Open Innovation Platform for IoT-Big data in Sub-Sahara Africa

Grant Agreement Nº 687607

Agriculture MVP

Low-Cost Weather station with Lora

Responsible Editor:	Unparallel Innovation, Lda
Contributors:	
Document Reference:	Agriculture MVP: Low-Cost Weather station with Lora
Distribution:	Public
Version:	1.0
Date:	January, 17th , 2017

Table of Contents

1. Introduction 4
2. Sensor node Main Architecture 5
   2.1. Core System controller 5
   2.2. Data Acquisition 6
   2.3. Data Communication 6
   2.4. Particular Specifications 11
3. Adafruit Feather 12
4. Weather Shield 14
   4.1. Wind and Rain Sensors Kit 15
5. Lora Communication 18
6. Build & Implementation 19
7. Testing & Implementation Results 26
   7.1. System Running 26
   7.2. Test Without LowPower 26
   7.3. Test With LowPower 28
8. LoRa Gateway Modifications 30

LIST OF FIGURES

Figure 1 - Overview of the overall solution for the Agriculture MVP 5
Figure 2 – Represented the Freeboard Cloud platform. 8
Figure 3 – Represented the Freeboard Cloud platform. 9
Figure 4 – Architecture model with the set specifications. 10
Figure 5 – Operation flow in LoRa communication protocol. 11
Figure 6 – Adafruit Feather 32u4. 12
Figure 7 – Adafruit Feather 32u4 with RFM9x LoRa module. 13
Figure 8 – Sparkfun Weather Shield. 14
Figure 9 – Sparkfun Weather Shield's schematic. 15
Figure 10 – Wind and Rain Sensors Kit. 17
Figure 11 – Schematic with the Pin's connection between Feather and Weather Shield. 20
Figure 12 – Some connection's pictures. 20
Figure 13 – Wind and Rain Sensors. 21
Figure 14 – 868 MHz Antenna. 21
Figure 15 – Connecting the 868 MHz Antenna to Feather. 22
Figure 16 – Implemented Weather Station. 22
Figure 17 – Step to include the Libraries. 23
Figure 18 – Steps to include the Libraries. 23
Figure 19 – Steps to Verify and Upload the software. 24
Figure 20 – Serial Monitor results. 24
Figure 21 – Weather Station test without LowPower during 4 minutes. 27
Figure 22 – Weather Station test without LowPower during 15 minutes. 27
Figure 23 – Weather Station test with LowPower during 4 minutes. 28
Figure 24 – Weather Station test with LowPower during 15 minutes. 29

LIST OF TABLES

Table 1 – Communication packages. 7
Table 2 - Feather 32u4's Specifications. 12
Table 3 - Weather shield measurements specifications. 14
Table 4 - ADC arrangement for 5V and 3.3V. 16
Table 5 - Pin's Connection. 19

1. INTRODUCTION

This document aims to describe and detail the development of a low cost weather station, based on the Adafruit Feather board. This board communicates with any LoRa intermediate gateway node. It describes the creation of a vigilant weather supervision, composed by multi-tech sensors as part of the IoT sensor node with LoRa network integration. This work was done in the context of WAZIUP Research Project, which has received funding from the European Union's H2020 Programme for research, technological development and demonstration under grant agreement No 687607.

With expert level architecture, the challenge here is to be able to gather data from the surrounding area and then make it available to the WAZIUP cloud. The transmission is carried out by the LoRa bi-directional communication, representing a low cost technologic long range communication. Also challenging is to provide a final sensor node as a light and scalable solution with low energy dependencies.

Nowadays there are many advanced and well-integrated weather stations. However, they were discarded for use in WAZIUP due to the need of having an IoT solution that could be qualified as "low-cost" and with built-in LoRa radio. So, and in order to decrease costs, we decided to use several low-cost components and integrate them to provide a WAZIUP Weather Station node. This node, was developed using the requirements received from the Project Use cases.

The last chapter also presents the modifications done to the WAZIUP LoRa Gateway[1], developed also within the project's scope, to enable the communication between the Gateway and two cloud repository services, such as the actual WAZIUP Platform, and a ThingsSpeak channel created for this Weather data.

This document provides a step-by-step tutorial, a simple user guide, on how built the WAZIUP Weather Station). The code used for this weather station is open source and hosted in the GitHub repository[2].

[1] https://github.com/CongducPham/LowCostLoRaGw

[2] https://github.com/unparallel-innovation/UI_Waziup_Weather_Station

2. SENSOR NODE MAIN ARCHITECTURE

The objective is to develop a low-cost and sustainable solution capable of reading real-time data typical of a weather station, using different sensors, and capable of communicating via LoRa. After a review of all known hardware available on the market, all the components strictly necessary to the solution were defined, which in turn fulfilled the requirements of the above: The hardware chosen was:

- Adafruit Feather 32u4 RFM95 LoRa Radio with female pin headers[3]
- Sparkfun Weather Shield with RJ11 female connectors[4]
- Wind and rain sensors kit[5]
- Antenna 868 Mhz and SMA cable

In , it's depicted the working model of the data acquisition and supervision system, constituted by the previously mentioned hardware, together with software capable of monitoring and supervising the sensory variables and the devices.

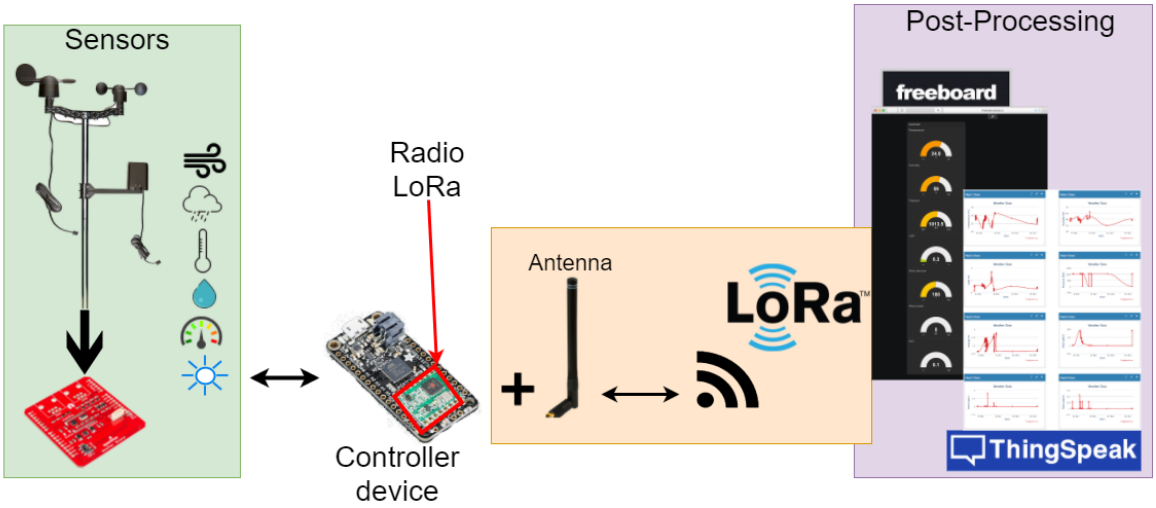

Figure 1 - Overview of the overall solution for the Agriculture MVP

2.1. Core System controller

In terms of the core system within the Weather Station solution, it's composed by the Feather32u4, which that takes a specialized role in the system where it performs control functions through software, with processing power, enabling the sensory devices to gather data from the environment, using specific libraries. This system also has built-in communication capabilities.

[3] https://www.adafruit.com/product/3078
[4] https://www.sparkfun.com/products/12081
[5] https://www.sparkfun.com/products/8942

2.2. Data Acquisition

The weather shield is an integrated module with several built-in sensors capable of collecting data, such as temperature, humidity, luminosity, barometric pressure and altitude. Along these sensors, the weather shield also enables the integration of three more different sensors to collect data regarding wind direction, wind speed and amount of rain.

Based on the proposed model, it becomes clear the connection between the controller and the Weather Shield. This connection is established with the I2C protocol that allow this digital integrated circuit to communicate with one or more masters. It's used this type of protocol because it's only intended short distance communications within a single device and only requires two signals to exchange information.

The software controller uses the library "Wire.h" that is dedicated to the I2C logic protocol. The embedded software requires the “WeatherConfig.h” library in order to call all the functions responsible for activating and reading the sensors signals coupled to the weather shield. This library also contains some logical functions dedicated to executing the desired specifications.

Until now, the communication system build between Adafruit Feather (ATmega 32u4) and all the sensors who take measurements (include WeatherStation library) working 100%.

2.3. Data Communication

Like as expected in the proposed system model, the controller will send data to the outstation, based on the information collected from the weather shield module. For this it makes use of the "SPI.h" library to run the communication with the radio module RFM9x LoRa 868/915. The LoRa radio must communicate with the LoRa gateway, specified by the system, and for that will interact with the "featherLora.h" library. This library was created because of the Adafruit Feather’s specifications and was originally developed by Congduc.

The data will be collected according to the time windows described, already considered in the project. At the end of each time window will be sent the package with the message containing the information collected.

In data sharing with outstation it was established to send an acknowledge information packet like a result of the incoming data from the different sensors. The typical message to be sent from one gateway to another is based on the type message as described in the following example (working 100%):

Example of the message send in the package:

\\!TC/18.6/HU/85/LU/56/DO/7.7/AZO/87/WD/90/WC/5.55/RC/0.55

The Table 1 outlines the type and content of the information sent in each package:

Table 1 - Communication package.

