Cégep André-Laurendeau  
Département de technologie du génie physique

**Laboratoire 5 : Transmission LoRa**

Par

Grégoire Trivalle

Félix St-Gelais

Travail présenté à

M Yanick Heynemand

**Dans le cadre du cours :**

Communication des objets 4.0 (244-470-AL)

**Table des matières**

[I. Introduction 3](#_Toc196340614)

[A. Matériel requis 3](#_Toc196340615)

[B. Dates des manipulations 3](#_Toc196340616)

[II. Manipulation 1 4](#_Toc196340617)

[A. Objectif 4](#_Toc196340618)

[B. Résultat 4](#_Toc196340619)

[III. Manipulation 1 4](#_Toc196340620)

[A. Objectif 4](#_Toc196340621)

[B. Résultat 5](#_Toc196340622)

[IV. Manipulation 3 6](#_Toc196340623)

[A. Objectif 6](#_Toc196340624)

[B. Résulat 8](#_Toc196340625)

[V. Conclusion 10](#_Toc196340626)

[VI. Piste de réflexion 11](#_Toc196340627)

[A. Capteurs supplémentaires pour enrichir le système 11](#_Toc196340628)

[B. Détection d’un capteur défectueux ou hors service 11](#_Toc196340629)

# Introduction

Le but de ce laboratoire est d’explorer la technologie de transmission sans fil LoRa, souvent utilisée dans le cadre de l’Internet des Objets (IoT). Il s’agit de comprendre le fonctionnement les limites et les applications de cette technologies, d’apprendre à configurer une communications entre deux dispositifs, et de transmettre des données vers un service infonuagique ThingSpeak.

## Matériel requis

* Carte protoTPhys
* Blindage Ethernet et son adaptateur pour protoTPhys
* Câble Ethernet
* Interface de programmation Arduino
* Capteur DHT11

## Dates des manipulations

* **9/04/2025 Manipulation 1 :** Communication LoRa simple entre deux dispositifs.
* **9/04/2025 Manipulation 2 :** Transmission de données des capteurs (DHT11 et LDR) via LoRa.
* **16/04/2025 : Manipulation 3 :** envoi des données reçues vers ThingSpeak via WiFi.

# Manipulation 1

## Objectif

Mettre en place une communication LoRa entre deux dispositifs (émetteur et récepteur) à l’aide de l’exemple de *Random Nerd Tutorials*. Vérifier que les messages courts sont bien transmis et reçus.

## Résultat

Une communication a été établie avec succès. Les messages étaient bien reçus toutes les 30 secondes, conformément à la réglementation ISM.

On pourrait rajouter une validation du receveur en renvoyant un message à l’envoyeur pour valider la transmission afin de s’assurer de la conformité des données reçu.

# Manipulation 2

## Objectif

Modifier l’émetteur pour envoyer des données dynamiques issues des capteurs DHT11 et LDR. Le récepteur doit afficher ces données sur l’écran OLED, avec informations supplémentaires (uptime, ID, compteur de messages).

Au cours de cette manipulation, nous avons commencé par réaliser la lecture des capteurs côté émetteur, à savoir le capteur de température et d’humidité DHT11 ainsi que le capteur de luminosité LDR. Les données obtenues ont ensuite été organisées sous forme d’un message structuré compact, en évitant l’usage de chaînes de caractères, afin d’optimiser la transmission.

Figure 1 : Section rajouter dans l’émetteur et le récepteur pour compacter l’information

struct packet\_frame {

  float light\_level, humidity, temperature;

  uint32\_t ttl, message\_id, device\_id;

};

union packet\_union {

  packet\_frame packet;

  uint8\_t raw[sizeof(packet\_frame)];

};

Du côté récepteur, il faut aussi utiliser le code de la figure 1 afin de correctement identifier chaque valeur du message. Ensuite les données reçues ont été affichées sur un écran OLED. À noter que la directive `packed` n’est pas utilisée parce que la structure est déjà à sa taille minimum possible par défault.

De plus nous avons rajouter mis en place une gestion d’erreur pour le DHT11 au cas où il ne renvoie pas de valeur (voir figure 2).

Figure 2 : Section du code pour vérifier si le DHT11 renvoi bien une valeur

*//Lecteur des valeurs du DHT11*

    float humidity = dht.readHumidity();

    float temperature = dht.readTemperature();

*//Boucle permettant de savoir si la valeur du DHT11 est valide, sinon  recommence jusqu'à obtenir une valeur valide*

    if (isnan(humidity) || isnan(temperature)) {

      if (retry\_MSG == false) {

        Serial.println("Failed to read from DHT sensor!");

      }

      if (millis() - lastMillis\_retry\_timer > retry\_TIMER || first\_pass) {

        Serial.println("retry...");

        lastMillis\_retry\_timer = millis();

        first\_pass = false;

      }

      retry = true;

      retry\_MSG = true;

      return;

    }

    retry = false;

    retry\_MSG = false;

Cette section permet d’afficher sur le moniteur s’il y a eu une erreur lors de la prise de mesure pour ensuite réessayer jusqu’à obtenir une valeur valide. Pour cela on écrit toute les 5 secondes sur le moniteur "retry…" afin de dire qu’il continue de réessayer à obtenir une valeur.

