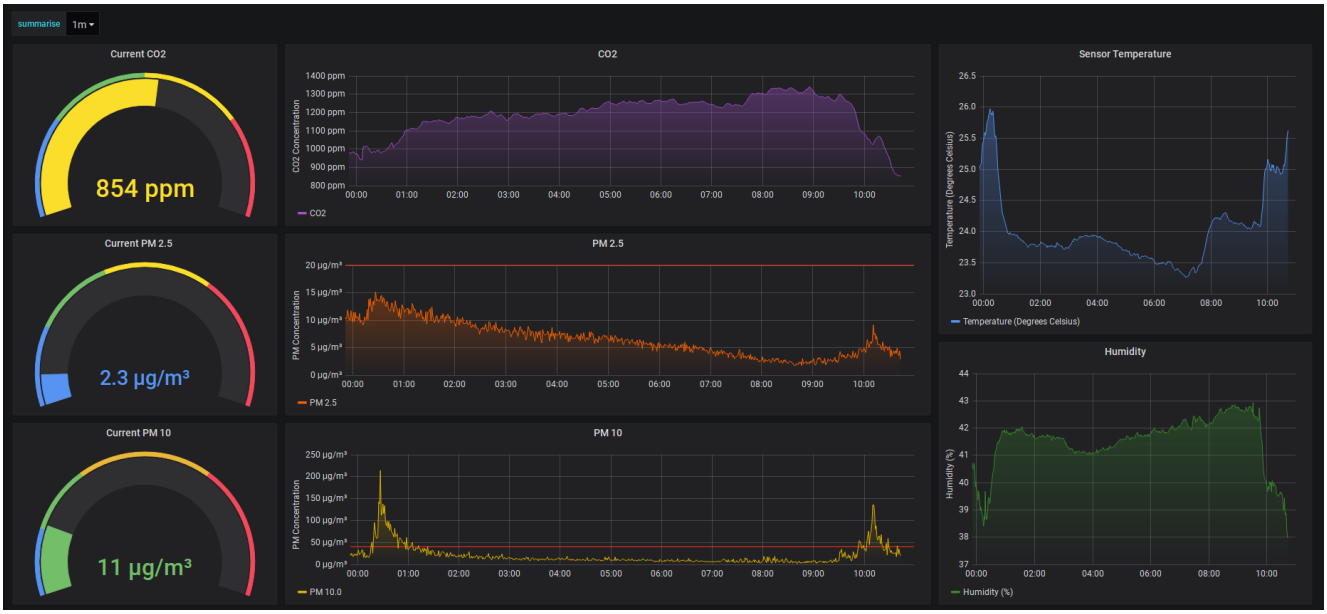
Finn Hobson
Computer Science Undergraduate
Computer Science Department
Merchant Venturers Building
Bristol, BS8 1UB
Email: fh16413@bristol.ac.uk

University of BRISTOL

Air Quality Research Project

Creating an Indoor Air Quality Data Dashboard



This guide describes how to build a dashboard that visualises the data collected by a Raspberry Pi air quality sensor.

The components of our air quality monitoring system:

- **Sensors** – Output current values for temperature, humidity, CO_2 concentration and particulate matter concentration. The sensors used during the creation of this guide were the Alphasense OPC-N3 particulate matter sensor and the MH-Z19 CO_2 sensor, but using other sensors should not change the method significantly. This guide assumes your sensors are already connected to the Raspberry Pi.

- **Raspberry Pi** – Receives the measurements from the sensors via the GPIO pins. Runs a Python script to send the data to the database.

- **InfluxDB** – A time series database to organise and store our measurement values.

- **Grafana** – A tool for creating graphs and dashboards from the data stored in the time series database.

The following steps should be completed on the Raspberry Pi. This requires a keyboard, a mouse, a monitor and a HDMI cable.

