Infrastructure Portfolio Semester 1

Fontys

University of Applied Sciences

Student Name: Divesh Balani
Student Number: 4788036
PCN: 506743
Date: 14/11/2022

Table of Contents

➔ Technical Skills

◆ Develop

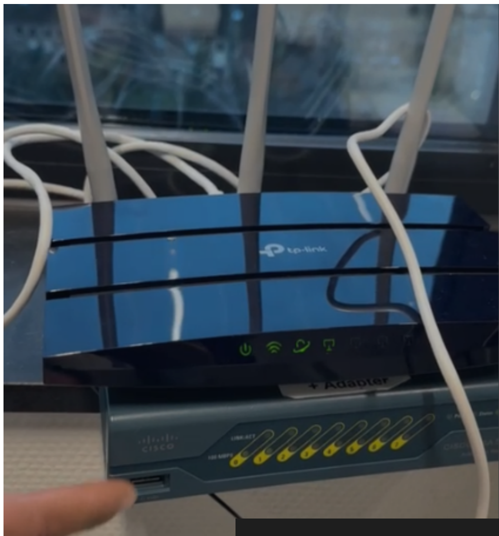
- Web Server: In the developmental stages of the semester for both the projects and workshops, we were able to develop a service for our project based on the workshops that were given by the teachers throughout the semester. In order to proceed with the development of the web service.

Flask was chosen as a web service because of our previous background knowledge with it. It allowed us to manipulate the site as we pleased. In order to get all of the information we needed.

- REST API: A NETIO cable and arduino were needed. The arduino was able to measure the light level, humidity and temperature for a thermostat and smart light system, and the NETIO cable was used to measure energy usage. In order for Flask and the NETIO cable to communicate, knowledge on REST API was needed. We learned about the REST API on week 15 day 2. In this, we learned that REST API is a type of API that is needed to communicate with http. As such, to implement the usage of REST API into the project, we used a get request for the NETIO IP and pulled information into a response.json file. From there, the data was rendered into an HTML template where data was able to be displayed onto our dashboard.

```
response = requests.get('http://192.168.4.6/netio.json')

voltage = response.json()['GlobalMeasure']['Voltage']
# print(voltage)

TotalCurrent = response.json()['GlobalMeasure']['TotalCurrent']
# print(TotalCurrent)

TotalEnergy = response.json()['GlobalMeasure']['TotalEnergy']
# print(TotalEnergy)

TotalLoad = response.json()['GlobalMeasure']['TotalLoad']
# print(TotalLoad)
```

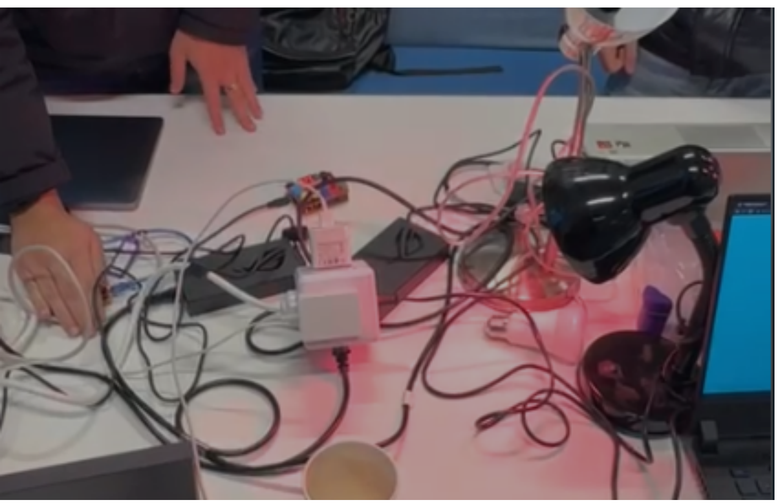
- Client Data: From there, a second client file was made to store client information and pull data from the arduino in particular. Using this data, a thermostat functionality was made for the web server.



- DHCP Server: Before running the flask server, a DHCP server was set up in the router settings. This allowed us to pre-define IPs for equipment such as the NETIO so we would not have to keep updating the get requests for it. Using the DHCP server, it also allowed us to set the primary DNS server to the raspberry pi, which was needed so that pihole can filter the inbound traffic and act as an adblocker.

