Limabora: A smart Farming Case Using LoRa modules, Gateway, TTN and Firebase

Leonard Mabele, MSc, B.Eng. Member, Engineers Board of Kenya.

Abstract

Internet of Things (IoT) has dominated the tech space headlines for the past five months here in Kenya. A number of IoT conferences have been organised throughout the period to present the concept of Internet of Things and try to have the C-suite understand what the buzz for IoT is all about and the future of it. @iLabAfrica Research Centre in Strathmore University has already started investing heavily in Machine-to-Machine (M2M) and IoT to present the value that these technologies are about to give (actually are already giving) to almost every vertical and sector of economy.

Out of this buzz, I started working on an IoT test scenario for smart farming three months ago to extract a case in one of the major verticals that IoT is to give a lot more value for Africa: Agriculture. My research was to test the reliability that Long Range Radio (LoRa) technology can give in trying to transmit data from greenhouse farm environments to real-time cloud and finally to an android mobile application that a farmer can use to see real-time data about their farm environment from anywhere. These data include the humidity, temperature and the soil moisture of the farm transmitted in real-time to an Android mobile application through Google's firebase. This paper presents a step by step approach of achieving this based on low power wide area network (LPWAN) LoRa technology.

Keywords: *M2M, IoT, LPWAN, LoRa*

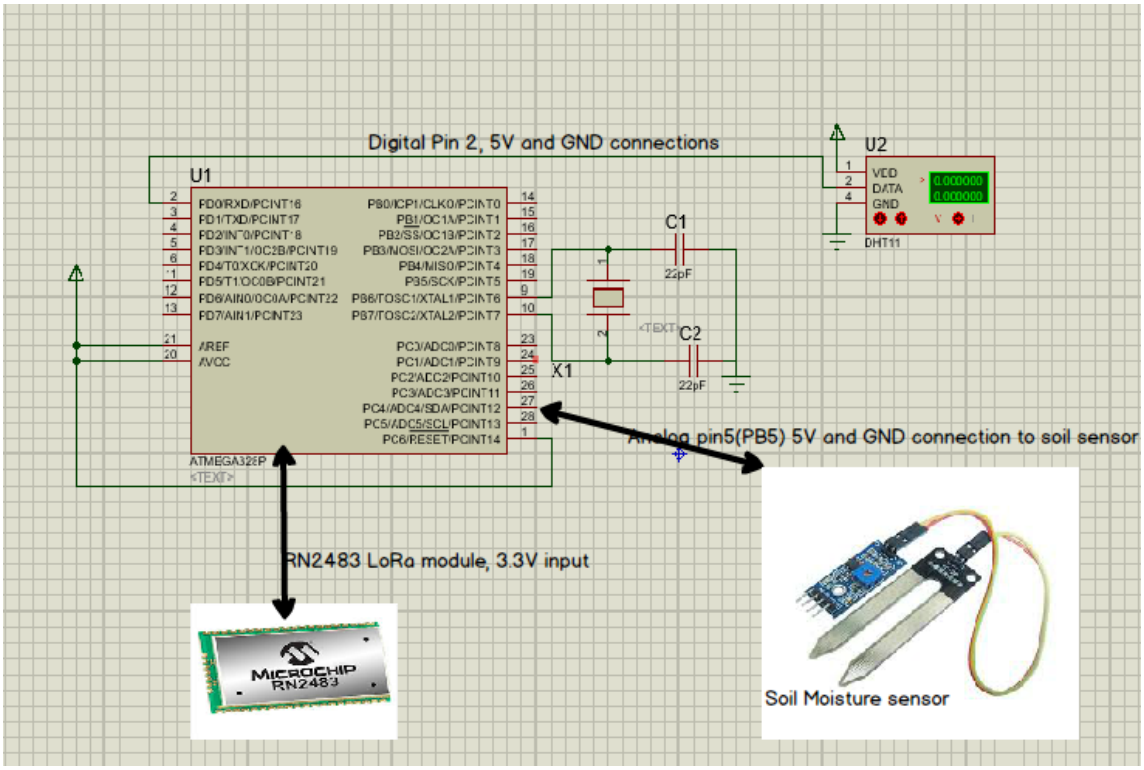
The Sensor Node Environment

In my setup, I used the following devices for my node:

