LoRaWan connection to The Things Network

By: Johnson Domacasse (#4471709)

27 March 2024

Contents

1. Introduction:

This research will be an extension of the previous LoRa implementation that used two LoRa MCUs to connect with each other. A recap on LoRa is that it is a protocol that can be used to transfer data at a slower data rate but with increased distance. This research will be focused around LoRaWAN and connecting this to the things network (TTN). What is important to know here is that we are using a gateway specially made to connect to this network. More on this in section 4.

Compared to LoRa, LoRaWAN is a Media Access Control (MAC) layer protocol built on top of LoRa modulation. It is a software layer which defines how devices use the LoRa hardware, for example when they transmit, and the format of messages. See figure 1 for the LoRaWAN architecture.

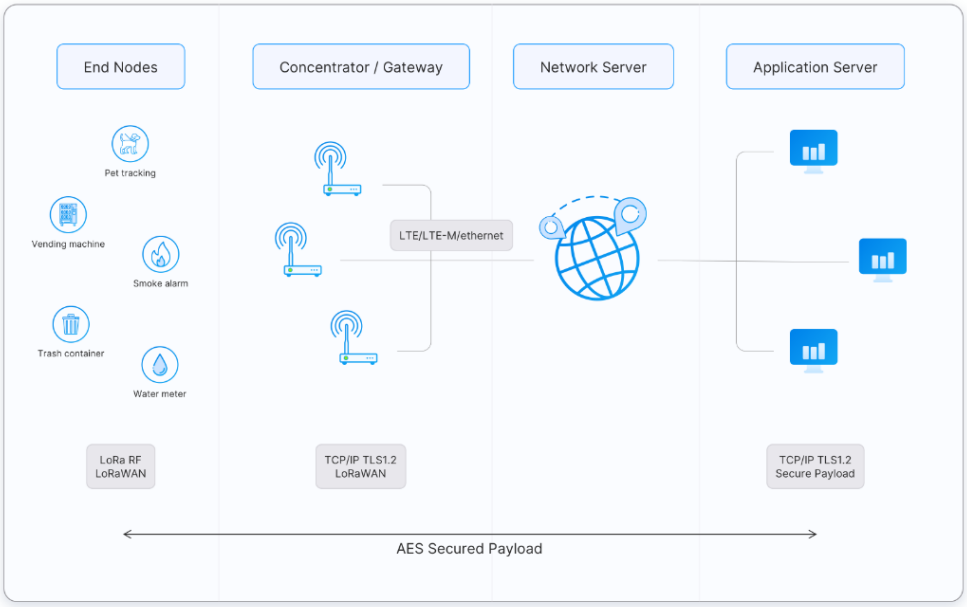

Figure 1: LoRaWAN architecture.

Here we can see that you have end nodes/devices. These act as your sensors. Via the gateway, this data is transported to the application server via the network. This type of communication is called Uplink. If the application server were to send a message to the end nodes then this would be called a downlink**[1].**

2. Methods:

This research was conducted in the Fontys R10 building. This is because there is a gateway here that offers good coverage. It is also possible to set up your own TTN gateway at home. For this research we used to following tools:

- 1x ESP32 LiLyGo TTGO LoRa MCU.
- A gateway to connect to TTN.
- Node-red
- A TTN website account.
- Code provided in the GIT.
- Arduino IDE
- MCCI LoRaWAN LMIC library.

2.1 Setting up the environment:

For the sake of this research, we will use the Arduino IDE as working with platform IO had its difficulties. See section 4. We are assuming in this scenario that you have the ESP32 boards installed in your environment. Install the LMIC library in your environment. After that we need to make some configurations. There is a file in the library that needs to be configured before starting. The file name is: lmic_project_config.h. Here you uncomment the area you are working from and comment the rest.

From there we traverse back into the TTN website account where we will set up here as well. Once signed up and logged in, we create an application. We use the manual addition option with the version being v1.0.3. The frequency here is the same as where you are working, in our case Europe. Finally using Activision by personalization, we generate the other required fields.

From there we upload the code that was provided to us in GIT. Be advised use the Arduino file code rather than the platform IO. See section 4 for more details on this. Here we change some important variable values. We change networks session key, the application session key, and the device address. These are all values that have been randomly generated when we created our application before. See figure 2.