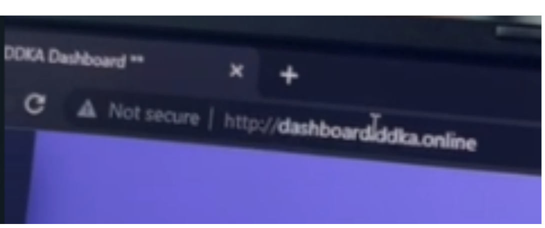
```python
def setup():
    global board

    board.set_pin_mode_dht(DHTPINREAD, sensor_type=11, differential=0.05)
    board.set_pin_mode_analog_input(LightLevel)
    board.set_pin_mode_analog_input(KNOB, callback=get_temp, differential=10)
    board.set_pin_mode_digital_output(REDLED)
    board.set_pin_mode_digital_output(GREENLED)


def get_data():
    return {"name": clientname,
            "room": roomnumber,
            "time": datetime.now().strftime("%I:%M%p"),
            "day": datetime.now().strftime("%A"),
            # "temperature": board.dht_read(DHTPINREAD)[1],
            "humidity": board.dht_read(DHTPINREAD)[0],
            "KNOB": round(board.analog_read(KNOB)[0] * MAX_TEMP / 1023.0, 2)
            }
```

- Extra Feature #1 Hue Lights: For the Phillips' Hue Light and bridge, the initial idea was to use REST API to make a controllable toggle of the lights via the web server. Unfortunately, due to time constraints, we were not able to successfully finish this feature. We did however, manage to use the API to make a smart light system to help save energy. We used an arduino light sensor to detect the light level. If it drops below a certain point, the lights turn on automatically and vice versa.

