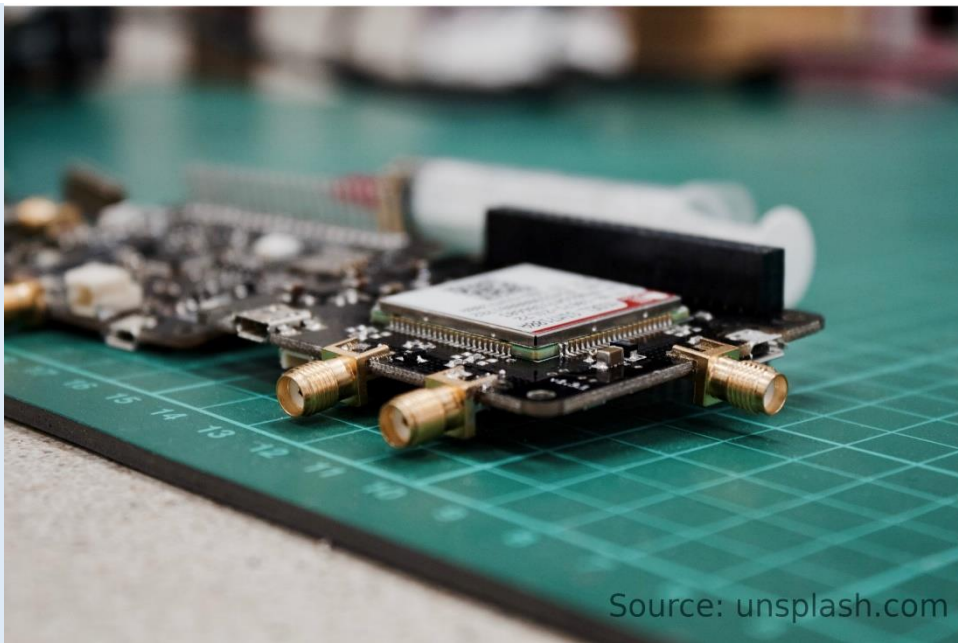




Wireless Sensor Networks

LoRaWAN & TheThingsNetwork



JULY 6



UNIVERSITY OF
WESTERN MACEDONIA

Delaportas Markos (1316)
Loukas Ioannis (1030)

Semester: 8th
Semester: 12th



Contents

1.	Introduction	3
2.	Components.....	4
3.	Schematics	4
4.	Code.....	6
5.	TTN	7
6.	NodeRed	8
7.	Future Improvements	9
8.	Lessons Learned	9
9.	Conclusions.....	10

TTN & NodeRed

Graphically Display Sensor Data w/ NodeRed

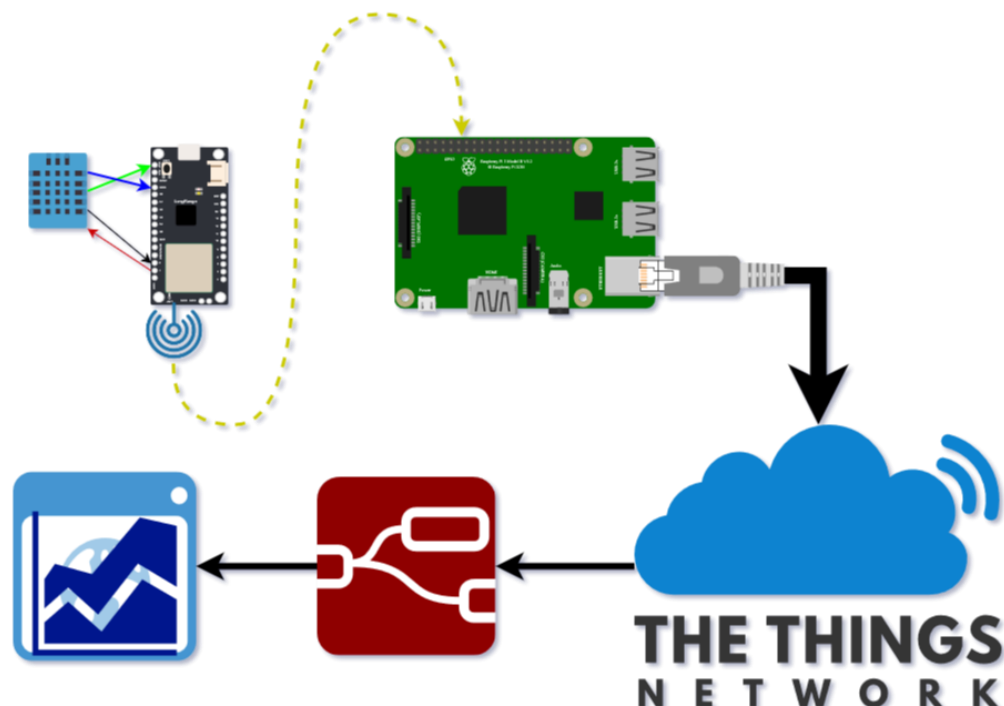
Abstract- Purpose of this project is to route data gathered from a temperature sensor to TheThingsNetwork using LoRaWan. Then sensor data are imported to a self-hosted NodeRed application which displays them graphically. Graphs are automatically updated every time a new packet arrives, which happens per one hour.