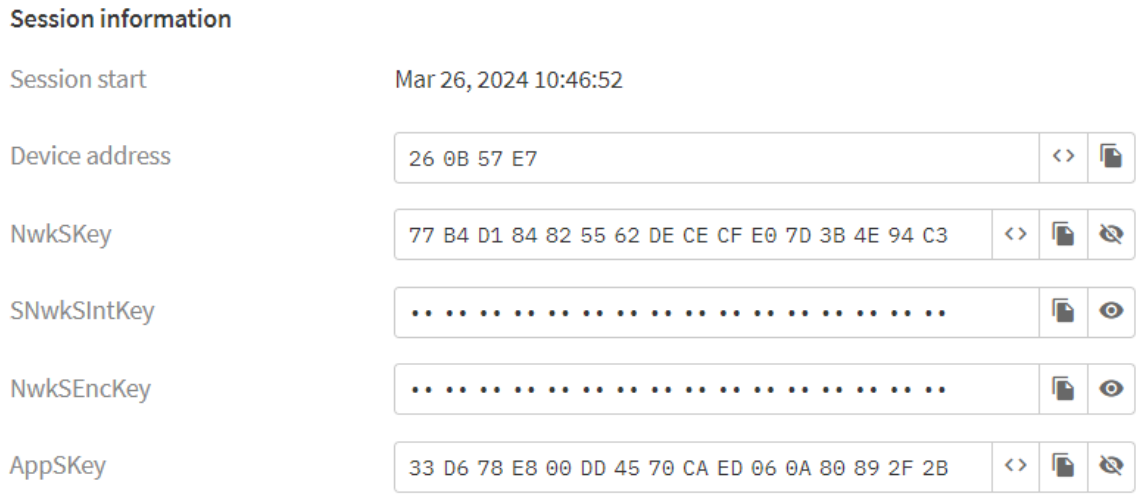

Figure 2: Generated keys used to establish a connection.

2.2 Running the program:

Be advised, upon building the code it will give you an error regarding HAL. This can be fixed by traversing to the source of the issue and renaming the HAL INIT function to something else.

After some time you will see the serial monitor being sent period messages that the packets have been queued. From here we can go back to our TTN dashboard, after some time has passed, you will notice that these packets are arriving to the network. These packets contain messages in hexadecimal values. See figure 3. This message is then copied into a hexadecimal to string converter so that we can have human readable text. Here we can see the “Hello, World!” string.

We then try to send a downlink via the dashboard to the device. We use a string to hexadecimal converter to create the hexadecimal value so this can be pasted. We then go back to our serial monitor to see if the message arrives. This takes some time before it arrives. See figure 4.

Finally we begin implementing the MQTT integration into this project so that these messages can be read in node red. To do this we need to set up the MQTT-in node, then we need to use a debug node to read the messages. We need to configure this MQTT node to receive from TTN. Inside your dashboard, under the integrations section, there is an MQTT integration. In there you will find the

URL to the server along with the code. What is important here is the topic. The topic needs to be specific to your hardware that is configured. A standard topic looks something like this: “v3/{application id}@{tenant id}/devices/{device id}”. From there we take in our application id. Our tenant id will remain: “ttn”. Finally, we put in our device id. After the device id, you have a series of command which can range from joining a topic, seeing the uplink message, the downlink message and many more. Additionally, you need to create a new API which will function as a connection password. These we will also include in the mqtt-in node so that it can connect to the network. All of this information can be found in the MQTT documentation**[2]**. See figure 5 for the initial flow.

2.3 Node-Red messages:

In your node-red application, you should see the received message from the device if everything was properly configured. See figure 5 for an example. Here what is important to us is the uplink message object. Within this object there is a key called: “frm_payload”. This is the key value we need to focus on as this is where our message is. See figure 6.

What is important to note here is that these characters combine to become a base64 encoded message. So we need to take this message and decode it. An external website that decodes this message can be used. Here you see the message being: “Hello to Smart Industry Students”. The next step is to take this message and convert it straight into a readable string in node-red it self. To do this, we take a function block and first traverse to the “frm_payload” key. We take the value and convert that into a string. From there we use the buffer function to decode the base64 message. Then we convert it back into a string. The results of the can be seen in figure 7.

3. Results:

The results that were mentioned in previous sections can be found here.

Figure 3: Hexadecimal message received from device via uplink.

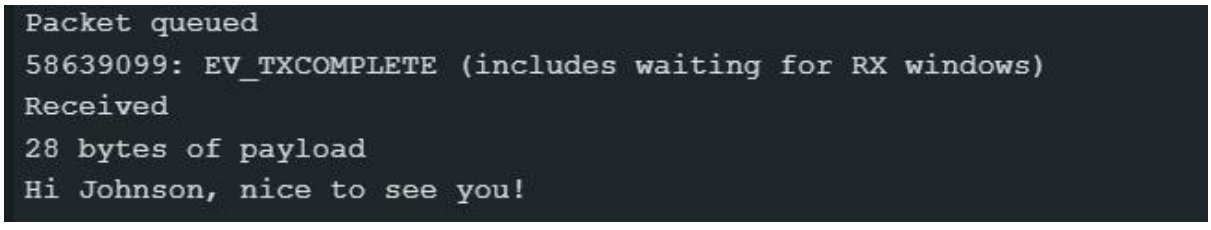

Figure 4: Hexadecimal messages sent via downlink.

