Sending environmental data from Raspberry Pi to Graphite

Raspberry Pi[1] is a mini-computer with an ARM processor designed specifically for teaching purposes. It is a low-cost device designed to improve pre-university hardware programming and understanding skills. Due to the small size and affordability, electronics manufacturers and researchers have integrated Raspberry Pi into projects that require more than one basic microcontroller. Raspberry Pi is slower than a modern desktop or laptop, but it is a complete Linux computer and can implement multiple features with low energy consumption.

The application is developed using Python, due to its fast implementation capability. The data measured with the Raspberry Pi 3B integrated sensors will be transmitted via StatsD[2] to Graphite. The connection of the peripherals to Raspberry Pi 3 B is presented in Fig. 1, and the module setup is shown in Fig. 2.

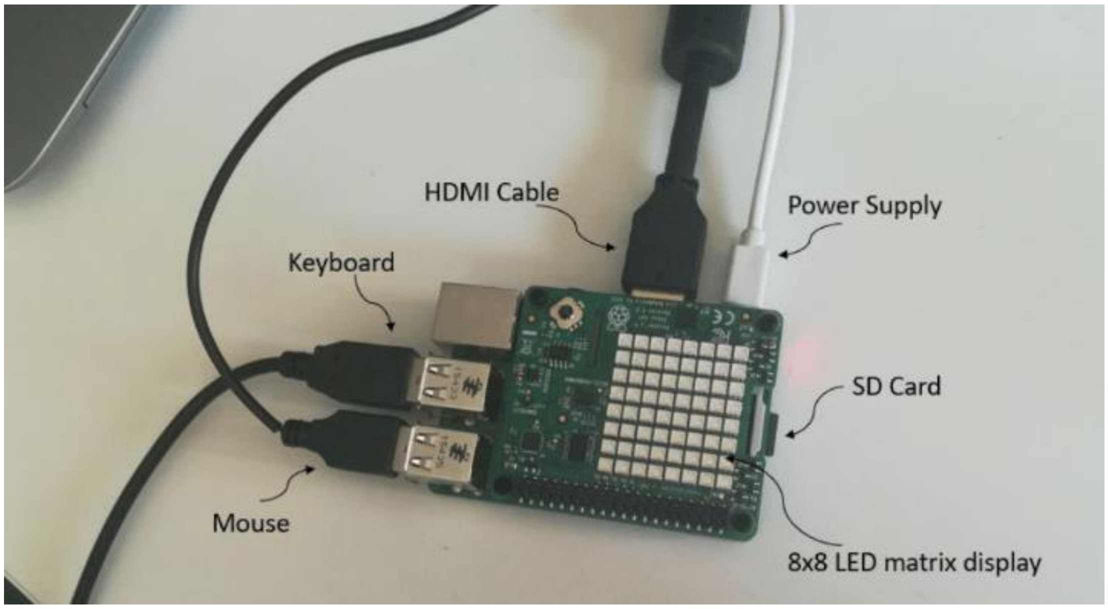

Fig. 1 *– Connecting the peripherals to Raspberry Pi*

[1] https://www.raspberrypi.org/
[2] https://www.datadoghq.com/blog/statsd/

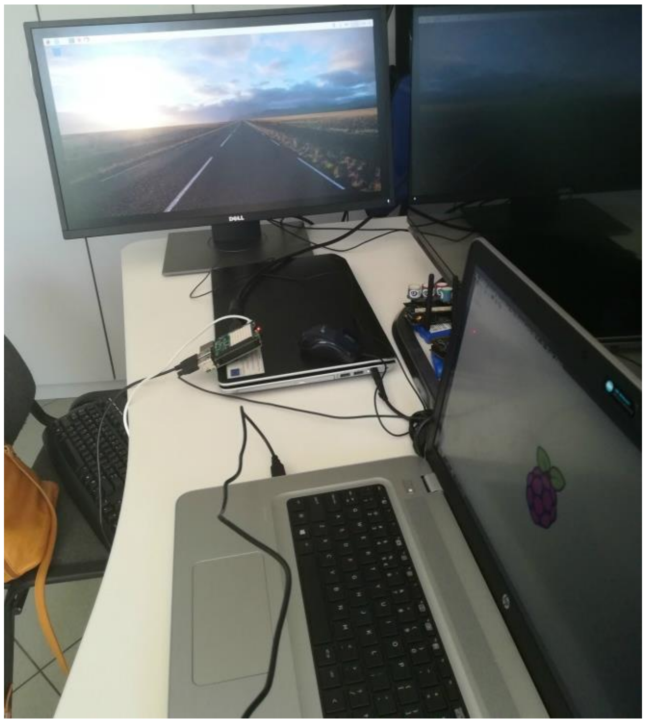

Fig. 2 *– Setting up the Raspberry Pi*

Graphite[3] is a graphic library that stores visual representations of data. It requires other applications for data collection and transfer, and works on the basis of being supplied with information through other services. StatsD sends data from Raspberry to Graphite efficiently because it works based on UDP[4] (User Datagram Protocol) packets. StatsD collects data and then sends it to Graphite in the format it expects to receive it. The flush interval is set by default to 10 seconds, but the it can be modified as needed. Fig. 3 describes the data transmission process.

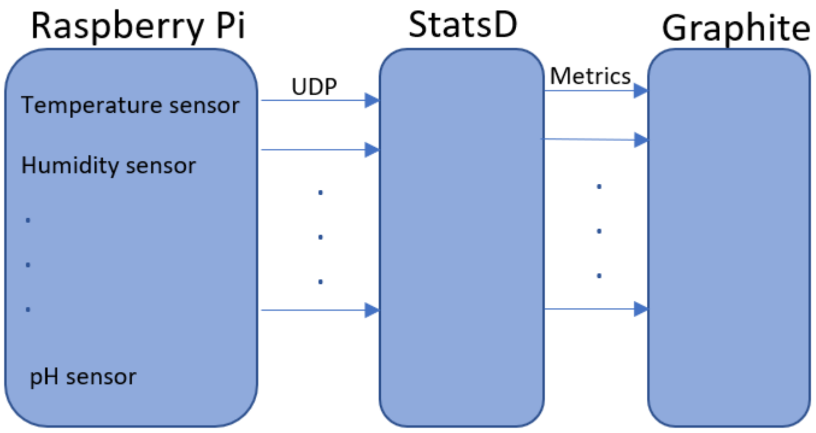

Fig. 3 *– Sending data to Graphite using StatsD*

