Faculty of Engineering & Applied Science

Experiment Name: Using Docker for Automated System Deployment on a Raspberry Pi

Experiment date: 11/23/2022

Group Number*: 4*

Section CRN: 44432

Course Instructor: ***Ramiro Liscano***

Lab TA: ***Sifatul Mostafi***

Student Name	**Student Id**
Preet Patel	*100708239*
Tiwaloluwa Ojo	*100700622*
Waleed El Alawi	*100764573*

Learning Objective

The objective from this lab was to learn and understand how to install docker and the design and goals from docker. We also learned how to do the setup of the docker environment on the Raspberry PI. We were able to run docker images from docker hub. Lastly, we design an automated image for IOT deployment.

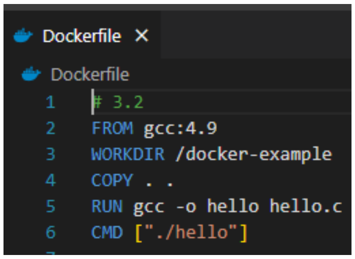
Deliverables

Task 3.2 Create docker file and run your own application

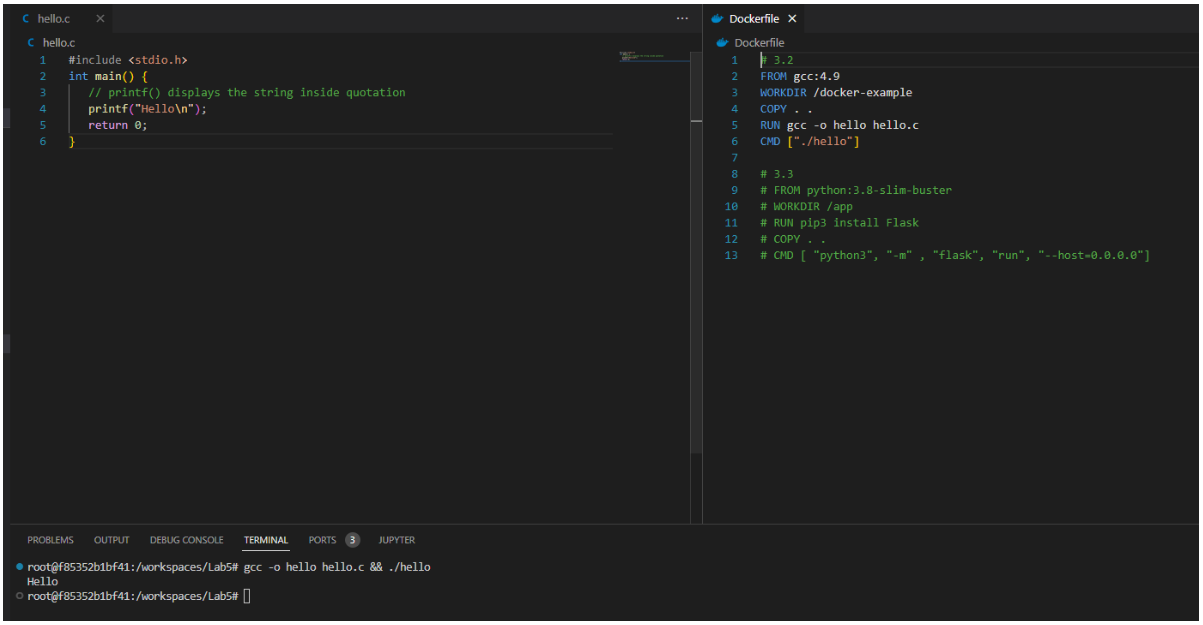
hello.c

```c
#include <stdio.h>
int main() {
   // printf() displays the string inside quotation
   printf("Hello\n");
   return 0;
}
```

Dockerfile



```dockerfile
# 3.2
FROM gcc:4.9
WORKDIR /docker-example
COPY . .
RUN gcc -o hello hello.c
CMD ["./hello"]
```



```c
#include <stdio.h>
int main() {
   // printf() displays the string inside quotation
   printf("Hello\n");
   return 0;
}
```

```dockerfile
# 3.2
FROM gcc:4.9
WORKDIR /docker-example
COPY . .
RUN gcc -o hello hello.c
CMD ["./hello"]

# 3.3
# FROM python:3.8-slim-buster
# WORKDIR /app
# RUN pip3 install Flask
# COPY . .
# CMD [ "python3", "-m" , "flask", "run", "--host=0.0.0.0"]
```

