# Lab no 03 Part 02 – Simple Application on Raspberry Pi

This lab introduces a simple application on Raspberry Pi.
In this Lab, you will control LED connected to Raspberry Pi using a webpage via Wi-Fi.

## Parts: -

1. Install Flask.
2. Connect the hardware.
3. Develop Python and HTML Code.

## Required Resources

- Raspberry Pi with a power adapter.
- SD card.
- Led.
- Resistor 330 ohm.
- Breadboard.

## Part 1: Install Flask.

Flask is used for developing web applications using python. To install Flask, you'll need to have pip installed.

**Open** Terminal on your Raspberry Pi.

**Run** the following commands to update your Pi and install pip:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install python-pip python-flask
```

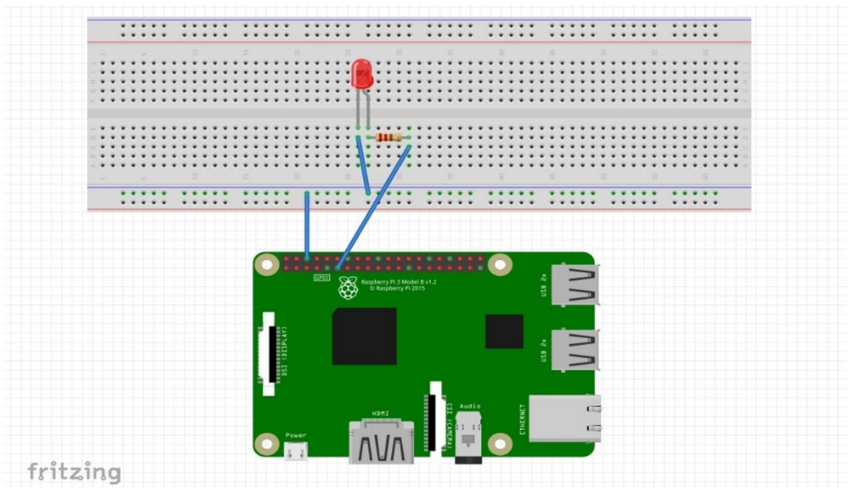
**Use** pip to install Flask and its dependencies:

```
sudo pip install flask
```

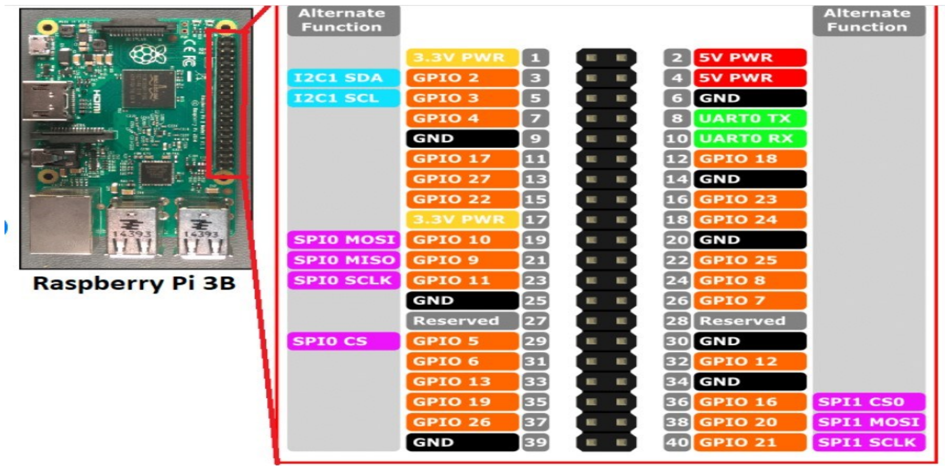
**Note**: **PyPI** is the Python Package index repository of python modules. **pip** is used to download and install packages directly from PyPI.

## Part 2: Connect the hardware

In the lab, our target is a simple application, to control a LED. The Figure below shows the circuit schematic. we connect LED to pin GPIO 17 of the Raspberry Pi.



**Note** GPIO 17 for the Raspberry Pi 3 +b. Check your kit version.


Raspberry Pi 3B
Alternate Function
Alternate Function
3.3V PWR 1 | 2 5V PWR
I2C1 SDA GPIO 2 3 | 4 5V PWR
I2C1 SCL GPIO 3 5 | 6 GND
GPIO 4 7 | 8 UART0 TX
GND 9 | 10 UART0 RX
GPIO 17 11 | 12 GPIO 18
GPIO 27 13 | 14 GND
GPIO 22 15 | 16 GPIO 23
3.3V PWR 17 | 18 GPIO 24
SPI0 MOSI GPIO 10 19 | 20 GND
SPI0 MISO GPIO 9 21 | 22 GPIO 25
SPI0 SCLK GPIO 11 23 | 24 GPIO 8
GND 25 | 26 GPIO 7
Reserved 27 | 28 Reserved
SPI0 CS GPIO 5 29 | 30 GND
GPIO 6 31 | 32 GPIO 12
GPIO 13 33 | 34 GND
GPIO 19 35 | 36 GPIO 16 SPI1 CS0
GPIO 26 37 | 38 GPIO 20 SPI1 MOSI
GND 39 | 40 GPIO 21 SPI1 SCLK

## Part 3: Develop Python and HTML Code.

- **Creating the Python Script**

1. **Create** a new folder and create a new file called app.py.

