PRACTICAL NO. 6

DESIGNING A WEB INTERFACE USING RASPBERRY PI

INSTRUCTOR : ENG D. D. G. R. KARUNARATHNE GROUP NO. : 4

GROUP MEMBERS :

D/ENG/22/0049 – J. KALISTAN

D/ENG/22/0066 – A. M. A. K. PRATHIBHATH

D/ENG/22/0085 – P. DANTHANARAYANA

D/ENG/22/0120 – M. A. E. WIJESURIYA

INTAKE : 39

STREAM : ET

COURSE: ET 4242 – INTERNET OF THINGS

DATE OF PERFORMANCE : 11.11.2025

DATE OF SUBMISSION : 14.11.2025

AIM

- To create a simple Python script to interface with the General Purpose Input/Output (GPIO) of the Raspberry Pi platform

OBJECTIVES

- To become familiar with the Apache server platform
- To utilize Python scripts to control and read hardware outputs and inputs

APPARATUS

- Raspberry Pi with:
    - Peripherals (Keyboard, Mouse)
    - Internet Access
    - Apache Server installed
- LED
- DIP Switch

PROCEDURE

1. First, the Raspberry Pi was setup and connected to the LED on GPIO Pin 7, and the DIP Switch on GPIO Pin 25. A Python script *script7_1.py* was created to switch the LED on and off using the DIP switch as a toggle.

2. Next, a square wave was generated by writing *script7_2.py*, that flashed the LED on and off at the desired frequency and duty cycle. The duty cycle and frequency was varied and the output observed.

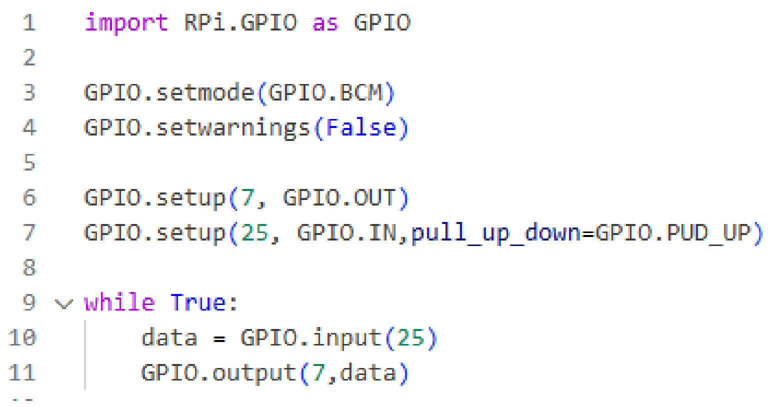
3. Then the Apache web server was installed onto the device and configured to serve web pages.

4. The webpage *lab7_3.htm* and the script *script7_3.py* were written and executed in order to create a form to toggle the LED on and off. The webpage was served locally through Apache.

5. Similarly, the webpage *lab7_4.htm* and the script *script7_4.py* were written and executed to read and display the value of the DIP switch when it was toggled.

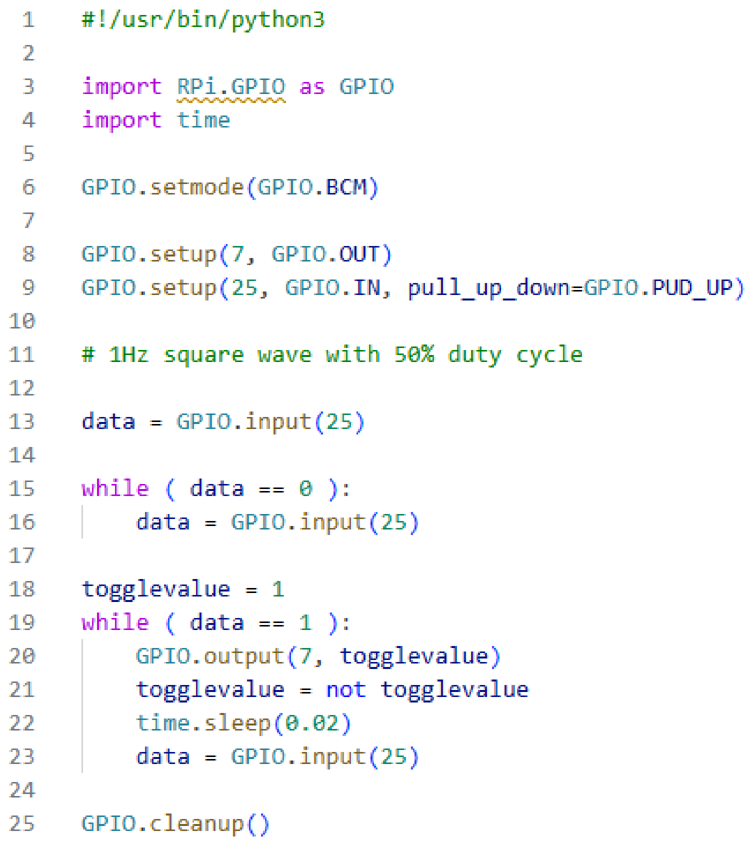
6. Finally, the functionality of both scripts were combined into a single webpage in *lab7_5.htm* and *script7_5.py*. This simultaneously showed the value of the DIP switch and toggled the LED on and off.