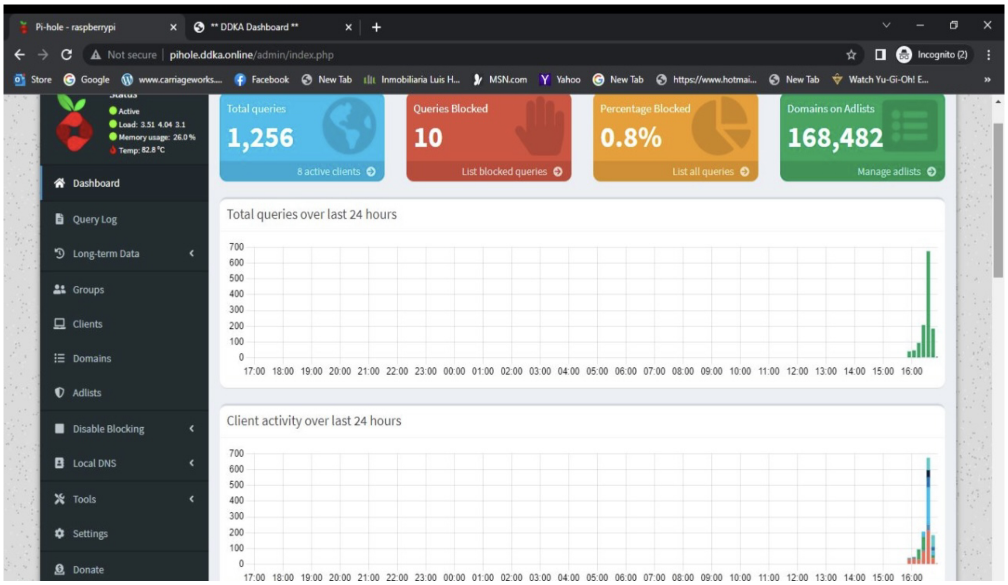


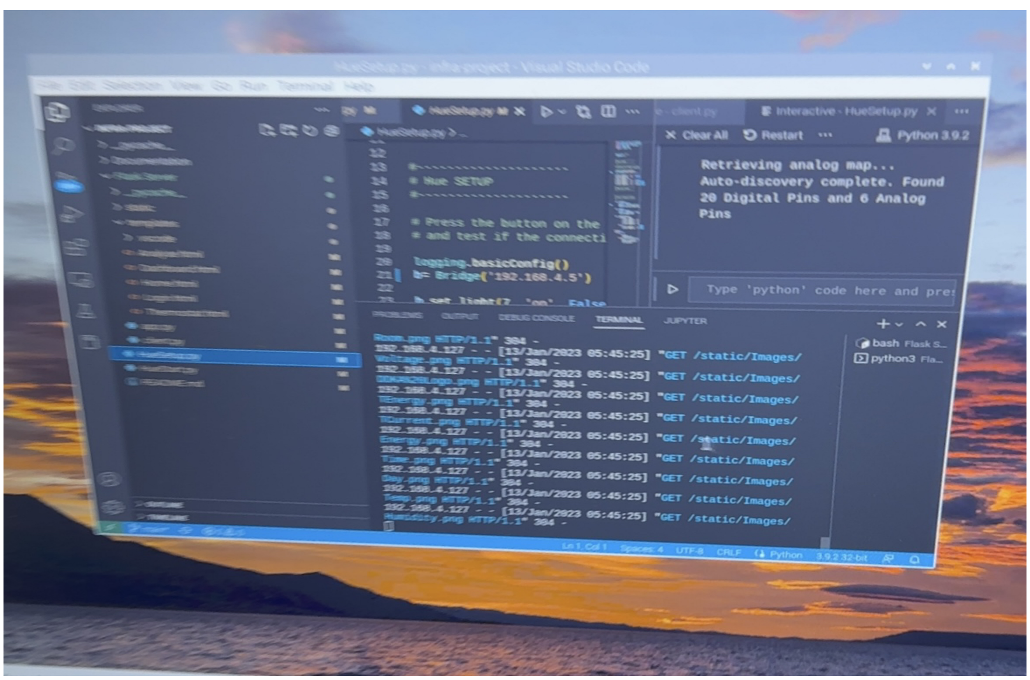
```python
logging.basicConfig()
b= Bridge('192.168.4.5')

b.set_light(7, 'on', False) # Turn off light 7
b.set_light(8, 'on', False) # Turn off light 8
time.sleep(1.5)
b.set_light(7, 'on', True)  # Turn on light 7
b.set_light(7, "xy", (0.5475,0.3113))
b.set_light(8, 'on', True)  # Turn on light 8
b.set_light(8, "xy", (0.5475,0.3113))

def setup():
    global board
    board = CustomPymata4(com_port = "COM3")
    board.set_pin_mode_analog_input(LDRPIN)

def loop():
    global value
    value, timestamp = board.analog_read(LDRPIN)
    time.sleep(0.01)

def lightcheck():
    if value < 300:
        b.set_light(7, 'on', True)  # Turn on light 7
        b.set_light(7, "xy", (0.5475,0.3113))
        b.set_light(8, 'on', True)  # Turn on light 8
        b.set_light(8, "xy", (0.5475,0.3113))
    else:
        b.set_light(7, 'on', False) # Turn off light 7
        b.set_light(8, 'on', False) # Turn off light 8

setup()
while True:
    try:
        loop()
        lightcheck()
    except KeyboardInterrupt: # crtl+C
        print ('shutdown')
        board.shutdown()
        sys.exit(0)
```

- DNS: The domains were set up locally using local DNS records. This was done by pihole due to the fact that pihole was the primary DNS server to filter traffic.