Communication package	
TC/18.6/	Temperature
HU/85/	Humidity
LU/56/	Luminosity
DO/7.7/	Pressor
AZO/87/	Supply
WD/90/	Wind direction
WC/5.55/	Wind Speed
RC/0.55/	Amount of rain

In that way, the LoRa gateway receivers this message by post-processing python scripts to be able to correctly and sequentially inspect all the information, in order to post in interface platform the values to be appreciated. Off course some parts need your own customization for cloud access.

The following example shows the result, for a short period of time, from weather station in the Cloud platform:

Figure 2 – Represented the Freeboard Cloud platform.[6]

[6] https://thingspeak.com/channels/184796

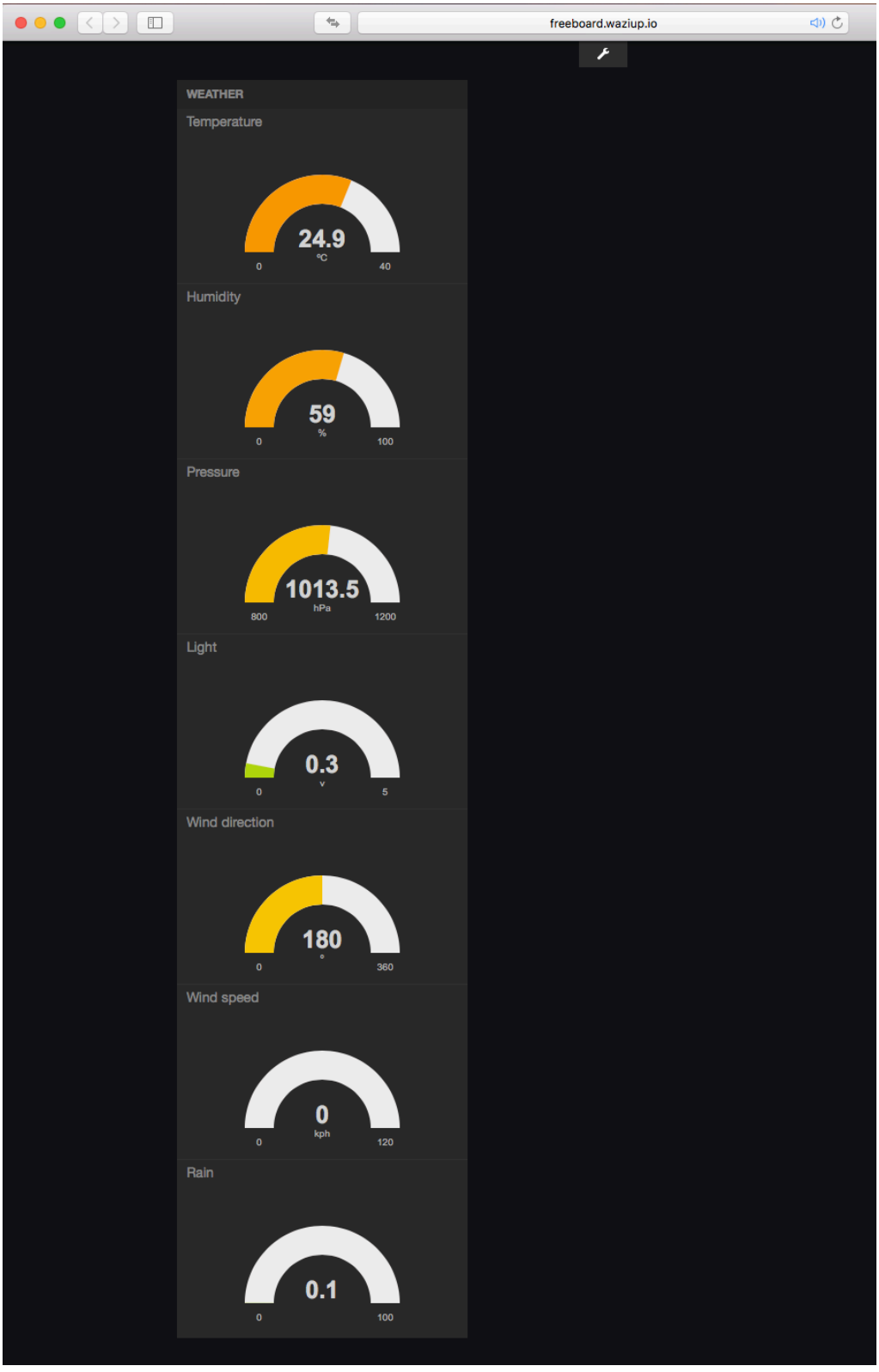

Figure 3 – Represented the Freeboard Cloud platform.[7]

[7] http://freeboard.waziup.io/index.html#source=http://thingproxy.freeboard.io/fetch/https://www.dropbox.com/s/ucx8p4wwktqiu43/UI_WEATHER.json?dl=1

So far, the LoRa communication between Feather (with RF95 integrated) and Raspberry PI (integrated with Hope RFM9X LoRa Radio) with the “FeatherLib” library works with a success of 100%. To develop and building LoRa end-devices you can follow Congduc’s tutorials.[8] [9]

The Figure 4 shows the architecture of proposed model and it’s operating flow with the set specifications:

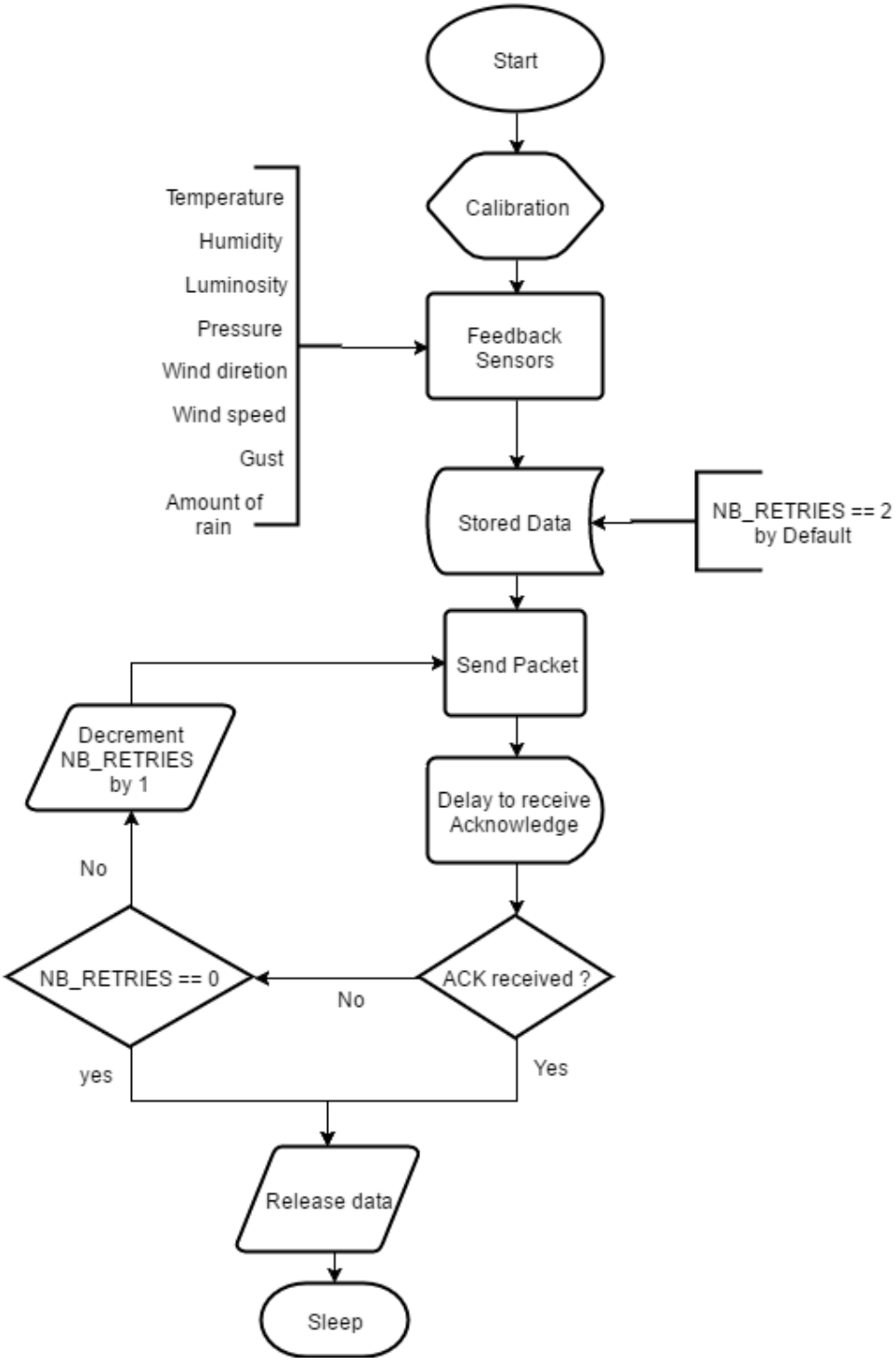

Figure 4 – Architecture model with the set specifications.

[8] http://cpham.perso.univ-pau.fr/LORA/LoRaDevices.html
[9] https://github.com/CongducPham/tutorials

It is important to understand how the LoRa communication protocol is constituted. It starts by sending a transmission with the master. Next, an acknowledge is expected. We can choose how many times the Feather tries to receive the ACK by the gateway (NB_RETRIES=2 by default). Only then the Slave fall asleep.