```
mkdir webserver
cd webserver
nano app.py
```

2. **Copy** and paste the following script to the Raspberry Pi terminal.

```
import RPi.GPIO as GPIO
from flask import Flask, render_template, request # used to generate output from a template file
app = Flask(__name__)
GPIO.setmode(GPIO.BCM)
# Create a dictionary called pins to store the pin number, name, and pin state:
pins = {17 : {'name' : 'GPIO 17', 'state' : GPIO.LOW}}
#Set each pin as an output and make it low:
GPIO.setup(pin, GPIO.OUT)
GPIO.output(pin, GPIO.LOW)
@app.route("/")
def main():
# For each pin, read the pin state and store it in the pins dictionary:
pins[pin]['state'] = GPIO.input(pin)
```

```
#Put the pin dictionary into the template data dictionary:
templateData = {'pins' : pins}
#Pass the template data into the template main.html and return it to the user
return render_template('main.html', **templateData)
#The function below is executed when someone requests a URL with the pin number and action in it:
@app.route("/<changePin>/<action>")
def action(changePin, action):
#Convert the pin from the URL into an integer:
changePin = int(changePin)
#Get the device name for the pin being changed:
deviceName = pins[changePin]['name']
#If the action part of the URL is "on," execute the code indented below:
if action == "on":
#Set the pin high:
GPIO.output(changePin, GPIO.HIGH)
#Save the status message to be passed into the template:
message = "Turned " + deviceName + " on."
if action == "off":
GPIO.output(changePin, GPIO.LOW)
message = "Turned " + deviceName + " off."
# For each pin, read the pin state and store it in the pins dictionary:
pins[pin]['state'] = GPIO.input(pin)
#Along with the pin dictionary, put the message into the template data dictionary:
templateData = {'pins' : pins}
return render_template('main.html', **templateData)
if __name__ == "__main__":
app.run(host='0.0.0.0', port=80, debug=True)
```

- **Creating the HTML File**

  Keeping HTML tags separated from your Python script is the key to keep your project organized.

  Flask uses a template engine called Jinja2 that can be uses to send dynamic data from the Python script to the HTML file. Follow the steps below:

  1. **Create** a new folder in the webserver folder called templates:

```
mkdir templates
cd templates
```

2. **Create** a new file called main.html.

```
nano main.html
```

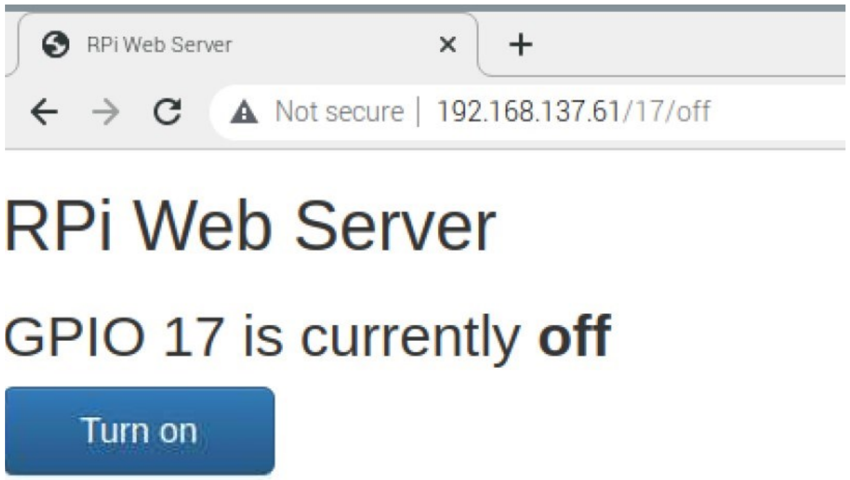
3. **Copy** and paste the following template to your Pi:

```
<!DOCTYPE html>
<head>
<title>RPi Web Server</title>
</head>
<body>
<h1>RPi Web Server</h1>
{% for pin in pins %}
<h2>{{ pins[pin].name }}
{% if pins[pin].state == true %}
is currently <strong>on</strong></h2><div class="row"><div class="col-md-2">
<a href="/{{pin}}/off" class="btn btn-block btn-lg btn-default" role="button">Turn off</a></div></div>
{% else %}
is currently <strong>off</strong></h2><div class="row"><div class="col-md-2">
<a href="/{{pin}}/on" class="btn btn-block btn-lg btn-primary" role="button">Turn on</a></div></div>
{% endif %}
{% endfor %}
</body>
</html>
```

4. **Run** the following command:

```
sudo python app.py
```

5. **Open** your Raspberry Pi address in your browser by entering its IP address (192.168.137.61).
6. **Control** the LED using Turn On/OFF Button



RPi Web Server

GPIO 17 is currently **off**