1. Introduction

As big data analysis is more essential day by day, as the need for distributed monitoring is getting bigger and bigger, the IoT industry is exploding. In other words, the ability to centralize data gathered from multiple sensors can affect both individuals and companies in ways such as upscaling production and/or improving the quality of life. All that become possible by programmatically manipulating metrics and extracting environmental conclusions.

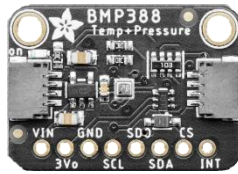
In our project a temperature sensor is connected to a microcontroller which transmits the data imported from the sensor using the LoRa protocol. Transmitted packets get captured by gateways within range which, being connected to the internet, undertake to route the data through TTN. We set up a Node-RED flow that imports each packet upon arrival and by extracting it's payload (aka the temperature) and it's timestamp we plot these values to a web interface.

The data flow can be seen below:

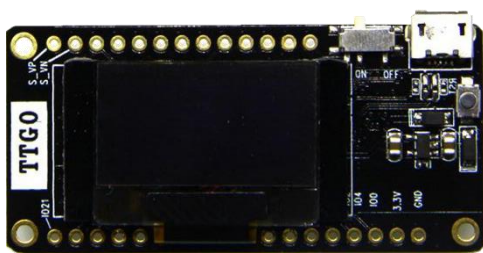


2. Components

The sensors used consist of a breakout board from adafruit built around the bosch bmp388 sensor, it collects temperature, pressure, and humidity data.



These data may be used to calculate the altitude –which needs to be initialized with the relative tide level that day.

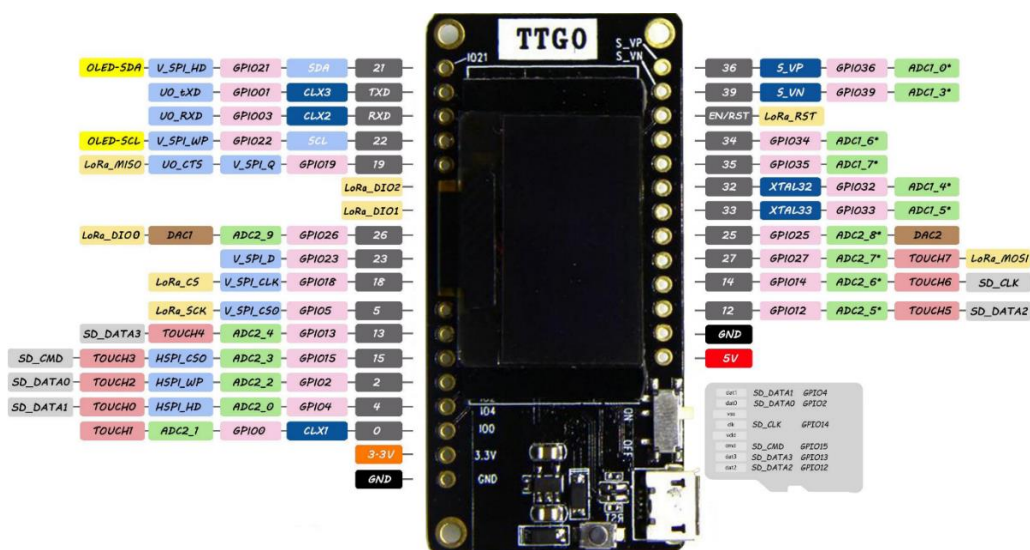


Amongst these data we collect the temperature which is then being read by an esp32 based board. This board utilizes the sx1276 chipset that emits LoRa packets. That is because LoRa gateways are in the range of 5km from the sensor and connected to TTN.