## Résultat

Les résultats obtenus lors de cette manipulation ont été concluants. L’émetteur a correctement mesuré les données environnementales provenant du capteur DHT11 (température et humidité) et du capteur de luminosité LDR, puis a structuré ces informations dans une union pour les transmettre efficacement via LoRa. Le message, une fois reçu, a été décodé par le récepteur et affiché de manière claire sur l’écran OLED, incluant les valeurs des capteurs avec aussi d’autres informations telles que l’identifiant unique du dispositif, le numéro du message transmis et le temps de fonctionnement depuis le démarrage. Le système a fonctionné de manière stable et fiable, répondant aux attentes de cette étape du laboratoire.

Encore une fois, il peut être intéressant que le récepteur renvoie un message à l’envoyeur pour signaler qu’il a bien reçu et validé l’information, en renvoyant les valeurs qu’il a obtenues, afin que l’émetteur puisse vérifier s’il n’y a pas eu d’erreur lors de la transmission. Par ailleurs, la vérification effectuée pour le DHT11 pourrait également être appliquée au capteur de luminosité LDR.

# Manipulation 3

## Objectif

L’objectif de cette troisième manipulation est de transformer le récepteur LoRa en une véritable passerelle de communication, capable de transmettre les données reçues vers une plateforme infonuagique, en l’occurrence ThingSpeak. Pour ce faire, il s’agissait de récupérer les informations envoyées par l’émetteur via le protocole LoRa réaliser dans les manipulations précédentes, puis de les reformater afin qu’elles puissent être transmises à nouveau par Wifi à l’aide d’une requête http sur la plateforme infonuagique. Cette étape permet de relier un réseau sans fil longue portée, de type LoRa, à un service Internet accessible à distance sur un réseau, créant ainsi un pont entre un système local et une application en ligne d’analyse de données.

Avant d'établir une connexion au réseau WiFi, plusieurs bibliothèques et éléments sont initialisés afin de configurer le module ESP32 pour qu’il puisse se connecter et interagir avec le service infonuagique ThingSpeak.

Figure 3 : Adresse MAC et bibliothèque pour la Wifi

#include "ThingSpeak.h"  
#include <WiFi.h>

*// #include <WiFiMulti.h>  // en complément pour supporter plusieurs config  Wifi*

#include <esp\_wifi.h>           //requis pour changer la MAC du Wifi

#include <ArduinoHttpClient.h>  //Librairie wrapper HTTP: arduinoHttpClient v0.6.1

byte mac[] = { 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA };

const char\* myWriteAPIKey = "XXXXXXXXXXX";

WiFiClient client

Il suffit donc d’entrer l’adresse MAC du module Wifi fournies par l’enseignant. La bibliothèque ArduinoHttpClient.h est utilisée pour l’envoi de requêtes http.

Ensuite, on prépare une clé API qui permettra d’écrire des données sur le canal ThingSpeak associé à l’équipe. La librairie ThingSpeak est en charge d’utiliser le client http et la clef d’api pour intéragire avec le serveur de l’enseignant.

Une fois les bibliothèques incluses et les variables configurées, l’ESP32 est préparé pour se connecter au réseau Wifi :

WiFi.setAutoReconnect(true);

  WiFi.mode(WIFI\_STA);

  esp\_wifi\_set\_mac(WIFI\_IF\_STA, mac);  *//requis pour changer la MAC du Wifi*

  WiFi.begin("MQTT", "");

  Serial.print("Connecting to WiFi ..");

  while (WiFi.status() != WL\_CONNECTED) {

    Serial.print('.');

    delay(1000);

  }

  Serial.println(WiFi.localIP());

ThingSpeak.begin(client);

Ce segment du code permet à l’ESP32 de se connecter automatiquement au réseau Wifi "MQTT". Il configure le module en mode client, applique une adresse MAC personnalisée, et tente la connexion. Une boucle assure que l’ESP32 reste bloqué jusqu’à ce que la connexion soit établie, affichant un point chaque seconde dans le terminal. Une fois connecté, l’adresse IP obtenue est affichée pour confirmer la réussite de la connexion. Et ensuite, initialise la communication avec la plateforme ThingSpeak.

Il ne manque plus qu’à transmettre au ThingSpeak, pour cela suffis simplement d’appliquer les lignes de codes de la figure 4.

