EXPERIMENT NUMBER : 8

EXPERIMENT NAME : GPIO AND ASSOCIATED PERIPHERALS INTERFACING

DATE : 10/12/2022, SATURDAY

9/10

* AIM :

To interface general - Purpose Input-Output (GPIO) and associated peripherals.

* INTEGRATED DEVELOPMENT ENVIRONMENT (IDE) :

Name - Thonny 4.0.1

Publisher - Aivar Annamaa

Support link - https://thonny.org

* IMPORT NECESSARY LIBRARIES :

① Import picamera, provides a pure Python interface to the Raspberry Pi camera module.

② Import RPi.GPIO as GPIO, module to control the GPIO on a Raspberry Pi.

③ Import time, provides various time-related functions.

④ Disable warnings

(a) LED Blinking Using Raspberry Pi

→ ALGORITHM -

① Select a pin to be used for connecting LED.

② Physical pin number numbering scheme is followed; for GPIO numbering, choose BCM.

③ Selected pin is configured as output.

④ Turn the LED ON; sleep for 1 second.

⑤ Turn the LED OFF; sleep for 1 second.

⑥ Clear GPIO pins.

PTO

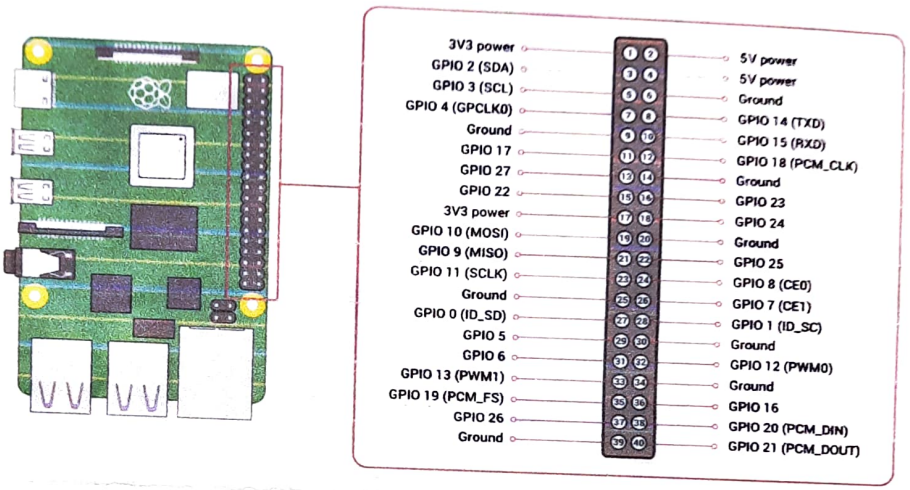


Figure 1 - Raspberry Pi Pin Diagram

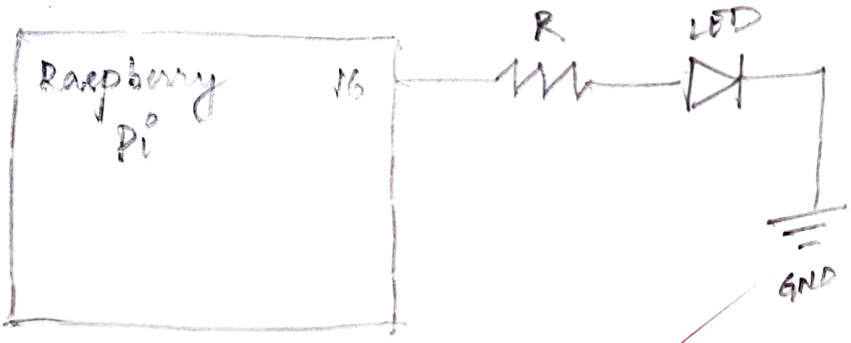


Figure 2 - LED Blinking Using Raspberry Pi

→ ALTHON CODE -

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)

# ledpin = 36
ledpin = 16

# GPIO.setmode(GPIO.BOARD)
GPIO.setmode(GPIO.BCM)

GPIO.setup(ledpin, GPIO.OUT)

while True:
        GPIO.output(ledpin, True)
        time.sleep(1)
        GPIO.output(ledpin, False)
        time.sleep(1)

