Contents

1 Creating a browser-based UI

One of the major advantages of using a single board computer is that it runs a full operating system, with existing libraries and software. In this lab, you will learn how to use that to create a browser-based interface through which users can interact with a Pi-based product.

1.1 By the end of this session...

You will be able to:

- Create and run a basic Flask app that interacts with a *virtual* "HAT library"

(a HAT is a hardware circuit that is attached to a Pi - **H**ardware **A**ttached on **T**op).

In a future lab assignment, we'll use this browser-based interface to control a real circuit connected to the Pi.

1.2 Lab report

You will submit your lab work in Gradescope. You will upload some screenshots and answer some questions as described in the Gradescope assignment. You do not have to include anything else (e.g. no description of procedure, etc.)

1.3 The VirtualHat library

Since we don't have any circuit attached to this Pi, we're going to use a VirtualHat library I created for this lab, which uses a "virtual circuit". Get the code for my HAT and install it on your VM with:

```
git clone https://github.com/ffund/virtualhat
cd virtualhat
sudo python3 setup.py install
```

The VirtualHat library includes the following functions:

- `setup()` initializes all of the peripherals and sensors in the VirtualHat
- `led_on()` turns on a virtual LED
- `led_off()` turns off a virtual LED
- `read_light_level()` reads the current light level from a virtual sensor

Read and then try running the `demo.py` file included with the library to see how these are used. To run the `demo.py` file, use

```
python3 demo.py
```

Now we are ready to create a basic browser-based interface to the VirtualHat library. We will use Flask, a lightweight web framework for Python applications.

Create a new directory called `lab1` in your home directory, and create an `app` inside *that* directory, then navigate to it:

```
mkdir /home/pi/lab1
mkdir /home/pi/lab1/app
cd /home/pi/lab1/app
```

Then, use your preferred terminal-based text editor to create a file `basic-flask-app.py` with the following contents:

```
from flask import Flask

app = Flask(__name__, static_folder='')

@app.route("/")
def hello():
    return app.send_static_file('index.html')

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True, threaded=True)
```

In this code, we:

- imported the `Flask` class from the `flask` library
- created an instance of the `Flask` class, calling it `app`
- set up a "route" that will be triggered when the user visits `/`, which is the root of the app; this is the route that is triggered when the user visits the home page of a site. We will run the function `hello()` when the route is triggered; this function just sends the static web page called "index.html" to the user. (We'll create `index.html` in just a moment.)
- set up the Flask app to run when this `basic-flask-app.py` file is called as a Python script (not when it is imported as a library).

As far as networking: we configured the app to run on the IP address `0.0.0.0`, which means that it will run on all IP addresses available on this host, and it will run on port 80, which is the default port for web pages that are not encrypted.

Save and close the Flask app file. Then, create a file `index.html` in the same directory, with the following contents, and save and close that file as well:

```
<!DOCTYPE html>
   <head>
      <title>Hello Flask!</title>
      <link rel="stylesheet"
          href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
   </head>
   <body>
   <div class="container">
    <h1>Hello Flask</h1>
   </div>
   </body>
</html>
```

(Note that we are loading the Bootstrap stylesheet, which will affect the appearance of the elements on our page. We'll talk more about Bootstrap soon.)

When you are ready, run the `basic-flask-app.py` Python script. You will need to run it as `sudo`, because only privileged applications can use the default web port, port 80:

```
sudo python3 basic-flask-app.py
```

You should see some output similar to the following;

```
 * Serving Flask app "basic-flask-app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 237-595-907
```

Copy and paste the Pi's address into the address bar of a browser, then hit Enter to load the site. You should see the following page: