WAZIUP: Co-funded by the Horizon 2020 Framework Programme of the European Union

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:	Confidential/Public/Restricted
Version:	2.0
Date:	April, 7th , 2017

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

This document aims to describe and detail the development of a low-cost Weather Station, based on the Teensy LC board and the Adafruit RFM9x LoRa Radio module. 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.

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 [1].

[1] https://github.com/bmpalmeida/UI_Waziup_Weather_Station/

2. 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:

- Teensy LC[2]
- Adafruit RFM9x LoRa Radio[3]
- Adafruit AM2315 Sensor[4]
- Adafruit MPL115A2 Sensor[5]
- Weather Meters Kit[6]
- Adafruit RTC DS3231[7]
- Antenna 868 Mhz and SMA adapter.

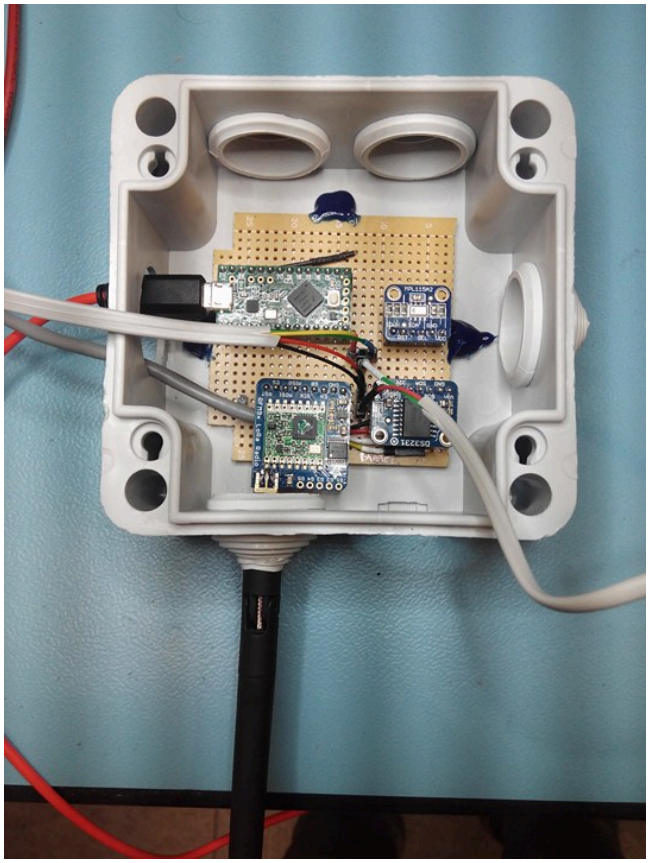

Figure 1 - Weather Station v2.0 - IoT solution for the Agriculture MVP.

[2] https://www.adafruit.com/products/2419
[3] https://www.adafruit.com/products/3072
[4] https://www.adafruit.com/product/1293
[5] https://www.adafruit.com/product/992
[6] https://www.sparkfun.com/products/8942
[7] https://www.adafruit.com/product/3013

In following figure, it is 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.

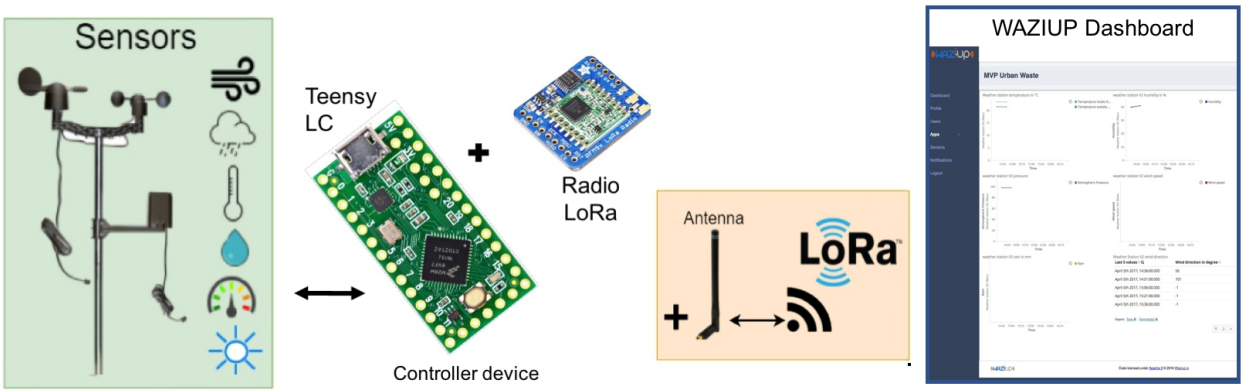

Figure 2 - Overview of the overall solution for the Agriculture MVP.

2.1. Core System controller

In terms of the core system within the Weather Station solution, it is composed by the Teensy LC, 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. In this system was also coupled communication capabilities.

2.2. Data Acquisition

With several external sensors this solution is capable of collecting data, such as temperature, humidity, barometric pressure, wind speed, wind direction and amount of rain.

Based on the proposed model, it becomes clear the connection between the controller and the sensors. This connection is established with the I2C protocol that allow this digital integrated circuit to communicate with one or more masters. It is 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 “Adafruit_AM2315.h” and “Adafruit_MPL115A2.h” libraries in order to call all the functions responsible for activating and reading the sensors signals coupled to the board.

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 sensors. 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 "SX1272.h" library. This library 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:

Example of the message send in the package:

\\!UT/1491383954/TO/18/HU/85/PA/101.10/TI/20/WD/90/WS/5.55/RA/0.55

The following table outlines the type and content of the information sent in each package:

Table 1 - Communication package.

\\!UT	Timestamp
TO/18/	Outside Case Temperature
HU/85/	Humidity
PA/101.10/	Pressor
TI/20/	Inside Case Temperature
WD/90/	Wind Direction
WS/5.55/	Wind Speed
RA/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.

2.4. Sending Data to ThingSpeak Cloud

To be able to send data information to a ThingSpeak channel it is need to take a few steps.

1) First of all you have to register on the platform[8] by creating an account. After signing in go on the top bar Channels->My channels and then on “New Channel” like in the following figure:

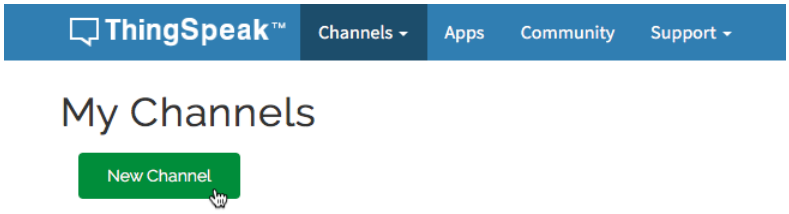

Figure 3 - Create a new channel in ThingSpeak.

2) Fill all the boxes with information about the new thingspeak channel (name, description, fields name, etc) and check the box “Make public”.

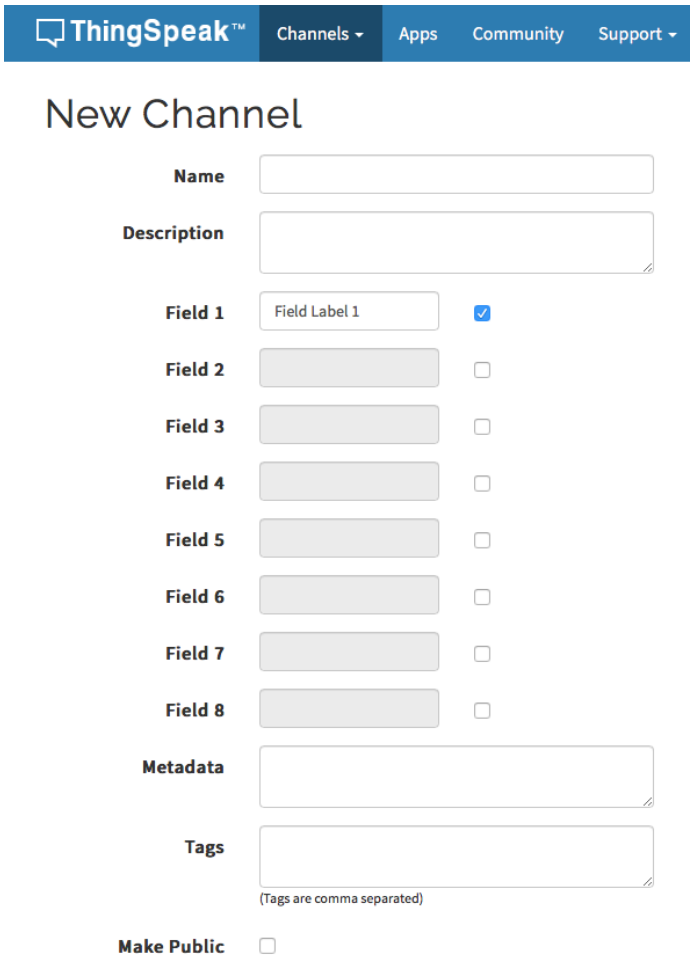

Figure 4 - Information about the new ThingSpeak channel.

[8] https://thingspeak.com/

3) After creating the channel, open the tab “API Keys” and take note of the “Write API Key”

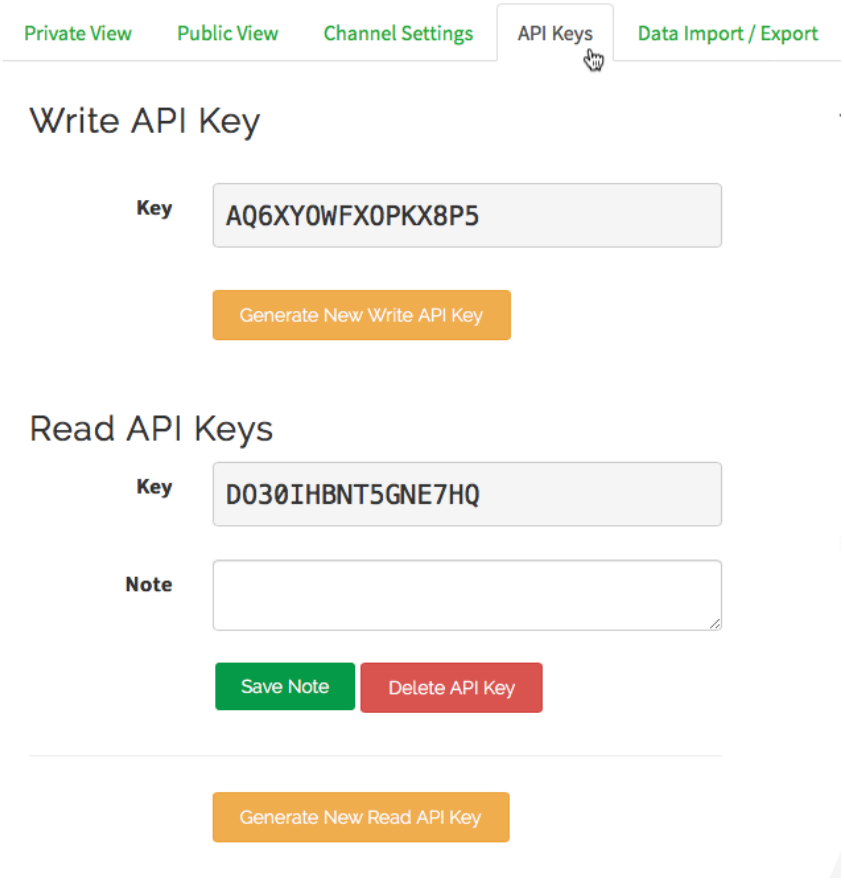

Figure 5 – ThingSpeak channel API Key.

4) On the gateway inside the folder “lora_gateway” edit the file “CloudThingSpeakWeatherStation.py” and replace ‘XXXXXXXXXXXXXXX’ by the api key on:

```
44 _def_thingspeak_channel_key='XXXXXXXXXXXXXXX'
```

The following examples shows the result, for a short period of time, from Weather Station in the ThingSpeak Cloud platform.

Figure 6 - Represented the ThingSpeak Cloud platform. [9]

[9] https://thingspeak.com/channels/228613

So far, the LoRa communication between Teensy LC (with RF95 integrated) and Raspberry PI (integrated with Hope RFM9X LoRa Radio) with the "SX1272" library works with success. To develop and building LoRa end-devices you can follow Congduc's tutorials.[10] [11]

The Figure 7 shows the architecture of proposed model and it's operating flow with the set specifications.

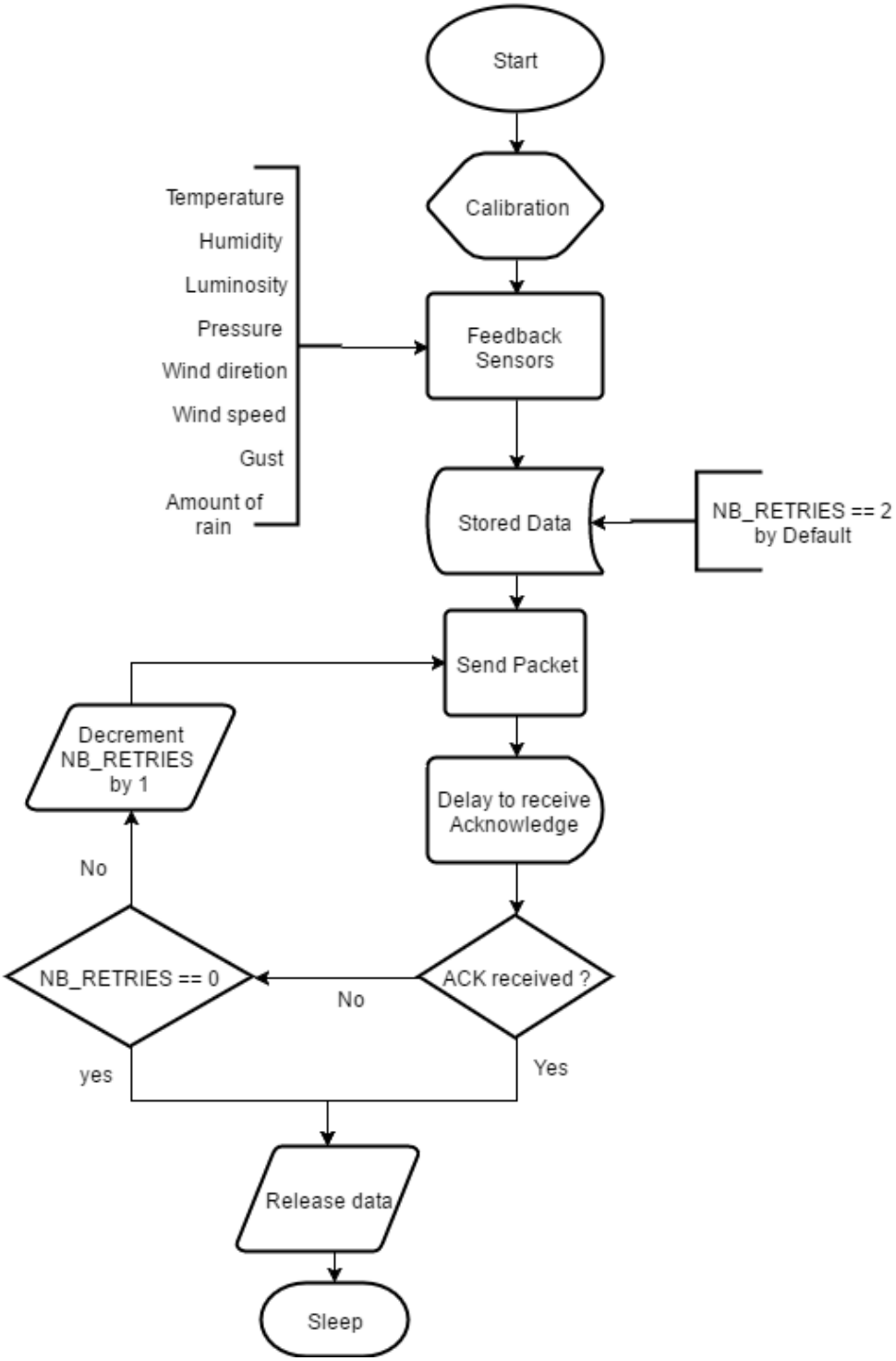

Figure 7 - Architecture model with the set specifications.

[10] http://cpham.perso.univ-pau.fr/LORA/LoRaDevices.html

[11] 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 Teensy tries to receive the ACK by the gateway (NB_RETRIES=2 by default). Only then the Slave fall asleep.

