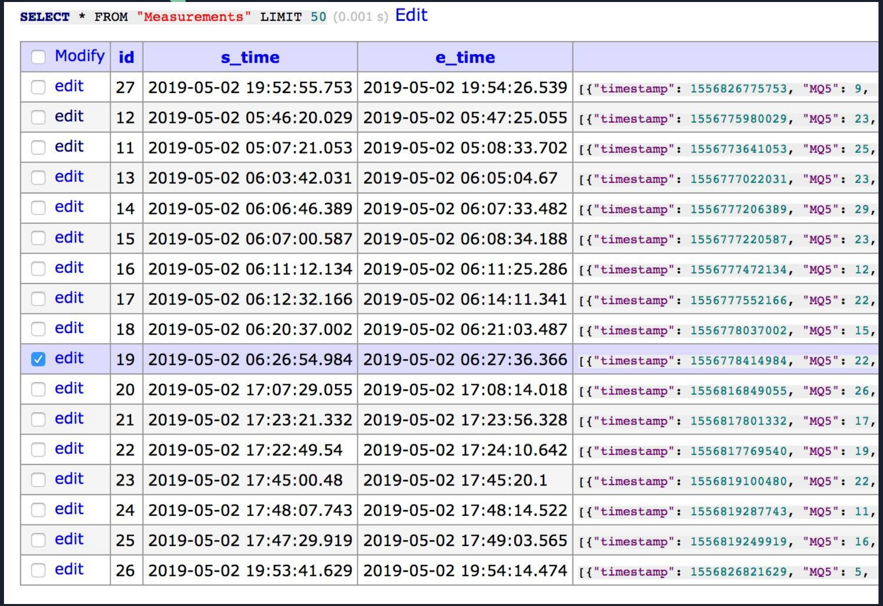
Air Quality Measurement Device

A Python and Raspberry Pi Implementation

Wesley Romary, Brandon Percin

- Problem
  - Analyzing air quality in a particular location is hard work, as one must keep track of temperature, dust particle count, and gas concentrations at many times during the day to understand how ‘good’ the air is.
- Solution
  - Use a Raspberry Pi along with four sensors (temperature, dust concentration, gas concentration, and relative air quality) to record data continuously
  - Send data to a PostgreSQL database on a Flask server
  - Query the server for data observed between two timestamps
  - Retrieve data from server on another device with a display
  - Display and analyze data in a user-friendly and graphical manner
    - Both sensor and display polling times are fully customizable

Backend Data Store



```
SELECT * FROM "Measurements" LIMIT 50 (0.001 s) Edit
```

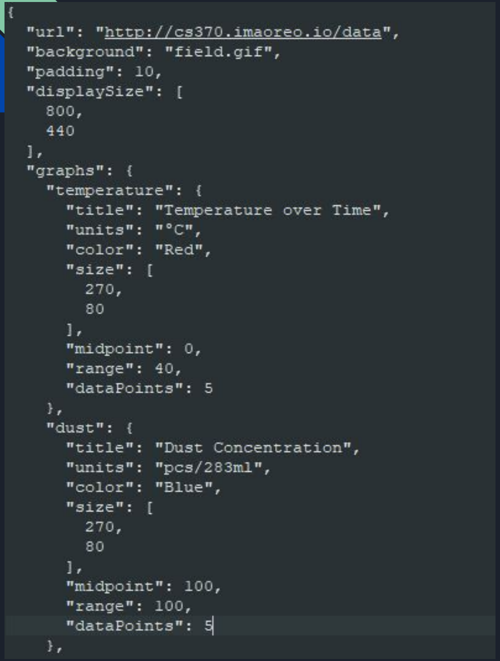
☐ Modify	id	s_time	e_time	
☐ edit	27	2019-05-02 19:52:55.753	2019-05-02 19:54:26.539	[{"timestamp": 1556826775753, "MQ5": 9,
☐ edit	12	2019-05-02 05:46:20.029	2019-05-02 05:47:25.055	[{"timestamp": 1556775980029, "MQ5": 23,
☐ edit	11	2019-05-02 05:07:21.053	2019-05-02 05:08:33.702	[{"timestamp": 1556773641053, "MQ5": 25,
☐ edit	13	2019-05-02 06:03:42.031	2019-05-02 06:05:04.67	[{"timestamp": 1556777022031, "MQ5": 23,
☐ edit	14	2019-05-02 06:06:46.389	2019-05-02 06:07:33.482	[{"timestamp": 1556777206389, "MQ5": 29,
☐ edit	15	2019-05-02 06:07:00.587	2019-05-02 06:08:34.188	[{"timestamp": 1556777220587, "MQ5": 23,
☐ edit	16	2019-05-02 06:11:12.134	2019-05-02 06:11:25.286	[{"timestamp": 1556777472134, "MQ5": 12,
☐ edit	17	2019-05-02 06:12:32.166	2019-05-02 06:14:11.341	[{"timestamp": 1556777552166, "MQ5": 22,
☐ edit	18	2019-05-02 06:20:37.002	2019-05-02 06:21:03.487	[{"timestamp": 1556778037002, "MQ5": 15,
☑ edit	19	2019-05-02 06:26:54.984	2019-05-02 06:27:36.366	[{"timestamp": 1556778414984, "MQ5": 22,
☐ edit	20	2019-05-02 17:07:29.055	2019-05-02 17:08:14.018	[{"timestamp": 1556816849055, "MQ5": 26,
☐ edit	21	2019-05-02 17:23:21.332	2019-05-02 17:23:56.328	[{"timestamp": 1556817801332, "MQ5": 17,
☐ edit	22	2019-05-02 17:22:49.54	2019-05-02 17:24:10.642	[{"timestamp": 1556817769540, "MQ5": 19,
☐ edit	23	2019-05-02 17:45:00.48	2019-05-02 17:45:20.1	[{"timestamp": 1556819100480, "MQ5": 22,
☐ edit	24	2019-05-02 17:48:07.743	2019-05-02 17:48:14.522	[{"timestamp": 1556819287743, "MQ5": 11,
☐ edit	25	2019-05-02 17:47:29.919	2019-05-02 17:49:03.565	[{"timestamp": 1556819249919, "MQ5": 16,
☐ edit	26	2019-05-02 19:53:41.629	2019-05-02 19:54:14.474	[{"timestamp": 1556826821629, "MQ5": 5,

