Lab 6
Xavier Thompson

Lab 6.1
ChatGPT

Prompt: Set up a customized Web Server running with in the
IP address space of your Local Area Network using Python and Flask. We will refer to the this web server as the “Weather Monitor”. It will perform
two tasks, which will both use the port: 1234.
You will also set up a HotSpot that will allow the ESP32C3 peripheral (the
Weather Station) to have WiFi access to the Internet (WAN), and to the
WebServer (LAN). You will have to make that the HotSpot can access the IP
addresses in your local area network.
We will refer to the ESP32C3 as the “Weather Station”. The “Weather Station”
will be directly accessing the website: www.wttr.in in order to get access to
outdoor weather information at numerous locations in the World.
The Hotspot allows the “Weather Station” (ESP32C3) to communicate with both
the Internet, and the Weather Monitor (web server) accessible from your Local
Area Network. We will test that this all works via the use of the program curl,
performing HTTP GET and POST requests based on the following commands:

```
weather_server=192.168.137.129:1234
curl http://$weather_server/readfile/location.txt
```

This should return a string with the location found in file: location.txt That file
should contain something like: “Scotts+Valley”. Curl should return the contents
of that file. You will have to customize your web server so this works.
You will also use an HTTP POST to have your mini-web server output a string.
By running the following script.
1

```
weather_server=192.168.137.129:1234
location="Scotts Valley"
contents="$location: Outdoors T:74F, H:36%, Indoors T:70F, H:40%\n"
curl -X POST -H "Content-Type: text/plain; charset=utf-8" \
-d "$contents" $weather_server/weather
```

The above HTTP POST will cause the “Weather Monitor” to generate output
that looks something like this:
"Scotts Valley: Outdoors T:47F, H:36%, Indoors T:70F, H:40%"
The \n should provide a line feed, so that it can be done multiple times, one

line after the other.
Finally running the next command should access the remote web site www.wttr.in
and get the outdoor weather parameters from whatever location you specify.
Here we are specifying the city of Scotts Valley.

```
loc="Scotts+Valley"
curl https://www.wttr.in/$loc?format=%l:+%t+%h
echo
```

It should generate output like this:

```
Scotts+Valley: +50°F 54%
```

Result: To achieve the setup described, you'll need to perform a few steps:

Install Flask:
Make sure you have Flask installed. If not, you can install it using the following command:

bash
Copy code

```
pip install Flask
```

Create the Flask Web Server:
Create a Python script for your web server, e.g., weather_monitor.py:

python
Copy code

```
from flask import Flask, request

app = Flask(__name__)

@app.route('/readfile/<filename>', methods=['GET'])
def read_file(filename):
    with open(filename, 'r') as file:
        return file.read()

@app.route('/weather', methods=['POST'])
def update_weather():
    data = request.data.decode('utf-8')
    with open('weather_data.txt', 'w') as file:
        file.write(data)
    return "Weather data updated successfully."
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=1234)
```

Run the Flask Web Server:
Execute the following command to run your Flask web server:

bash
Copy code

```
python weather_monitor.py
```

Create a Hotspot:
Set up a hotspot on your computer or use a dedicated hotspot device. Ensure that it has internet access.

Configure ESP32C3 to Connect to Hotspot:
Program your ESP32C3 to connect to the hotspot using the appropriate credentials.

Test with curl Commands:
Use the following curl commands for testing:

bash
Copy code

```
# Read file from Weather Monitor
weather_server=192.168.137.129:1234
curl http://$weather_server/readfile/location.txt

# Update weather data on Weather Monitor
location="Scotts Valley"
contents="$location: Outdoors T:74F, H:36%, Indoors T:70F, H:40%\n"
curl -X POST -H "Content-Type: text/plain; charset=utf-8" \
-d "$contents" $weather_server/weather

# Access remote website for outdoor weather parameters
loc="Scotts+Valley"
curl https://www.wttr.in/$loc?format=%l:+%t+%h
echo
```

Make sure to replace the IP address (192.168.137.129) with the actual IP address of your machine running the Flask server. Additionally, customize the file paths and contents in the Flask script as needed.

Note: This setup assumes that your machine is acting as both the Flask server and the hotspot provider. Adjustments may be needed based on your specific network configuration.

Lab 6.2

Prompt: How can I modify this C string to make a post command using a custom contents field:

```
static const char *REQUEST = "POST " WEB_PATH " HTTP/1.0\r\n"
    "Host: "WEB_SERVER":"WEB_PORT"\r\n"
    "User-Agent: esp-idf/1.0 esp32\r\n"
    "\r\n";
```

Result: To modify the C string for a POST command with custom content, you need to add the necessary HTTP headers for the POST request and include the content in the request body. Here's an example of how you can modify the string:

c

Copy code

```
static const char *CUSTOM_CONTENT = "your_custom_content_here";

static const char *REQUEST = "POST " WEB_PATH " HTTP/1.0\r\n"
    "Host: "WEB_SERVER":"WEB_PORT"\r\n"
    "User-Agent: esp-idf/1.0 esp32\r\n"
    "Content-Type: application/x-www-form-urlencoded\r\n" // Specify the content type
    "Content-Length: %d\r\n" // Include the length of your custom content
    "\r\n"
    "%s"; // Insert your custom content here using the format specifier

// Calculate the length of the final request string
int request_length = snprintf(NULL, 0, REQUEST, strlen(CUSTOM_CONTENT),
CUSTOM_CONTENT);
char *final_request = malloc(request_length + 1); // +1 for the null terminator

// Populate the final request string
snprintf(final_request, request_length + 1, REQUEST, strlen(CUSTOM_CONTENT),
CUSTOM_CONTENT);

// Now you can use `final_request` for your POST request
```

Replace "your_custom_content_here" with your actual custom content. This example assumes that your custom content is a string. Adjust the Content-Type header accordingly if your content is in a different format.

Note: Make sure to free the allocated memory for final_request when you're done using it.