Institut Universitaire de Technologie,
Aix-Marseille Université

INTERNSHIP REPORT
Diplôme Universitaire de Technologie
Spécialité Réseaux et Télécommunications

Wireless sensors connection to a cloud

Andréa Voisin-Grall

University of the West of Scotland

Project supervisor: Zeeshan Shakir

Academic supervisor: Houssain Corrine

2019

Table of contents

1 Introduction

1.1 Introduction to the company

The University of the West of Scotland (UWS) is a Scottish public university with four campuses in south-western Scotland, in the towns of Paisley, Hamilton, Dumfries and Ayr.

It is one of Scotland's most vocational higher education institutions, having strong links with industrial and commercial partners. Computing is a key strength in UWS, now formally acknowledged by an excellent teaching assessment. The UWS offer a wide range of academic, social and sporting facilities and deliver a range of taught courses (Certificate, Diploma, Bachelor degrees (BA, BAcc, BEng, BSc), Postgraduate Diplomas, and Masters (MA, MBA, MSc) and research degrees (Master of Research (MRes), Master of Philosophy (MPhil) and Doctor of Philosophy (PhD)).

The university has 15,955 students, with approximately 1300 staff, spread across six schools of learning, and it is known to be one of the the best universities of Scotland.

1.2 Introduction to the Project

The Internet of Thing (IoT) refer to the interconnection between different objects and devices that are able to communicate between each other's and to send data without any kind of human interaction. From a simple sensor to a connected watch IoT is able to provide solutions to improve human life and improve industrialization by performing in an intelligent way different little tasks.
It has been already invented few years ago but it is already a key sector in the industry and will probably be one of the principal subjects of research in the future.
But any kind of exchange of data mean a potential intrusion, meaning that security will also be one of the main challenges that people will be facing when they will deal with IoT.

The goal of this project is to set up a board full of sensors remotely and to create an interface, allowing users to have an access in real time to all the data that the sensors are measuring, without the constraint of having wires everywhere. The project will be using Pycom products, such as a Pysense shield* and a Fipy development module. The data that it will be measuring will be temperature, humidity, pressure altitude and light. And they will be all printed on a Wia cloud* using lines charts and text widgets. To transfer those data, we will use the Wi-Fi protocol.

2 Devices and software

In this part we will be explaining the different tools we used to develop this project and the reason why we decided to use them.

2.1 Pycom Ltd

Pycom Ltd is a small company that provide hardware and software solutions for IoT, they propose different kind of devices that allow you to work with a large panel of specifications such as Wi-fi, LoRa*, Arduino, Raspberry PI…
Most of the Pycom products and every one of those that we will be using are written in Python but we will talk about Python in a next part.

Since Pycom is not a really known company not a lot of work linked to their devices has been published which has been one of the challenges we faced during the development of this project.

2.1.1 Pysense

Pysense is a shield from Pycom company that is equipped with different sensors that can be combined with a Pycom development module (e.g Lopy, Fipy, Gpy).
The sensors and feature that Pysense own are:

- Barometric pressure sensor
- Humidity sensor
- 3 axis 12-bit accelerometer
- Temperature sensor
- Ambient light sensor
- Ultra-low power operation (~1uA in deep sleep*)
- USB port with serial access
- LiPo* battery charger
- MicroSD card compatibility

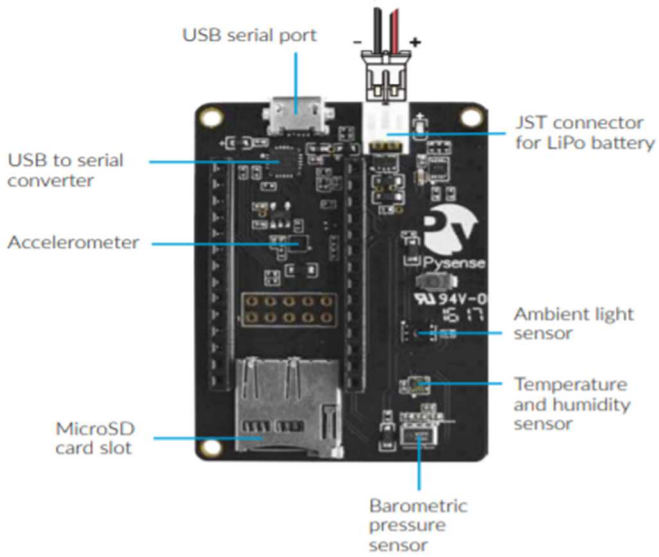

Figure 1: A Pysense Shield from Pycom Ltd

The interesting thing about this device is that it allows us to have a multitude of sensors at the same place avoiding the multitude of wires that we would have to use we were using an Arduino or a Raspberry Py. It also provides us a solution to create a single code to set up all of the sensors at once which make us spare a lot of time.

This device cannot work alone because it doesn't own any microcontroller, this is one of the reasons why we will be using a FiPy development module.

2.1.2 FiPy

FiPy is a development module for Pycom shields that provide a Wi-Fi, Bluetooth (BLE*), SigFox*, LoRa and dual LTE-M* (CAT M1 and NBIot) connectivity.
It has a low consumption but provide a quick connectivity which is what we were looking for during this project.
Fipy might require an antenna to work properly and provide a good signal (Appendix C), especially for LoRa but since it was working fine without one, we decided to not use it for the Wi-Fi (we will also explain why we decided to choose Wi-Fi rather than the other protocols)

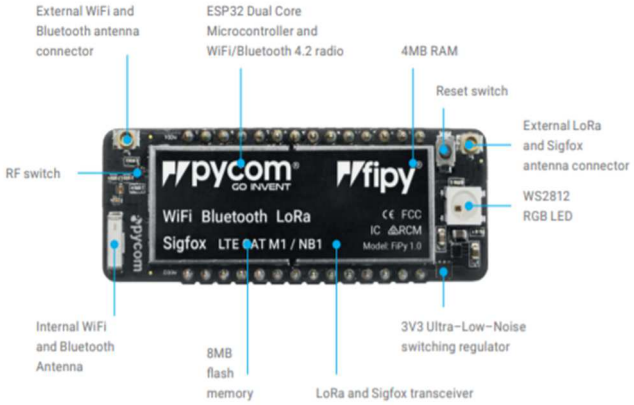

Figure 2: A FiPy development module from Pycom Ltd

FiPy is quite easy to setup because you only need to plug it on your Pysense to connect it (Cf. figure 3)

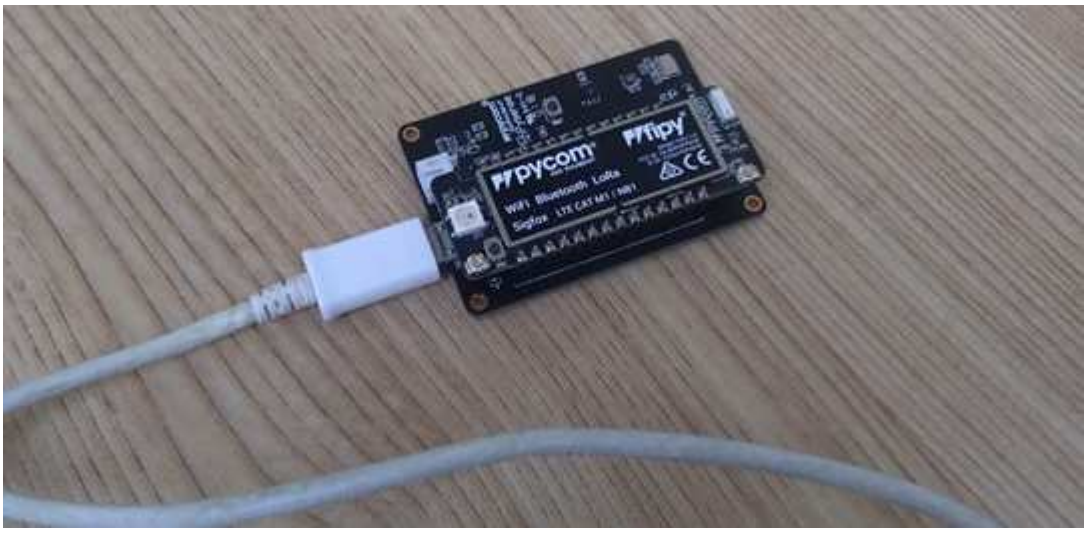

Figure 3: A Fypy and a Pycom plugged together

2.2 Python

As we said previously Pycom products are written in Python so we had no other choices but to use it.

