Rapport de Travaux Pratiques - 2024.09.29

Internet des objets - Protocoles LoRaWAN pour la reception en longue distance de données ESP8266 et par modem LoRa E5

Table des matières

Introduction

Dans le cadre de notre parcours universitaire en Électronique et Informatique, nous sommes amenés à interagir avec des objets connectés évoluant dans des environnements peu couverts en WiFi, comme sur le site de Saint-Cyr, ou encore dans des environnements où les ondes radio sont particulièrement pertinentes du fait de leurs basses fréquences et donc meilleurs pénétrations à travers les bâtiments. Ces ondes peuvent se propager sur de longues distances tout en consommant peu d'énergie. Ainsi, nous allons nous familiariser avec le protocole LoRaWAN.

La technologie LoRa est très pertinente dans les cas où l'environnement est ouvert et où il est nécessaire de transmettre des informations sur de longues distances sans devoir installer des antennes à intervalles courts et réguliers. Par exemple, nos copains d'AGRAL pourraient avoir besoin de mettre en réseau des arroseurs automatiques pour des champs intelligents capables d'adapter l'irrigation à partir de capteurs mesurant le taux de CO2 ou l'humidité du sol, plutôt que de le faire de manière inutilement fréquente. Toujours dans une optique environnementale et économique, d'autant plus importante avec la hausse des prix de l'électricité suite aux événements en Ukraine, un éclairage public ultra-adapté pourrait fonctionner grâce à de multiples capteurs, n'éclairant que quelques lampadaires à l'avance pour les véhicules ou piétons, tout en réduisant la pollution visuelle nocturne pour le voisinage.

Pour résumer, le protocole LoRaWAN est très pertinent pour gérer les échanges entre capteurs et serveurs, notamment pour des relevés de données ou la détection de pannes et fuites énergétiques. Ce protocole permet aux capteurs de communiquer avec une passerelle, puis d'envoyer les données vers un réseau plus large. Il nous permet ainsi d'assurer une grande autonomie des capteurs à distance, tout en optimisant leur consommation d'énergie.

Pour une meilleure gestion, portativité et communication de notre projet, nous avons, sur les conseils de Thibault, mis en place un suivi avec GitHub et une remise en ligne via Google Doc.

Aide rédactionnelle, au débogage, au code et soutien : Yulin, Maxime, Ayman, Sékouba, Victor, Quentin, Nicolas, ChatGPT, HARIAN Elyoth, DOUZE Yann, Benjamin et l'incroyable Monsieur VIATEUR Sylvain.

1.Test du matériel - Tests unitaires.

Après avoir configuré l'environnement de travail disponible dans l'annexe A. Mise en place de l'environnement de travail, nous allons vérifier le capteur DHT22 et le module LoRa-E5. Dans un second temps, nous enverrons des messages depuis l'ESP pour pouvoir enfin transmettre des relevés via le capteur DHT22.

1.1. Test du capteur DHT22

Nous avons connecté le capteur, à la borne D7 du microcontrôleur, correspondant d'après la documentation technique à la sortie hardware GPIO 13, pour tester le bon fonctionnement du capteur DHT22. En annexe, nous avons la A.2. Configuration de Arduino IDE pour échanger avec l'ESP.

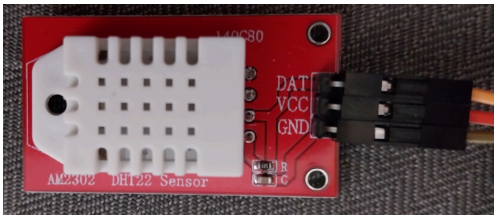


Figure 1.1.1 : Extrait de code montrant la collecte d'une variable et sa restitution.

```
27    float humidite = dht.readHumidity();
28    float temperature = dht.readTemperature();

37    // Test des donnees
38    Serial.println("Temperature : " + String(temperature)+" °C");
39    Serial.println("Humidite : " + String(humidite)+" %");
```