CODES



```python
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

GPIO.setup(7, GPIO.OUT)
GPIO.setup(25, GPIO.IN,pull_up_down=GPIO.PUD_UP)

while True:
    data = GPIO.input(25)
    GPIO.output(7,data)
```

Figure 1: script7_1.py



```python
#!/usr/bin/python3

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO.setup(7, GPIO.OUT)
GPIO.setup(25, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# 1Hz square wave with 50% duty cycle

data = GPIO.input(25)

while ( data == 0 ):
    data = GPIO.input(25)

togglevalue = 1
while ( data == 1 ):
    GPIO.output(7, togglevalue)
    togglevalue = not togglevalue
    time.sleep(0.02)
    data = GPIO.input(25)

GPIO.cleanup()
```

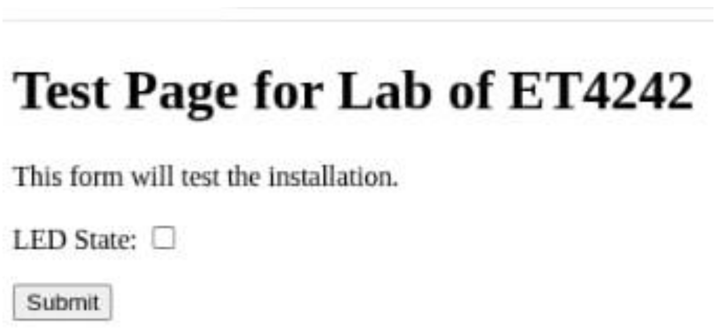
Figure 2: script7_2.py

```python
#!/usr/bin/python3
import cgi, cgitb
import RPi.GPIO as GPIO

cgitb.enable()

GPIO.setmode(GPIO.BCM)
GPIO.setup(7, GPIO.OUT)

form = cgi.FieldStorage()
userin = form.getvalue('led')

print("Content-type: text/html\r\n\r\n")
print("<html><body>")
if (userin == "power_on" ):
    print("<h1>LED: Power On!<br></h1>")
    GPIO.output(7, GPIO.HIGH)
else:
    print("<h1>LED: Power Off!<br></h1>")
    GPIO.output(7, GPIO.LOW)

print ("</body></html>")
```

Figure 3: script7_3.py

```html
<!DOCTYPE HTML>

<html>
<boy>
<h1>Test Page for Lab of ET4242</h1>
<p>This form will test the installation.</p>
    <form action="/cgi-bin/script7_3.py" method="POST">
        <p>
        LED State: <input type="checkbox" name="led" value="power_on"/></p>
        <p><input type="submit"></p>GP
    </form>
</body>
</html>
```

Figure 4: lab7_3.htm



Test Page for Lab of ET4242

This form will test the installation.

LED State: ☐

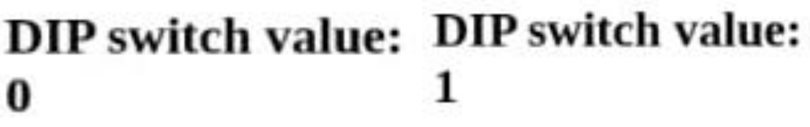
Figure 5: Output of lab7_3.htm

```python
#!/usr/bin/python3
import cgi
import cgitb
import RPi.GPIO as GPIO

cgitb.enable()

GPIO.setmode(GPIO.BCM)
GPIO.setup(25, GPIO.IN, pull_up_down=GPIO.PUD_UP)

data = GPIO.input(25)
GPIO.cleanup()

print("Content-type: text/html\r\n\r\n")
print("<html><body>")
print("%i" % data)
print("</body></html>")
```

Figure 6: script7_4.py

```html
<!DOCTYPE HTML>

<html>
<body>
    <h2>
    DIP switch value:
    <div id="dipdata"></div>
    </h2>

    <script type="text/javascript">

    setInterval (loaddata, 500);
    function loaddata()
    {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function(){
            if( this.readyState == 4 && this.status == 200){
                document.getElementById("dipdata").innerHTML =
                    this.responseText;
            }
        };

        xhttp.open("GET", "/cgi-bin/script7_4.py", true);
        xhttp.send();
    }
    </script>
</body>
</html>
```