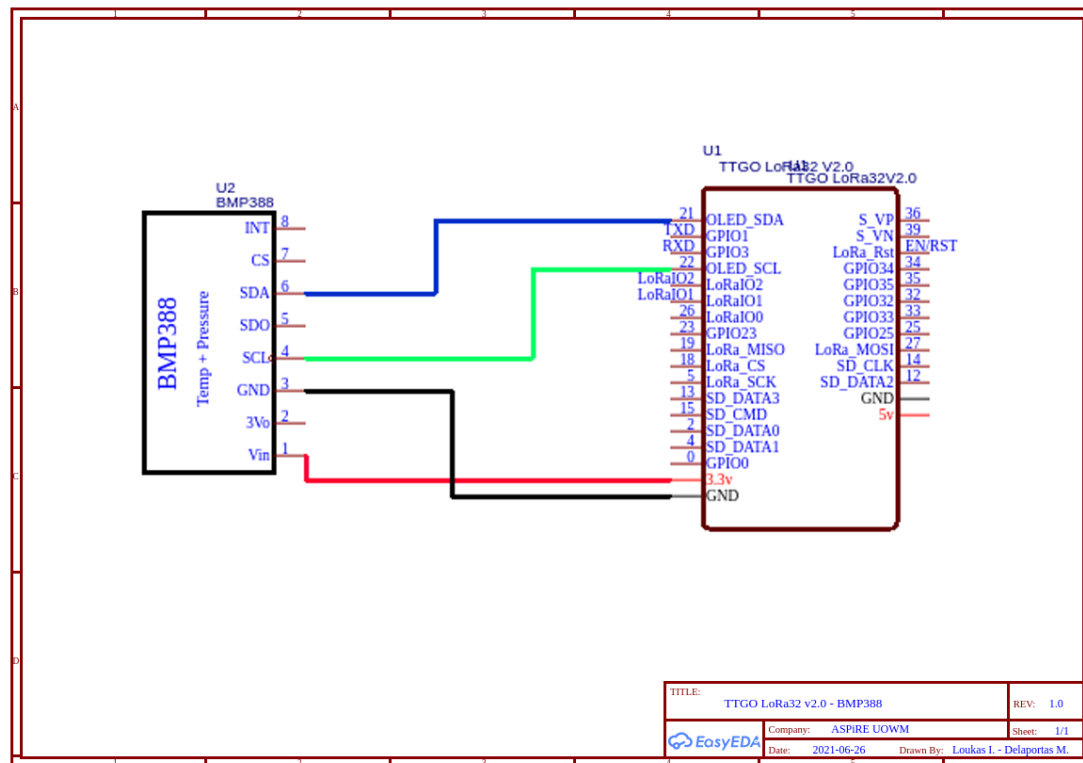
In greater detail these gateways receive LoRa packets and route them to a network server and can thus be accessed from a web interface. This is particularly useful because it enables us to import them to another public server.

3. Schematics

The microcontroller used is the TTGO LoRa32 v2.0 which can be connected with a BMP388 via SPI or I²C. In our project, we choose to connect via I²C, because of the simplicity of use. Specifically, I²C requires two cables for data transmission and synchronization, one serial data (SDA) and one serial clock (SCK), unlike the SPI which utilizes three.



