IoT Worksheet 3: Web Server

By Nishaal Naseer

Student ID: 22023475

Contents

1. Introduction

The purpose of this assessment is to design and create a program that entails the essence of Internet of Things. Some options were given and I have chosen to create a web server that stores and visualizes data from an IoT device.

Touch, temperature and light from a Microbit are stored and visualized in this project.

2. Requirements

This project can be viewed as a framework to deploy an IoT related project and therefore the devices, modules and code that can be used are not strict. However, if you would have to change the code and devices accordingly.

The simplest form of this would be a database and a python FastAPI server, on the same computer.

Regardless, in this documentation I am going to explain my setup and how you can do it too. My set up below:

1. Computer (To run webserver and database server)
2. Raspberry Pi 4B (To collect data from Microbit and send through an HTTP request)
3. Microbit device (To collect data)
4. Last but not least the Pi and the computer should be on the same network and be capable of talking to each other.

3. Code Dependencies

3.a Computer

I have used MariaDB 10.11.2 with HeidiSQL 11.3. The SQL code to create database can be seen at server/init/init.sql

I used Python 3.11.4 for the server, and I think you should too, because I tried running in Python 3.9 and some of the type annotations were not supported. The dependencies for the server are listed in the server/requirements.txt

3.b Raspberry Pi

pi_stuff/requirements.txt has all the modules needed for this part of the project. Honestly, if not for Pydantic I think the code could run out of the box without any pip installs. Because the default raspberry pi is custom tailored to IoT related projects.

You can use python 3.9 for the raspberry pi.

4. Reasoning on Choices of Technology

4.a MariaDB

I have used MariaDB because I have grown very familiar with it over the past year, but its also very easy to use and set up. But other than that there is no absolute reason to use it, if you wish you can go ahead and change the DBMS, but you'll have to find a corresponding asynchronous driver for it.

4.b FastAPI

After careful consideration, I have opted to employ FastAPI as the foundation for my web server. FastAPI stands out as a robust, asynchronous, and production-ready solution, boasting seamless development and a smooth transition to the production environment. Leveraging the power of Pydantic, FastAPI facilitates seamless integration of third-party type checking, enabling the transformation of Python object models into JSON. JSON, renowned for its unambiguous nature, inherently showcases its schema within the transmitted data, adding further clarity to the process. By adopting FastAPI, I am are ensuring a professional, high-performance, and efficient web server, primed to meet the demands of our development and production endeavors.

FastAPI documentation (http://server_url:port/docs) provides an interface to inspect debug and showcase data.

4.c Raspberry Pi

I have my friends Raspberry Pi; I can use it to make my IoT project more interesting and I will not hesitate to do so.

The devices are connected through USB, so I do not see any restrictions in this aspect, even if you decide to use something else, as long as the devices, ports and cable are working perfectly fine.

5. **Installation**

5.a Microbit

You can connect the Microbit to your computer and copy the hex file to it, if you wish to change the code you can change, test and make a new hex file from https://python.microbit.org/ , the site also has valuable documentation for the device.

5.b Raspberry Pi

On Linux you can create a folder called iot-worksheet-3 and cd there. Create another dir called pi_stuff and cd there.

Copy all the files in pi_stuff.

Edit the contents of config.json, make sure the value for server has the correct protocol, ip, port and doesn't end with a forward slash. Follow the contents in Troubleshooting Raspberry Pi to get the name of the port of Microbit. Once you have it adjust the contents of config.json accordingly.

Run install.sh . Don't forget to give the file execution rights.

What the script does is the following

1. The script checks if it has sudo (root) permissions.

2. It creates a Python virtual environment/

3. Installing dependencies.

4. Creates a run.sh file and gives it execution rights

5. Creates a service called "worksheet-3-pi_stuff.service", reloads the daemon and starts the service.

6. Shows status of service.

If you wish you can modify or change it entirely, I believe this structure gives has a lot of flexibility. This script also assumes your username is ‘pi’.

However, those instructions are for Debian based systems only. (I think, I don’t know how systemd works on other linux distributions).

For other operating systems you can follow the structure of the install.sh file. Although the specific steps maybe different I believe in all operation systems our goal is to do the same, create a program that will boot with OS and will not be interrupted easily.

5.c MariaDB

You can download MariaDB and it’s connectors from https://mariadb.org/

For Linux you can follow these instructions https://www.digitalocean.com/community/tutorials/how-to-install-mariadb-on-ubuntu-20-04

Afterwards, it is not compulsory to create the database and table.

5.d FastAPI

Copy the contents to a dir $HOME/Desktop/iot-worksheet-3/server

Contents in src/config.json are the credentials and configurations required to connect to the database server, adjust them accordingly.

Run install.sh after chmod +x install.sh . Please note script assumes your username is pi

The script does the same things as the raspberry pi script with an extra step.

After dependencies are installed to the venv it attempts to establish a connection to a MariaDB server as entailed by the src/config.json script. Once connection succeeds a Database called worksheet-3 is attempted to be created, and in it a table called periodic_report. The schema for this is also available in init/init.sql .

The service created in this case is “worksheet-3-server.service”

Again, this is only a framework (sort of), it can be adjusted depending on your set up.