Python is an open source programming language; it is an interpreted language which mean that it doesn't need to be compiled to run. Python is not really hard to learn and is easy to practice which was quite an advantage for the program we developed on this project.

Since I had no real knowledge of Python, I had to learn it through different websites that were quite helpful and complete.

2.3 Atom

To write our different programs in Python we had to use a text editor, there was a lot available but in the end we decided to use Atom because of two main reasons.

The first one was that most of the few previous Pycom users were using Atom so using the same software was a good way to spare a good amount of time and energy.
The second reason was that Atom provide a plug-in developed by Pycom Ltd called Pymakr (Cf. figure 4) that allow you to have access to a console to print your data in real time but also to upload and run your programs directly into your Pysense/FiPy (as long as you have a serial connection).

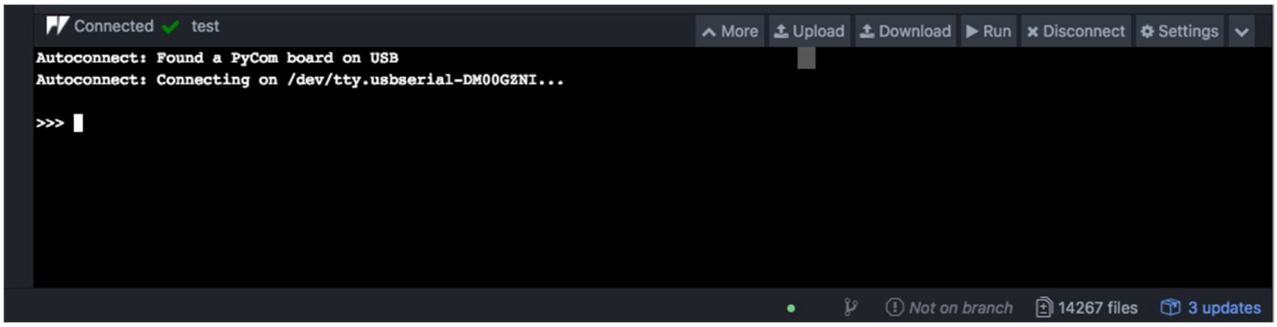

Figure 4: Pymakr plug-in

2.4 Wi-Fi

Wi-Fi is a wireless network technology based on the norm IEEE 802.11. It allows to connect by radio wave different informatic devices in the same network and to send data between them.
There were different protocols that we could have been using since Fipy is compatible with a lot of different protocols but to transfer our measures but Wi-Fi was one of the simplest one. We also already had a gateway configured and ready to be used.

2.5 Wia

Wia is a cloud platform backend for IoT solutions, it is compatible with Raspberry Pi, Pycoms products, Arduino, and a lot of other development boards. It only requires a Wi-Fi connection and few lines of code to work and will then provide a scalable dashboard with all the tools required. Such as a debugger, an events console, and plenty charts to print the data measured…

There was a lot of cloud available that we could have used as well but we decided to choose this one because it was one of the most user-friendly and the explanation on the website were quite complete.

3 Realization

In this part we will be explaining our procedure to realize this project.

The first thing to do was to connect our devices to a computer to inject some programs but for that we had to upgrade our firmware first.

3.1 Firmware update

To allow our Pysense/Fipy to be recognized by the PC and Atom we had to update the firmware, we found all the information about this procedure on the official Pycom website, and few tutorials were available on the internet. This was quite a difficult thing to do but we will explain why later.

Once the device was detected by the Pymakr plug-in we were now ready to implement our first code.

3.2 Testing the sensors

Now that we were able to put some programs into our devices we had to test if the Pysense sensors were working properly.