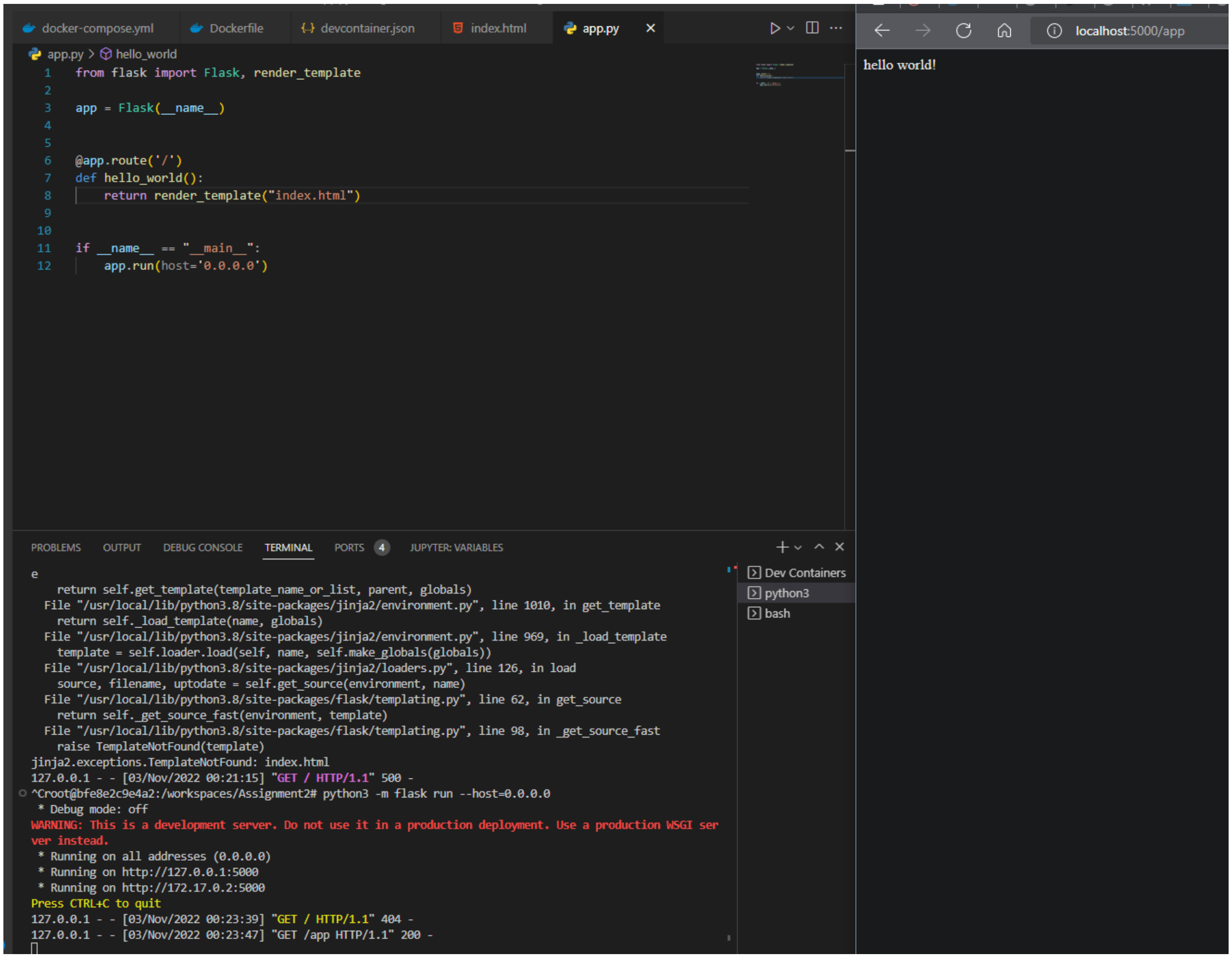
```
root@f85352b1bf41:/workspaces/Lab5# gcc -o hello hello.c && ./hello
Hello
root@f85352b1bf41:/workspaces/Lab5#
```