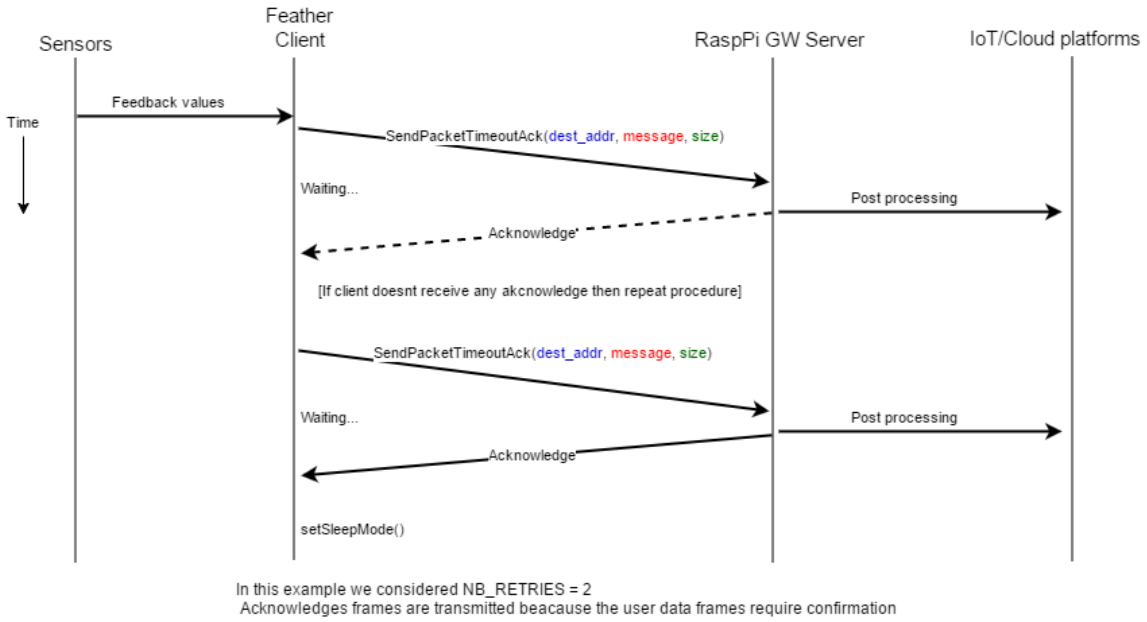

Figure 5 – Operation flow in LoRa communication protocol.

2.4. Particular Specifications

After some analysis about the requirements and implications to be had in this system, some reserved variables containing useful parameters for the developed model were verified.

- We can choose how long the feather sleep between two communications (four minutes by default).
- We can choose to show in console the information about what is happening on communication in real time (warning: increase high load memory).
- When somehow the wires in the circuit were built wrong it returns to warning with a message "I2C communication to sensors is not working. Check solder connections "and red led from Feather device start blink.

3. ADAFRUIT FEATHER

The Figure 6 shows the Adafruit Feather 32u4 LoRa Radio (RFM9x) that is a development board from Adafruit. It's a portable microcontroller with new standards, great for communications in networks without high power requirements of WI-FI, it has a "Long Range" (LoRa) packet radio transceiver built in USB and battery charging.

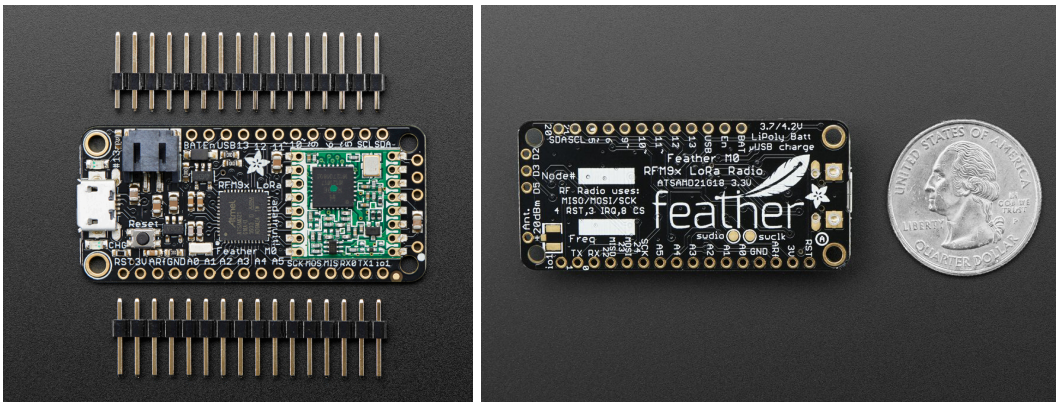

Figure 6 – Adafruit Feather 32u4.[10]

The 900 MHz radio version can be used for either 868MHz or 915MHz transmission/reception. The exact Radio Frequency is determined when you load the software. The Feather 32u4 have a ATmega32u4 processor clocked at 8 MHz with a 3.3V logic, 32K of flash and 2K of RAM, with built in USB. It have a USB-to-Serial program & debug capability built in with no need for an FTDI-like chip.

To make it easy to use for portable projects was added a connector for any of 3.7V Lithium polymer batteries and built in battery charging. Don't need a battery because it will run just fine straight from the micro USB connector. With a battery we can take it on the go, then plug in the USB to recharge. The Feather will automatically switch over to USB power when its available. We also tied the battery thru a divider to an Analog Pin, so we can measure and monitor the battery voltage to detect when you need a recharge. In the Table you can find the Feather 32u4's specifications:

Table 2 - Feather 32u4's Specifications.

Feather 32u4's Specifications	
Measures	51mm x 23mm x 8mm
Weight	5.5 grams

[10] https://www.adafruit.com/product/3078

Microcontrollers	ATmega32u4 @ 8MHz with 3.3V logic/power
Supply	3.3V regulator with 500mA peak current output
USB support	Comes with USB bootloader and serial port debugging
GPIO pins	x20
Hardware	Serial support, I2C support, SPI support
PWM pins	x80
Analog inputs	x10
Reset button	Yes

This Feather uses a RFM9x LoRa 868/915 MHz radio module. These radios aren't good for transmitting audio or video, but they do work quite well for small data packet transmission when you need more range than 2.4 GHz. That provides ultra-long range spread spectrum communication and high interference immunity whilst minimising current consumption.

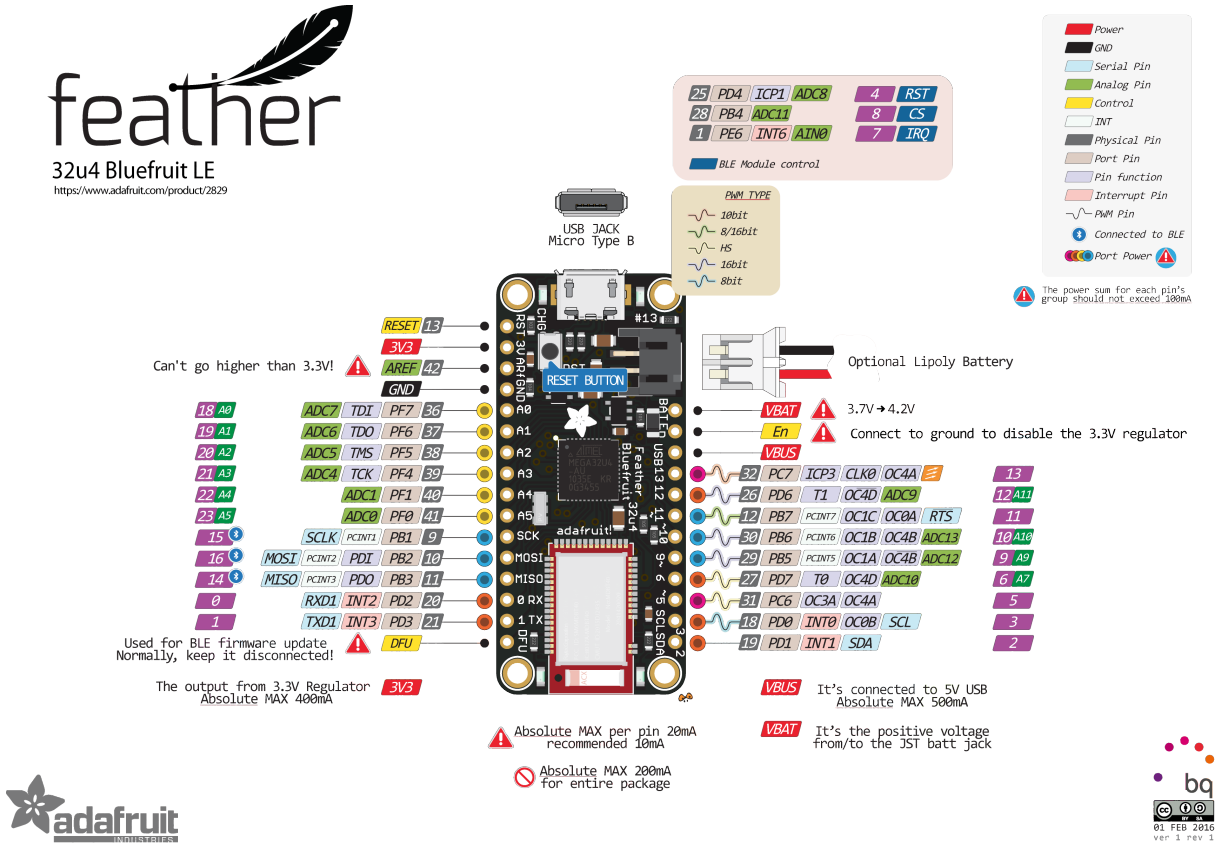

Figure 7 – Adafruit Feather 32u4 with RFM9x LoRa module.[11]

[11] https://www.adafruit.com/product/3078

4. WEATHER SHIELD

The Weather Shield, in Figure 8, was designed by the SparkFun. Is an easy to use Arduino shield that grants you access to temperature, luminosity, barometric pressure, altitude and humidity. There are also connections on this shield to sensors as wind speed, direction, rain gauge and GPS for location and super accurate timing.