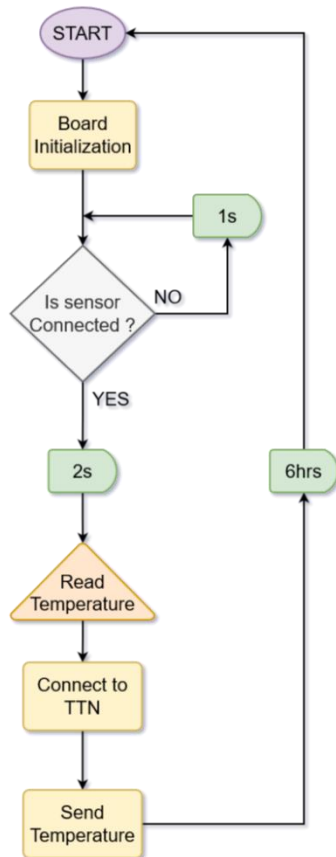
Wiring Diagram



BMP 388	TTGO LoRa32
V _{in}	3.3v
GND	GND
SCL	GPIO22
SDA	GPIO21

4. Code

The code can be generally divided into two main parts. In greater detail there is the code that runs on the microcontroller and the code described in the Node-RED flow.



The code running on the MCU can be described in this flowchart.

In greater detail for the temperature sensor to output realistic results a delay of two seconds shall be implemented before we import any measurements. That is because the bmp388 breakout sensor needs this time in order to initialize, which happens regardless if it is connected correctly. As the sensor gets powered it outputs the initial value of 24.01°C for the first five loops, even then the values imported have big fluctuation and thus a delay (can be 'nop') of around 2 seconds shall be implemented for the measurements to stabilize.

Moving on to the MCU, the board utilizes the Imic library- which is essentially a real time operating system for embedded devices- it doesn't run the standard 'void loop()' function, instead it implements its own 'os_run_loop_once()' function which then does all the necessary 'jobs'. That means that there is not much flexibility in the commands executed in each clock cycle (aka each 'loop()'), but we decided that the measurements had to be imported and transmitted periodically to TTN. In order to get around this problem, a complete reset of the MCU has to take place before packet transmission. Yet the board we used did not have a reset pin, so hardware reset (through raising the RST pin 'high') is not an option. For this reason, software reset is what we ended up implementing and thus packet transmission is done periodically in respect to the desired interval.

Another issue that shall be noted is, the difference between the value type imported from the sensor function and the type of data that TTN accepts. In greater detail, the sensor returns a floating-point number that represents the temperature, but TTN accepts an unsigned integer of 8 bits type. Conversion from one to the other cannot happen directly and will cause a core panic upon execution. In other words, we cannot just map each byte of the temperature

value to an array of the appropriate bytes, even if the size is correct and no overflow takes place.

To resolve the above issue an intermediate step had to be introduced in which the floating-point values are first converted to a string. A string buffer takes over to store every digit of the value to an address of character type. Here it is important to note that floating-point numbers have a precision of six digits, adding the dot and the first two digits produces 9 characters, but considering that each string is required to be null terminated (aka ends with ‘\0’). So the total buffer size, as well as the payload are ten bytes in size and not four, which is the size of a float. It is then that the array of characters can be copied to the payload byte by byte and be passed as an argument to the `lmic` function that undertakes data transmission.

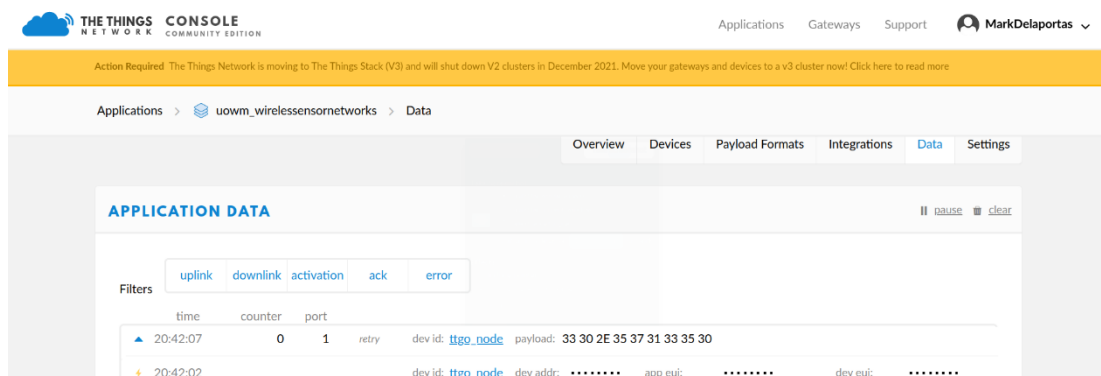
For reference the code below was what resolved this issue

```
sprintf(buff, "%f", Temperature_Print());  
for(int i=0; i<10; i++) mydata[i] = buff[i];
```