[3] https://graphite.readthedocs.io/en/latest/
[4] http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/udp.html

StatsD

A StatsD server has the role of retrieving information from multiple clients and sending them at a pre-set time for storage and viewing. By using StatsD, it is possible to both receive custom metrics and to do real-time monitoring of the system's operating state. StatsD is a daemon that can be used to transmit information to Graphite. UDP packages will be listened to through a dedicated interface. In this way, StatsD can receive a large amount of information that can be stored in a buffer if there is no backend connectivity. Unified values will be sent to Graphite. The StatsD structure contains the following components: the client, the server and the backend.

The client is a library called each time certain values collected by the StatsD server are transferred. These data are first aggregated by the server, and then sent to backends, which perform data operations. For example, Graphite backend allows real-time visualization of graphs for measured measurements.

Graphite

Graphite is a set of software used to store, collect, and graphically display temporal information. The collected data is transferred through specialized applications designed to retransmit this information in the format supported by Graphite. Many companies are implementing Graphite to monitor the development plan and the production of e-commerce services. The functions performed by Graphite are:

- Data storage in time-dependent series;
- Delivering data graphs.

The graphite tool structure is as follows:

- Carbon - the daemon listening to time-dependent data series;
- Whisper - Database for data storage in time-dependent series;
- Webapp - a (Django) webapp used to deliver graphics on demand.