For that we used a simple program that was collecting the data and printing them on the console (Cf. figure 5).

```
import time
from pysense import Pysense
from LIS2HH12 import LIS2HH12
from SI7006A20 import SI7006A20
from LTR329ALS01 import LTR329ALS01
from MPL3115A2 import MPL3115A2,ALTITUDE,PRESSURE

py = Pysense()
mp = MPL3115A2(py,mode=ALTITUDE) # Returns height in meters. Mode may also be set to
PRESSURE, returning a value in Pascals
mpp = MPL3115A2(py,mode=PRESSURE) # Returns pressure in Pa. Mode may also be set to
ALTITUDE, returning a value in meters
si = SI7006A20(py)
lt = LTR329ALS01(py)
li = LIS2HH12(py)

while(True):
    print("Temperature: " + str(mp.temperature()))
    print("Altitude: " + str(mp.altitude()))
    print("Pressure: " + str(mpp.pressure()))
    print("Humidity: " + str(si.humidity()))
    print("Light: " + str(lt.light()))
time.sleep(15)
```

Figure 5: Sensors testing program

We will now explain in the easiest way possible the code above:

- The green part corresponds to the different libraries we are using, each one corresponds to a different sensor except the import time that is here to create our loop.
- In the blue part we define the variables that we will will use to retrieve the measures.
- The red part allows us to create a loop that will repeat each 15 seconds.
- The purple part is getting the value from the different variables and print them on the console.

And this is the results we get on the Pymakr console:

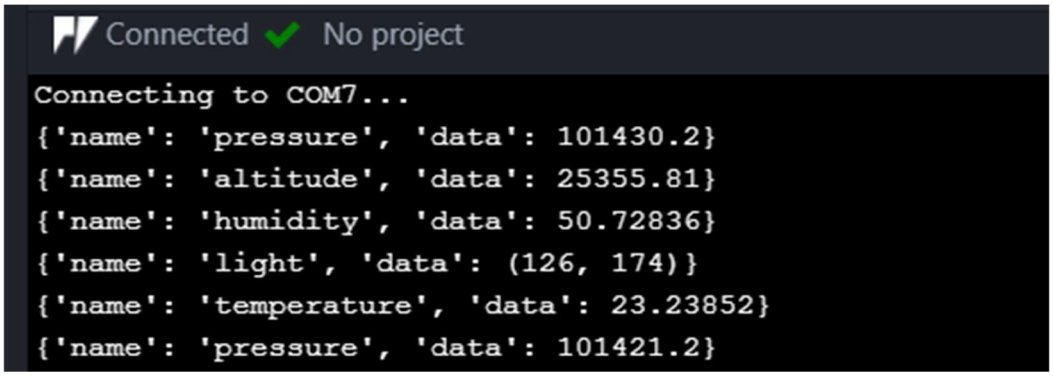

Figure 6: Sensors results

As you can observe, we are not using all of the sensors that the Pysense provide (like the accelerometer for instance), the reason is that we decided to put only the sensors that could fit in a living space or in a working area. Since the Pysense/FiPy are standing still we don't see any utility of using the accelerometer. We still kept the altitude sensor because it was linked to the barometric sensor and was used to debug it.

If you take a look at the result on the console you can see that the altitude values are wrong and way too high (should have been something about 13 meters), but once again we will explain why later.

3.3 Testing the Wi-FI

The next step was to control if our FiPy module was able to connect to any network with Wi-Fi.

To achieve that we only needed a classic router that would be connected to the internet.

It was important to know the SSID* and the Key of this router to connect it to the device.

This is the code we used:

```
from network import WLAN
from pysense import Pysense

# Wifi credentials
WIFI_SSID = 'mySSID'
WIFI_KEY = 'myWIFI_KEY'
```

```
wlan = WLAN(mode=WLAN.STA)
nets = wlan.scan()

# Connect to the WiFi network
for net in nets:
    if net.ssid == WIFI_SSID:
        print('Network found!')
        wlan.connect(net.ssid, auth=(net.sec, WIFI_KEY), timeout=5000)
        print('Connecting...')
        while not wlan.isconnected():
            machine.idle() # save power while waiting
        print('WLAN connection succeeded!')
        break
```

Figure 7: Wi-Fi testing program

Once again, we will explain the code above in an easy way:

- Green part corresponds to the libraries we are using.
- The blue part corresponds to the creation of our variables, this is where we put the router SSID and its key. Having them into the code can be quite dangerous so encrypt them would be a great thing to do if someone decide to work on that project in the future.
- The purple part corresponds to a loop where the program will try to search for a network with the same SSID we put, and try the Wi-Fi credentials we gave. If the program found it will print “Network found”, “connecting’ and once it is connected “WLAN* connection succeeded!”.