Task 3.3 Web server

Creating a web server printing “Hello World”

app.py code

```python
from flask import Flask, render_template

app = Flask(__name__)


@app.route('/')
def hello_world():
    return render_template("index.html")


if __name__ == "__main__":
    app.run(host='0.0.0.0')
```

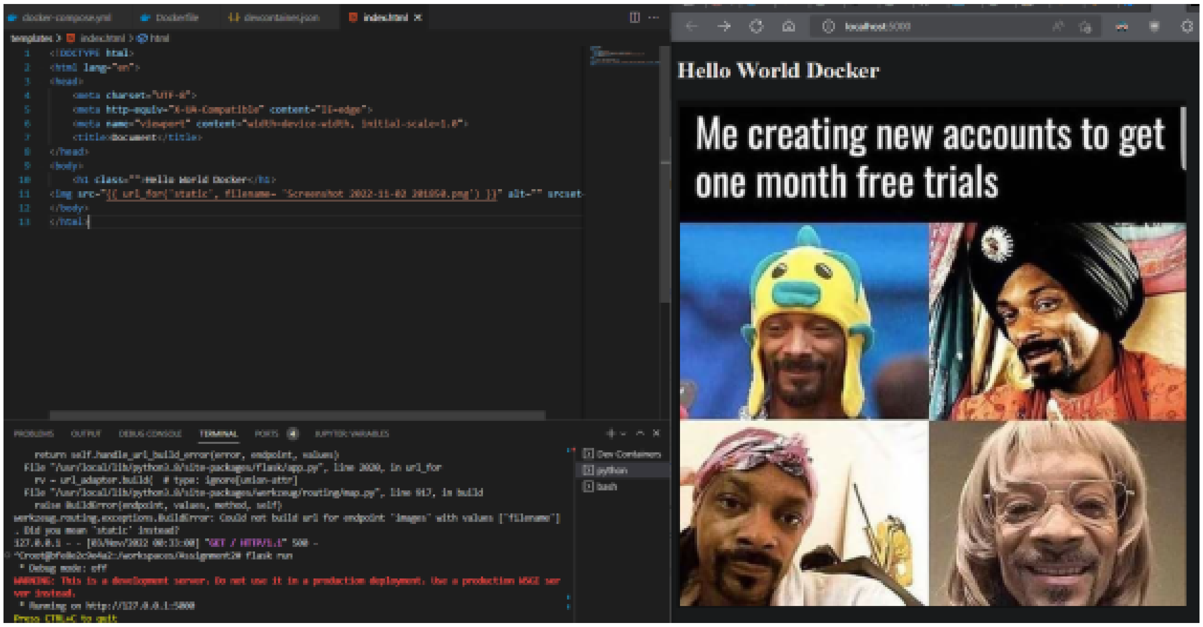


```python
from flask import Flask, render_template

app = Flask(__name__)


@app.route('/')
def hello_world():
    return render_template("index.html")


if __name__ == "__main__":
    app.run(host='0.0.0.0')
```

hello world!

```
e
    return self.get_template(template_name_or_list, parent, globals)
  File "/usr/local/lib/python3.8/site-packages/jinja2/environment.py", line 1010, in get_template
    return self._load_template(name, globals)
  File "/usr/local/lib/python3.8/site-packages/jinja2/environment.py", line 969, in _load_template
    template = self.loader.load(self, name, self.make_globals(globals))
  File "/usr/local/lib/python3.8/site-packages/jinja2/loaders.py", line 126, in load
    source, filename, uptodate = self.get_source(environment, name)
  File "/usr/local/lib/python3.8/site-packages/flask/templating.py", line 62, in get_source
    return self._get_source_fast(environment, template)
  File "/usr/local/lib/python3.8/site-packages/flask/templating.py", line 98, in _get_source_fast
    raise TemplateNotFound(template)
jinja2.exceptions.TemplateNotFound: index.html
127.0.0.1 - - [03/Nov/2022 00:21:15] "GET / HTTP/1.1" 500 -
^Croot@bfe8e2c9e4a2:/workspaces/Assignment2# python3 -m flask run --host=0.0.0.0
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI ser
ver instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
127.0.0.1 - - [03/Nov/2022 00:23:39] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [03/Nov/2022 00:23:47] "GET /app HTTP/1.1" 200 -
```

Adding the image to the web server.



