IS4151 Pervasive Technology Solutions and Development

Individual Assignment 2

**Protocol Document**

Prepared by:

Lim Jian Li (A0168138X)

[**1. Source Files Structure**](#_gbpkxojd6xhy) **3**

[1.1 Folder Structure](#_qwk9ub8k6atf) 3

[1.2 source Folder](#_7bwnkphaql9h) 4

[1.2.1 cloudserver Folder](#_4qzmuo2kp2uf) 4

[1.2.2 district1 & district2 Folders](#_olmlizmc4wu6) 7

[**2. Protocol & Design**](#_9752kqutegr9) **9**

[2.1 Cloudserver](#_s97szdaxbhr3) 9

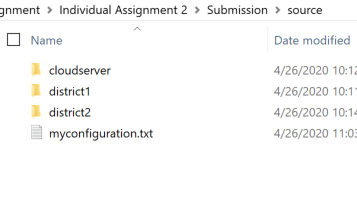
[2.2 Edge Processor (Raspberry Pi)](#_xt988lkv5t5n) 11

[2.3 Radio Controller and Tracker Timeline](#_2ouek05iswxb) 13

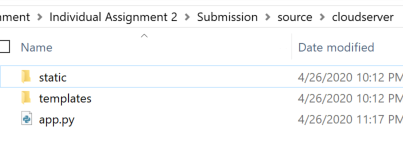
[2.4 Database](#_q96nqhrrigam) 13

# 1. Source Files Structure

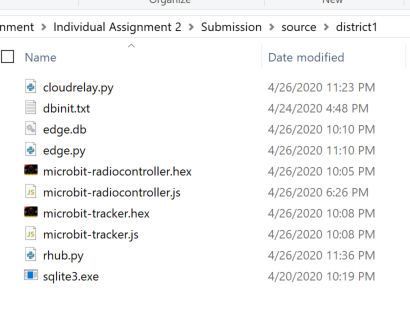
## 1.1 Folder Structure



The **source** folder contains: 3 folders and 1 **.txt** file describing my setup.

****

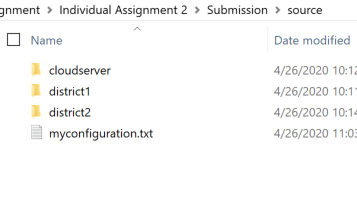
**cloudserver**: This contains the cloud server Flask implementation.



**district1**: This contains the files pre-configured as District 1.

**district2:** This folder has the same files as District 1. Some values are changed to represent District 2.

## 1.2 source Folder



This folder contains **myconfiguration.txt** which lists the configuration of my IP addresses and ports used:

IP Addresses:

cloudserver: 192.168.1.54

district1edge: 192.168.1.23

district2edge: 192.168.1.178

Ports:

district1radiocontroller: /dev/ttyACM0

district2radiocontroller: /dev/ttyACM0

### 1.2.1 cloudserver Folder

## 

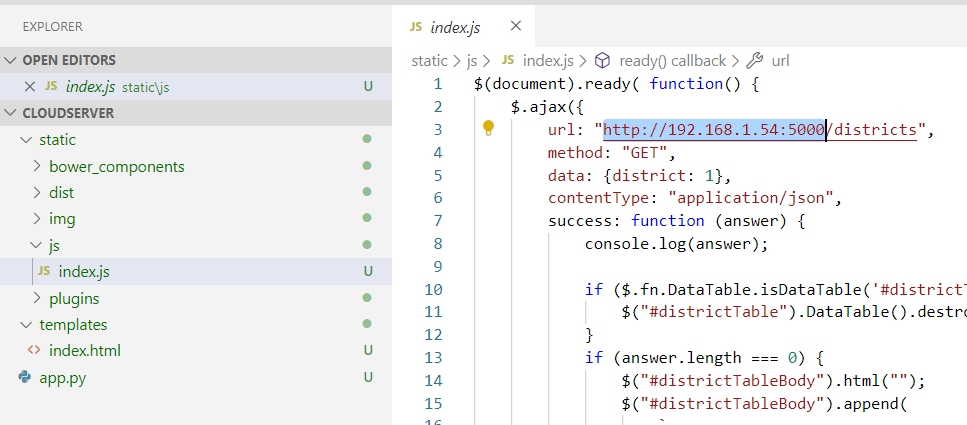
## 

**app.py**: Contains the Flask server for the cloud server. IP addresses here must be changed to the IP address of the Raspberry Pi.