6. <u>Troubleshooting</u>

6.a FastAPI

View documentation and test endpoints at f”{protocol}://{server}/docs”

Please get access to the server terminal, because that’s where all requests and errors are logged. You can view some of it with, sudo systemctl status worksheet-3-server.service . Else just stop the service and run the run.sh file directly.
If you can’t connect to the server, check your’ firewall, the server terminal will show if you are receiving any requests and its codes.

If you are getting server 500, view the traceback from server terminal and handle it accordingly.

6.b Raspberry Pi

It should be noted the Microbit device needs to be disconnected and connected on reboot of the Pi.

Confirm it is working by checking the process terminal, you can follow the same advice as that for the server.

Confirm the port is correct, you can note the devices available with python3 -m serial.tools.miniterm

Disconnect the device and run python3 -m serial.tools.miniterm again.

If there is no difference in the list of available devices then there is a connection issue, else copy the name of the port that disappeared and copy it to config.json “port” value.

To make sure if data from the device is being received run python3 -m serial.tools.miniterm /dev/ttyACM0 115200

Note:

1. The first argument is python path variable
2. It is required by python
3. Python serial module
4. Port
5. Baud rate

You can change the port name if it is different as per the above step, if you do receive data, but its unintelligible, there might be an issue with the baud rate and or the character set of the encoded data. I believe by default it is ASCII.

7. **Running**

If you have installed with the script files then most likely they are already running. Or you can run sudo systemctl restart service_name for both services just to make sure they are running.

You can open HeidiSQL and view the data in the table and see if it is being populated.

To get a graph of the data you can follow the guideline here.

The path is ‘/report’

The query parameters are:

1. report_type
    a. ‘PINS’
    b. ‘TEMPS’
    c. ‘LIGHT’
2. start, format YYYY-MM-DD
3. end, format YYYY-MM-DD (optional)
4. HTTP Method: GET
5. Set header like: -H 'accept: text/html'

Examples, they are in a python formatting since I find them rather easy to understand.

Without end date: f"{protocol}://{server}{path}?report_type={type}&start={start_date}"

With end date:

f"{protocol}://{server}{path}?report_type={type}&start={start_date}&end={end_date}"

8. Short Comings and Further Improvements

The data I collect from the Microbit device are: accelerometer, temperature, light level, touch. However, I can only use the temperature to create 2D graphs as the light and touch sensors seem to be broken. (Light level is always 0 and touch sensors always False). If I receive a proper device I could create proper graphs in all these aspects.

Further improvements in these can include, but not limited to, require moving the sensors outdoors, and designing neural networks to attempt to predict the weather.

9. TODO Before Production

I think weather proofing the devices the sensors and Pi are required, even if not outdoors soot can build up on the circuit board of the devices and act as a conductor before frying the devices.

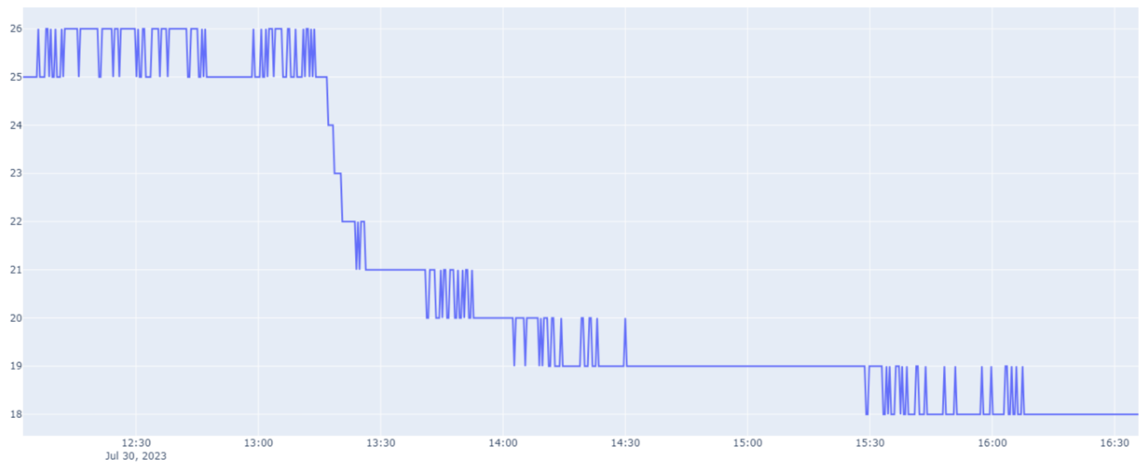
SSL verification and user authentication. This would make sure only verified users get to post and get data while the data that is transmitted is secure. The instructions on how to do so can be found at: https://fastapi.tiangolo.com/

10. Conclusion

In this project, I have developed a web server that can collect data from an IoT device and generate graphs for it. The web server is implemented using the Python programming language, and it uses the Plotly library to generate graphs. The web server is able to collect data from a variety of IoT devices, and it can generate graphs for a variety of data types.

The web server is a valuable tool for visualizing and analyzing IoT data. It can be used to track trends, identify patterns, and troubleshoot problems. The web server is also easy to use, and it can be accessed from any web browser.

Below is a PNG of the graph generated. You can see the moment I cranked up the AC.


12:30
Jul 30, 2023
13:00
13:30
14:00
14:30
15:00
15:30
16:00
16:30