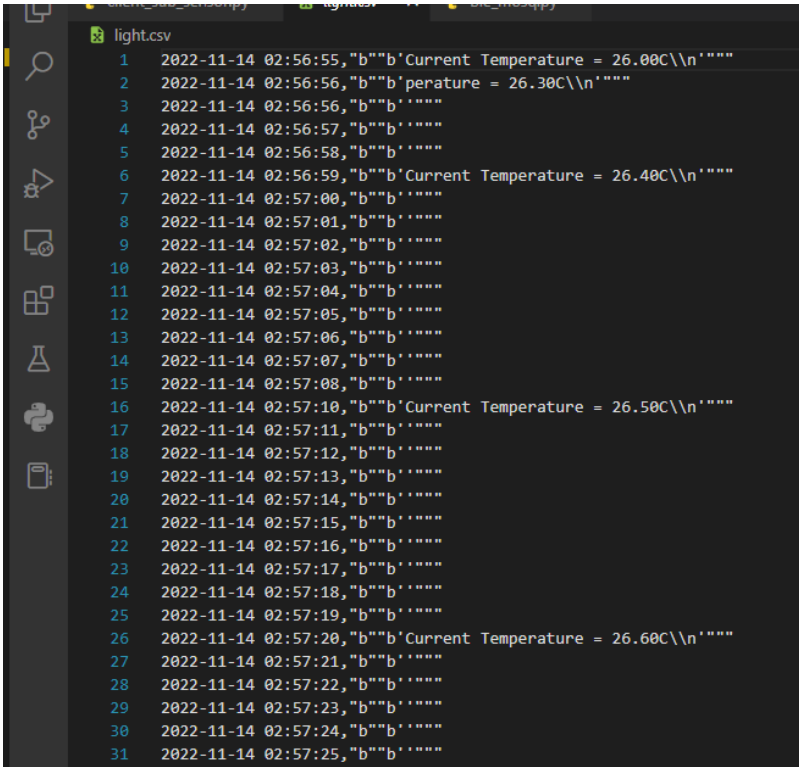
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1 class="">Hello World Docker</h1>
<img src="{{ url_for('static', filename= 'Screenshot 2022-11-02 201050.png') }}" alt="" srcset=
</body>
</html>
```

```
    return self.handle_url_build_error(error, endpoint, values)
  File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2020, in url_for
    rv = url_adapter.build(  # type: ignore[union-attr]
  File "/usr/local/lib/python3.8/site-packages/werkzeug/routing/map.py", line 917, in build
    raise BuildError(endpoint, values, method, self)
werkzeug.routing.exceptions.BuildError: Could not build url for endpoint 'images' with values ['filename']
. Did you mean 'static' instead?
127.0.0.1 - - [03/Nov/2022 08:33:08] "GET / HTTP/1.1" 500 -
^Croot@bfe8e2c9e4a2:/workspaces/Assignment2# flask run
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI ser
ver instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Hello World Docker

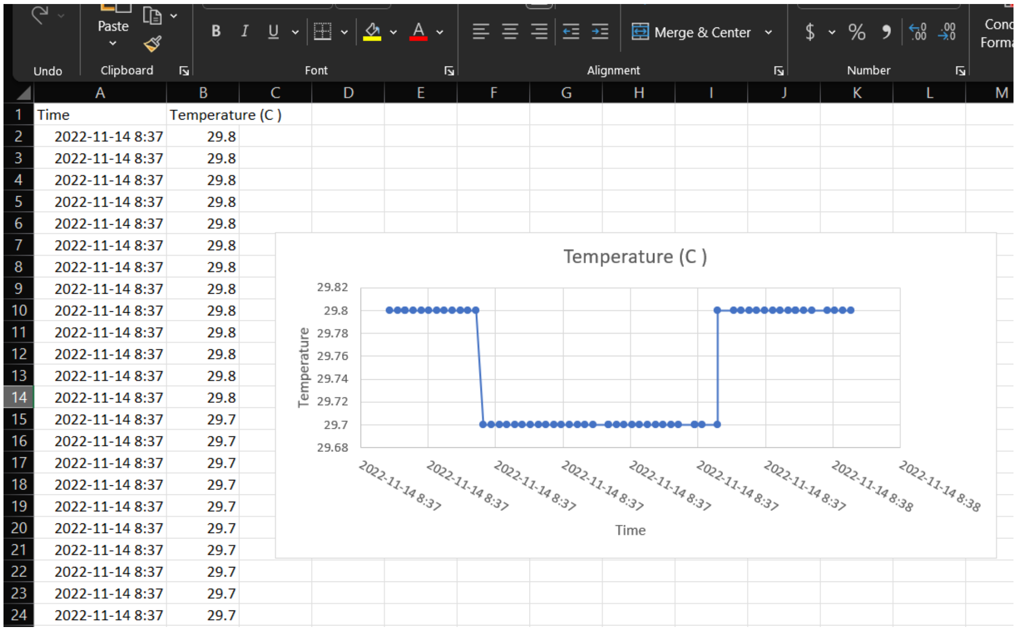
Task 3.4 Display DHT11 sensor data on the web page built on task 3