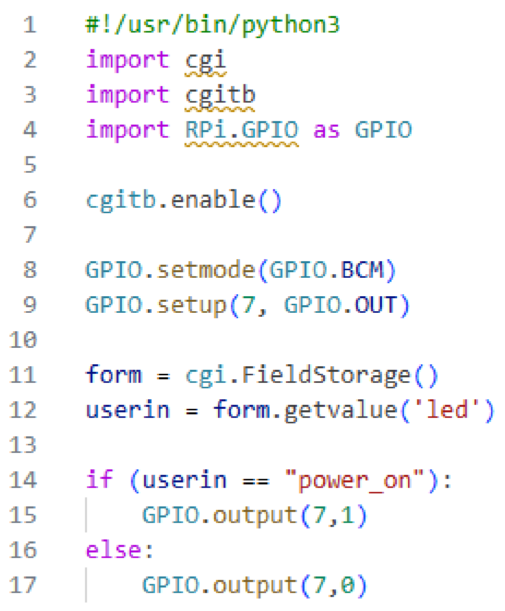
Figure 7: lab7_4.htm



DIP switch value: 0

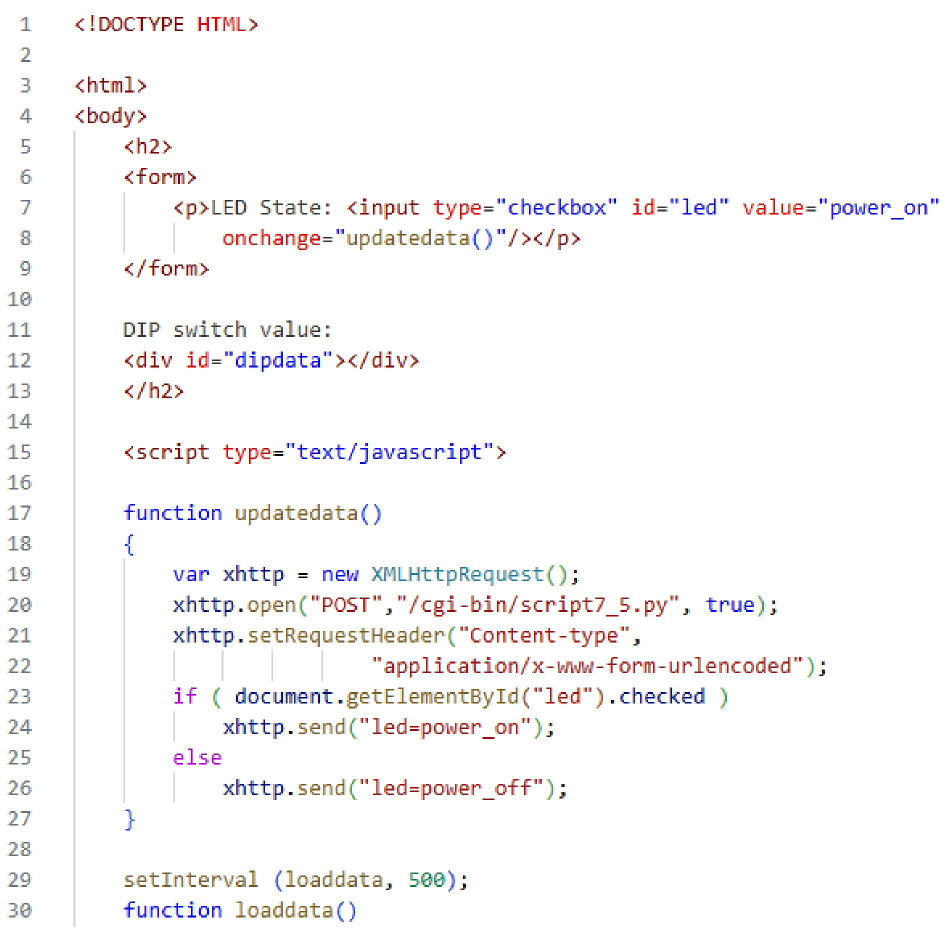
DIP switch value: 1

Figure 8: lab7_4.htm output

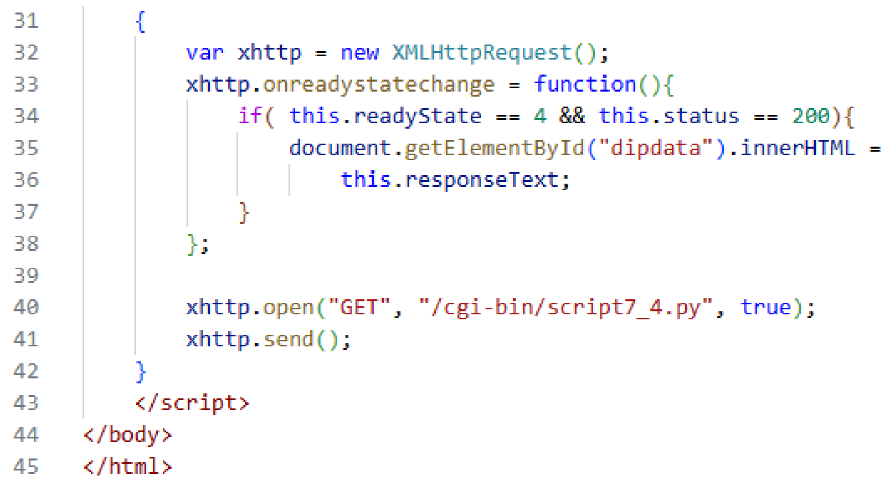


```python
#!/usr/bin/python3
import cgi
import cgitb
import RPi.GPIO as GPIO

cgitb.enable()

GPIO.setmode(GPIO.BCM)
GPIO.setup(7, GPIO.OUT)

form = cgi.FieldStorage()
userin = form.getvalue('led')

if (userin == "power_on"):
    GPIO.output(7,1)
else:
    GPIO.output(7,0)
```

Figure 9: script7_5.py

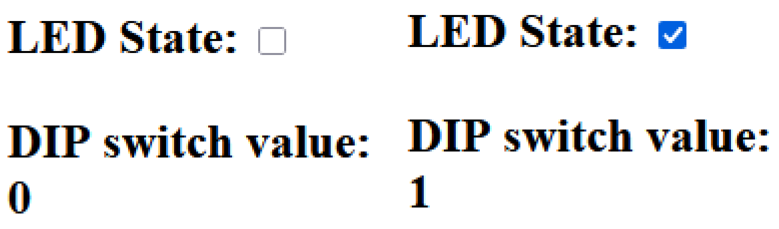


```html
<!DOCTYPE HTML>

<html>
<body>
    <h2>
    <form>
        <p>LED State: <input type="checkbox" id="led" value="power_on"
            onchange="updatedata()"/></p>
    </form>

    DIP switch value:
    <div id="dipdata"></div>
    </h2>

    <script type="text/javascript">

    function updatedata()
    {
        var xhttp = new XMLHttpRequest();
        xhttp.open("POST","/cgi-bin/script7_5.py", true);
        xhttp.setRequestHeader("Content-type",
                        "application/x-www-form-urlencoded");
        if ( document.getElementById("led").checked )
            xhttp.send("led=power_on");
        else
            xhttp.send("led=power_off");
    }

    setInterval (loaddata, 500);
    function loaddata()
```



```
    {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function(){
            if( this.readyState == 4 && this.status == 200){
                document.getElementById("dipdata").innerHTML =
                    this.responseText;
            }
        };

        xhttp.open("GET", "/cgi-bin/script7_4.py", true);
        xhttp.send();
    }
    </script>
</body>
</html>
```

Figure 10: lab7_5.htm



LED State: ☐

DIP switch value: 0

LED State: ☑

DIP switch value: 1

Figure 11: Output of lab7_5.htm

DISCUSSION