## 

There are a total of 6 IP addresses that require a change.

**cloudserver/static/js/index.js**: This file uses AJAX calls to the server to perform tasks and display the information on the cloud webpage. IP addresses here must be changed to the IP address of the Raspberry Pi.

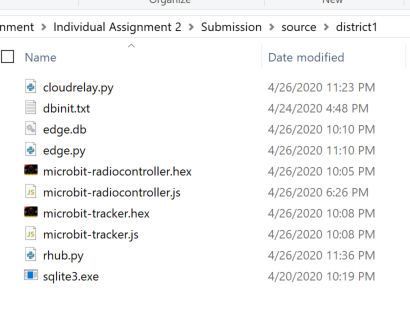


There are **8** occurrences of this IP address.

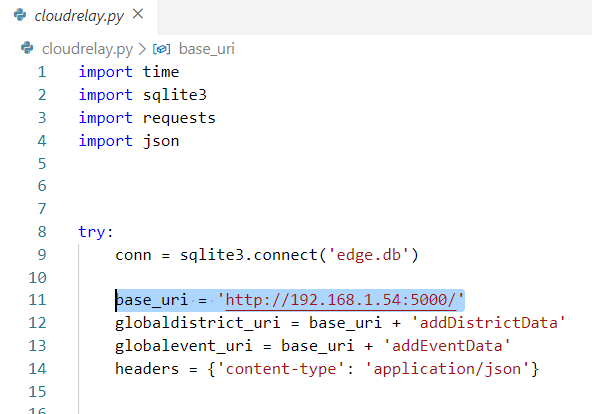
This IP address should be the IP address of the cloud server.

## 

### 1.2.2 district1 & district2 Folders



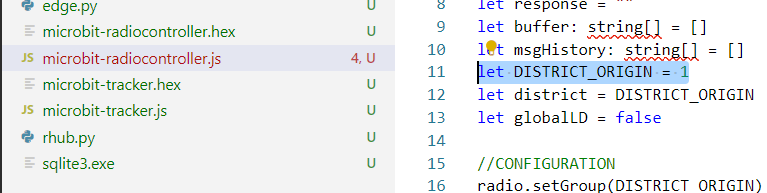
**cloudrelay.py:** This sends data to the cloud server. Change IP address in **base\_uri** to cloud server IP.



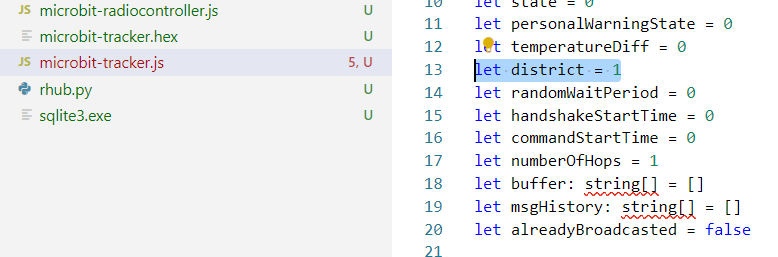
**dbinit.txt**: This contains information about the 2 tables used in the system: `trackers` and `events`. There is no need to use this document as the tables have been initialised in the **edge.db** file already.