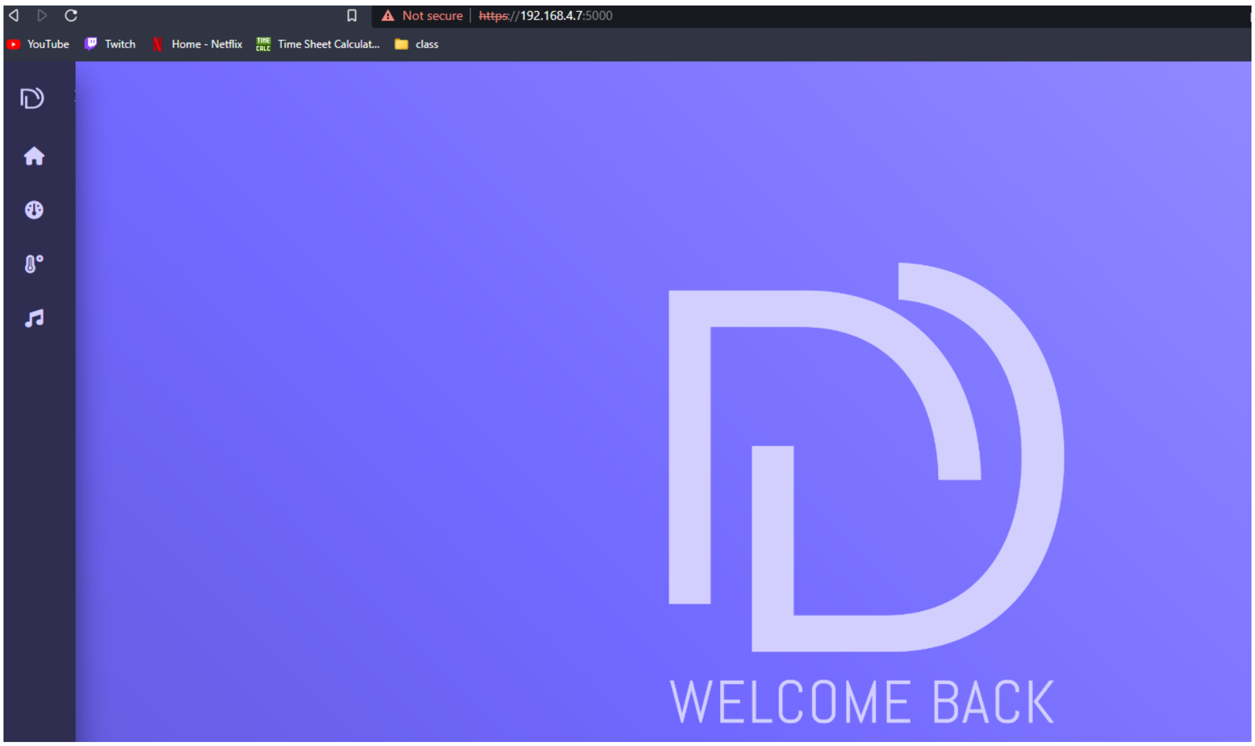


- DNS, Pihole & DHCP: The Raspberry Pi was set up and all the contents needed to run the flask server and hue lights were installed, including the libraries for the code. Additionally, Pihole was also installed and was successfully running as the DHCP server's primary DNS. From then on, Pihole and the flask server were successfully running on the Raspberry Pi.

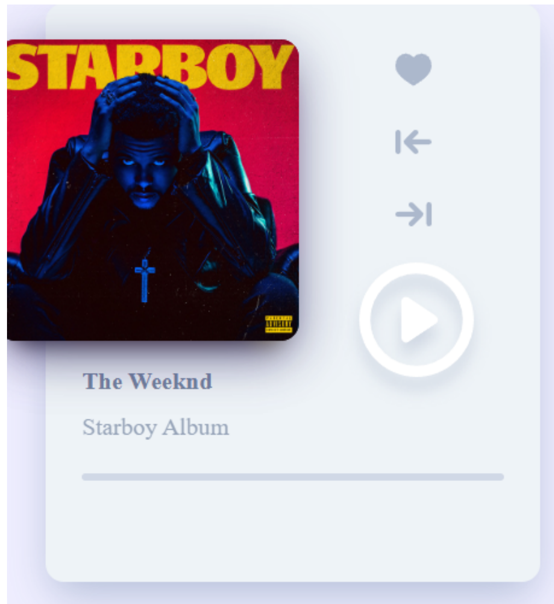




- Security (SSL): Although the web server was successfully running, the next step was figuring out how to secure the site. We decided not to use a client login page due to security purposes. We would rather not have a login page at all than one that is not secure. We decided to allow everyone on the same network to monitor their own energy consumption and fill out their name and room number on their own terms. From here, we used openssl to provide a self signed certificate but it would not be as secure as a paid and recognized ssl certificate. As provided in the screenshot below, it is seen that the web server is using https as a protocol but the browser still does not trust it due to it being a self signed certificate.