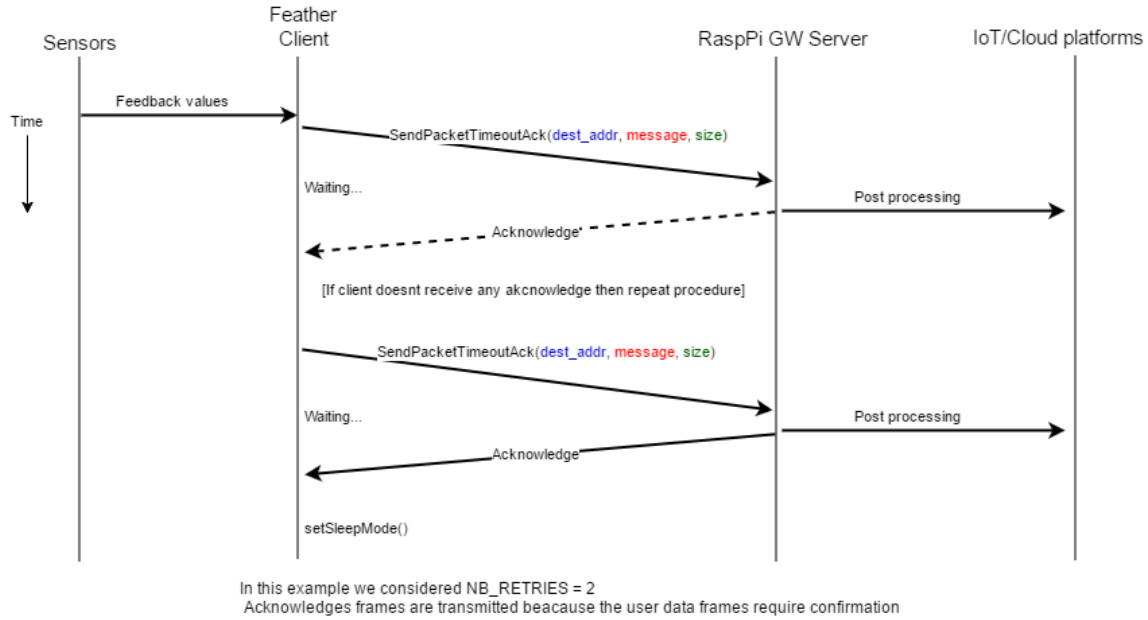

Figure 8 - Operation flow in LoRa communication protocol.

2.5. 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 teensy sleep between two communications.
- We can choose to show in console the information about what is happening on communication in real time.

3. TEENSY LC

Teensy-LC (Low Cost) is a powerful 32 bit microcontroller board, with a rich set of hardware peripherals that delivers an impressive collection of capabilities to make modern electronic projects simpler. It features an ARM Cortex-M0+ processor at 48 MHz, 62K Flash, 8K RAM, 12 bit analog input & output, hardware Serial, SPI & I2C, USB, and a total of 27 I/O pins. Maintains the same form-factor as Teensy 3.1, with most pins offering similar peripheral features.

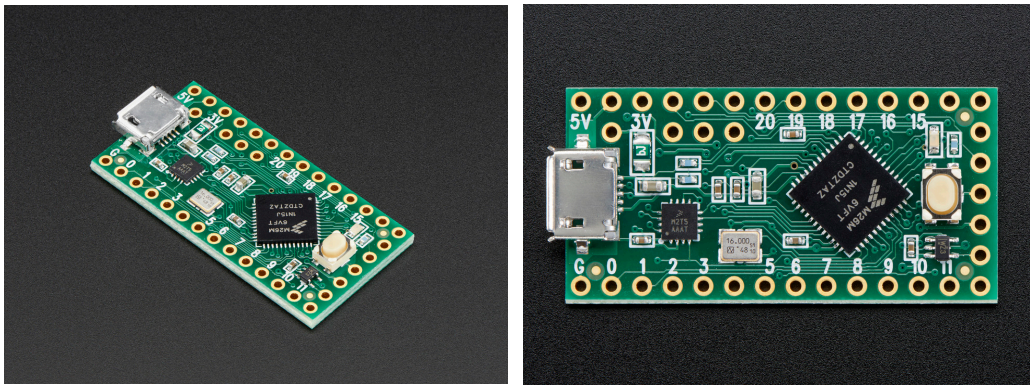

Figure 9 - Teensy LC. [12]

Most modern projects involve serial communication with sensors, other chips, other systems, or even the internet. Hardware serial ports greatly simplify projects and enable excellent performance. Teensy-LC provide plenty of serial connectivity: 2 SPI ports, 2 I2C, and 3 Serial ports. All 3 serial ports are supported by high quality drivers in Teensyduino, with both transmit and receive buffering, and even support for RS485 transmitter enable.

Teensy-LC supports USB Serial, MIDI, Keyboard (international layouts), Mouse, Joystick, and RawHID protocols. A full set of 16 bidirectional USB endpoints are supported by the hardware, allowing any type of USB device. As more USB protocols are added to Teensyduino, despite its low cost, Teensy-LC will be up the task. Many Arduino libraries require a hardware timer. Teensy-LC has a total of 7 timers, all of them with 16 or more bits of resolution, to allow excellent compatibility with easy-to-use libraries. Many combinations of popular libraries, which would normally conflict, can seamlessly run together on Teensy LC and Teensy 3.1.

The Cortex-M0+ processor is a powerful, full 32 bit CPU, designed for lower power, lower cost devices. Cortex-M0+ has fewer instructions and a simpler bus structure than the more powerful Cortex-M4 on Teensy 3.1. For simple code, M0+ often runs at similar speed, when running at the same clock frequency.

In the following table and figure you can find the Teensy's LC specifications and the respective pin assignments.

[12] https://www.adafruit.com/products/2419

Table 2 - Teensy LC's Specifications.

Measures	18mm x 37mm x 4mm
Microcontrollers	Cortex-M0+ @ 48MHz
USB support	Comes with USB bootloader and serial port debugging
Hardware	Serial support, I2C support, I2S support, SPI support
PWM pins	x10
Analog inputs	x13
Reset button	Yes