Figure 1.1.2 : Vu du Serial Monitor du test capteur pour les variables

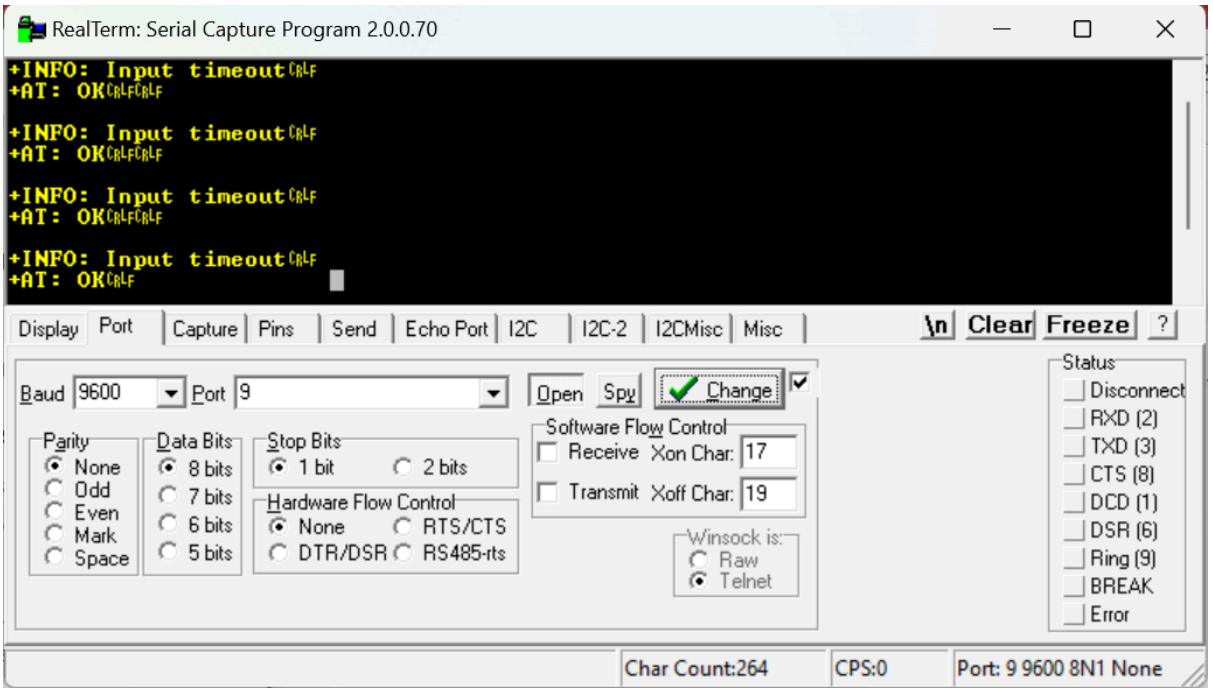
```
Output   Serial Monitor ×
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM3')
Debut des relevees capteur DTH22 avec un intervalle de mesure de 0,5 s
Temperature : 24.10 °C
Humidite : 80.70 %
Temperature : 24.10 °C
Humidite : 80.70 %
Temperature : 24.10 °C
Humidite : 82.60 %
```

Les données du capteur d'humidité et de température DHT22 fluctuent de manière cohérente et s'affichent correctement sur le Serial Monitor, ce qui valide le test.

1.2. Test et configuration du module LoRa-E5

Nous allons connecter le Grove LoRa-E5 à un convertisseur sous 3,3V pour la liaison série vers USB afin de pouvoir visualiser et communiquer avec le LoRa-E5 de manière filaire. Nous utiliserons Realterm comme terminal pour la communication série et les commandes AT du module LoRa-E5 disponible au lien suivant : AT Commandes.