**edge.db**: This contains the 2 SQLite tables (`trackers` and `events`).

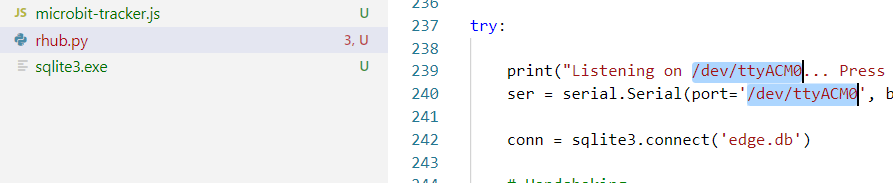
**edge.py**: This is the Python Flask script to be run on the Raspberry Pi edge processor for the local district webpage. It is the same file for both **district1** and **district2**.



**microbit-radiocontroller.hex & microbit-radiocontroller.js**: These are the microbit files for the radio controller. They are preconfigured to the district number.



**microbit-tracker.hex & microbit-tracker.js**: These are for the COVID-19 trackers. They are preconfigured with the district number.

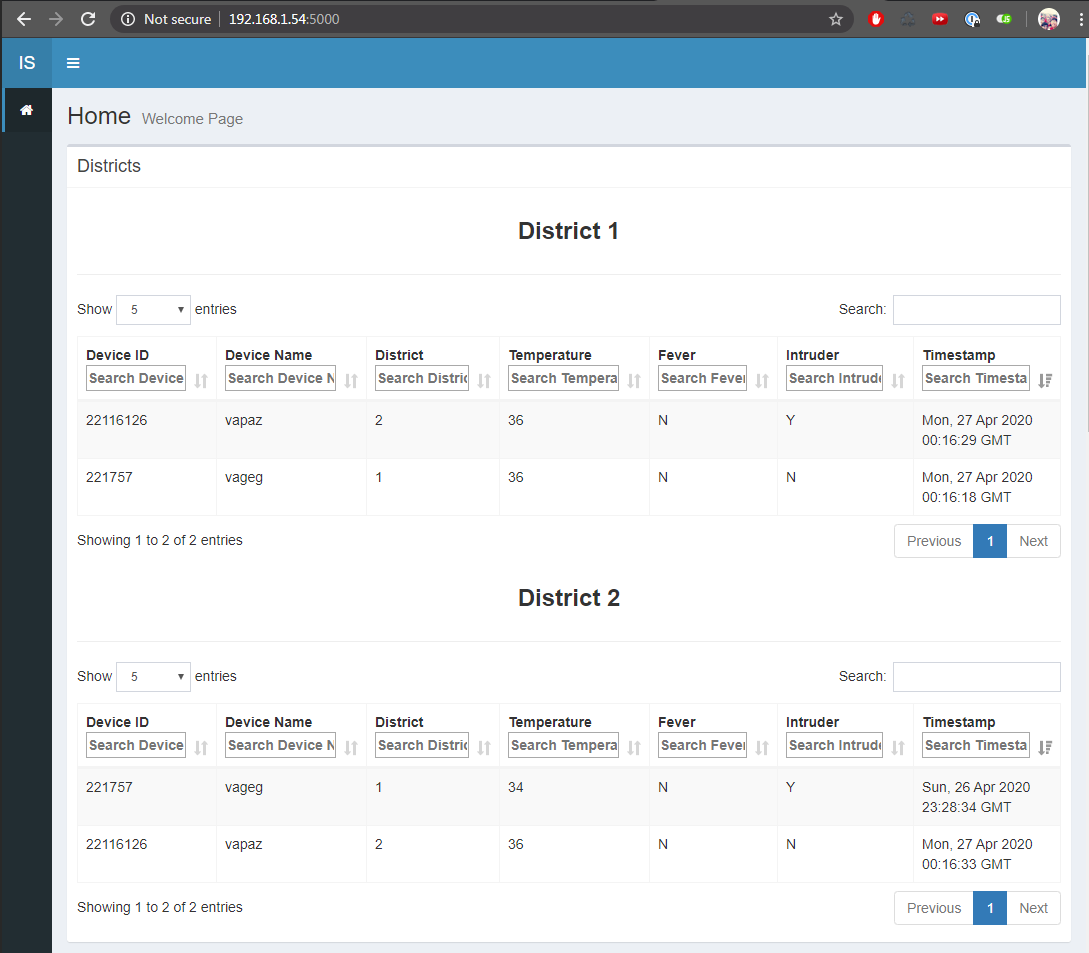


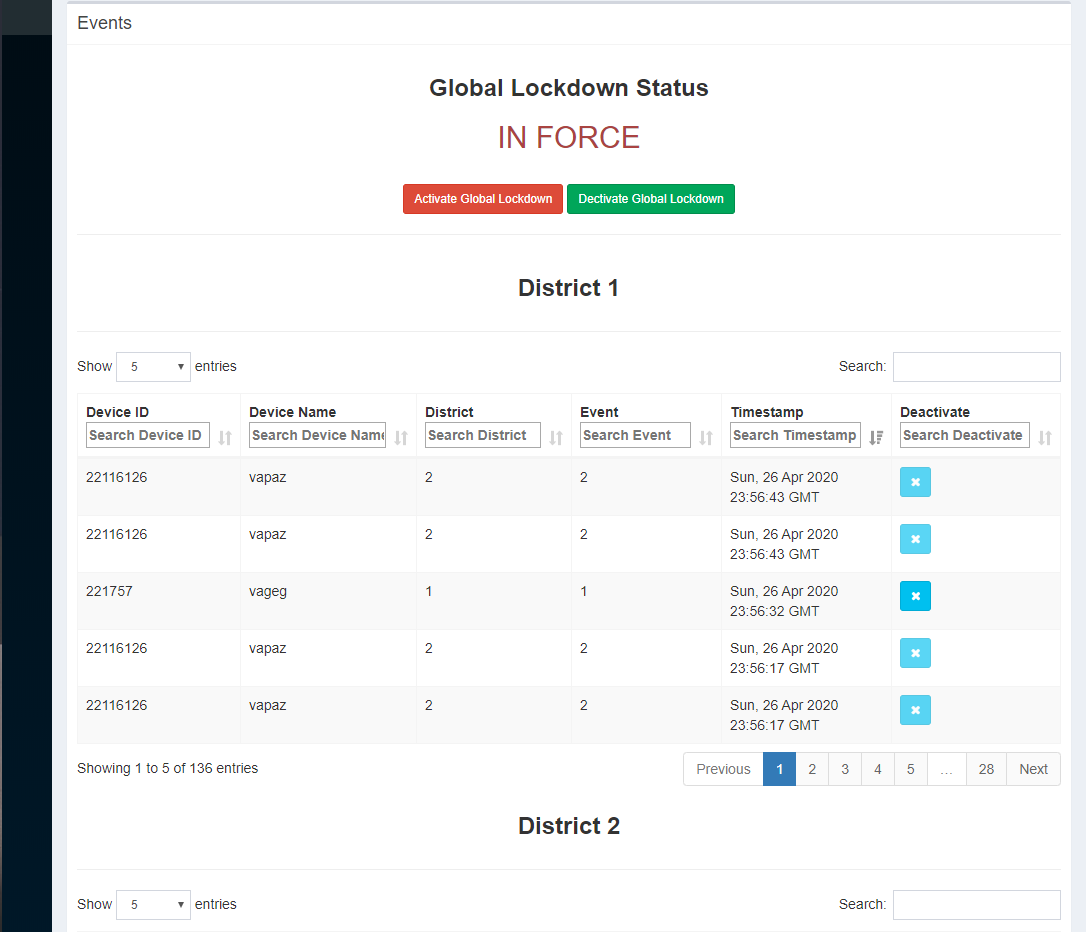
**rhub.py**: This file is to be run on the edge processor Raspberry Pi. The preconfigured listening port is **/dev/ttyACM0**.