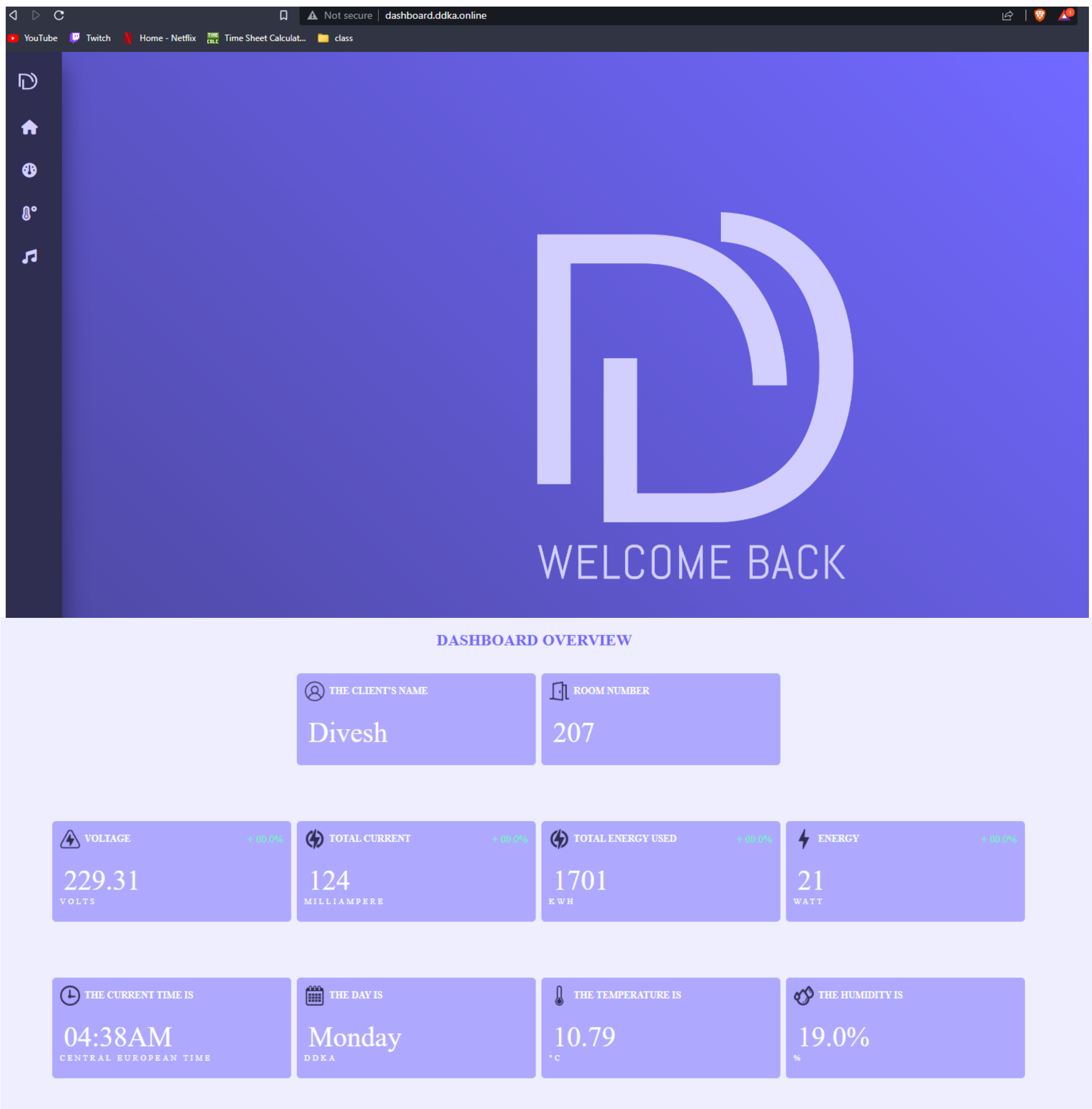


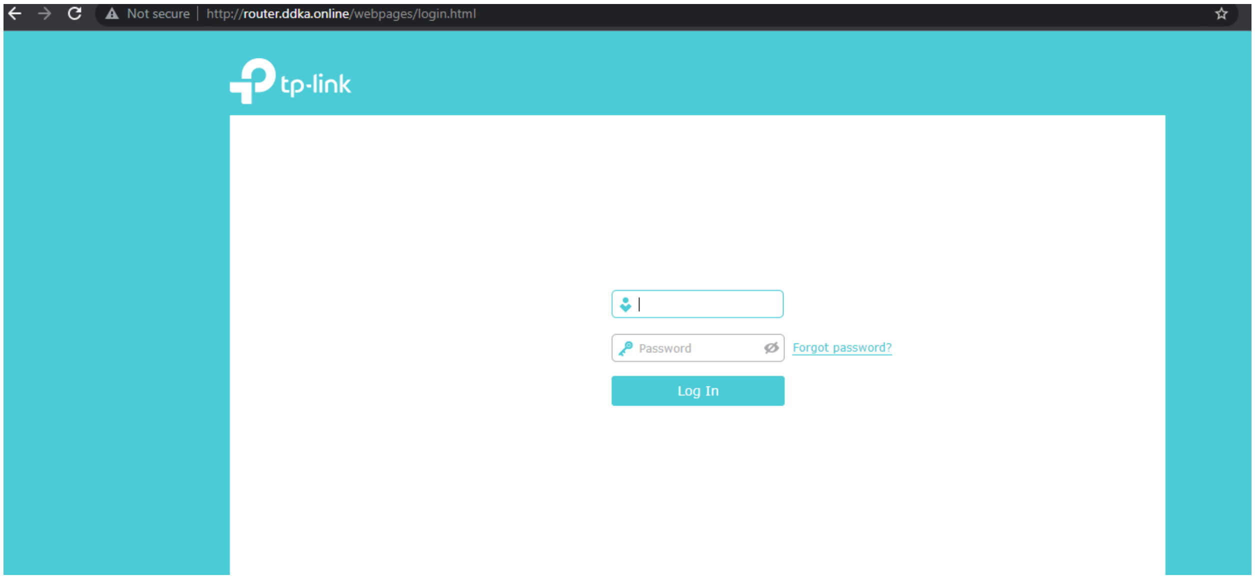
- Extra Feature #2 Thermostat: A thermostat page was included so clients can monitor and change their temperature. The arduino knob is used as a physical temperature change as you would have a thermostat knob in the room realistically.
- Extra Feature #3 Music Player: A music player was added to the dashboard just for a developmental miscellaneous product. The songs are downloaded and stored in the server.