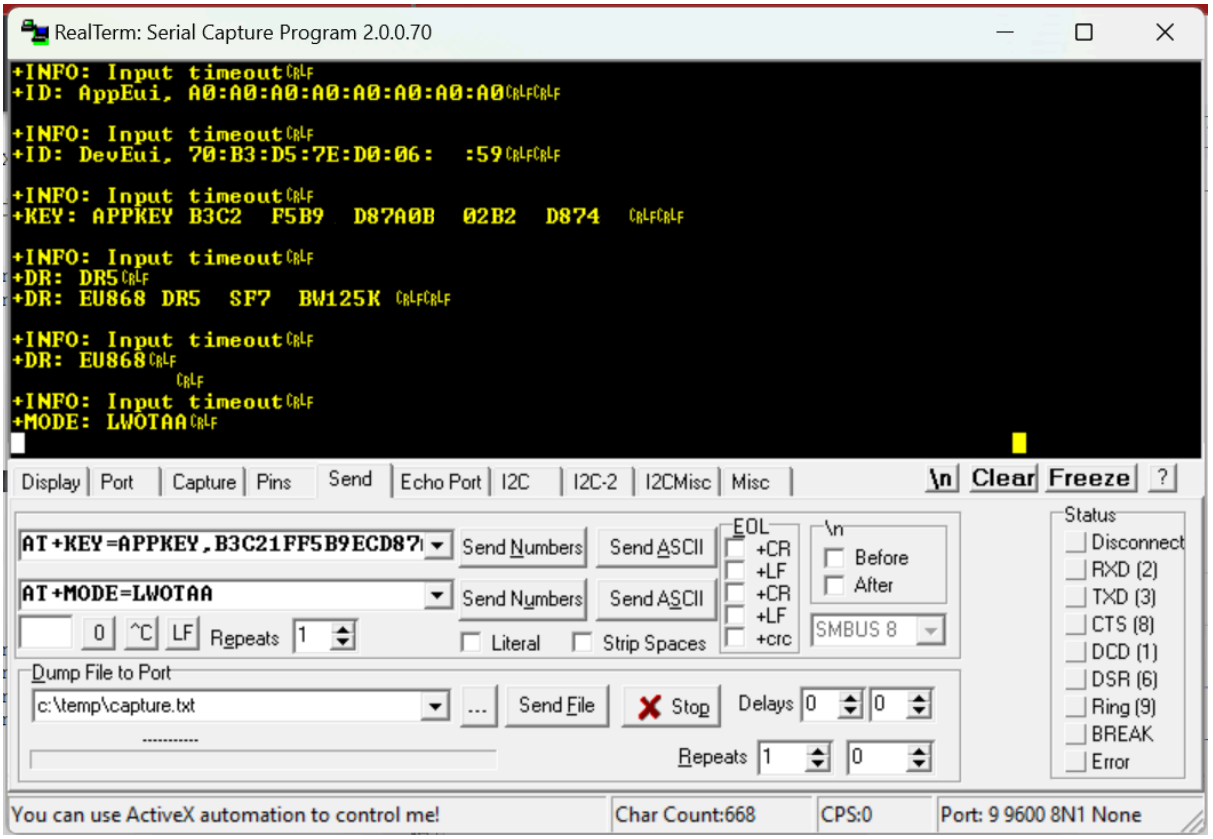
Figure 1.2.1 : Terminal de communication série montrant les paramètres de communication utilisés.



Pour tester la liaison et le bon paramétrage du terminal de communication, nous avons utilisé la commande simple “AT”, qui renvoie “OK”. Ainsi, nous pouvons conclure à la bonne connexion du convertisseur et du module LoRa avec le PC via la liaison série.

On note par ailleurs les caractères CR LF (Carriage Return Line Feed) sur le terminal, qui indiquent simplement une fin et un retour à la ligne.

Figure 1.2.2 : Terminal de communication série montrant le paramétrage du module LoRa-E5.



Pour pouvoir se connecter au réseau, nous avons relié les données du réseau TTN et du module LoRa. On note que DevAddr est l'adresse locale assignée automatiquement par le réseau via OTAA.