Data has been collected using DHT11 sensor



```
light.csv
2022-11-14 02:56:55,"b""b'Current Temperature = 26.00C\\n'"""
2022-11-14 02:56:56,"b""b'perature = 26.30C\\n'"""
2022-11-14 02:56:56,"b""b''"""
2022-11-14 02:56:57,"b""b''"""
2022-11-14 02:56:58,"b""b''"""
2022-11-14 02:56:59,"b""b'Current Temperature = 26.40C\\n'"""
2022-11-14 02:57:00,"b""b''"""
2022-11-14 02:57:01,"b""b''"""
2022-11-14 02:57:02,"b""b''"""
2022-11-14 02:57:03,"b""b''"""
2022-11-14 02:57:04,"b""b''"""
2022-11-14 02:57:05,"b""b''"""
2022-11-14 02:57:06,"b""b''"""
2022-11-14 02:57:07,"b""b''"""
2022-11-14 02:57:08,"b""b''"""
2022-11-14 02:57:10,"b""b'Current Temperature = 26.50C\\n'"""
2022-11-14 02:57:11,"b""b''"""
2022-11-14 02:57:12,"b""b''"""
2022-11-14 02:57:13,"b""b''"""
2022-11-14 02:57:14,"b""b''"""
2022-11-14 02:57:15,"b""b''"""
2022-11-14 02:57:16,"b""b''"""
2022-11-14 02:57:17,"b""b''"""
2022-11-14 02:57:18,"b""b''"""
2022-11-14 02:57:19,"b""b''"""
2022-11-14 02:57:20,"b""b'Current Temperature = 26.60C\\n'"""
2022-11-14 02:57:21,"b""b''"""
2022-11-14 02:57:22,"b""b''"""
2022-11-14 02:57:23,"b""b''"""
2022-11-14 02:57:24,"b""b''"""
2022-11-14 02:57:25,"b""b''"""
```