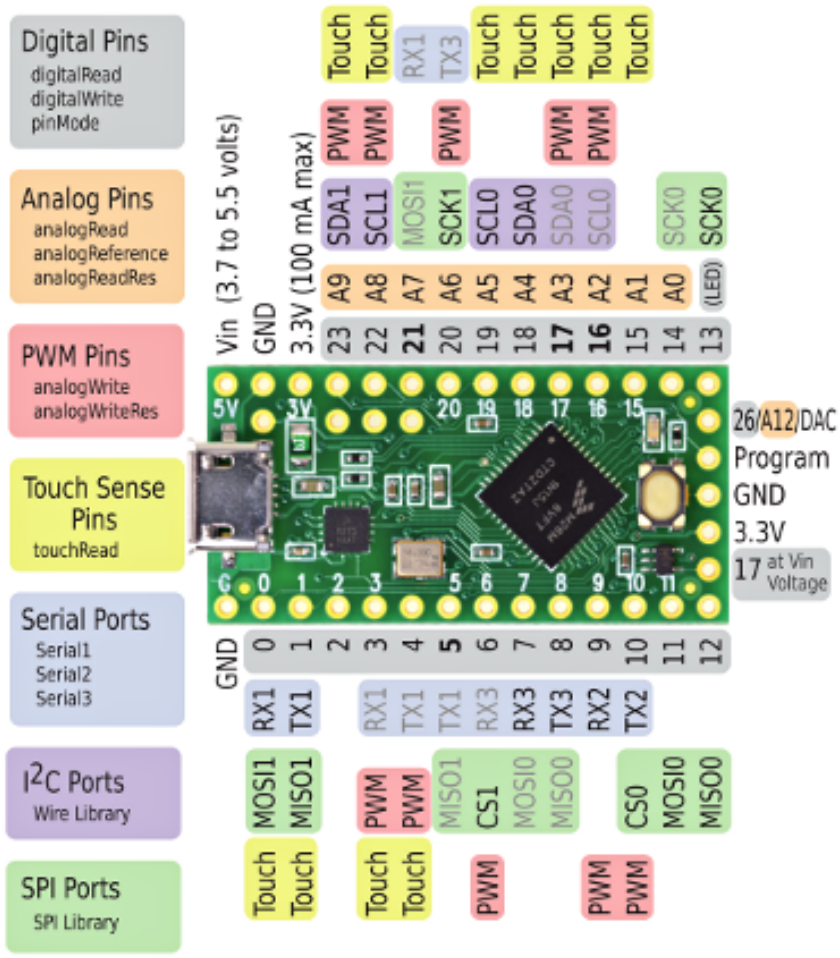

Figure 10 - Teensy LC Pin Assignments.

4. ADAFRUIT RFM9X LORA RADIO

Sending data over long distances is like magic, and now you can be a magician with this range of powerful and easy-to-use radio modules. These modules are great for use with Arduinos or other microcontrollers, say if you want a sensor node network or transmit data over a campus or town.

These radio modules come in four variants (two modulation types and two frequencies). The LoRa radios are exciting and more powerful but also more expensive.

These are +20dBm LoRa packet radios that have a special radio modulation that is not compatible with the RFM69s *but* can go much farther. They can easily go 2 Km line of sight using simple wire antennas, or up to 20Km with directional antennas and settings tweaking's:

- SX1276 LoRa® based module with SPI interface;
- +5 to +20 dBm up to 100 mW Power Output Capability (power output selectable in software);
- ~100mA peak during +20dBm transmit, ~30mA during active radio listening;
- The RFM9x radios have a range of approx. 2 km **line of sight** with tuned uni-directional antennas. Depending on obstructions, frequency, antenna and power output, you will get lower ranges - *especially* if you are not line of sight.

All radios are sold individually and can only talk to radios of the same part number. Each radio comes with some header, a 3.3V voltage regulator and level shifter that can handle 3-5V DC power and logic so you can use it with 3V or 5V devices. Some soldering is required to attach the header.

You will need to cut and solder on a small piece of wire (any solid or stranded core is fine) in order to create your antenna. Optionally you can pick up a uFL or SMA edge-mount connector and attach an external duck.

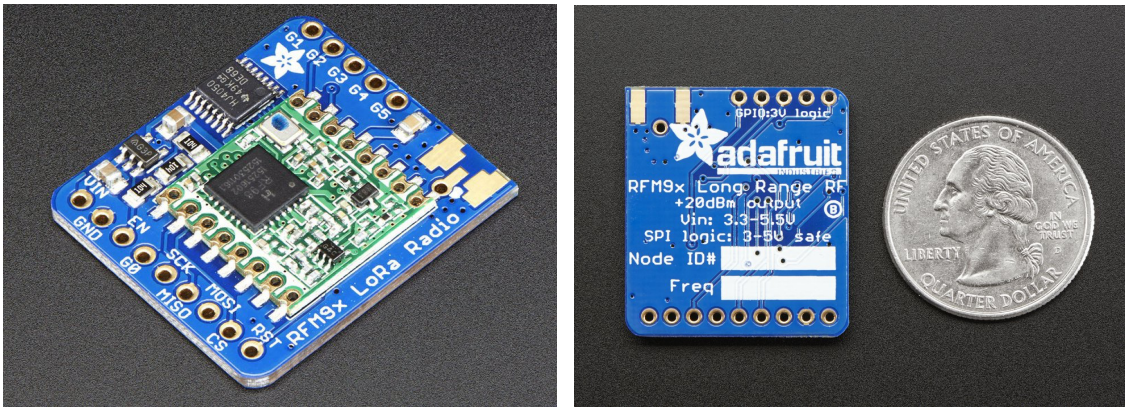

Figure 11 - Adafruit RFM9x LoRa Radio. [13]

[13] https://www.adafruit.com/products/3072

In the following table you can find the RFM9x LoRa Radio Specifications specifications.

Table 3 - Adafruit RFM9x LoRa Radio Specifications.

Measures	29mm x 25mm x 4mm
Weight	3.1 grams
Supply	Comes with a 3.3V voltage regulator and levelshifter that can handle 3-5V DC power and logic.

5. ADAFRUIT AM2315 SENSOR

This sensor contains a DS18B20 temperature sensor and a capacitive humidity sensor. A small microcontroller inside does the readings and provides a simple I2C interface for reading the finished & calibrated output data. Especially nice is that this sensor is in a rugged case with mounting bracket, which makes it way superior to a normal PCB-mounted sensor.