1. Install/Configure InfluxDB & Grafana

Instructions for installing and configuring InfluxDB and Grafana on your Raspberry Pi can be found on the following webpage:

https://www.circuits.dk/install-grafana-influxdb-raspberry/

Once you have installed InfluxDB on your Raspberry Pi, you can start the InfluxDB service using the following command in a terminal:

```
sudo service influxdb start
```

InfluxDB will start running on HTTP port 8086 by default.

We will now use Influx command line interface (CLI) to create a new database for our sensor data. Start the CLI using the following command:

```
influx -precision rfc3339
```

You should then see the following appear in the terminal:

```
$ influx -precision rfc3339
Connected to http://localhost:8086 version 1.7.x
InfluxDB shell version: 1.7.x
>
```

We can now enter input in the form of Influx Query Language (InfluxQL). To create a new database, enter the following InfluxQL statement with any name of your choosing. Database names can only contain ASCII letters, digits, or underscores and must not begin with a digit:

```
> CREATE DATABASE my_database
>
```

Now that you have created a new database, use the `SHOW DATABASES` statement to display all existing databases. You should see `name`, `_internal` and the database you just created:

```
> SHOW DATABASES
name: databases
---------------
name
_internal
my_database

>
```

We have now created a database and it is ready to store the data from our sensors. Exit the Influx shell by typing the command `exit`.

2. Write Python Script to Send Data to InfluxDB

The Python script created alongside this guide can be found on GitHub:

https://github.com/finnhobson1/Air-Quality-Project/blob/master/airqualityinflux.py

This script receives measurement values from the Alphasense OPC-N3 particulate matter sensor and the MH-Z19 CO_2 sensor every 4-5 seconds and writes these data points to the InfluxDB database we created in the previous step.

Before writing your own script, you must install the InfluxDB Python library. This can be done using the following command in a terminal:

```
sudo apt-get install python-influxdb
```

Here are some of the key elements you need to include in your Python script to write the measurement values to your InfluxDB database:

At the start of the script, make sure to import the InfluxDB library along with any other libraries you might need (lines 3-8):

```
from influxdb import InfluxDBClient
import time
import datetime
```

Before you can send write data to your InfluxDB database you must set some parameters (lines 25-33). Make sure to set `dbname` to the exact name of the database you created earlier. You can set `session` to anything you like such as "test1" or "experiment1":

```
host = "localhost"
port = 8086
user = "root"
password = "root"
dbname = "my_database"
session = "session_name"
```

After these parameters have been set and before you start collecting measurement values, you must initialise a InfluxDBClient object with the parameters (line 96):

```
client = InfluxDBClient(host, port, user, password, dbname)
```

You now need to write a function to get the data points that you want to write to your database. In the example code, there is a function called `get_data_points()` which returns a variable containing a new set of measurement values ready to write to the InfluxDB database.

In this function, first create variables for each measurement value and create a timestamp variable (line 56):

```
timestamp = datetime.datetime.utcnow().isoformat()
```

You now need to package your data points in a variable in JSON format. Add all of the individual measurement values as a key-value pair in the "`fields`" section. For example (lines 70-86):

```
datapoints = [
                {
                      "measurement": session,
                      "tags": {
                      },
                      "time": timestamp,
                      "fields": {
                            "temperaturevalue":temperature,
                            "humidityvalue":humidity,
                            }
                      }
                ]
```

You can also add tags to your data. In the example code, the data is tagged by "`run_num`" which is the time that the script started running.

Once you have a variable containing some datapoints in JSON format, write them to your database (line 111):

```
result = client.write_points(datapoints)
```

The `client.write_points()` function returns `True` if the data is written to your database successfully, and `False` otherwise. Use this returned value to check for errors.

Create a loop that repeatedly takes new measurements and writes these new data points to your database. You can set your chosen sampling rate using the `time.sleep()` function (lines 106-118).

Save your script, then open a terminal and navigate to the directory where you script is saved. Run the script using the following command, replacing '`my_script.py`' with the name of your script.

```
sudo python my_script.py
```

If you want your script to run automatically when your Raspberry Pi boots, there are a few different methods for doing this. These methods are explained on the following webpage:

https://learn.sparkfun.com/tutorials/how-to-run-a-raspberry-pi-program-on-startup/all

3. Create Grafana Dashboard

If you have installed Grafana on your Raspberry Pi, start the Grafana server and configure it to start automatically at boot using the following commands:

```
sudo service grafana-server start
sudo update-rc.d grafana-server defaults
sudo systemctl enable grafana-server
sudo systemctl start grafana-server
```

Grafana will start running on HTTP port 3000 by default. You should be able to access the Grafana web GUI by typing the following into the address bar of your web browser:

```
http://localhost:3000
```

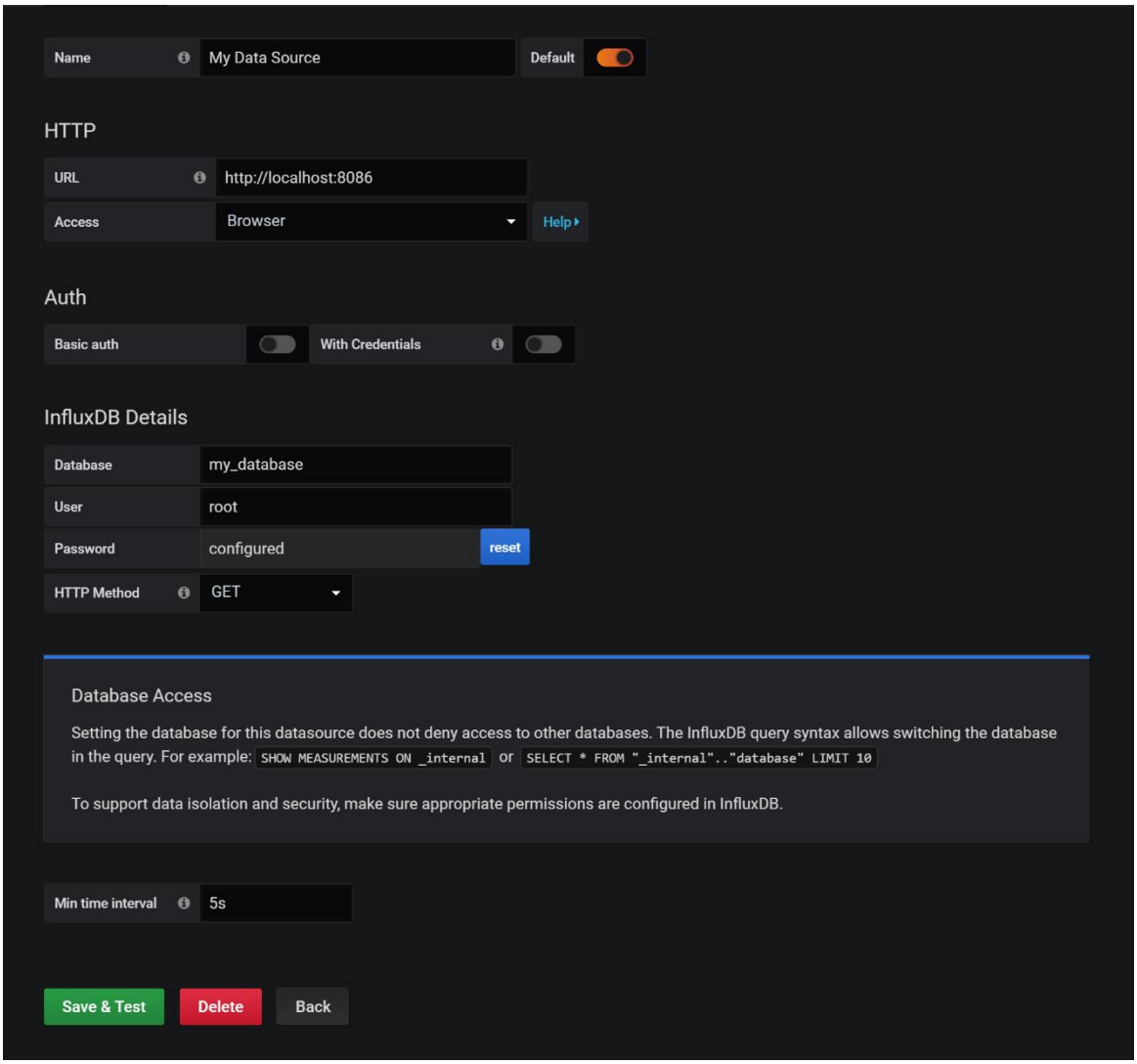
You should then see the Grafana login page. The default username is **admin** and the default password is **admin**. Change the password when you are invited to:

Once you have logged in, you first need to add your database as a data source. In the side menu, go to Configuration > Data Sources. Then click on the 'Add data source' button.

You should see a list of database options. Select InfluxDB.

You will then need to fill in the details of the InfluxDB database you created:

- You can give the data source any name you would like. Click the ‘Default’ option next to the ‘Name’ field.
- The URL should be the address of your InfluxDB server. This is **http://localhost:8086** by default.
- Make sure the name in the ‘Database’ field is the exact name you gave to the database you created earlier.
- Enter **root** into the ‘Username’ and ‘Password’ fields.
- Select **GET** as the ‘HTTP Method’.
- Set the ‘Min time interval’ to approximately the sampling rate from your Python script.



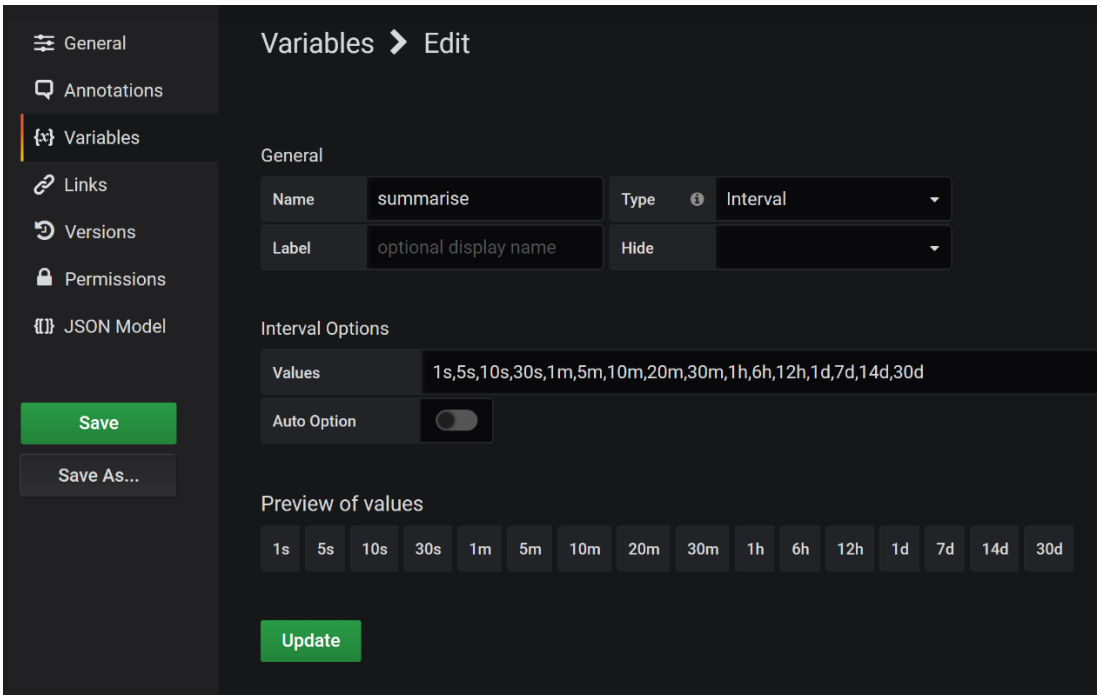
Once you have entered these details, click ‘Save & Test’. If everything has been entered correctly, you will see a ‘Data source is working’ message appear in a green box.