And this is the results we get:

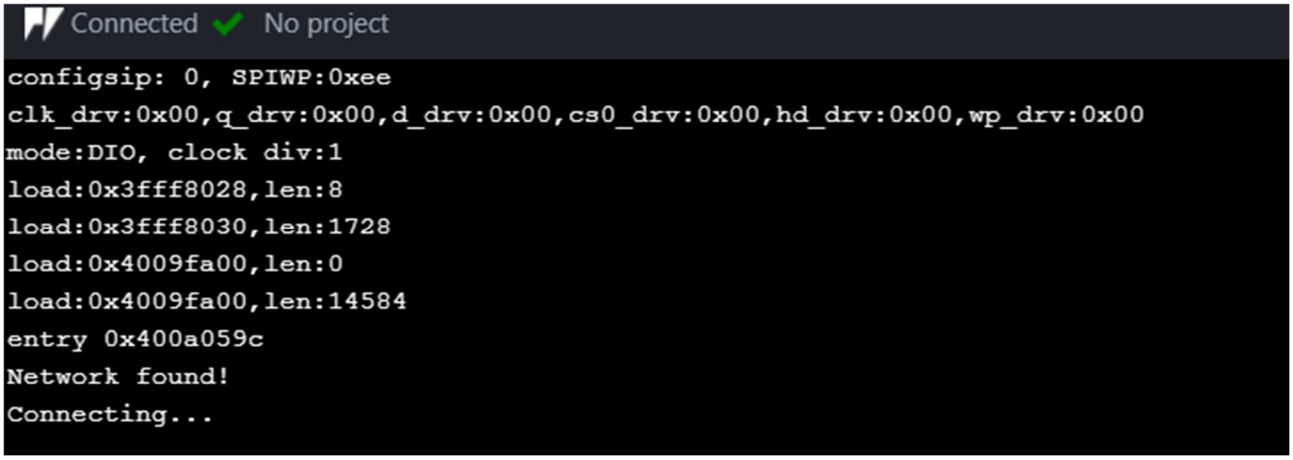

Figure 7: Wi-Fi results

Network is found and the Fipy is connected to the Wi-Fi network.

3.4 Connecting and posting to Wia

Once our devices were connected to the network, we now had to write the program that would connect our device to Wia via the Wi-FI and then send our measurements.

First thing to do to achieve that was to create an account on Wia and to save a device, which would give us a secret key that we included on our code. By doing that we increase a little bit the security of the project.

```
# Wia secret key to protect the device from any intrusion
DEVICE_SECRET_KEY = '...'
```

Then we used this code from Wia website that enabled us to create a function type that we would use later to send our data to the Wia cloud.

```
url = "https://api.wia.io/v1/events"
headers = { "Authorization": "Bearer " + DEVICE_SECRET_KEY, "Content-Type":
"application/json" }

# Create an Event for Wia
def post_event(name, data):
    try:
        json_data = { "name": name, "data": data }
        print(str(json_data))
        if json_data is not None:
            req = requests.post(url=url, headers=headers, json=json_data)
            return req.json()
        else:
            pass
    except:
        pass
```

And finally, we can use the function of above to send the different measures of our sensors.

The code below uses the same principles of working that the first program we used to test the sensor.

```
while True:
    post_event("temperature", si.temperature())
    post_event("pressure", mpp.pressure())
    post_event("altitude", mpa.altitude())
    post_event("humidity", si.humidity())
    post_event("light", lt.light())
    time.sleep(10)
```

Once these lines of code were included in our whole program, we uploaded it to our Fipy/Python.

To see if that was working, we used a tool from the Wia cloud called Debugger that would allow us to get real time feedback from our devices.

This is what we get from the debugger.