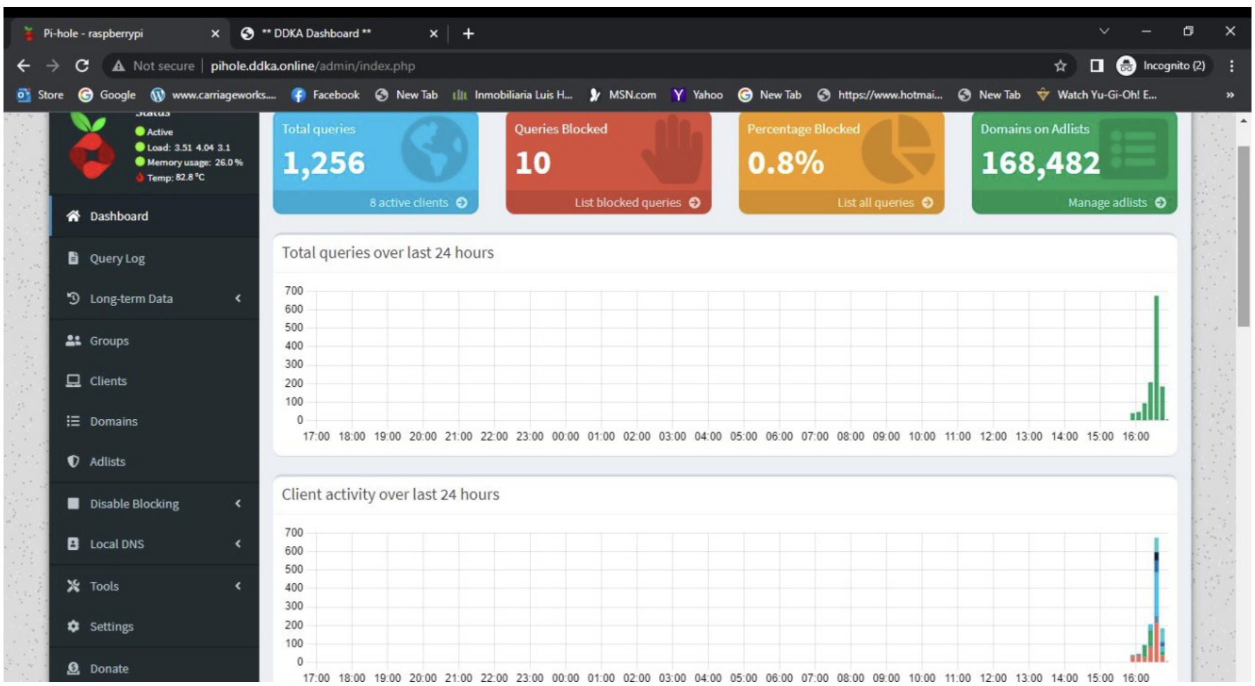


◆ Demonstrate:

- With all of these features and necessities fully functioning, it was seen that our web server was finally ready after an intense amount of hard work, dedication and time management. The DHCP server was correctly set up. The DNS server and records were correctly set up and pointed to the following domains, (*"router.ddka.online"* points to the default gateway, *"pihole.ddka.online"* points to Pihole, and *"dashboard.ddka.online"* points to the web server for the client dashboard). The arduino data and the NETIO data using REST API was also able to be displayed on the dashboard successfully. The following images below depict a fully functioning web server.