Carbon refers to one or more types of daemons that make up the graphite installation backend. For simple installations, usually there is only one daemon called carbon-cache.py. This document provides a brief description of the role of each daemon, and how to use it to build more sophisticated storage backends.

- Whisper is a fixed-size database that allows fast and reliable data storage over time. Whisper allows sharing larger resolutions of recent data in smaller resolutions to keep the history of data for a long time.
- WebApp is a Django app running under Apache / mod_wsgi, and generally provides an API reference point based on a URL to retrieve data and generate graphs and a user interface to navigate through the metrics.

The data displayed by Graphite is transmitted by Carbon and Carbon-Relay (configuration file used for deletion and replication). The Graphite web interface will reproduce the information in the cache or directly from the disk.

The method of data transfer differs according to the process through which the application or script through which the data was sent was made.

- There are working tools and APIs useful for transmitting data to Carbon;
- For a single script or for data testing, the most useful method is to use the Plaintext protocol.

Below is presented the code[5] used for taking measured data with Raspberry Pi sensors, and their transmission to Graphite.

```
#!/usr/bin/env python
from sense_hat import SenseHat
import time
from statsd import StatsClient
sense = SenseHat()

client = StatsClient(host="docker-master.beia-consult.ro",
port=8125)
while True:
    t = sense.get_temperature()
    p = sense.get_pressure()
    h = sense.get_humidity()

    t = round(t, 1)
    p = round(p, 1)
    h = round(h, 1)

    msg = "Temperature = {0}, Pressure = {1}, Humidity =
{2}".format(t,p,h)

    client.gauge("beia.raspberry.temperature", t)
    client.gauge("beia.raspberry.pressure", p)
    client.gauge("beia.raspberry.humidity", h)
    print (msg)
    time.sleep(15)
```

StatsD configuration consists of setting 2 parameters: host and port[6].

```
import statsd
client = StatsClient(host="docker-master.beia-consult.ro", port=8125)
```

host = "docker-master.beia-consult.ro" – StatsD server adress.

port = 8125 – StatsD server port, is set by default for both the server and the client .

[5] https://github.com/beia/beialand/blob/ODSI/projects/ODSI/raspberry_sensor_data.py

[6] https://github.com/etsy/statsd

Fig. 4 shows the pressure values visualization using Graphite. It can be noticed that the room pressure value is constant, and the graph form is due to the interruption of the data transmission to Graphite.