# 2. Protocol & Design

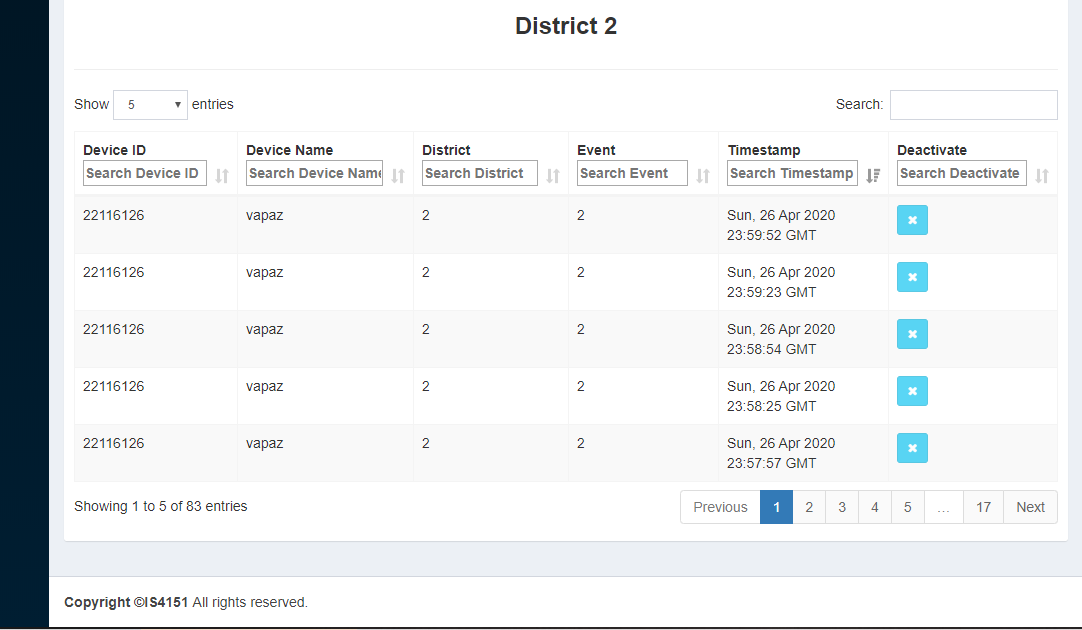
## 2.1 Cloudserver

The cloudserver is a Flask app that runs a single html page. The html page imports a few Javascript plugins and also a script (**cloudserver/static/js/index.js**) that makes AJAX calls to the server API.