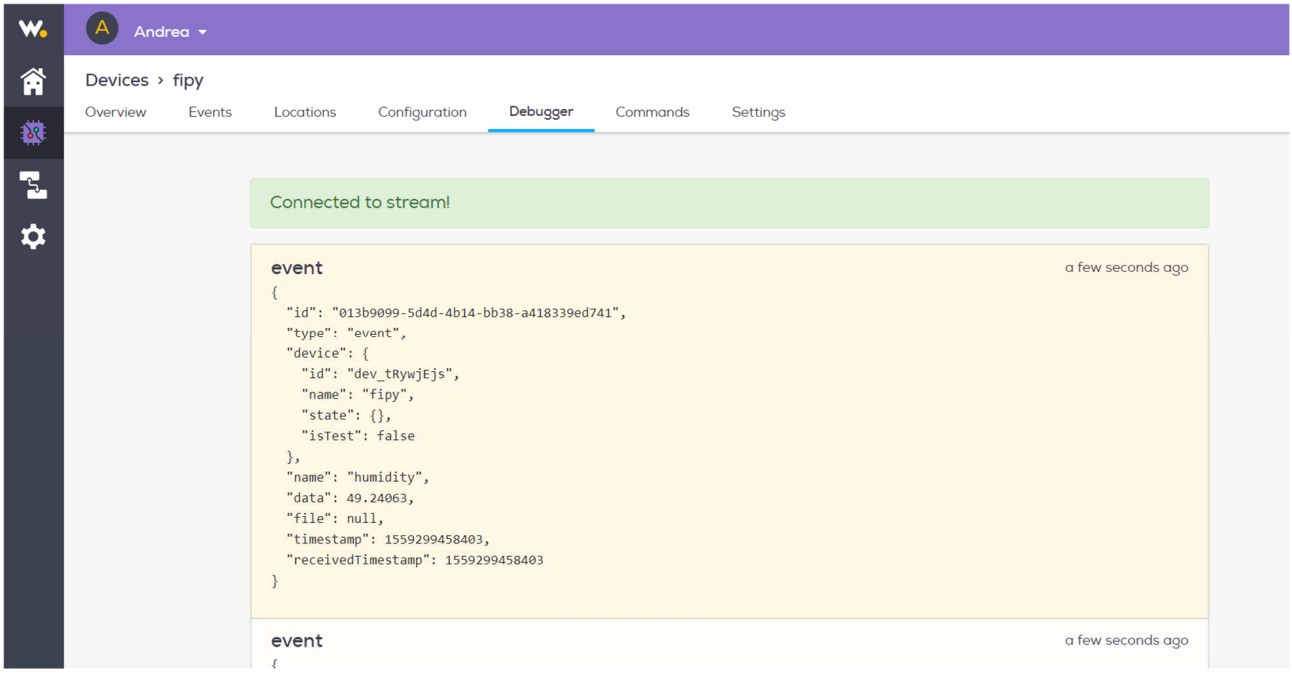

Figure 8: Wia Debugger

Note that all the events are stocked in the events page as well which is great to have an historic of all the measurements you have made.

3.5 Final step

Our last step was to gather all of the different programs and parts of program to a single one.

You can find the complete version of the program in the appendix A.

You will find as well in the appendix B a file name boot.py, this file allows you to run your program without having to do it from a computer, you just need to plug the device to a power supply (which make this project wireless).

The very last thing to do was to add different widgets to the Wia dashboard (Overview tab) to set our data in an easier way to read them.

After adding different widgets this is what the Wia dashboard looks like.