Now that your data source is set up, in the side menu go to Create > Dashboard.

You will see a blank dashboard with a ‘New Panel’ ready to create. Before we create our first graph, we will add a drop-down menu to our dashboard that will let us choose the time-resolution of our data (so we can choose to show a 10-second average, a 1 minute average, a 5 minute average etc).

Click on the ‘Dashboard settings’ button (the cog symbol) then go to ‘Variables’ and click on the ‘Add variable’ button.

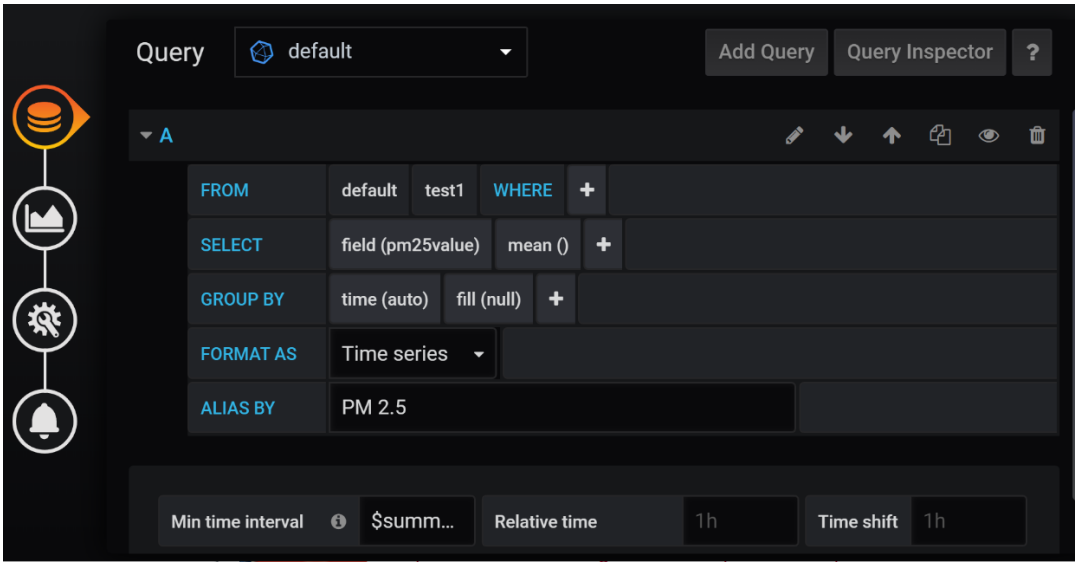
We will create a new variable called ‘summarise’. Enter the details shown below. In the ‘Values’ field, enter all of the time intervals that you want to choose between for averaging your data:



Click the ‘Add’ button then go back to the dashboard.

Click on the ‘Add Query’ button to begin creating a graph from the one of the data values in your database. The query should be similar to the screenshot shown below:

- Select a measurement and a value from your database.
- Enter **auto** into the ‘time()’ field.
- Enter an appropriate name for that data value in the ‘Alias By’ field.
- Enter **$summarise** into the ‘Min time interval’ field.



A line graph should now appear showing the data for your chosen value. You can change the appearance of the graph, add a title/axis labels, add threshold lines and more by going to the 'Visualisation' and 'General' tabs.

You can change the range of your data and refresh your dashboard in the top right corner of the screen.

Go back to your dashboard and try creating some more panels for your other data values using the 'Add panel' button. You can also use different visualisation formats such as gauges to show the current reading for a value.

Try to fill your dashboard with as many useful panels as possible. You can change the layout of your dashboard by dragging the panels around and changing their size. You can change the colour of the graph data by clicking on the coloured line next to the name of the data.