Figure 12 - Adafruit AM2315 sensor. [14]

While it is not rated as “weatherproof”, this sensor would do much better for sensing where there might be wind, rain, zombies, etc. than SHT PCB-breakout sensors, and the i2c interface makes it easier to interface with microcomputers that can't do the delicate timing of the DHT sensors.

Simply connect the red wire to power, black to ground, yellow wire to your I2C data pin, and the white wire to the I2C clock pin. You cannot change the I2C address so only one sensor per I2C bus. Two ~10Kohm pullup resistors are required for use, connect from the SDA and SCL lines to the power wire:

The following table shows the measurements specifications:

Table 4 - AM2315 sensor measurements Specifications.

Temperature	-20 to 80°C temperature readings ±0.1°C typical accuracy
Humidity	0-100% humidity readings with 2% accuracy

[14] https://www.adafruit.com/product/1293

6. ADAFRUIT MPL115A2 SENSOR

This pressure sensor from Freescale is a great low-cost sensing solution for measuring barometric pressure at 1.5 hPa resolution. It's great for basic barometric pressure sensing. The sensor is soldered onto a PCB with 10K pull-up resistors on the I2C pins.

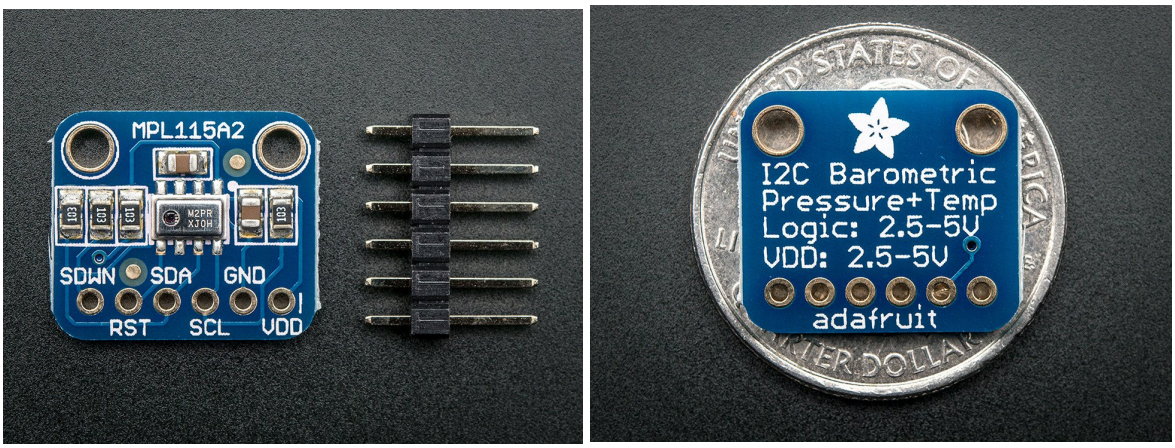

Figure 13 - Adafruit MPL115A2 sensor. [15]

This chip is good for use with power and logic voltages ranging from 2.4V to 5.5V so you can use it with your 3V or 5V microcontroller. This chip looks and sounds a whole lot like the MPL3115A2 but this is the less precise version, best for barometric sensing only.

Using the sensor is easy. If you're using an Arduino, simply connect the VDD pin to the voltage pin, GND to ground, SCL to I2C Clock and SDA to I2C. Then download the “MPL115A2” Arduino library and example code for temperature, pressure and basic altitude calculation. Install the library, and load the example sketch. Immediately you'll have the temperature, pressure and altitude data printed in the serial console.

The following table shows the measurements specifications:

Table 5-MPL115A2 sensor measurements Specifications.

Pressure	500 - 1150 hPa (up to 10Km altitude)
Altitude	1.5 hPa / 50 m

[15] https://www.adafruit.com/product/992

7. WEATHER METERES 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 0.2794mm (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 2.4 m/h (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. For more specification you can read the paper available at the link in the notes.[16] 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 metres per hour. 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 diagram bellow. Cause the circuit was built for 5V supply we need to rewrite arrangement values.

Table 6 - ADC arrangement values.

	Angle (º)	ADC with 5Volts (Default version)	ADC with 3.3Volts (Our Version)
North	0	913	905
	22.5	680	657
	45	746	697
	67.5	393	382
West	90	414	390
	112.5	380	365
	135	508	476
	157.5	456	423

[16] http://cpham.perso.univ-pau.fr/LORA/resources/RPIgateway.pdf

South	180	615	563
	202.5	551	529
	225	833	809
	247.5	801	790
East	270	990	991
	292.5	940	927
	315	967	960
	337.5	878	856

- **Rain**: This sensor rain works with interrupts and return values in each data transmission. The rain gauge is a self-emptying tipping bucket type. Each 0.2794 mm (0.011”) of rain causes one momentary contact closure that can be recorded with a digital counter or microcontroller interrupt input.

Figure 14 - Weather Meteres Kit. [17]

[17] https://www.sparkfun.com/products/8942

8. ADAFRUIT DS3231 RTC

This is a great battery-backed Real Time Clock that helps in microcontroller projects to keep track of time even if it is reprogrammed, or if the power is lost. A real time clock is basically just like a watch: it runs on a battery and keeps time for you even when there is a power outage.

Using an RTC, you can keep track of long timelines, even if you reprogram your microcontroller or disconnect it from USB or a power plug. Most microcontrollers, including the Arduino, have a built-in timekeeper called millis() and there are also timers built into the chip that can keep track of longer time periods like minutes or days. The biggest reason for using a RTC is because millis() only keeps track of time since the Arduino was last powered.

That means that when the power is turned on, the millisecond timer is set back to 0. The Arduino doesn't know that it's 'Tuesday' or 'March 8th', all it can tell is 'It's been 14,000 milliseconds since I was last turned on'.

So what if you wanted to set the time on the Arduino? You'd have to program in the date and time and you could have it count from that point on. But if it lost power, you'd have to reset the time. Much like very cheap alarm clocks: every time they lose power they blink 12:00.