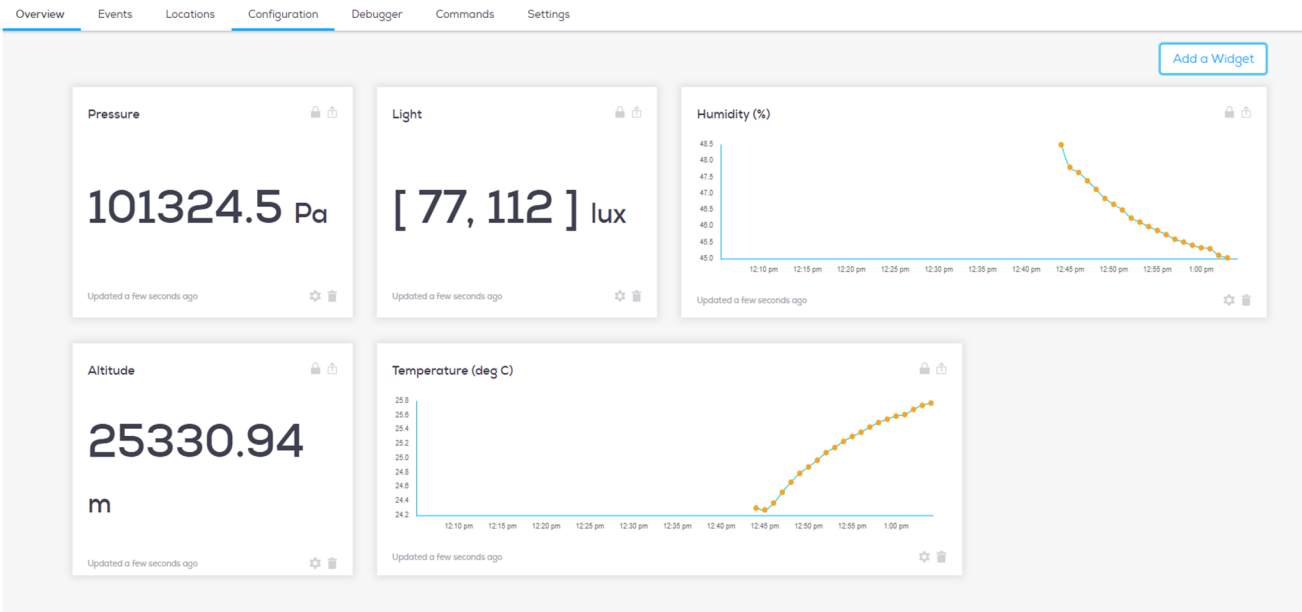

Figure 9: Wia Dasboard with all the measures

Everything is working fine, but once again you can see that the altitude value is wrong. We will explain that in the following part.

3.6 Major issues encountered

In this part we will outline the different issues we faced during the realization of this project and how we solved them.

The first trouble we had was when we tried to upgrade the firmware, most of the tutorials found on the web where not complete and most of the time we had different results from the others users. This problem was solved somehow by repeating different steps changing few variables each time. We won't develop on that because it would compel us to explain all the procedure and the plenty of software we used during this project which would be quite too long.

Second issue we faced was, as we indicate previously, that the altitude value from the sensor was printing wrong. The value is obviously too high and should have been something like 13 meters according to the few specialized websites we consulted. From what we understood, this error come from the fact that both altitude and pressure sensors are actually the same one, which mean that the

pressure and altitude numeric values are stored on the same registers, but the thing is that the numeric function has a scale of 1:4 between them so when you retrieve a value from altitude or pressure the other one will be 4 time increased or reduced. We spent a lot of time trying to fix this but there were probably no solutions. However, one time during a test period both values printed correctly but then everything turned back to the wrong ones and there was no way to get these results back.

Last issue we've encountered was not really an issue but more like a difficulty, indeed the bigger challenge of that project was that the documentation and help available was really rare. This mean that we had to search really deeply to get few the information and solutions we were looking for and obviously that we had to do a lot of test by ourselves to figure out our own solutions.

4 Conclusion

Those ten weeks of internship have been an incredible experience for me, I've learned so many things, from the way to work as a searcher making reports, having timelines and deadlines to the way of working on subjects almost unexplored. I have in a way discovered and started to learn the language Python which will be something useful in my future and I have acquired notions of electronic and programming. I have learned how to do things by myself and especially how to search information in a more efficient way which allowed me to become more autonomous.

It was also an incredible way to improve my English orals and written skills thanks to a complete immersion.

Completing this project was quite a challenge because it wasn't really my field of working but in the end, it was a real pleasure to achieve to connect the devices between each other's. This whole work allowed me to rediscover the Internet of Things although I had worked on it in my past but that wasn't as interesting as there.
I now have a more complete vision of the thing and even started considering the idea of continuing my studies on that domain.