Database Schema with data

The data storage system lives within 2 docker containers:

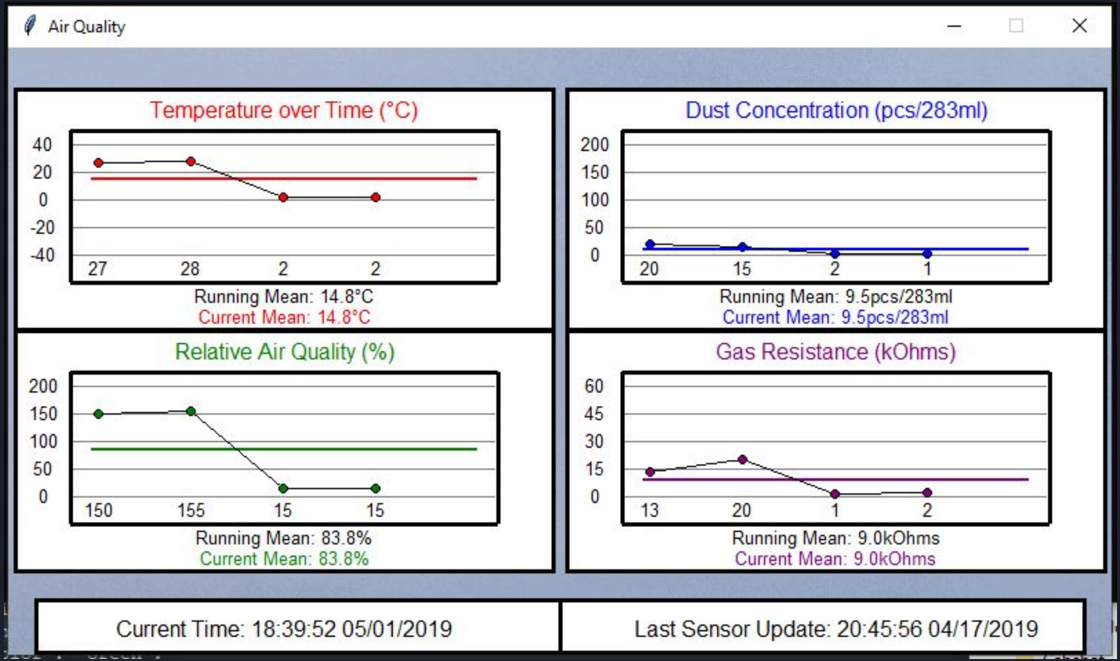
- PostgreSQL - Database
- Flask/uWSGI
  - Flask handles all the requests and database actions
  - uWSGI orchestrates the flask instances

The Pi and the server interface with each other via a REST API. This prevents unauthorized access to the server and allows for data sanitation.



```
{
  "url": "http://cs370.imaoreo.io/data",
  "background": "field.gif",
  "padding": 10,
  "displaySize": [
    800,
    440
  ],
  "graphs": {
    "temperature": {
      "title": "Temperature over Time",
      "units": "°C",
      "color": "Red",
      "size": [
        270,
        80
      ],
      "midpoint": 0,
      "range": 40,
      "dataPoints": 5
    },
    "dust": {
      "title": "Dust Concentration",
      "units": "pcs/283ml",
      "color": "Blue",
      "size": [
        270,
        80
      ],
      "midpoint": 100,
      "range": 100,
      "dataPoints": 5
    },
```

Snippet of display.py's config.json, which is used to change aspects about the graphs and window to be displayed



Temperature over Time (°C)

Running Mean: 14.8°C
Current Mean: 14.8°C

Dust Concentration (pcs/283ml)

Running Mean: 9.5pcs/283ml
Current Mean: 9.5pcs/283ml

Relative Air Quality (%)

Running Mean: 83.8%
Current Mean: 83.8%

Gas Resistance (kOhms)

Running Mean: 9.0kOhms
Current Mean: 9.0kOhms

Current Time: 18:39:52 05/01/2019

Last Sensor Update: 20:45:56 04/17/2019

Example of displayed data after reading in 4 data points from the server. Total number of points to be specified and the expected range of values is specified in the config.json file. The display program can run on any machine with both a display and capabilities to run Python3.