While this sort of basic timekeeping is OK for some projects, some projects such as data-loggers, clocks, etc will need to have consistent timekeeping that doesn't reset when the Arduino battery dies or is reprogrammed. Thus, we include a separate RTC! The RTC chip is a specialized chip that just keeps track of time. It can count leap-years and knows how many days are in a month.

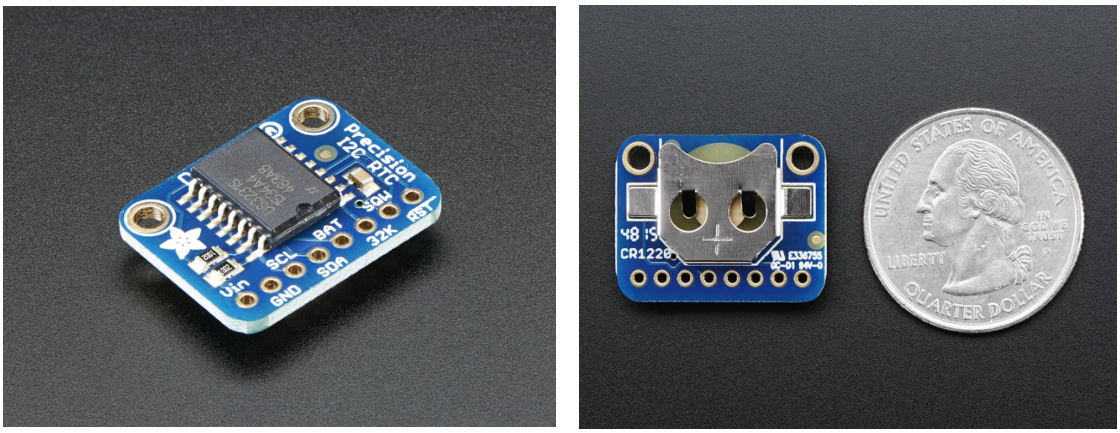

Figure 15 - Adafruit DS3231 RTC. [18]

[18] https://www.adafruit.com/product/3013

Most RTC's use an external 32kHz timing crystal that is used to keep time with low current draw. And that's all well and good, but those crystals have slight drift, particularly when the temperature.

This DS3231 Adafruit RTC is in a beefy package because the crystal is inside the chip! And right next to the integrated crystal is a temperature sensor. That sensor compensates for the frequency changes by adding or removing clock ticks so that the timekeeping stays on schedule.

This is the finest RTC you can get, and become with a coin cell plugged into the back so we can get years of precision timekeeping, even when main power is lost. Great for datalogging and clocks, or anything where you need to really know the time. Comes as a fully assembled and tested breakout plus a small piece of header. In the following table you can find the Adafruit DS3231 RTC's specifications:

Table 7 - Adafruit DS3231 RTC Specifications.

Measures	23mm x 17.6mm x 7.2mm
Weight	2.1 grams
Supply	From 2.3V to 5.5V. Do not need regulator. Use the same power as the logic level of microcontroller.

9. 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 "SX1272" library that is the base of the Congduc library to support this kind of communication.

10. 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) **Connect Teensy with the external Sensors**:

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 8 - Pin Connection between components.

Teensy LC	RFM9x LoRa module	AM2315 sensor	MPL115A sensor	DS3231 RTC	Weather Meters
G	GND	Black cable	GND	GND	-
3V	VIN	Red cable	VIN	VIN	-
-	EN	-	-	-	-
-	GO	-	-	-	-
13	SCK	-	-	-	-
12	MISO	-	-	-	-
11	MOSI	-	-	-	-
10	CS	-	-	-	-
4	RST	-	-	-	-
19 (SCL0)	-	White cable	SCL	SCL	-
18 (SDA0)	-	Yellow cable	SDA	SDA	-
20 (A6)	-	-	-	-	Wind Direction
21 (A7)	-	-	-	-	Rain
22 (A8)	-	-	-	-	Wind Speed

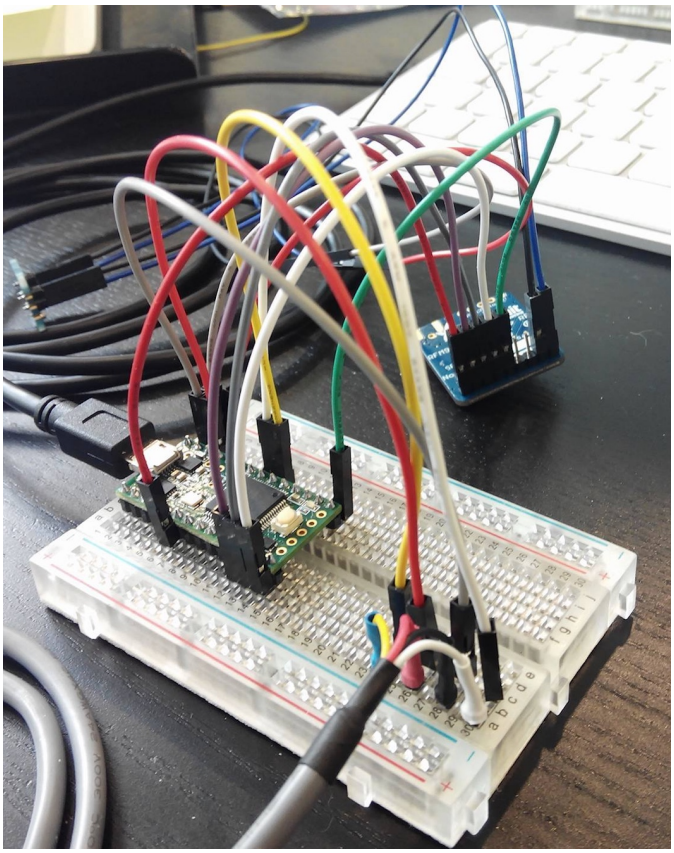

Figure 16 - Some connection's pictures in testing phase.