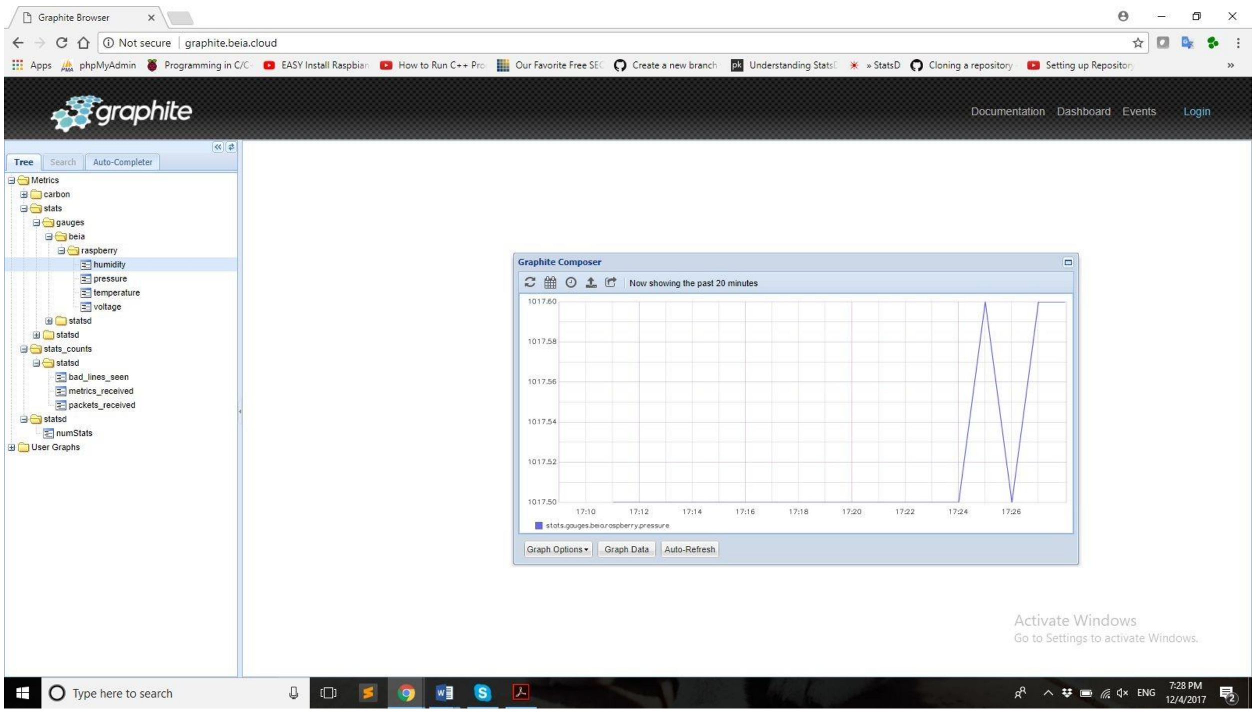

Fig. 4 *– Pressure graph displayed using Graphite.*

Also, Fig. 5 shows the graphs for humidity and temperature values sent during the last 20 minutes.