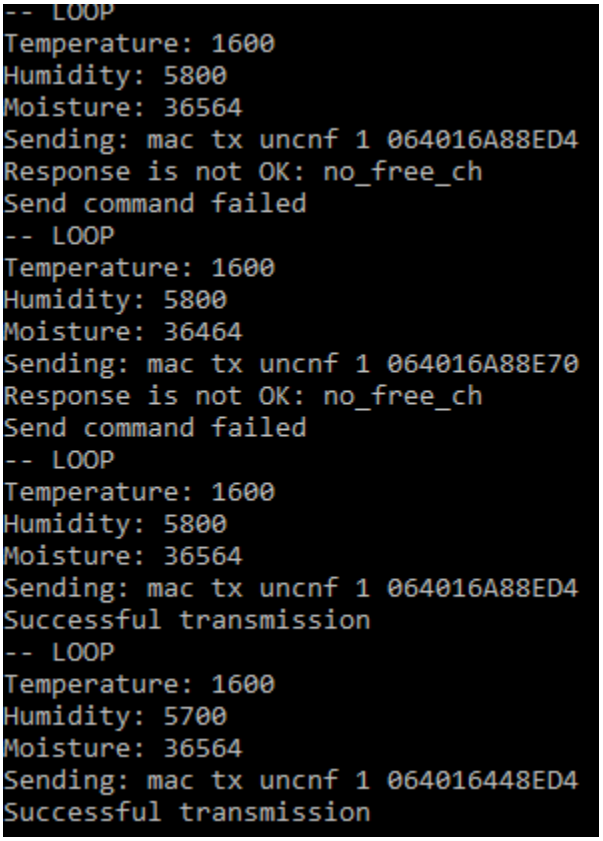
1. ATmega328P MCU
2. Soil moisture sensor from Sparkfun
3. DHT11 sensor module for temperature and humidity sensing
4. An RN2483 LoRa Module

The soil moisture sensor connects to the ATmega328P microcontroller unit (MCU) using one of the analog pins of this MCU chip while the DHT11 uses any of the digital pins. In this case, pins A5 and D2 were used. The RN2483 module uses the Universal Asynchronous Receiver Transmitter (UART) interface to connect to the MCU. One can use hardware serial or software serial to make this connection. In my case, I used the software serial. The diagram of the connection is as shown below. Both sensors use 5V power input while the RN2483 communication module uses 3.3V. Since the ATmega 328P MCU used is the 16MHz, 5V input, I used the LD1117 regulator to get the 3.3V for the RN2483 LoRa module.

The ATmega328P was programmed with Platformio platform using the C++ language. Platformio is added as a plugin to your preferred text editor such as vim, eclipse or atom. I prefer the atom environment though I upload my code and view serial from the terminal on linux or cmd on windows (I like it that way, but you can do this from atom). (Kravets, 2017). The diagram of this connection is as shown below. The code is shared on github.



On transmission of data from the sensors to the things network, I have added the image below to show a serial excerpt of platformio on windows cmd as data gets transmitted on the LoRa network both on fail and successful transmission.



The Communication Architecture

Obtaining sensor data is not that much of a challenge in M2M and IoT as long as one understands the sensor(s) involved and how to connect them appropriately to the MCU. I always advise one to thoroughly read the datasheet and the documentation of every sensor. The real challenge in IoT is transmitting this data and getting to present it to the end user who does not, and certainly should not understand the technical implementation involved in achieving it. In this case, with the end user being the farmer, Wi-Fi, Bluetooth and GSM are the alternative communication infrastructures that one would use to transmit the data to the farmer's mobile application. These three have their major pros and cons and certainly are usable but the context of my solution led me to adopt the LoRa infrastructure not only to make the solution a little more sophisticated but to create new knowledge in the development of IoT solutions in Africa as well.