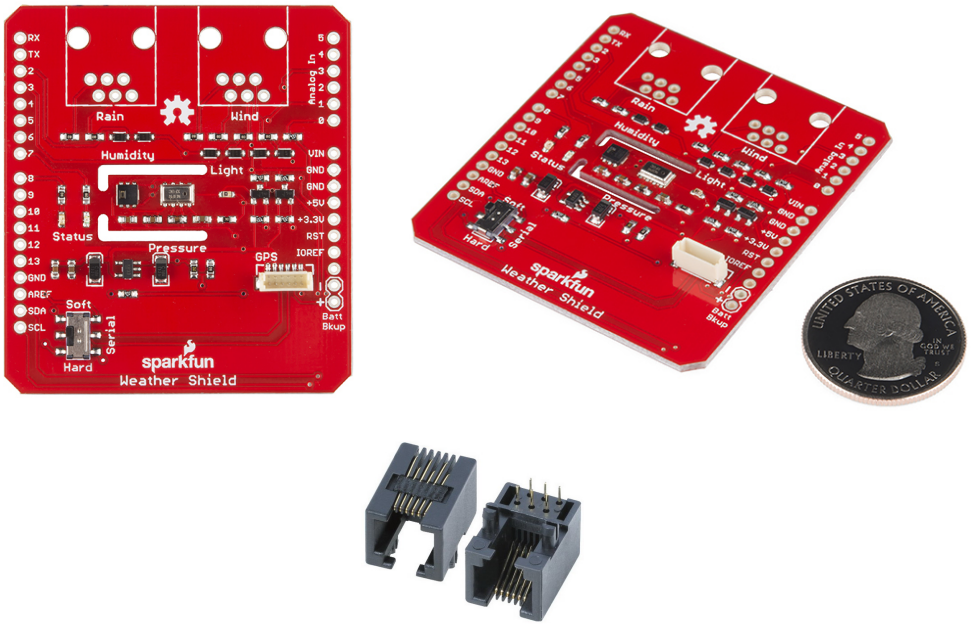

Figure 8 – Sparkfun Weather Shield.[12]

These Weather Shields utilize the “HTU21D Humidity”, “MPL3115A2 Barometric Pressure”, and “ALS-PT19” light sensors and relies on the “HTU21D” and “MPL3115A2” Arduino libraries. Each shield comes with two unpopulated RJ11 connector spaces (for hook up of rain and wind sensors). Finally, it can operate from 3.3V up to 16V and has built in voltage regulators and signal translators.

The Table 1 shows the measurements specifications:

Table 1 - Weather shield measurements specifications.

Measurements Specifications	
Temperature	Return a float containing the temperature in degrees [-40ºC; +125ºC] and maximum error of +/-0.4ºC.
Humidity	Return a float containing the humidity as a percentage with maximum error of +/-3%.

[12] https://www.sparkfun.com/products/12081

Pressure	Return a float with barometric pressure in hPa (1 hPa = 100 Pa) with maximum error of +/-0,5 hPa.
Altitude	Not used in this project.
Light	Return a float with range between 0 and 3.

In Figure 9 you can consult the Sparkfun Weather Shield's schematic:

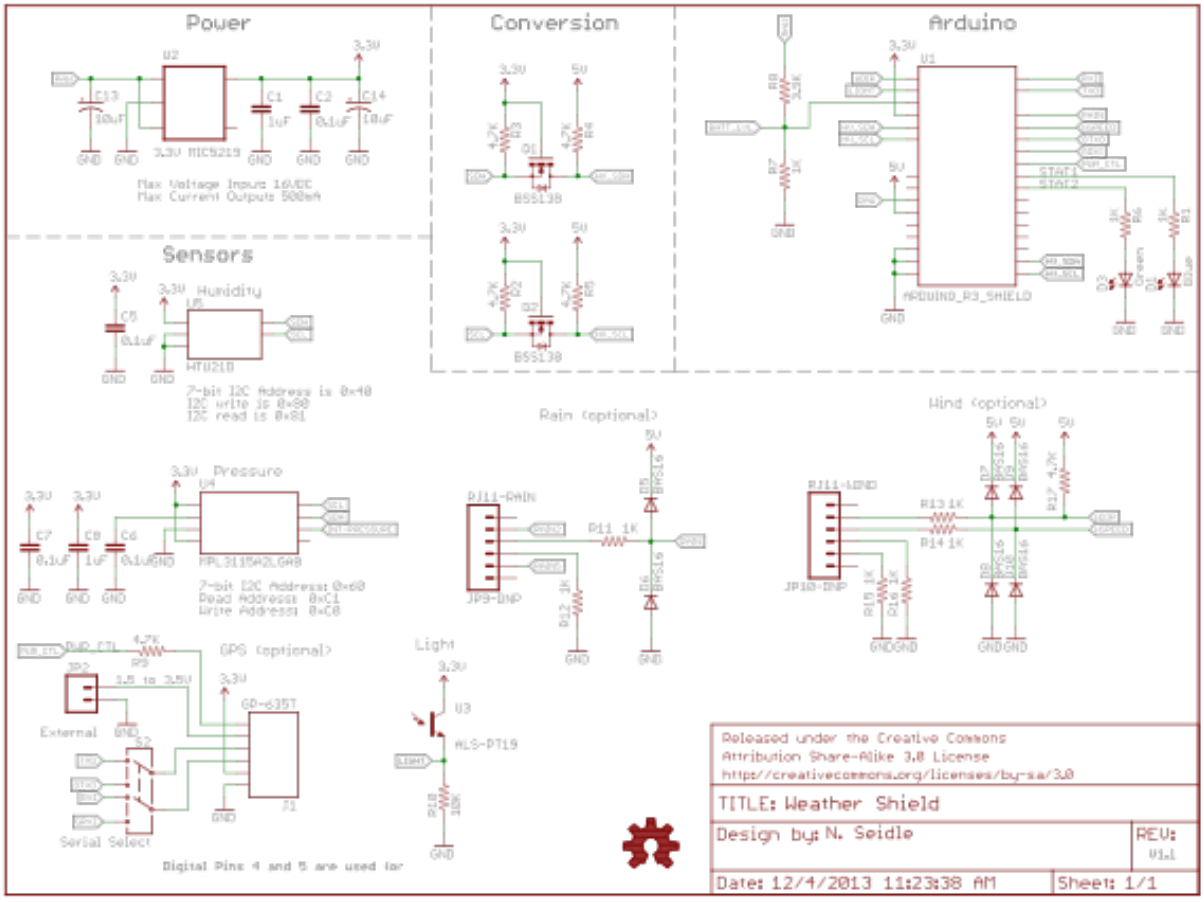

Figure 9 – Sparkfun Weather Shield's schematic.

4.1. Wind and Rain Sensors Kit

This kit from Argent Data System represents the three core components of weather measurement: wind speed, wind direction and rainfall. None of the sensors in this kit contain active electronics, instead they use sealed magnetic reed switches and magnets so we'll need to source a voltage to take any measurements.

The positive side of this is that the sensors are easy to interpret. The rain gauge is a self-emptying bucket-type rain gauge which activates a momentary button closure for each 0.011" of rain that are collected. The anemometer (wind speed meter) encodes the wind speed by simply closing a switch which each rotation. A wind speed of 1.492 MPH produces a switch closure once per second.

Finally, the wind vane reports wind direction as a voltage which is produced by the combination of resistors inside the sensor. The vane's magnet may close two switches at once, allowing up to 16 different positions to be indicated.

It's necessary to pay attention to the measurements specifications of wind, gust wind and rain shown below:

- **Wind**: The sensor wind works with interrupts and return values of direction and speed in kilometres per hour every five minutes (by default). The cup-type anemometer measures wind speed.
  The wind vane has eight switches, each connected to a different resistor to measures the direction.
  The ADC arrangement is shown in the Table 2. Cause the circuit was built for 5V supply we needed to rewrite arrangement values.

Table 2 - ADC arrangement for 5V and 3.3V.

	Angle (º)	ADC with 5Volts (Old Version)	ADC with 3.3Volts (Our version)
North	0	913	747
	22.5	680	483
	45	746	522
	67.5	393	252
West	90	414	258
	112.5	380	239
	135	508	325
	157.5	456	283
South	180	615	399
	202.5	551	368
	225	833	638
	247.5	801	617
East	270	990	873
	292.5	940	776
	315	967	821
	337.5	878	689

- **Gust**: This parameter starts from sensor wind who return maximum values of wind in kilometres per hour every five minutes.

- **Rain**: This sensor rain works with interrupts and return values in inches every five minutes. The rain gauge is a self-emptying tipping bucket type. Each 0.011" (0.2794 mm) of rain causes one momentary contact closure that can be recorded with a digital counter or microcontroller interrupt input. Cause the circuit was built for 5V supply we needed to force the Pull Up.