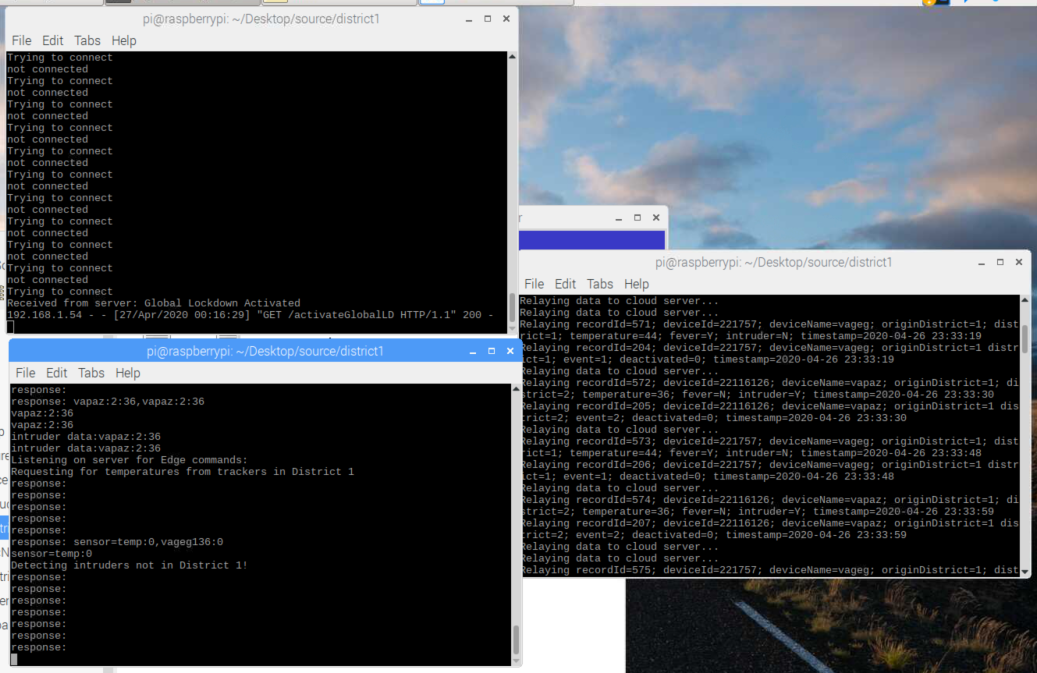


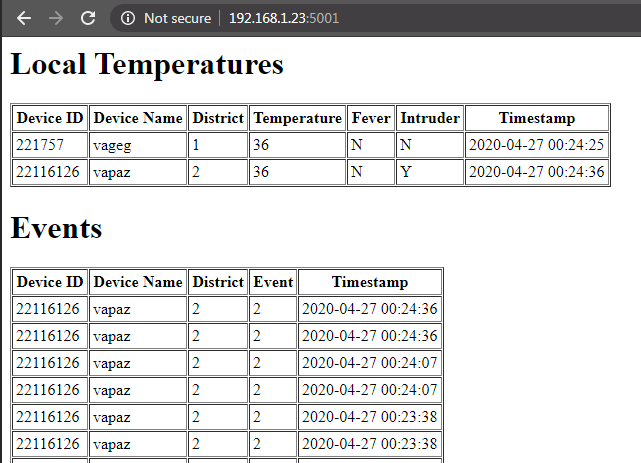
Note: Only the **Deactivate** button for Type 1 events are active. The rest are disabled.



## 2.2 Edge Processor (Raspberry Pi)

The Raspberry Pi runs **3** programs: **edge.py**, **rhub.py**, **cloudrelay.py**.





**edge.py**: This is the Flask server that displays the local temperatures and events for the district. It also hosts 3 API calls (deactivate local lockdown, activate global lockdown and deactivate global lockdown).

These rely on sockets to send data to **rhub.py**.

**rhub.py**: This is the Python program that communicates with the radio controller via serial. It also writes data to the local SQLite database. The **rhub.py** program uses a while loop with time.sleep(x) to send commands to the radio controller.

The **rhub.py** loop:

1. Send handshake
2. Wait for response loop (0.1s interval)
3. FIRST CYCLE: Wait 5s and send temperature request command  
   OTHER CYCLES: Wait 10s and send temperature request command
4. Wait for response:  
   5a. If no response is received for 25 loops, restart from step 3.  
   5b. If response is received, parse data and insert into array.

## 2.3 Radio Controller and Tracker Timeline

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time (s)** | | **Edge Processor** | **Radio Controller** | **Tracker** |
| 0s |  | Send out temperature command to **Radio Controller** | Receives temperature command by serial from **Edge Processor**  Forward to **Tracker** | Receives temperature command from **Radio Controller** by radio  Send out temperature by radio  Forwards command to other **Trackers**  Forward received command from other **Trackers** |
| 5s | Waits for response from **Radio Controller**:  A. Response received  B. 25 cycles passed | Serial write line of temperature array to **Edge Processor** | X Clear current memory |
|  | Writes data received to database  Send district switching (radio group switching) command to **Radio Controller** | (Intruder detection)  Receives radio group switching command from **Edge Processor**  Send out temperature command to **Trackers** | Receives temperature command from **Radio Controller**  Send out temperature by radio  Forwards temperature to other **Trackers** |
|
| 10s | Write data received by serial from **Radio Controller** to SQLite database | Serial write line of intruder information to **Edge Processor** | X (5s) Clear current memory |
|
|  |

### 2.4 Database

### 

The MySQL database and the SQLite database are the same. There are additional parameters not shown in the website table (**originDistrict** for both tables, **deactivated** for events table).