GPIO.cleanup()
```

(b) LED Control using Switch:

→ ALGORITHM -

① Physical pin number numbering scheme is followed.
② Selected pin is configured as output.
③ Enable Pull.
④ Read status of pin/port and assign to variable switch.
- Turn the LED ON.
- Turn the LED OFF.
⑤ Clear GPIO pins.

PTO

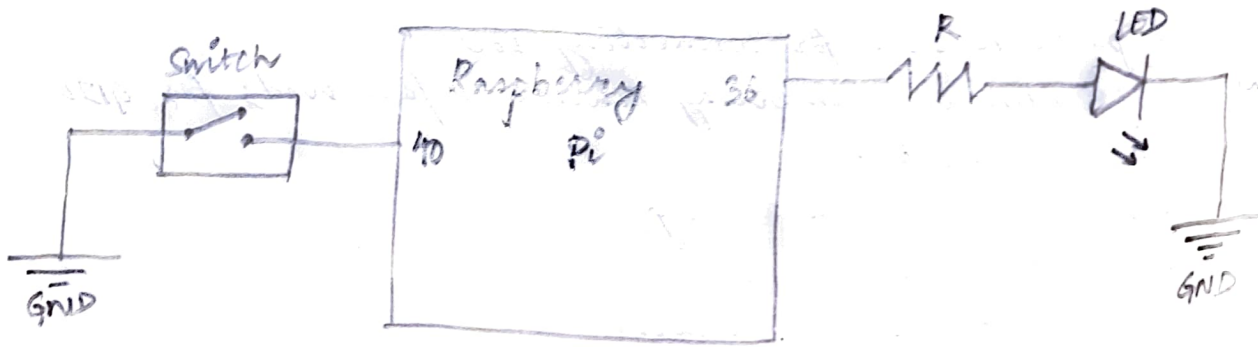


Figure 3 - LED control using Switch

→ PYTHON CODE -

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)

ledpin = 36
switch = 40

GPIO.setmode(GPIO.BOARD)
GPIO.setup(ledpin, GPIO.OUT)
GPIO.setup(switch, GPIO.IN, pull_up_down = GPIO.PUD_UP)

while True:
    status = GPIO.input(switch)
    if (status):
        GPIO.output(ledpin, True)
        print("LED ON: Button Pressed!")
    else:
        GPIO.output(ledpin, False)
        print("Button NOT Pressed!")

GPIO.cleanup()
```

(C) LED Fade In and Fade Out using PWM:

→ ALGORITHM -

① Physical pin number numbering scheme is followed and selected pin is configured as output.

② Select pin number and frequency in Hertz. Start PWM signal generation with 0% duty cycle.

③ Increase & Decrease duty cycle from 0 to 100 and 100 to 0 consecutively.
(i) Change duty cycle.
(ii) Suspend execution for a given number of seconds.

④ For exception handling, turn PWM off and reset GPIO pins.

→ PYTHON CODE -

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)

ledpin = 32
GPIO.setmode(GPIO.BOARD)
GPIO.setup(ledpin, GPIO.OUT)

dimmer = GPIO.PWM(ledpin, 50)
dimmer.start(0)
try:
    while(True):

        for i in range(0,100,5):
            dimmer.ChangeDutyCycle(i)
            time.sleep(0.2)

        for i in range(100,0,-5)
            dimmer.ChangeDutyCycle(i)
            time.sleep(0.2)

except KeyboardInterrupt:
        dimmer.stop()
        GPIO.cleanup()
```

(9) Direction Control of Servo motor using PWM:

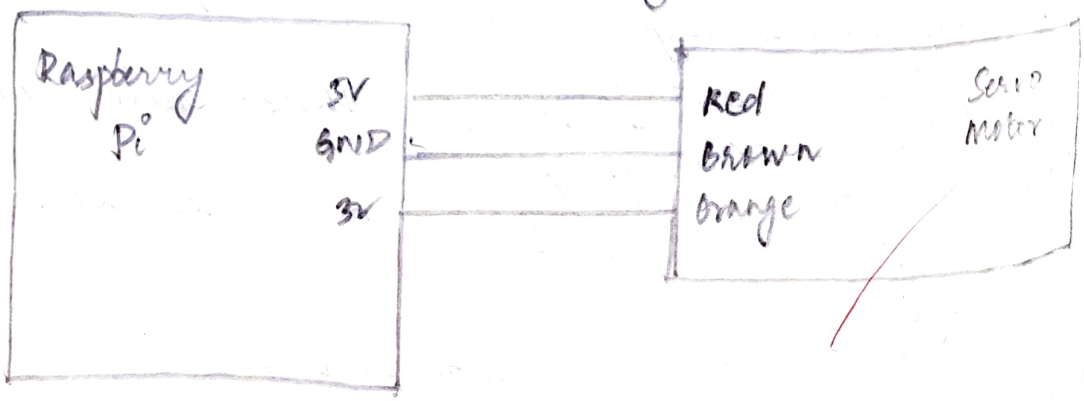
→ ALGORITHM -

① Pin to be connected to motor : GPIO 18. The selected pin is configured as output. Select pin number and frequency in Hertz.

② Start PWM signal generation with 0% duty cycle. Increase duty cycle from 0 to 100 and change duty cycle. Suspend execution for a given number of seconds.

③ For exceptions handling, turn PWM off and reset GPIO pins.

Figure 4 - Direction control of Servo Motor Using PWM



→ PYTHON CODE -

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)

motorpin = 12
GPIO.setup(motorpin, GPIO.OUT)
servo = GPIO.PWM(motorpin, 50)
servo.start(0)

try:
    while(True):
        for i in range(0, 100, 5):
            servo.ChangeDutyCycle(i)
            time.sleep(0.2)

except KeyboardInterrupt:
    print("Motor stopped!")
    servo.stop()
    GPIO.cleanup()
```

(e) Assignment:

Capture image when button is pressed.
Save the image as well.