Apache Web Server is an open-source software package that serves web pages and operates on a wide variety of platforms. Initially released in 1995, it now serves approximately 26.4% of all websites, according to a survey conducted in March 2025. As a lightweight platform, it is ideal for running on resource-constrained devices such as the Raspberry Pi, in order to locally host web interfaces or monitor and control devices.

As Apache has a modular architecture, this allows it to be integrated easily with PHP or Python scripts, as demonstrated in this practical. This setup allows the Python script to handle the sensor data collection, or device control, while the Apache server can expose these scripts as dynamic web pages, allowing users to monitor and control these IoT devices. Additionally, it has reliable performance, even on limited hardware due to its extensive documentation and stability.

Further development of the fundamentals demonstrated in this practical can be shown by integrating more complex frameworks, such as Flask or Django for more complex IoT applications. These make it easier to visualize sensor data in more intuitive and user-friendly ways, or send more complex commands to edge devices.

CONCLUSION

Through the successful completion of this practical, we have gained real-world experience in setting up and configuring Apache Web Servers in order to monitor and control IoT devices. The LED and the DIP Switch served as basic examples of how both input and output devices could be interfaced with a local web server in order to monitor and control these devices, respectively.

Writing the scripts and the web pages allowed us to clearly observe the differences in frontend and backend programming, and how the two sections must be connected together correctly for the proper functioning of the device as a part of the Internet of Things. We gained an intuitive understanding of how to build user-friendly interfaces to provide real-time monitoring and control for edge devices.

By using these open source tools such as Apache and Python, we are able to build powerful, efficient devices that can be added to the Internet of Things for a wide variety of practical applications.