Figure 10 – Wind and Rain Sensors Kit.

5. LORA COMMUNICATION

LoRa RF it's a radio modulation format that gives significantly longer range than straight FSK modulation, a technique that compete with others technologies, specific intended for wireless battery operated Things.

Lora significantly improves the receiver and as with other spread-spectrum modulation techniques uses the entire channel bandwidth to broadcast the signal, making it robust to channel noise and insensitive to frequency offsets caused from the use of low cost crystals. The selection of the data rate is a trade-off between communication range and message duration.

The LoRa gateways are designed to use in long range start network architectures and are utilized in a LoRaWAN system. The gateways use different RF components than the end-point to enable high capacity and serve as a transparent bridge relaying messages between end-devices and a central network server in the backend.

The Gateways are connected to the network server via standard IP connections while the end-devices use single-hop wireless communication to one or many gateways.

The LoRa modulation is defined by the following basic parameters:

- The Bandwidth (BW), meaning the difference in minimum and maximum frequency;
- The Spreading Factor (SF), is a measure for the number of bits encoded per symbol;
- The Coding Rate (CR), is a measure for the amount of forward error correction;
- Preamble Length and SyncWord value;
- Whether an explicit header is sent with the message, this header contains information about the parameter of the rest of the message (payload length, the CR parameter, CRC presence);
- Presence of a 16-bit CRC and;
- The maximum packet length is 256 bytes in LoRa mode.

Instead of using a Received Signal Strength Indicator (RSSI) method to identify if a signal is present the Channel Activity Detector (CAD) is used to detect the presence of a LoRa signal.

The CAD detection has the advantage of being much faster than a typical RSSI detection and the capability to differentiate between noise and a desired LoRa signal. The CAD process requires two symbols, and if the CAD is detected, the "CAD_Detected" interrupt will be asserted and the device will stay in RX mode to receive the data payload.

Regarding software, we use "FeatherLib" library that is the base of the Congduc's library to support this kind of communication. This library has some specific features to the hardware used in this case, and uses a modified and improved version of the "SX1272" library originally developed by Libelium for its SX1272 based LoRa module.

6. BUILD & IMPLEMENTATION

Here we will show you step by step how to build a Low Cost Weather Station with LoRa communication to collect data and send them to gateway. First of all, you need to assembling the hardware, and then programming the software and tested it.

1) **Connecting the Adafruit Feather 32u4 with Sparkfun Weather Shield**:

Try using different colours when connecting the circuits. You can use the following colours, like the next table. You have to go through some delicate but simple soldering task to attach female header pins to both devices. It's not difficult, but you have to train a little before.

Table 3 - Pin's Connection.

Pin's Connection	
Feather 32u4	**Weather Shield**
Vin	3V
GND	GND
SCL (3)	SCL
SDA (2)	SDA
12	7
13	8
Int2 (0) in Vin with a 10KΩ resistor	-
Int2 (0)	2
Int3 (1)	3
A0	A0
A1	A1
A2	A2
A3	A3

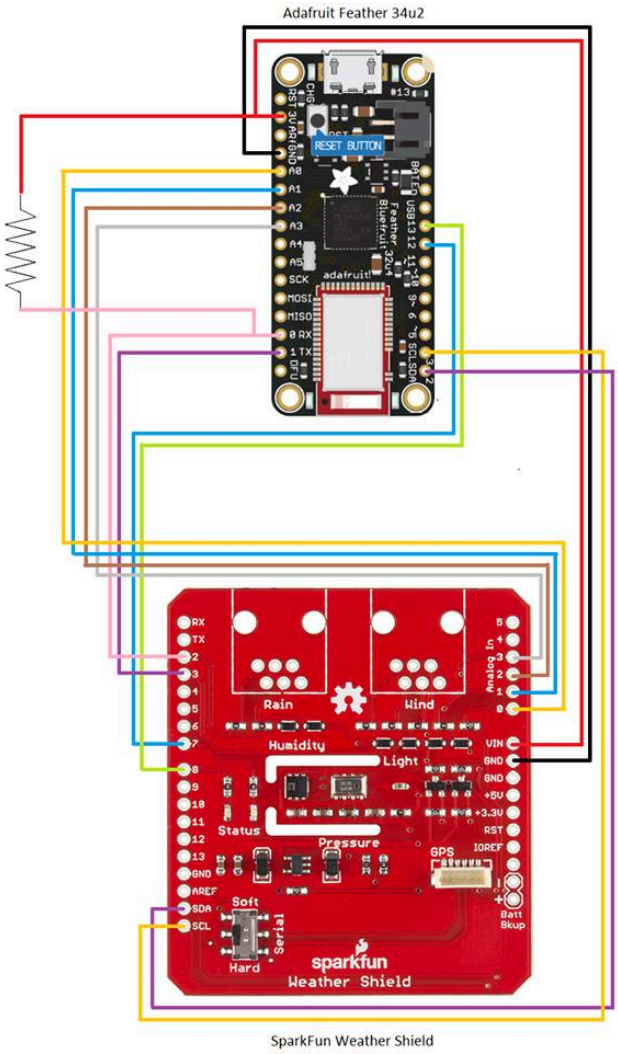

Figure 11 – Schematic with the Pin's connection between Feather and Weather Shield.

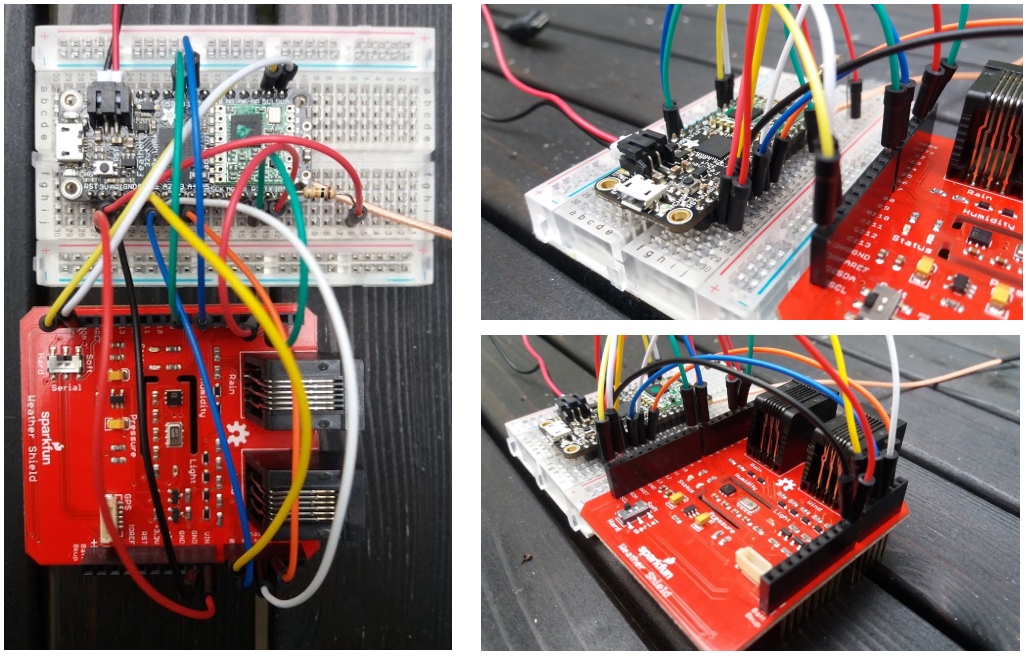

Figure 12 – Some connection's pictures.

2) **Connecting the other sensors**:

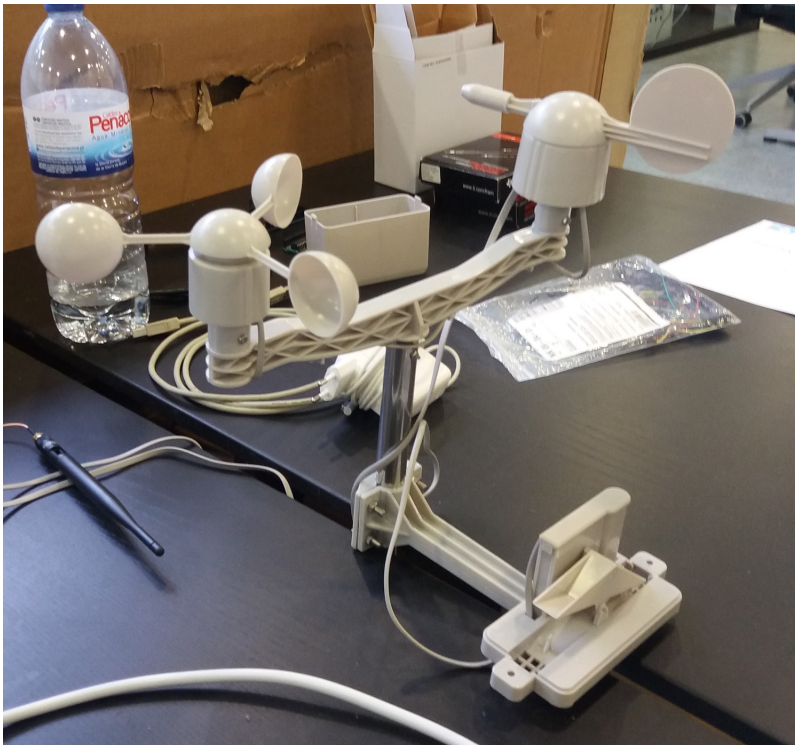

Figure 13 – Wind and Rain Sensors.