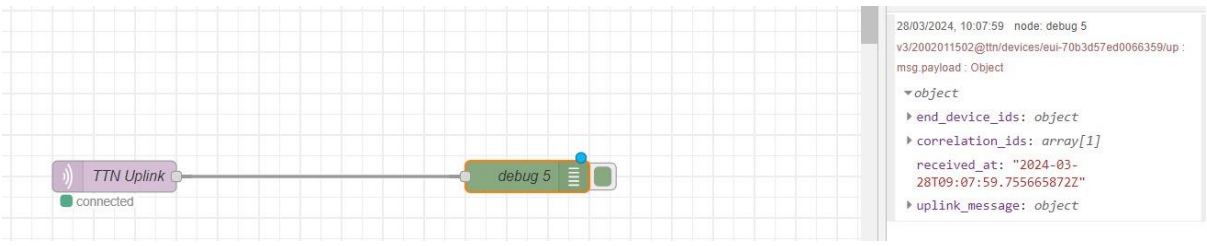

Figure 5: MQTT flow with message object.

Figure 6: Complete message object

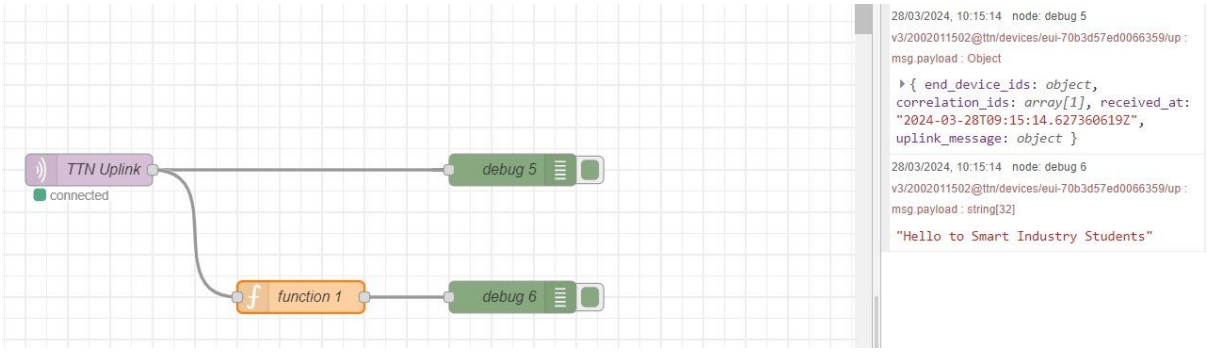

Figure 7: MQTT flow with decoded message.

4. Discussion:

The focus of this section is to discuss certain topics that came up when working on this research. These can range from problems that were found or interesting findings.

4.1 Problems occurred:

I would not classify this completely as a problem but more so as a note. As mentioned in the introduction section, this research was taken place at Fontys University, where there is a working gateway to TTN. I have not tried myself to configure my own TTN gateway however I have found a few article explaining how this can be done**[3]**.

This could be a personal problem during my own attempt however I was unable to perform this research in my own usually platform io environment. I followed the exact same steps described here and still no results. I was patiently waiting in both the serial monitor and TTN dashboard for some results. Because I received no answers, I changed to the Arduino ide for this project and followed the same steps. Be advised, Instead of using the c file in the main folder of the provided code, I used the Arduino file provided in the other directory. This proved to work.

This issue was never fixed! The display on the MCU should be showing the amount of packets that it has sent or received however this does not do so. I checked with other students performing this research and the results was the same for them.

Finally, the key to performing this researching is **Patience**. When the MCU has had the code uploaded on it for awhile and powered on, the results were showing in the serial monitor but not the TTN dashboard. The first time for me it started showing almost immediately. The second time it took around 10 minutes and the third time nearly an hour. So Patience here is the key.

5. Bibliography:

[1] - *Getting started*. The Things Stack for LoRaWAN. (n.d.-b). https://www.thethingsindustries.com/docs/getting-started/

[2] - *Mqtt Server*. The Things Stack for LoRaWAN. (n.d.). https://www.thethingsindustries.com/docs/integrations/mqtt/

[3] - *Gateways*. The Things Stack for LoRaWAN. (n.d.-a). https://www.thethingsindustries.com/docs/gateways/