The advantages of the LoRa technology include:

1. It uses 868 MHz/ 915 MHz ISM bands which is available worldwide.
2. It has very wide coverage range about 5 km in urban areas and 15 km in suburban areas.
3. It consumes less power and hence battery will last for longer duration.
4. Single LoRa Gateway device is designed to take care of 1000s of end devices or nodes.
5. It is easy to deploy due to its simple architecture.
6. It uses Adaptive Data Rate technique to vary output data rate/RF output of end devices. This helps in maximizing battery life as well as overall capacity of the LoRaWAN network. The data rate can be varied from 0.3 kbps to 27 Kbps for 125 KHz bandwidth.
7. The physical layer uses robust CSS modulation. CSS stands for Chirp Spread Spectrum. It uses 6 SF (spreading factors) from SF 7 to 12. This delivers orthogonal transmissions at different data rates. Moreover it provides processing gain and hence transmitter output power can be reduced with same RF link budget and hence will increase battery life.
8. It uses LoRa modulation which has constant envelope modulation similar to FSK modulation type and hence available PA (power amplifier) stages having low cost and low power with high efficiency can be used.

In the development of Limabora, I constructed a LoRa Gateway using a Raspberry Pi 3 and the iC880a (purchased from IMST) concentrator. The iC880a is able to receive packets of different end devices send with different spreading factors on up to 8 channels in parallel. The LoRa nodes with larger distances from the concentrator must use higher spreading factors while the nodes closer to the concentrator must use lower spreading factors eliciting the concept of dynamic data rate. The larger the distance, the lower the data rate and vice versa.

The LoRa node connected to the ATmega328P MCU packages the payload of the two sensors and transmits them to the Internet through the iC880a concentrator sitting on the raspberry Pi 3. I want to applaud the amazing team of the things network (TTN) as I used their platform to register my gateway and my node to carry out this test. I used their platform as well to decode my payload from hexadecimal to float. The payload is packaged as bytes in the microcontroller code. (Network, 2017)

The two images below shows how the communication architecture is developed in trying to transmit the data to the cloud. The RN2483 packages the data and sends to the iC880a concentrator on the raspberry pi. This is what is called uplink transmission. The concentrator has LoRa on one end and Internet on the other end through the Raspberry Pi. In this case I have used the Ethernet cable on the Pi but you can use Wi-Fi as well by editing the network interfaces on the Raspberry Pi. This is also provided by the Raspberry Pi team on their forum here.("Setting WiFi up via the command line - Raspberry Pi Documentation," 2017)

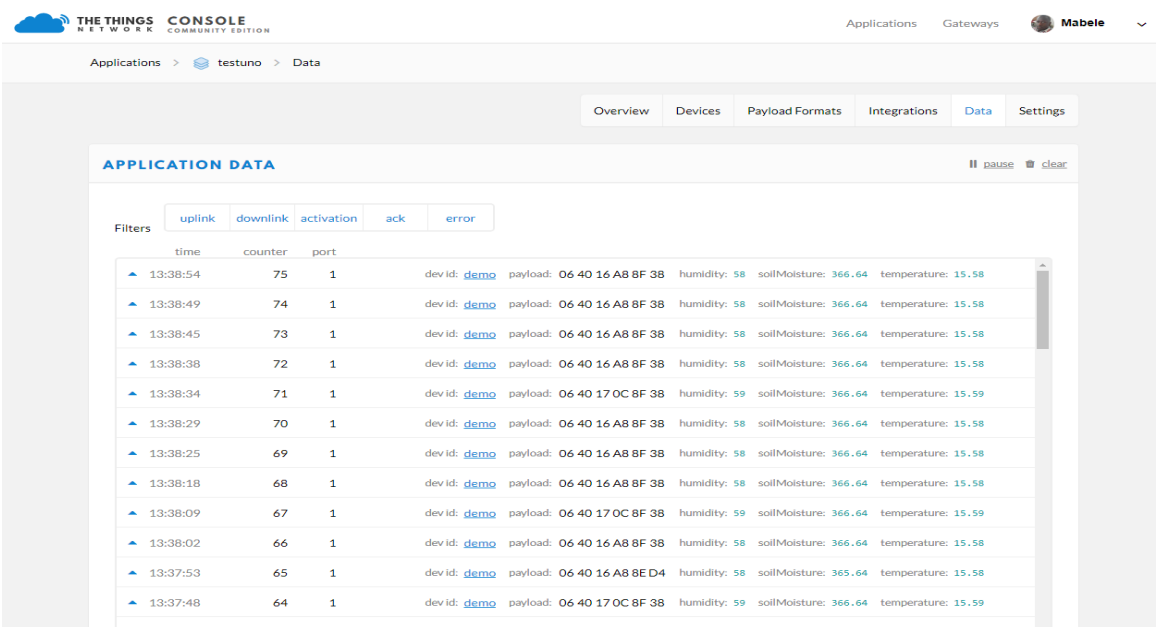
Image of the sensor node with the RN2483 LoRa module