3) **Connecting the Antenna 868 MHz in Feather**:

The Feather 32u4 comes fully assembled radio and tested. You will need to cut and solder the SMA plug cable in order to create your antenna.

Figure 14 – 868 MHz Antenna.

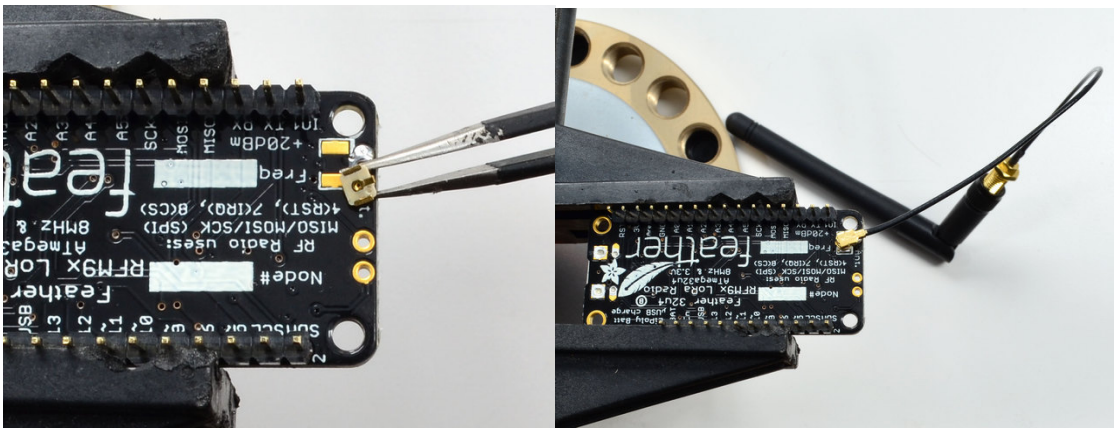

Figure 15 – Connecting the 868 MHz Antenna to Feather.

4) **Weather Station built with off-the-shelves components, so far**:

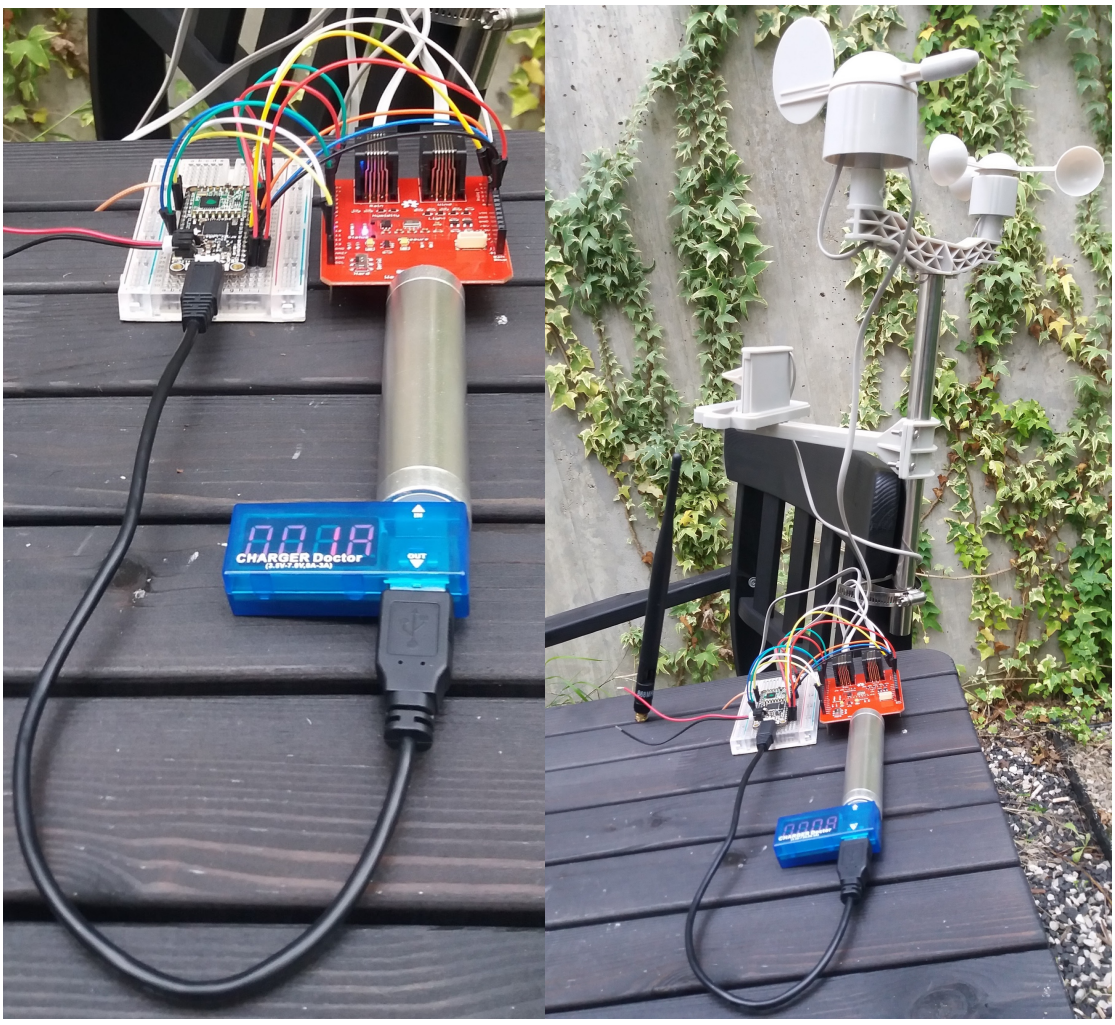

Figure 16 – Implemented Weather Station.

5) **Download the IDE and specific software**:

First, you will need to download the Arduino IDE 1.6.6 or later. Using the Library Manager from Arduino IDE to include 'SPI.h' and 'Wire.h' for protocol communication. Found in the "Sketch" menu under 'Include Library', 'Manager Libraries...'.

When you open the 'Include Library' or even 'Library Manager' you will find a large list of Libraries ready for one-click install call. Click in that button, and the library should install automatically. When installation finishes, close the Library Manager.

Next, you'll need to get "FeatherLib" and "WeatherStation" libraries from our Github.[13] The "FeatherLib" library includes LoRa Communication and was built because of its specification. The "WeatherConfig" library includes all necessary sensor control functions. For these libraries you will need to install by 'add .ZIP Library ...

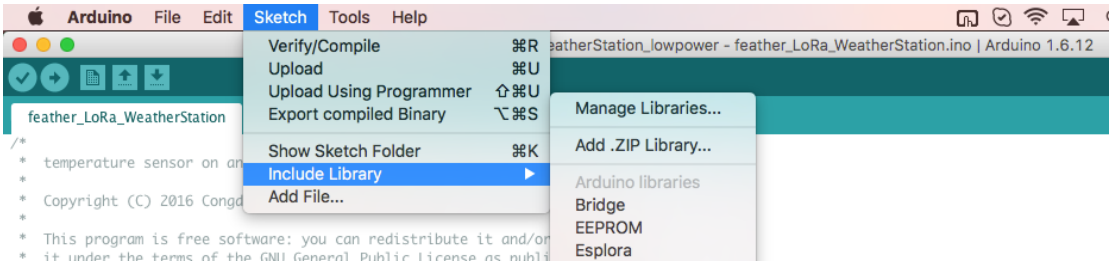

Figure 17 – Step to include the Libraries.

6) **Compiling the software**:

Write a script that include both of this libraries and with "SPI" and "I2C" libraries. In Github you will find an example of this project.

Copy the "Feather_Lora_WeatherStation.ino" file into your "sketch" folder. This archive contains the latest sketch (firmware) that the USB Feather board uses to sample the sensors and output data with Lora communication.

You can use this sketch as is, modify it to suit your own purposes, or write your own sketches. Connect the USB end to your computer and the USB port should be detected in the Arduino IDE. For that follow the same steps of the Figure 18:

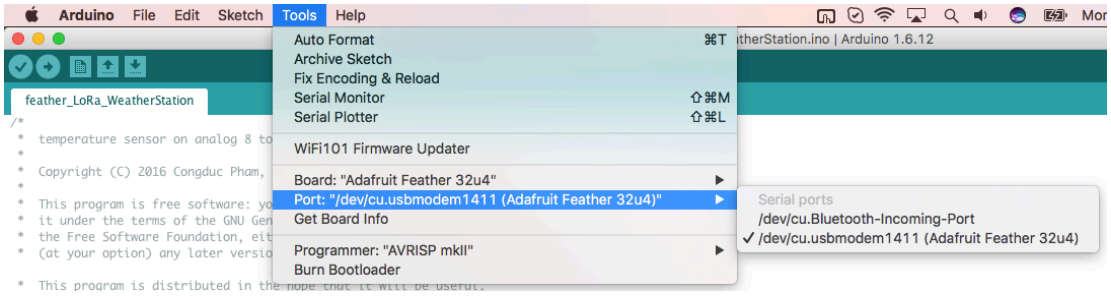

Figure 18 – Steps to include the Libraries.

13 https://github.com/bmpalmeida/UI_Waziup_Weather_Station/tree/master/weather%20station%20client/lib/WeatherStation

Then, click on the «verify» button. After it says “Done compiling” Select the serial port for your device. It may have another name than what is shown in the example. Then click on the «upload» button.