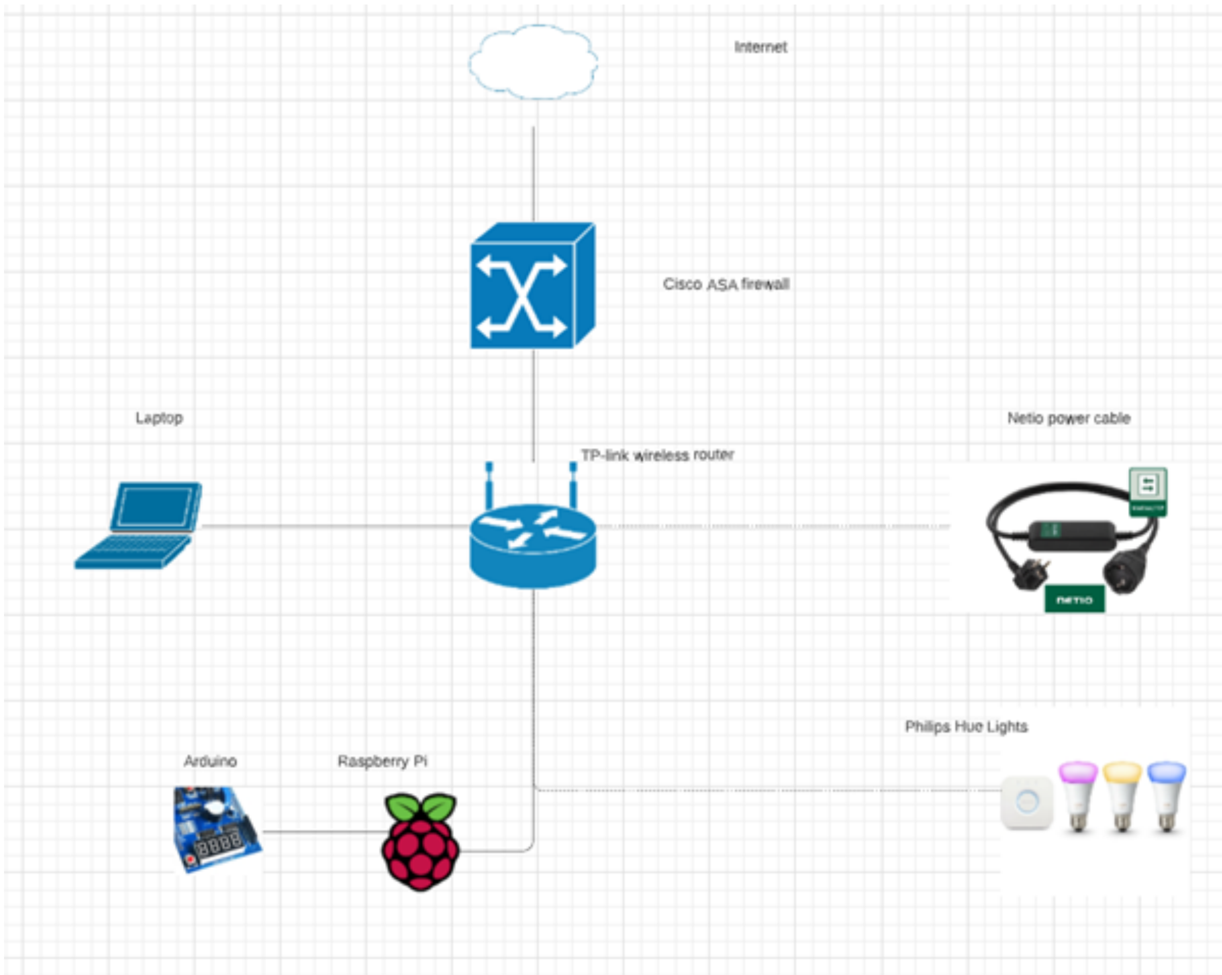






◆ The Server and the Network Environment:

- Before working on the web server, a network drawing was a requirement to be made for the project. Below is the image of the network drawing



- **Security:**
  - Assessing the web server using the CIA Triangle
    - Confidentiality:
      - Confidentiality does not score too high with the web server. A login page with a MySQL database was in the making however, due to the lack of time we had, we could not go through with this. We also only had a self signed ssl certificate which would not provide the best security possible. This would in turn, put our users at risk of data leaks. We opted to only allow people on the same network to connect to the web server. The only disadvantage is that the web server cannot be accessed off site.
    - Integrity:
      - There is no login page except for the default gateway to the router and Pihole. Pihole and the gateway use http instead of https and there was not much we could do because this was provided by a third party, however, these

are only accessible on the same LAN which makes it a little more secure. These passwords are not stored and therefore remain in their authentic state. All other data is taken as a live statistic from the NETIO cable and arduino and are not stored.

- Availability
  - Setting up this web server is a strenuous and time consuming task due to all the hardware it needs to function and run. You cannot run the web server if you are missing one component such as the NETIO cable or arduino because all the components interact and communicate with each other in some way, shape or form. This means that if one piece of hardware fails, the service will not run until it is repaired.This could lead to long lasting downtimes.

- Conclusions & Solutions to the assessment
  - Confidentiality:
    - The way we approached this was the best possible way when looking at the assessment. It limits access to the web server to clients on the same networks which makes the system a little more secure. You can distribute further access using static NATs should you choose to do so. Port forwarding is not recommended due to the lack of a login page.
  - Integrity
    - Since client data is not stored and is taken as a live statistic, this allows the data to remain in its authentic state and doesn't put the clients at risk. The main web server was managed to be put through a https protocol, however, only a self signed certificate could be used.
  - Availability:
    - In order to make the system as advanced as we could, all the data needed to interact with each other. The hardware failure is the only disadvantage to the system that we have set in place. Aside from that possibility, the long process to start up the web server is a hassle but can be fixed with a simple script to run all the commands if they are prewritten in the code.

Project Implementations

- ➔ My Role
  - ◆ My main role was a web server developer for the group. This meant making the foundation of the Flask Server and maintaining the web server to add more data variables, however, I helped out and did as much as I could.
- ➔ Contributions
  - ◆ In this project, I worked on the web server, DNS, security (in particular the ssl certificates), I made the smart hue light system using Phillips' API and I helped to configure the DHCP server to make sure the DNS server functioned correctly.