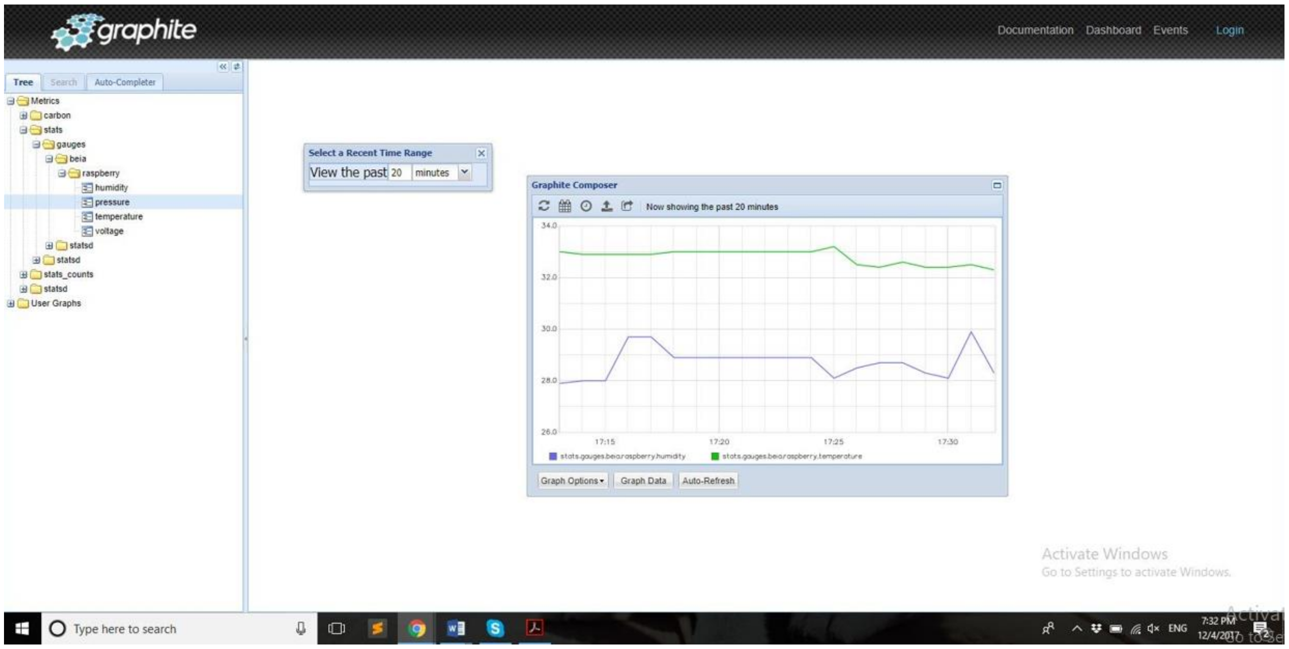

Fig. 5 *– Humidity and temperature data graphs for the last 20 minutes*

Fig. 6 presents the graphs for humidity and temperature values sent during the last 4 hours, before the transmission interruption.

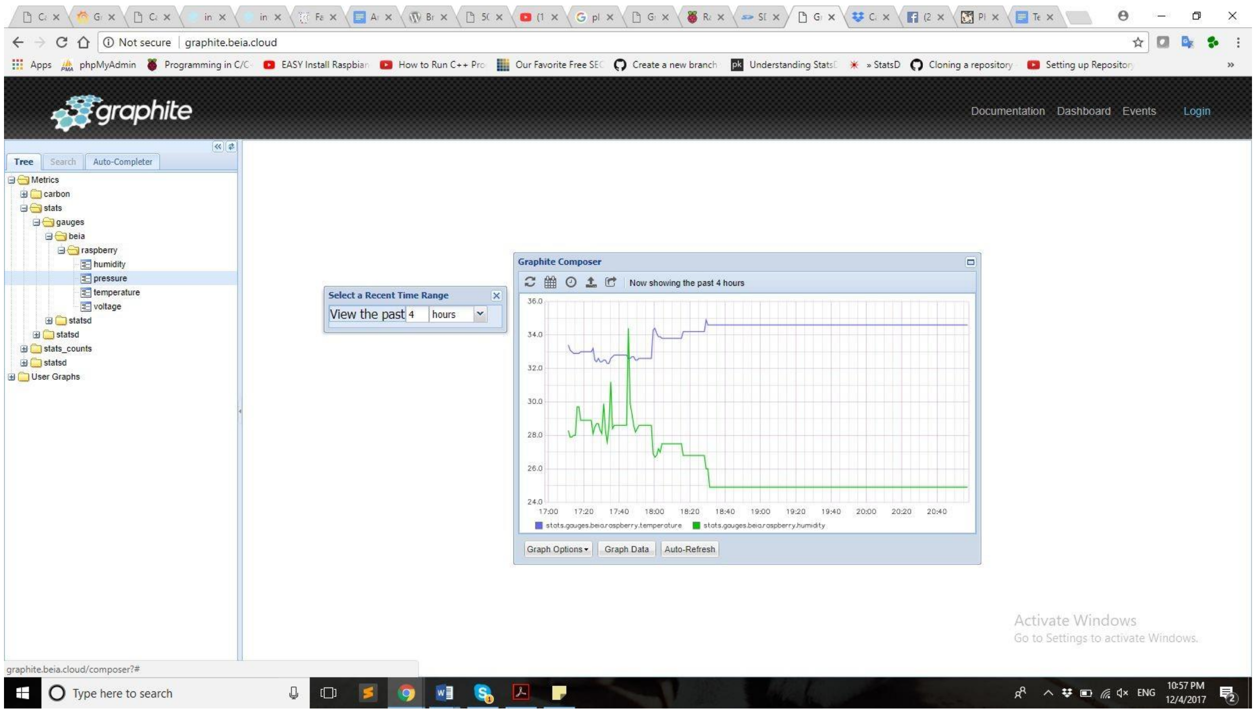

***Fig. 6** - Humidity and temperature data graphs for the last 4 hours*