IoT Weather Machine

Jordan Gribben

CMP408: System Internals & Cybersecurity

Introduction

The purpose of this mini project is to build a full-stack system that will display weather data on the Raspberry Pi Zero. This will be achieved by a city being entered that will then be transferred to the Pi via MQTT , once the Pi has received this communication it will receive weather data for that city using an online weather source, this data will then be displayed on the Pi. This will be achieved by completeing the following objectives:

•Create a webpage using an amazon web services (AWS) EC2 instance that allows users to input a city.

•Publish this data using an MQTT broker.

•Retrieve this data using the Raspberry Pi, get the weather data for the city retrieved and display it on the Pi.

•Use LEDs to show the current state of the Pi.

•Implement security features

This project also contains various security features, these include:

•RSA key.

•Usernames & passwords.

•Input sanitisation.

These features improve the overall security of the project. The RSA key ensures that only authorised users can SSH into the instance. Usernames and passwords ensure that only those with this information can publish and subscribe using the MQTT broker. The input sanitisation helps protect against malicious user inputs.

Methodology

This project's purpose is to be an IoT weather machine by, taking in a city and displaying that city's weather data. Once a user has submitted a city from a webpage the weather data for that city will then be displayed on the Raspberry Pi.

The project can be split into two key sections these being the Raspberry Pi setup/hardware, and the AWS EC2 instance which acts as both the webpage and the MQTT broker. Figure A contains a diagram of how to set up the raspberry Pi and its peripherals.