Github Repository: github.com/johnylouk1997/T*IGO2TTNwBMP388

5. TTN

The Things Network is a platform which, in its core is a web interface for LoRaWAN traffic. In greater detail users that utilize this protocol can login in the platform and monitor the data transmitted by their LoRa modules. Obviously non authorized users cannot view these data and that's because each packet is encrypted using a set of keys specific to the individual and the application in TTN. These credentials should be present in the source code for the packet validation to complete. This way the gateways that are connected to TTN (and within range) are able to decrypt the packets and forward them to TTN interface, it is then that TTN's backend presents the appropriate packets to the correct user in respect to his application credentials.

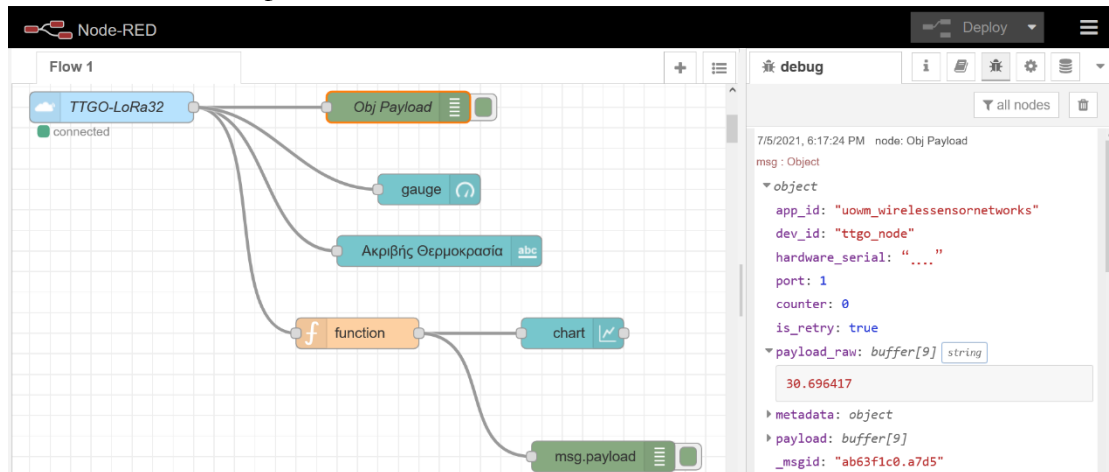


The screenshot displays the 'THE THINGS NETWORK CONSOLE' interface. At the top, there's a navigation bar with 'Applications', 'Gateways', 'Support', and a user profile 'MarkDelaportas'. Below this, a yellow banner states: 'Action Required The Things Network is moving to The Things Stack (V3) and will shut down V2 clusters in December 2021. Move your gateways and devices to a v3 cluster now! Click here to read more'. The main content area shows the breadcrumb 'Applications > uowm_wirelessensornetworks > Data'. A tabbed interface at the top of the data section includes 'Overview', 'Devices', 'Payload Formats', 'Integrations', 'Data' (selected), and 'Settings'. The 'APPLICATION DATA' section features filters for 'uplink', 'downlink', 'activation', 'ack', and 'error'. Below the filters, a table lists data points with columns for time, counter, port, and payload. The first entry shows a time of 20:42:07, counter 0, port 1, and a payload of 33 30 2E 35 37 31 33 35 30. The second entry shows a time of 20:42:02 and a dev id of tigo_node.