→ ALGORITHM -

① Physical pin number numbering scheme is followed and pull is enabled.

② Read status of pin/port and assign to variable switch.

③ Retrieves or sets the current rotation of the camera's image and set the opacity of the renderer.

④ suspends execution for the given number of seconds. Capture an image from the camera, storing it in output.

⑤ Hide the preview overlay and finalize the state of the camera. Clear GPIO pins.

→ PYTHON CODE -

```
import picamera
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)

switch = 40
GPIO.setmode(GPIO.BOARD)
GPIO.setup(switch, GPIO.IN, pull_up_down = GPIO.PUD_UP)

while True:
    status = GPIO.input(switch)
    if (status):

        picam = picamera.PiCamera()
        picam.rotation = 180
        picam.start_preview(alpha = 200)
        time.sleep(5)

        picam.capture("Rotated_Image.png")
        picam.stop_preview()
        picam.close()

GPIO.cleanup()
```

* RESULT:

Thus GPIO and other peripherals interfacing demonstrated. All the simulation results were verified successfully.

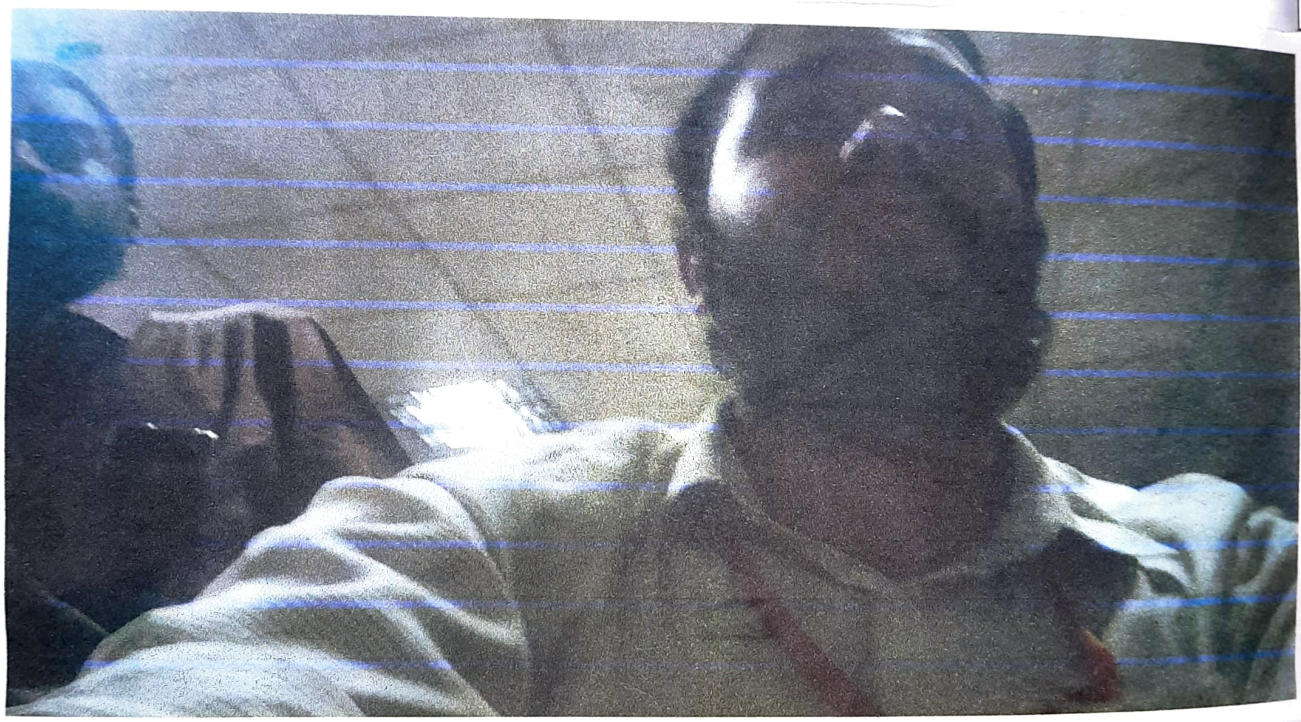


Figure 5 - Assignment output