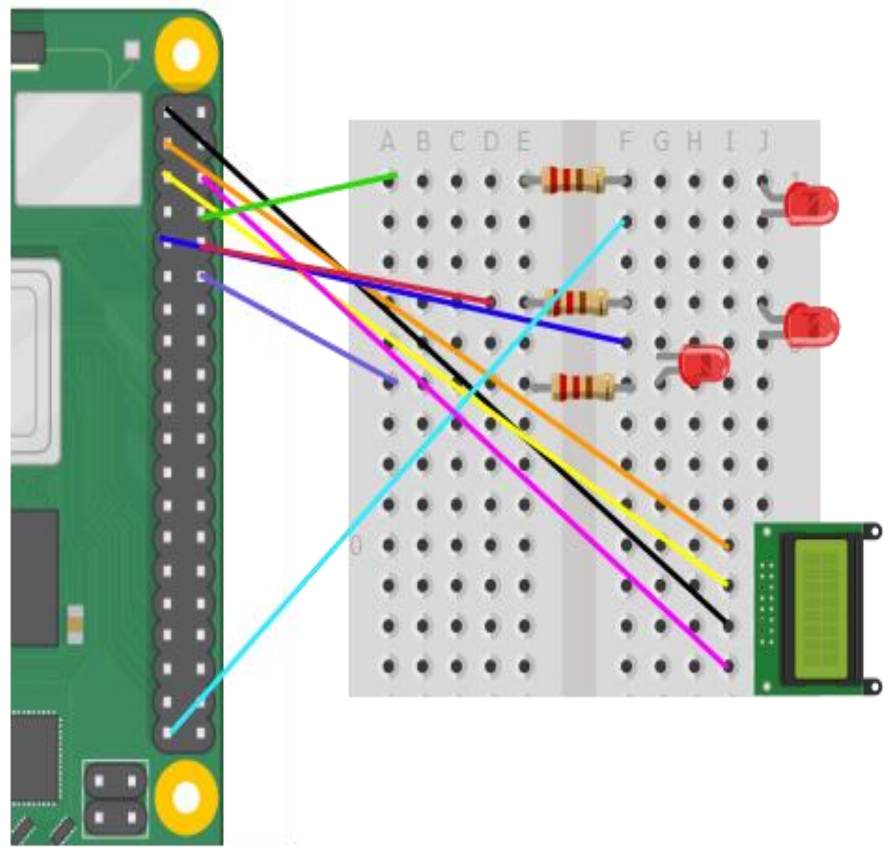


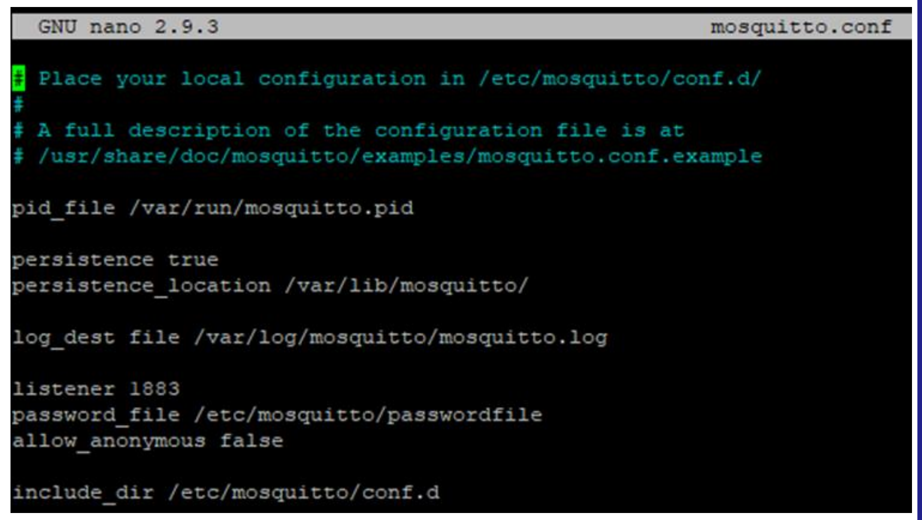
Figure A – Pi Physical Setup

Once the Pi was wired like in figure A a python script was then created. The main purpose of this script is to use OpenWeather to retrieve the weather data based of a city, the city name will be entered using a webpage on the AWS EC2 instance. The Pi will receive this city using MQTT, the Pi will subscribe to a set topic allowing it to receive any data published to that topic.

The python script can be broken down into the following parts:

- Connect to MQTT broker
- Subscribe to MQTT topic
- Get weather data
- Display weather data
- Have LED's light up based on current state of the Pi

For AWS a ubuntu EC2 instance was created, this instance is used for both the webpage and the MQTT broker. The webpage is run using flask while Mosquitto is used as the MQTT broker. A new security group should also be set up for this instance allowing ports, 22, 80, and 1883, for SSH, HTTP and MQTT respectively. An RSA key is created for the instance, this is to ensure that only users with this key can SSH into the instance, helping to keep it secure. Two more security features are added to the instance before the python script is created, these are adding username/passwords to MQTT and having input sanitation for the user input field. A password file was created for Mosquitto, this file contains the username and password of each authorized user, once created the "mosquitto.conf" file must be changed. This amended file can be seen in figure B (Eclipse, 2021). When creating the input field for the webpage it was important to ensure that the data being entered is safe for both the Pi and the webpage, to prevent against attacks such as cross site scripting.

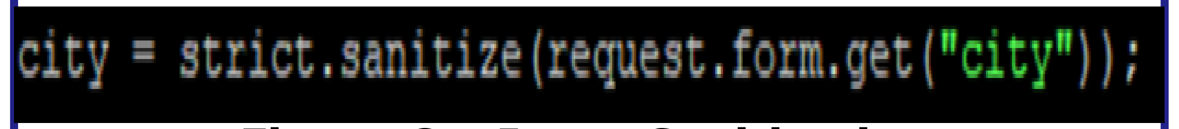


```
  GNU nano 2.9.3                                        mosquitto.conf

# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

listener 1883
password_file /etc/mosquitto/passwordfile
allow_anonymous false

include_dir /etc/mosquitto/conf.d
```

Figure B – mosquito.conf

Project Highlights

While this project has features a specific highlight of this project is the input sanitisation. This feature allows for any inputted tags to be removed, making the user input safe for both the webpage and the Pi. This security feature helps protect against potential cross site scripting attacks. This sanitisation is done using the "pybluemonday" library, figure C below shows the line of code that sanitises the input. (DevOps Tools, 2021)



```
city = strict.sanitize(request.form.get("city"));
```

Figure C – Input Sanitisation

Future Work

- Linux Kernel module (LKM) – A LKM could be added to control the LED's connected to the Pi.
- Security Features – Additional security features could be added to make the project more secure such as running on HTTPS over HTTP.

References

DevOps Tools, 2021. *A library for sanitizing HTML very quickly via bluemonday.* [Online] Available at: https://pythonawesome.com/a-library-for-sanitizing-html-very-quickly-via-bluemonday/
[Accessed 10 January 2022].

Eclipse, 2021. *mosquitto.conf man page.* [Online] Available at: https://mosquitto.org/man/mosquitto-conf-5.html
[Accessed 8 January 2022].