2) Connect the Wind and Rain sensors:

Figure 17 - Wind and Rain Sensors.

3) Connect the Sntenna 868 MHz in RFM9x LoRa module:

Figure 18 - 868 MHz Antenna and SMA adapters.

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

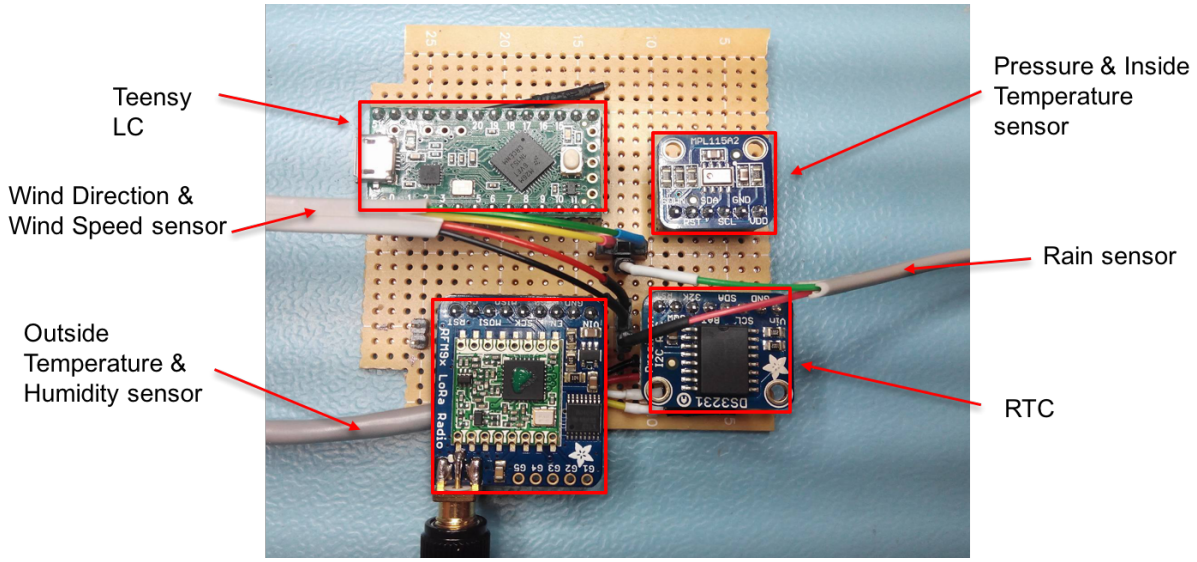

Figure 19 - Implemented Weather Station in final prototype.

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 “Snooze”, “SX1272”, “MPL115A”, “AM2315” and “RTCLib” libraries from our Github.[19]
The “Snooze” library allow to put the microcontroller in low-power mode. The "SX1272” library includes LoRa Communication and was built because of its specification; The “RTCLib” allow the interaction with the DS3231 RTC and; The “MPL115A” and “AM2315” libraries includes all necessary sensor control functions.

6) **Compiling the software**:

Write a script that include both of these libraries and “SPI” and “I2C” libraries. In Github you will find an example of this project. Copy the “WeatherStation_v2.0.ino” file into your “sketch” folder. This archive contains the latest sketch (firmware) that the USB Teensy 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. 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 «upload» the code.

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.

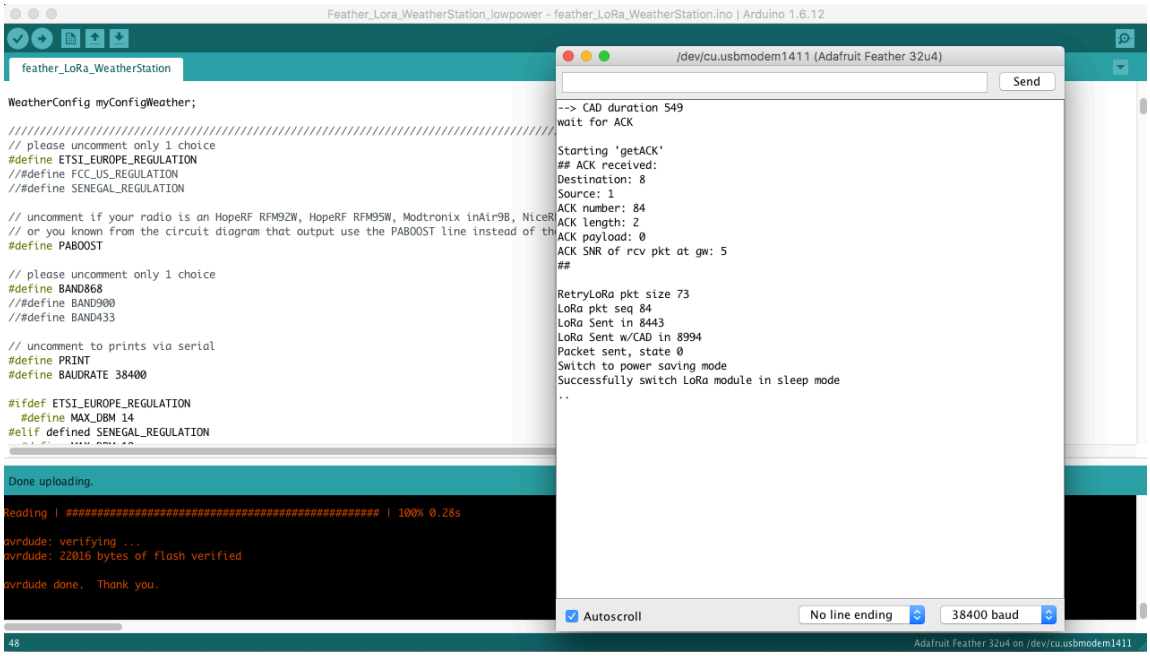

Figure 20 - Serial Monitor results.

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

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.