Enfin, nous allons chercher une passerelle pour nous connecter au réseau. Les passerelles TTN sont répertoriées à l'adresse internet : www.thethingsnetwork.org/map

Figure 1.2.3 : Carte locale des passerelles du 5ème arrondissement de Paris (France), bâtiment Esclagon, du réseau TTN.

Annexe

A. Mise en place de l'environnement de travail

A.1. Configuration du matériel TTN par étapes

Pour garantir une communication correcte entre nos relevés de capteurs et notre serveur, il est essentiel de configurer le réseau TTN (The Things Network).

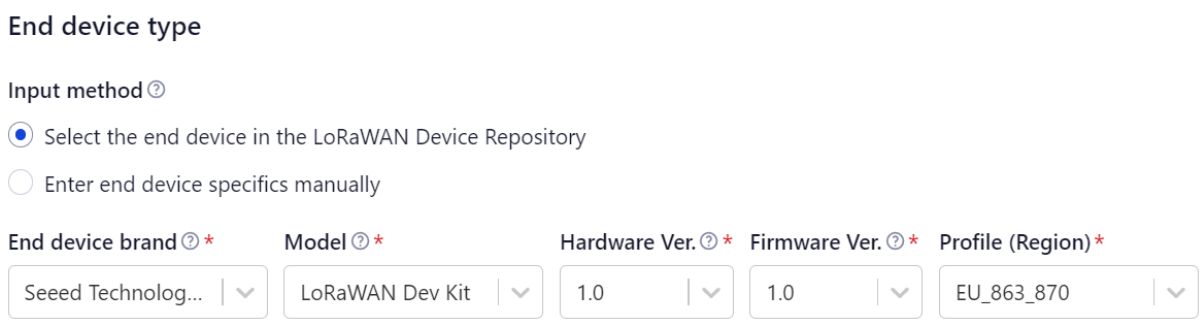
1) Création d'une application depuis le réseau communautaire The Things Network :

https://eu1.cloud.thethings.network/console/applications

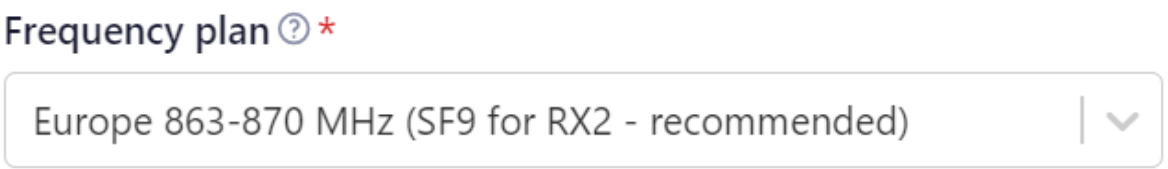
2) Ajout des modules utilisés : End devices -> Top end devices -> Register end device

https://eu1.cloud.thethings.network/console/applications/echange-lorae5-esp8266/devices/add