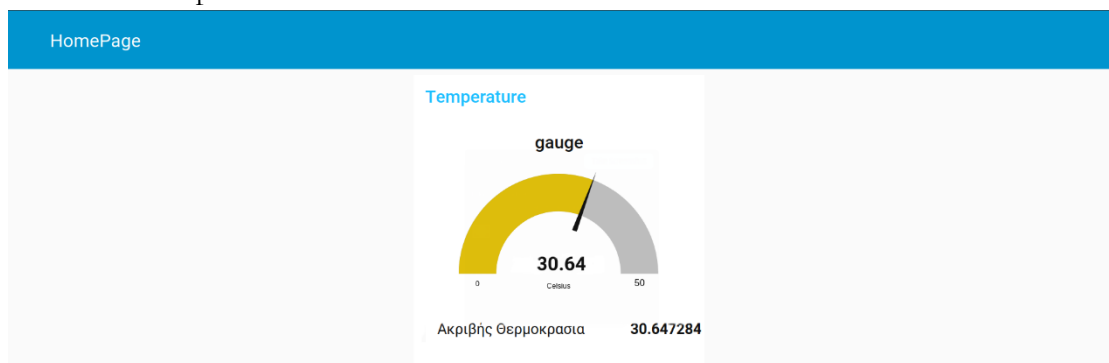
6. NodeRed

According to the architecture of this project all measurements need to be outputted to a web interface as well as be dynamically updated whenever a new packet arrives to TheThingsNetwork. To achieve this, we installed Node-Red in a virtual private server of ours and added the packages that import data from TTN to the flow. After we successfully import every new packet arrived, we treat it as a json object and extract the payload (aka the temperature) and the timestamp (aka the time of arrival), we use these two values to plot the temperature to a graph in respect to the time it was collected.

The flow below imports the data as they arrive to TTN, extracts the payload and the timestamp.



It then plots above data to a user interface.



It should be noted that no RTC is on the esp32, so the timestamp is gathered at the time a new packet arrives to the Gateway, thus the actual timestamp of each value is around 100ms-500ms, earlier. But given the fact that this is outdoor temperature we're talking about, and data are collected only twice a day we figured it's not that big of a deal.

7. Future Improvements

To accomplish a successful project, it must be able to upgrade. For this reason, we quote the below:

- A. Replace existing equipment:
 - a. BMP388 with BME280 (or something similar from the same category), to receive beyond temperature and humidity. This way we will have a larger volume of data that we can utilize.
 - b. TTGO LoRa32 with Raspberry Pi (or something similar from the same category), to have more processing power in our microcontroller.
 - i. External LoRa module
 - c. PCB omnidirectional antenna with a Bidirectional antenna, to reduce latency and increase range.
- B. Add more microcontrollers in different locations, to have a larger volume of data that we can utilize.
- C. Add a GPS module like PA1010D, to know the exact coordinates of our microcontrollers.
- D. Creating a machine learning algorithm and retrieve with NoteRed API from 3rd party resources, to make predictions for future temperatures or forecasts, in combination with our data.
- E. Add a solar panel, per node, to become completely autonomous in terms of power delivery, thus can be placed where no electricity is available.
- F. Can be used in the field of smart homes, adapting the already existing LAN protocols.
- G. Can be used in agricultural fields (pun intended), an area that is rapidly growing today (pun intended), in combination with other systems on the market.

8. Lessons Learned

A. Verifying the Results

Considering this was not the first time we undertook a project of this nature we continued studying about IoT, embedded systems and the governing principles influencing the design decisions. During the design phase, we utilized commonly used tools in the Arduino community, like Arduino IDE, GitHub, etc. This provided valuable insights, not only in regard to the routing LoRa packets through TTN per se, but also about IoT as a whole.

B. Project Management Lessons Learned

From the very beginning we putted effort to create a solid GANTT chart considering all the constrains. Additionally, we clarified what aspects of the project each one had to undertake, that is because we have adopted a system in

which the project is divided into the smallest possible pieces in order to be managed with ease and also to embrace continuous communication between us members. After all it is clear that coding is a solitary process, but it still needs collaborative skills.

All in all, if we had to summarize all lessons learned into one phrase it would be that “Five hours of debugging can save you five minutes of documentation reading”!

9. Conclusions

Having to deal with the current pandemic, provided an unprecedented experience for both of us. The first challenge presented was the need for remote work. After finding the right tools for communicating remotely, it became apparent that the greatest challenge would be the way to substitute physical access to university labs. Getting around that required significant help from mentors as well as the university professors. After almost three months of lockdown, we managed to return to the labs, but with extremely limited time until the final documentation was required. This proved very challenging.

References

- A. github.com/mcci-catena/arduino-lmic
- B. thethingsnetwork.org/forum/t/big-esp32-sx127x-topic-part-3/18436
- C. primalcortex.wordpress.com/2017/11/24/the-esp32-oled-lora-ttgo-lora32-board-and-connecting-it-to-ttn/
- D. thethingsnetwork.org/
- E. nodered.org/
- F. github.com/MartinL1/BMP388_DEV