iC880a concentrator on Raspberry Pi

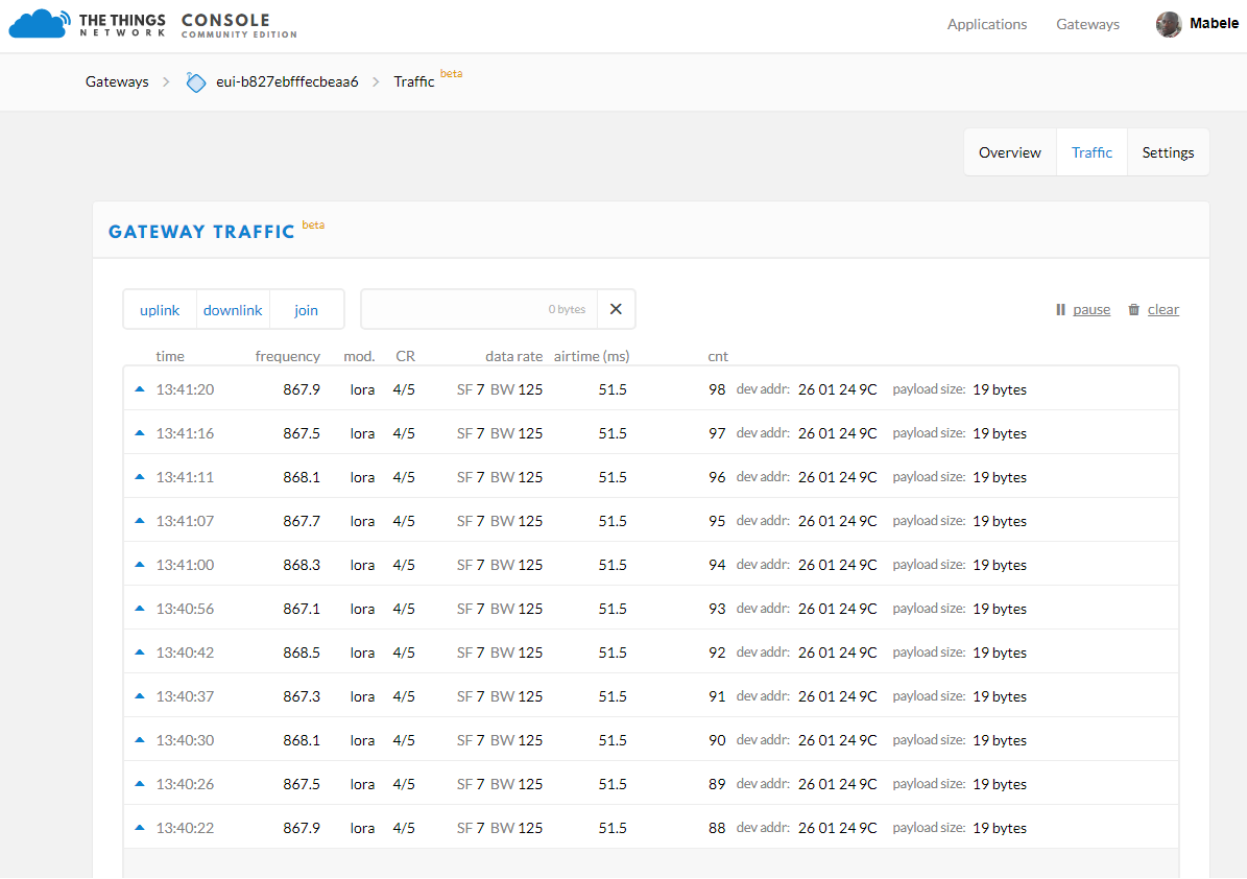
The team at TTN have already provided a very nice documentation on how to go about setting up a LoRa gateway using the iC880a concentrator and the Raspberry Pi on their GitHub page. (ic880a-gateway, November 22, 2015/2017)