5 Recommendations

I would like to express my sincere gratitude to Zeeshan Shakir who supervised me, giving me instructions and guideline all along the realization of this project. But also, to Shahriar Al-Ahmed who helped me to solve a lot of the problems I was facing and who totally manage the setting up of the Wi-Fi in the laboratory.

I would also like to thanks Duncan Thomson and my English teacher Corinne Houssain who both offered me this amazing opportunity to learn and to travel in a foreign country.

And finally, I would like to tanks the whole University of The West of Scotland for accepting me as a one of their trainees.

6 Glossary

Shield, a shield or a board is a modular circuit that you put onto your device to install it with extra functionality.

Cloud, a technology that allow you to store your data on the internet rather than on the hard drive of your computer.

Wi-Fi, a wireless network technology based on the norm IEEE 802.11. It allows to connect by radio wave different informatic devices in the same network and to send data between them.

LoRa, or **LoRaWAN** (Long Range Wide Area Network) is a low-cost telecommunication protocol using gateways and which is used frequently in IoT solutions.

Deep sleep, a mode on the Pysense that allow you to reduce the power consumption.

LiPo, (Lithium Polymer battery), mean a rechargeable battery that use polymer electrolyte.

BLE, (Bluetooth Low Energy), a variant of the Bluetooth technology that is cheaper and that consume less than Bluetooth keeping the same range of communication.

SigFox, a long range and low debit French network for little transmissions.

LTE-M, a type of low power wide area network radio technology.

SSID, (Service Set Identifier), the name of a wireless network under the norm IEEE 802.11

WLAN, (Wireless Local Area Network), a local network where the devices don’t need any kind of wires to be connected.

7 Bibliography

https://www.siuk-saudi.com/en/profiles/university/west-scotland/

https://en.wikipedia.org/wiki/Wi-Fi

https://github.com/pycom/pycom-libraries

https://forum.pycom.io/topic/1970/pysense-pressure-altitude-odd-readings

https://forum.pycom.io/topic/2738/pysense-nonsense-altitude-readings

https://developers.wia.io/docs/pycom-pysense

https://docs.pycom.io/pytrackpysense/installation/firmware.html

8 Appendix

Appendix A

```
from network import WLAN
from pysense import Pysense
from MPL3115A2 import MPL3115A2,ALTITUDE,PRESSURE
from SI7006A20 import SI7006A20
from LTR329ALS01 import LTR329ALS01
import urequests as requests
import socket
import time



py = Pysense()

si = SI7006A20(py)
mpa = MPL3115A2(py,mode=ALTITUDE)
mpp = MPL3115A2(py,mode=PRESSURE)
lt = LTR329ALS01(py)

# Wifi credentials
WIFI_SSID = 'IoT-24'
WIFI_KEY = 'IoTlabUWS'

# Wia secret key to protect the device from any intrusion
DEVICE_SECRET_KEY = 'd_sk_19aMFM2qLo5tcvBFufci3m3b'

url = "https://api.wia.io/v1/events"
headers = { "Authorization": "Bearer " + DEVICE_SECRET_KEY, "Content-Type":
"application/json" }

wlan = WLAN(mode=WLAN.STA)
nets = wlan.scan()

# Connect to the WiFi network
for net in nets:
    if net.ssid == WIFI_SSID:
```

```
        print('Network found!')
        wlan.connect(net.ssid, auth=(net.sec, WIFI_KEY), timeout=5000)
        print('Connecting...')
        while not wlan.isconnected():
            machine.idle() # save power while waiting
        print('WLAN connection succeeded!')
        break

# Create an Event for Wia
def post_event(name, data):
    try:
        json_data = { "name": name, "data": data }
        print(str(json_data))
        if json_data is not None:
            req = requests.post(url=url, headers=headers, json=json_data)
            return req.json()
        else:
            pass
    except:
        pass

# Main loop that will post the data on Wia
while True:
    post_event("temperature", si.temperature())
    post_event("pressure", mpp.pressure())
    post_event("altitude", mpa.altitude())
    post_event("humidity", si.humidity())
    post_event("light", lt.light())
    time.sleep(10)
```

Appendix B

```
from machine import UART
import machine
import os

uart = UART(0, baudrate=115200)
os.dupterm(uart)

machine.main('main.py')
```

Appendix C

Appendix D