Plotting the data using excel

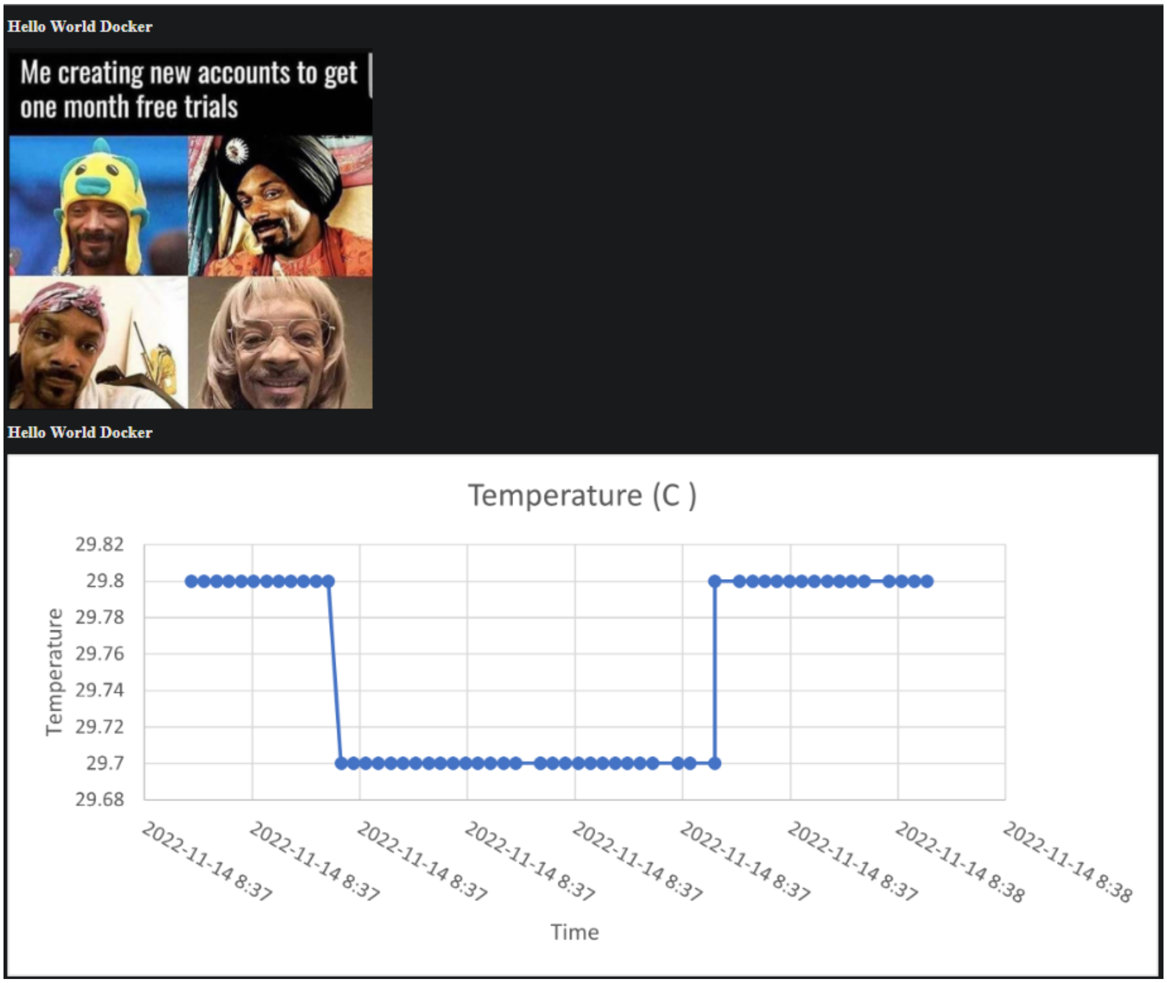


	A	B
1	Time	Temperature (C)
2	2022-11-14 8:37	29.8
3	2022-11-14 8:37	29.8
4	2022-11-14 8:37	29.8
5	2022-11-14 8:37	29.8
6	2022-11-14 8:37	29.8
7	2022-11-14 8:37	29.8
8	2022-11-14 8:37	29.8
9	2022-11-14 8:37	29.8
10	2022-11-14 8:37	29.8
11	2022-11-14 8:37	29.8
12	2022-11-14 8:37	29.8
13	2022-11-14 8:37	29.8
14	2022-11-14 8:37	29.8
15	2022-11-14 8:37	29.7
16	2022-11-14 8:37	29.7
17	2022-11-14 8:37	29.7
18	2022-11-14 8:37	29.7
19	2022-11-14 8:37	29.7
20	2022-11-14 8:37	29.7
21	2022-11-14 8:37	29.7
22	2022-11-14 8:37	29.7
23	2022-11-14 8:37	29.7
24	2022-11-14 8:37	29.7

Index.html Code

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1 class="">Hello World Docker</h1>
<img src="{{ url_for('static', filename= 'images/Screenshot 2022-11-02 201850.png')
}}" alt="" srcset="">
<h1 class="">Hello World Docker</h1>

<img src="{{ url_for('static', filename= 'images/temperature-data.png') }}" alt=""
srcset="">
</body>
</html>
```

Here is a demonstration of the sensor data plot image on the html page



Hello World Docker

Hello World Docker

Conclusion

In conclusion, in this lab we were able to install and learn about Docker. We understood what Docker is and how it works. In the lab, we learned how to create a dockerfile which can help in the generation of a docker image. Next, we ran the created docker image to create a docker container to view the output of the application. Then, we used python's web framework flask and docker to set up a web server which shows the output of the application we created on a web page. For this task, we also learned how to add an image to the flask web page which was useful for the next task. In the final task of this lab, we plotted the data collected from the temperature and humidity sensor and displayed the image of the plot on the web page that we built on task 3. VScode was heavily relied upon as well as its Dev Containers extension. It allowed us to use VScode as a containerized workspace