The two interfaces below show how one can view the real-time data from the sensor node environment as gateway traffic and on the application registered on the things network. It is critical for one to activate the node device before adding it to the things network.

Application Data:



Gateway Traffic:



Transmitting the Data to Firebase

The real value of IoT lies in the data and the insights that can be derived from it through various analytics algorithms. Whether descriptive or predictive. In this scenario, the key interest is to try and store our sensor data to wherever location we would like to. I first tried this with MySQL and successfully managed to store the data in a relational table. I used the update query as the data will be too much since it is transmitting in seconds.

The things network team under the integrations tab on the application interface have also provided an easy way to do this. I leveraged on this and added a PUT method on HTTP integration pointing to my Firebase url where I had created the database called project e. The payload has to be in json format. (“Firebase,” 2017)

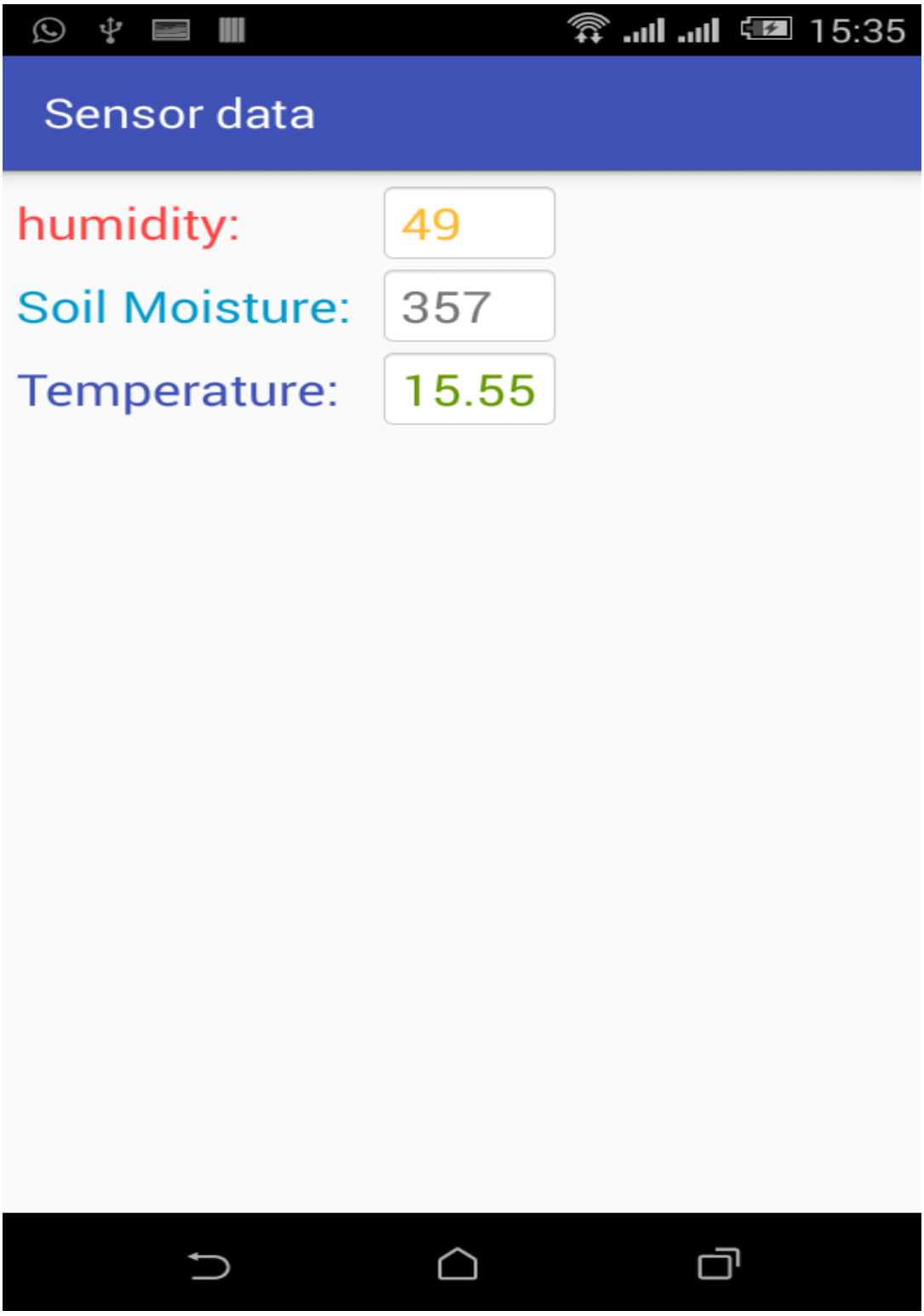
The result is as shown below:

The app_id, counter, device_id and the downlink url are all fetched from the things network. The payload is received in json which is the format used by firebase. One can decode the payload_raw to obtain the sensor values for viewing it on a web interface or mobile App. The payload_raw is in base64 format.

Presenting the Data on Android Mobile Application

In this Limabora project, I have developed a small Android application that presents the data from the greenhouse to the farmer in real-time. More features are to be added to the application and improve its user interface and possibly integrate to the actuators on watering the greenhouse though in this case the major focus was to use LoRa infrastructure to transmit data and visualize it on Android mobile application while storing it in Firebase. Historical graphs of the data and the predictive capability are also to be integrated to the application.

The code is on Github and can be accessed with the following link. The Android interface that presents this data is shown below:


15:35
Sensor data
humidity:
49
Soil Moisture:
357
Temperature:
15.55

Conclusions

The development of the Limabora solution poses a low-cost sophisticated development due to the low-power long-range capability provided by LoRa technology. In realizing this solution:

1. One needs to setup the node environment and the gateway with the best line-of-sight between them.
2. One should use an SMA-antenna with a pigtail cable to connect to the gateway.
3. Placing the gateway on a raised surface or top of a building will provide better transmission of data from the nodes as the interference is minimised.
4. The iC880a uses SPI interface on the Raspberry Pi with 5V input, hence a keener look on the pin connections is really critical to avoid damaging the concentrator.
5. LoRa technology uses Chirp spread spectrum technique. One needs to understand this properly on developing a solution using this technology.
6. LoRa infrastructure auto-sets the spreading factors, bandwidth and the coding rate between the nodes and the gateway using the Dynamic data rate concept.
7. LoRa is based on wireless RF technology and can be used in various applications of M2M and IoT such as energy monitoring, smart cities, environment monitoring etc.
8. One only needs to setup a firebase database and integrate to the things network.

Recommendations

- One should always test the nodes and the gateway to determine the best location points of placing the gateway.
- Read the datasheet and the manual of the LoRa concentrator and the RN2483 modules before diving into development. The module have a 3.3V while the gateway has a 5V input. The modules use UART interface while the concentrator uses SPI interface.
- Ensure your Wi-Fi signal is strong enough to guarantee transmission of sensor data to the cloud. Otherwise if you are in a building with good cabling, I would strongly advise one to use the Ethernet cable on the Raspberry Pi.
- Allocate a static IP to Raspberry Pi for ease of managing the gateway.

References

Firebase. (2017). Retrieved from https://firebase.google.com/

ic880a-gateway: Reference setup for iC880a gateways running The Things Network. (2017). Shell, The Things Network Zurich. Retrieved from https://github.com/ttn-zh/ic880a-gateway (Original work published November 22, 2015)

Kravets, I. (2017). PlatformIO: An open source ecosystem for IoT development. Retrieved from http://platformio.org

Network, T. T. (2017). The Things Network. Retrieved from https://www.thethingsnetwork.org/

Setting WiFi up via the command line - Raspberry Pi Documentation. (2017). Retrieved August 1, 2017, from https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md