off the relay and clean up GPIO on program exit
set_relay(False)
GPIO.cleanup()
```

3. Open a terminal on your Raspberry Pi and navigate to the folder where you saved the script.
4. Run the script using the command:

```
python temperature_control.py
```

We will read the temperature from the DS18B20 sensor and control the 230V device using the relay module based on the specified threshold temperature. Make sure to adjust the GPIO pin numbers, sensor path, and other settings based on your actual setup.