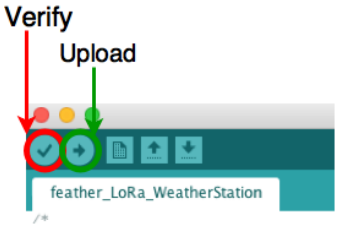

Figure 19 – Steps to Verify and Upload the software.

Then, click on the «verify» button. After it says “Done compiling” Select the serial port for your device. It may have another name than what is shown in the example. Then click on the «upload» button.

7) **Serial Monitor**:

You can see the sensors output and LoRa communication state if you use serial communication. Use the Arduino IDE «serial monitor» to get such output, just to verify that the sensor is running fine. Be sure to use 38400 bauds.

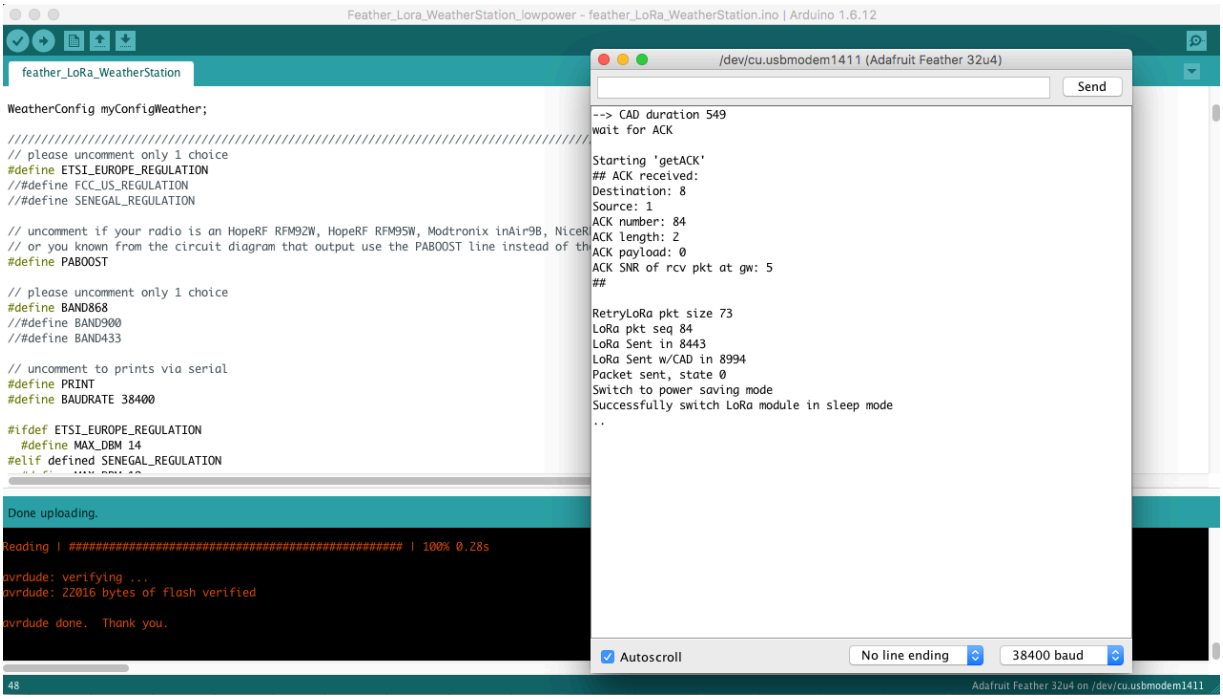

Figure 20 – Serial Monitor results.

Default Configuration:

The following code it's a default configuration. Other types of default configuration, like I/O pins or other defines, steels like the way it is.

```
// please uncomment only 1 choice
#define ETSI_EUROPE_REGULATION
//#define FCC_US_REGULATION
//#define SENEGAL_REGULATION

// uncomment if your radio is an HopeRF RFM92W, HopeRF RFM95W, Modtronix inAir9B, NiceRF1276
// or you know from the circuit diagram that output use the PABOOST line instead of the RFO line
#define PABOOST

// please uncomment only 1 choice
#define BAND868
//#define BAND900
//#define BAND433

// uncomment to prints via serial
//#define PRINT
#define BAUDRATE 38400

// CHANGE HERE THE LORA MODE, NODE ADDRESS
#define LORAMODE  1
#define DEFAULT_DEST_ADDR 1
#define node_addr 8

// CHANGE HERE THE TIME IN MINUTES BETWEEN 2 READING & TRANSMISSION
unsigned int idlePeriodInMin = 4;


#ifdef WITH_APPKEY

// CHANGE HERE THE APPKEY, BUT IF GW
CHECKS FOR APPKEY, MUST BE

 // IN THE APPKEY LIST MAINTAINED BY GW.

uint8_t my_appKey[4]={5, 6, 7, 8};


#ifdef WITH_ACK

#define NB_RETRIES 2
```

7. TESTING & IMPLEMENTATION RESULTS

7.1. System Running

1) Power-on the Feather. The script starts automatically when feather is powered on;

2) At the start and during its automatic sensors calibration process the green led appears on (take about 5 seconds).

3) Wakes-up every 4 min, take all the measurements and send to GW.

4) At the moment of reading values by the sensors and send a message to LoRa gateway the blue led turns on and then:

   I- If feather does not receive acknowledge then the blue led turns off and Feather sleeps.

   II- If feather receive acknowledge from master gateway then the blue led switch to green for 2 seconds, and feathers start sleeping.

5) When Feather sleeps all led's are off.

6) Feather wake up after four minute (by default), take the measurements and send back to master gateway until sleeping again.

7.2. Test Without LowPower

- Sketch "Feather_Lora_WeatherStation.ino";
- 70% of full memory load;
- With radio LoRa Sleep mode policy;
- Without feather low-power policy;
- Wake-up's 4 minutes at a time;
- +14dbm of power;
- 10mA with radio in sleep mode;
- 115mA when is active and sending;
- 20mA when waiting for ACK;
- 10mA average.

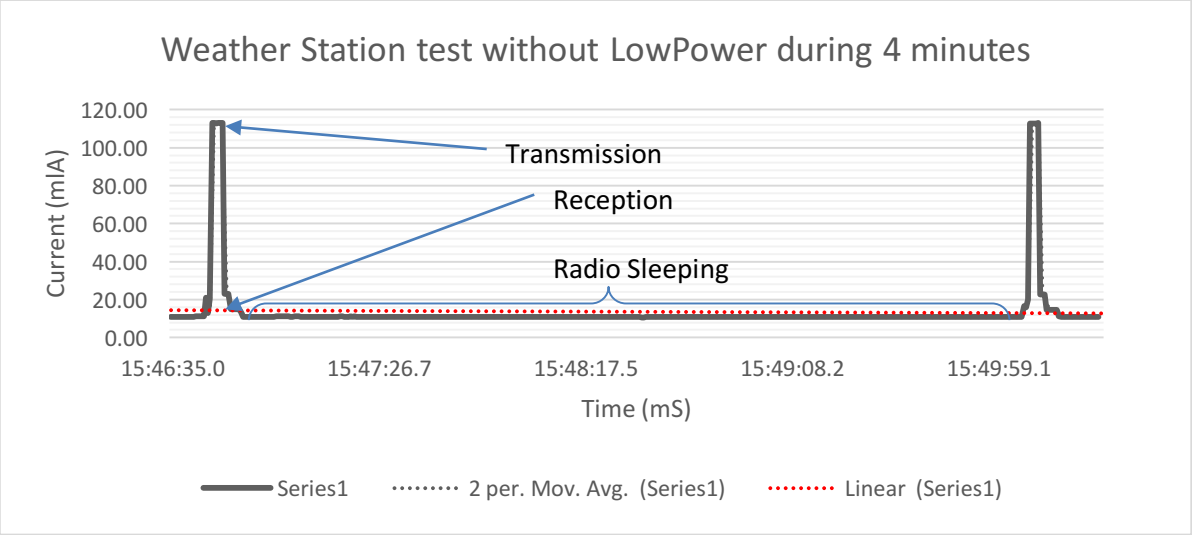

Figure 21 – Weather Station test without LowPower during 4 minutes.

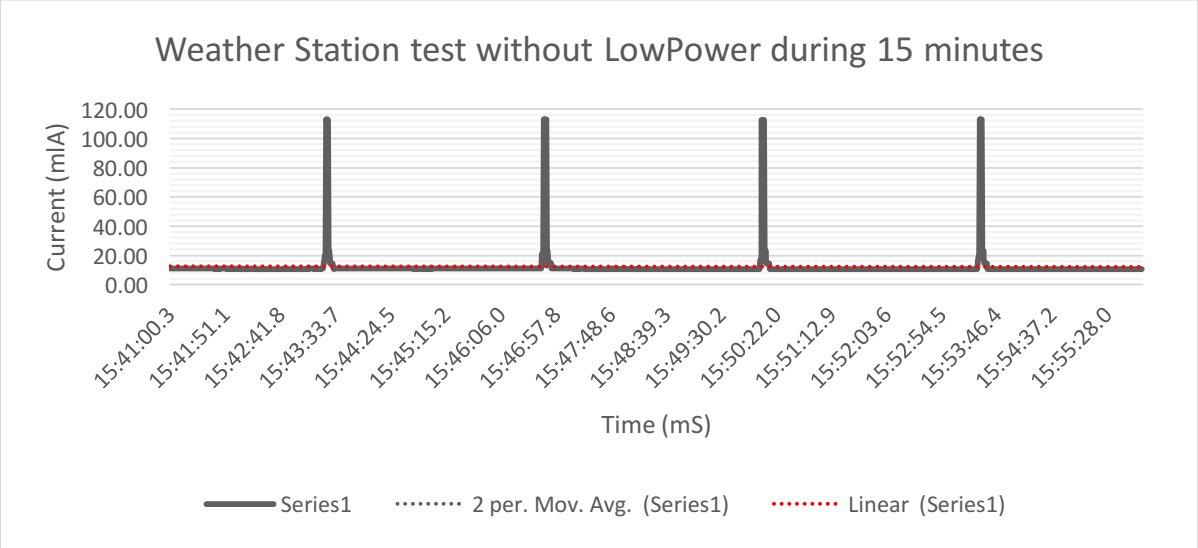

Figure 22 – Weather Station test without LowPower during 15 minutes.