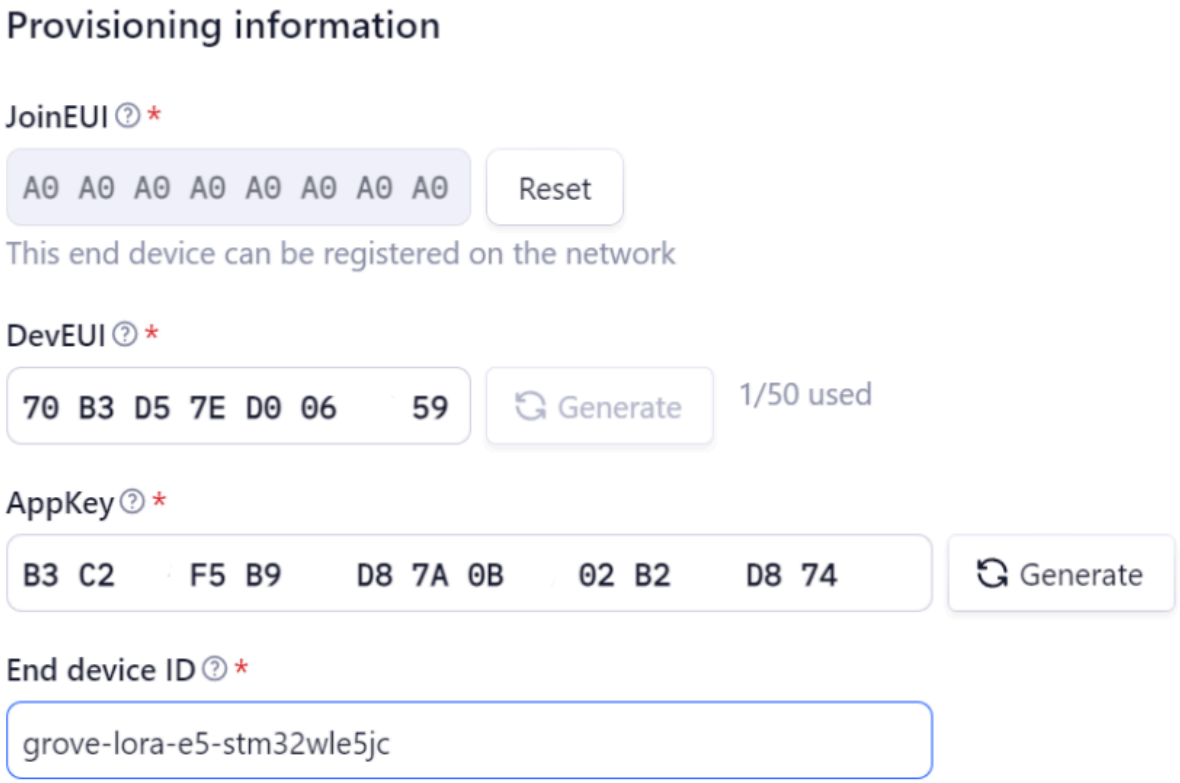
3) Selection du type du module tel que :



4) Choix de la bande de fréquence utilisée ici standard européen :



5) Generation d'identifiants réseau aléatoire pour pouvoir se connecter à l'application :



A.2. Configuration de Arduino IDE pour échanger avec l'ESP

1) Installation de Arduino IDE depuis :

https://downloads.arduino.cc/arduino-ide/arduino-

2) Dans Arduino IDE, indiquer le module à utiliser depuis File->Preferences…-> et indiquer dans « Additional boards manager URLs : » l'url depuis recherche internet «esp8266 url for arduino ide » :
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json / https://dl.espressif.com/dl/package_esp32_index.json,
http://arduino.esp8266.com/stable/package_esp8266com_index.json

sur le site
https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/ ou celui de l'ESP8266
3) Installation de la carte ESP depuis Tools->Board->Boards Manager… : « Arduino Nano ESP32 » et dans notre cas ESP8266

4) Après avoir relié le capteur avec le microcontrôleur, Tools->Board : Node MCU 1.0 (ESP-12E Module) -> esp8266 -> Node MCU 1.0 (ESP-12E Module)

5) Installation librairie DTH :

a)
https://www.gotronic.fr/art-capteur-d-humidite-et-de-t-grove-101020011-18963.htm

b) fiche technique :
http://wiki.seeedstudio.com/Grove-TemperatureAndHumidity_Sensor/

c) « **Software**

- **Step 1.** Download the Seeed DHT library » -> code -> download zip

d) Inclusion de la librairie téléchargée : Sketch -> Include Library -> Add .ZIP Library… -> Grove_Temperature_And_Humidity_Sensor-master.zip

5) Installation du driveur (si le port n'est pas détecté) depuis
https://randomnerdtutorials.com/getting-started-with-esp8266-wifi-transceiver-review/ ->
https://randomnerdtutorials.com/install-esp32-esp8266-usb-drivers-cp210x-windows/ ->
https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads ->
https://www.silabs.com/documents/public/software/CP210x_Windows_Drivers.zip

6) Connecter le Port : Tools -> Port…-> COM3