Note, again, that in this case we didn't realize any advanced power saving mechanisms such as putting the micro-controller in deep sleep mode or lower frequency, or performing ADC reduction, just powering off the radio module using your sleep mode.

It's expected that the baseline consumption can be further decreased with more advanced power management policy.

Can run for 1 week with four batteries 2600mAh with this specification.

7.3. Test With LowPower

- Sketch “Feather_Lora_WeatherStation_LowPower.ino”;
- 71% of full memory load;
- With radio LoRa Sleep mode policy;
- Wake-up’s 4 minutes at a time;
- +14dbm of power;
- 1mA with radio in sleep mode and feather too;
- 120mA when is active and sending;
- 20mA when waiting for ACK;
- 1mA average.

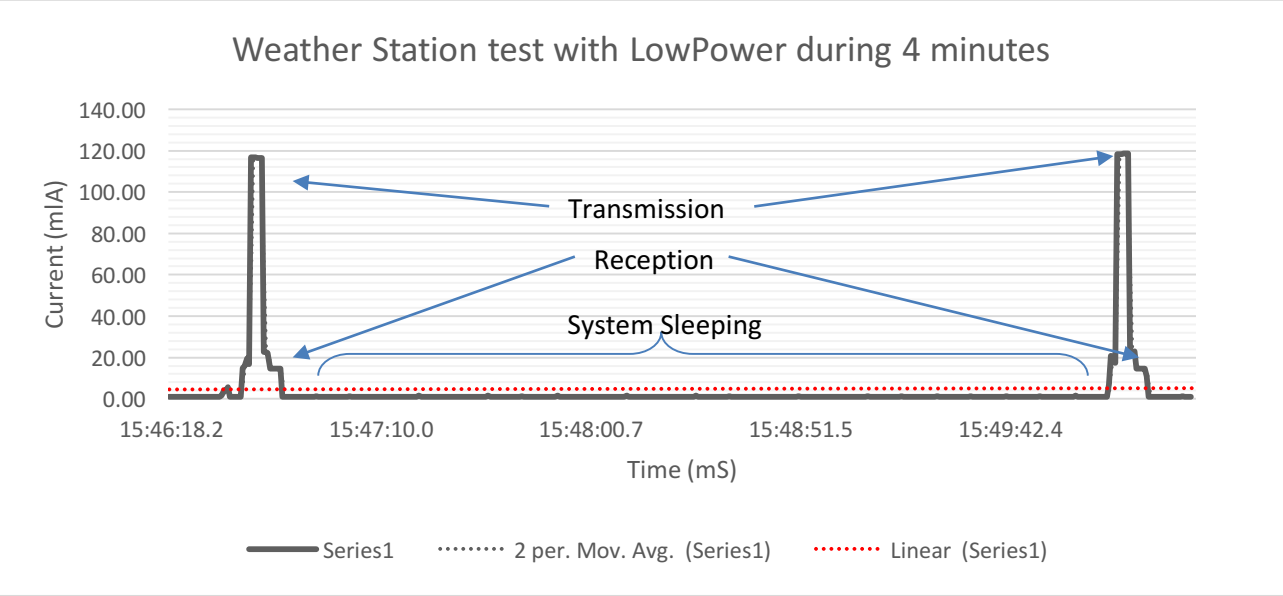

Figure 23 – Weather Station test with LowPower during 4 minutes.

After removing the energy consumed by the feather, we found that an entire cycle for data acquisition, encoding and transmission consumes about 120mA.

The largest consumed energy part on the system comes from weather shield (about 80% of then). The receiving communication actually consumes less than half that amount of energy.

Can run for 10 weeks with four batteries AA 2600mAh with this specification.

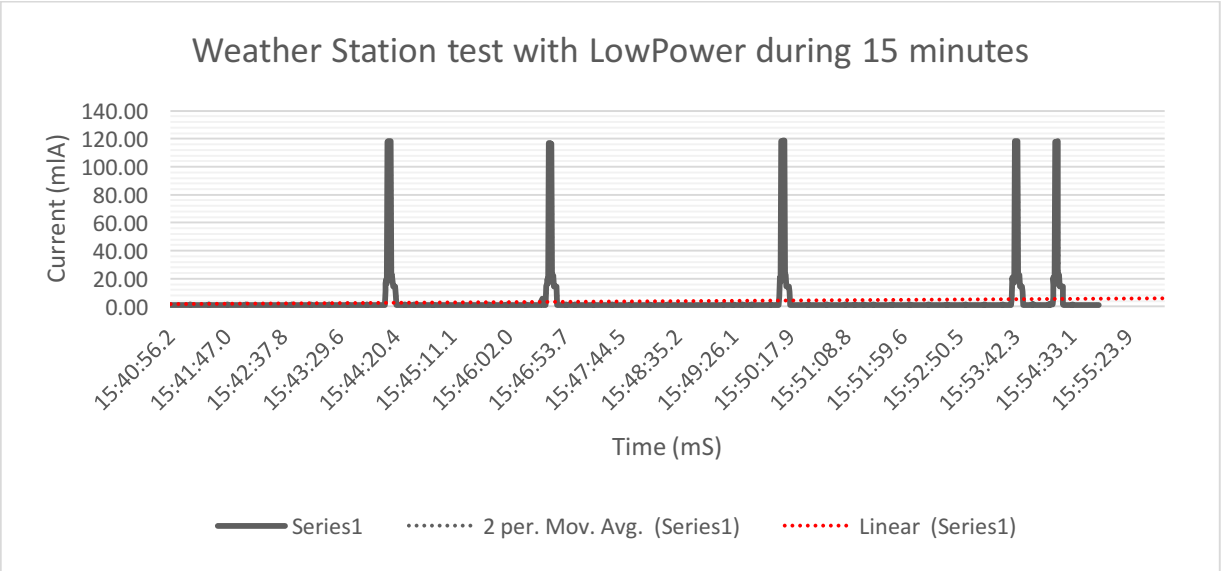

Figure 24 – Weather Station test with LowPower during 15 minutes.

In this case, you can see that the sleeping intervals aren't being respected. This is because the relationship between the sensor interrupts and LowPower hasn't yet been resolved, and the whole system wakes up at the moment an interrupt occurs.

8. LoRa Gateway Modifications

This chapter describes the modifications made to the WAZIUP Low-cost LoRa Gateway, so that the gateway can push the weather data to two Cloud repositories, such as the actual WAZIUP Platform, and a ThingsSpeak channel created for this Weather data. The WAZIUP Gateway was installed by following the tutorials "*newCloudDesign"* available on GitHub[14].

Now the folder "WAZIUP Gateway Modifications" contains files in the format "Cloud(…).py" and each one together with the cloud configuration file ("clouds.json", from the original LoRa Gateway) tells the gateway how to communicate with a specific cloud.

The data structure defined in the context of WAZIUP, and shown here as an example of the Weather Station data: "\!TC/18.6/HU/85/LU/56/DO/7.7/AZO/87,…" is used to represent the data sent from the sensor nodes to the gateway. This represents weather parameters like Atmospheric temperature, Humidity, Barometric pressure, etc. After parsing this information the gateway reads the cloud configuration file ("clouds.json"), and pushes the data the ThingSpeak channel and to the WAZIUP Platform. This, only happens if the user includes the 2 python scripts available in the GitHub repository[15], and adds the following entries to the original "clouds.json":

```
(...)
{
    "name":"WAZIUP Orion cloud",
    "script":"python CloudWAZIUP.py",
    "type":"iotcloud",
    "write_key":"",
    "enabled":true
},
{
    "name":"ThingSpeak cloud",
    "script":"python CloudThingSpeakWeatherStation.py",
    "type":"iotcloud",
    "write_key":"",
    "enabled":true
},
(...)
```

When the WAZIUP Gateway receives weather data from the sensor nodes, it will execute the scripts defined on this configuration file, and this ensures that the weather data is sent simultaneously to both Thingspeak channel and WAZIUP Platform.

[14] https://github.com/CongducPham/LowCostLoRaGw

[15] https://github.com/unparallel-innovation/UI_Waziup_Weather_Station/tree/master/WAZIUP%20Gateway%20Modifications

ACKNOWLEDGEMENT

This document has been produced in the context of the H2020 WAZIUP project. The WAZIUP project consortium would like to acknowledge that the research leading to these results has received funding from the European Union's H2020 Research and Innovation Programme under the Grant Agreement H2020-ICT-687670.