ThingSpeak.setField(1, int(p\_u.packet.device\_id));

      ThingSpeak.setField(2, p\_u.packet.humidity);

      ThingSpeak.setField(3, p\_u.packet.temperature);

      ThingSpeak.setField(4, p\_u.packet.light\_level);

      ThingSpeak.setField(5, int(p\_u.packet.message\_id));

      ThingSpeak.setField(6, int(p\_u.packet.ttl));

      int x = ThingSpeak.writeFields(1, myWriteAPIKey);

      if (x == 200) {

        Serial.println("Channel update successful.");

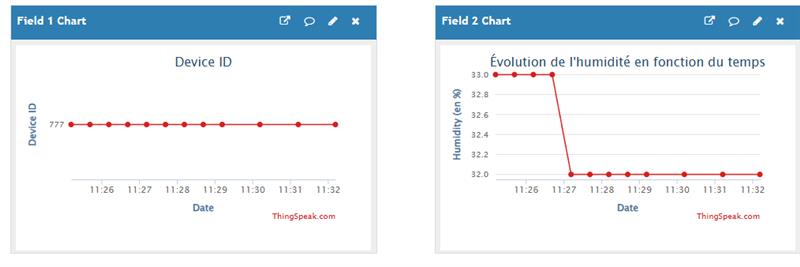
      } else {

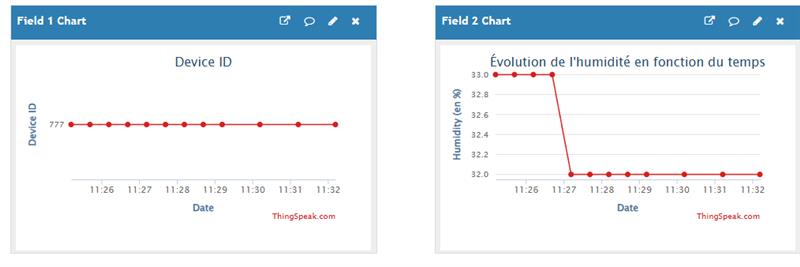
        Serial.println("Problem updating channel. HTTP error code " +  String(x));

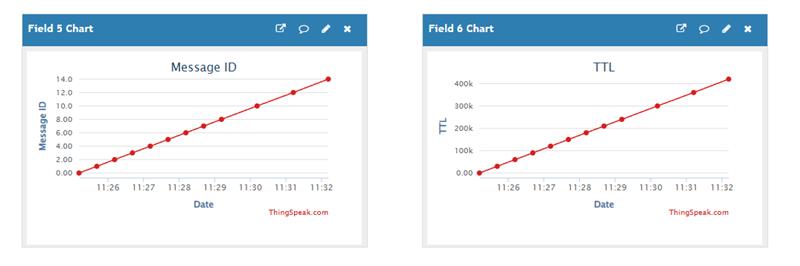
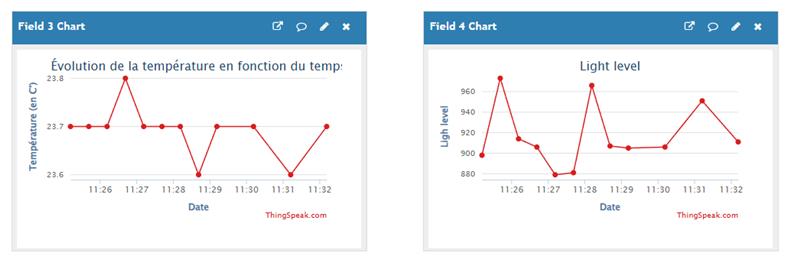
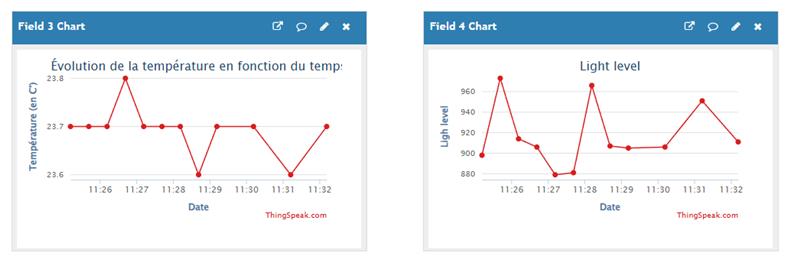
      }

    }

## Résulat

La réception des données via LoRa s’est déroulée avec succès, confirmant le bon fonctionnement de la communication sans fil entre l’émetteur et la passerelle. Une fois les données reçues, celles-ci ont été transmises correctement vers la plateforme ThingSpeak grâce à la connexion Wifi. Le réseau sans fil utilisé, nommé MQTT, nécessitait une adresse MAC spécifique qui a été configurée manuellement conformément aux consignes du laboratoire. Voici ci-dessous les figures des graphiques du ThingSeapk démontrant la bonne réception des données, ainsi que le lien du ThingSpeak lui-même : <https://thingspeak.mathworks.com/channels/2875451>





# Conclusion

Ce laboratoire nous a permis d’aborder de manière concrète la technologie de transmission sans fil LoRa, largement utilisée dans les systèmes de l’Internet des Objets (IoT). À travers les trois manipulations, nous avons progressivement mis en place un système de communication fonctionnel, allant d’une simple transmission de messages entre deux modules, jusqu’à l’intégration d’une passerelle connectée à une plateforme infonuagique.

La première manipulation nous a initiés au fonctionnement de base du module LoRa RFM95, en établissant une communication stable entre deux dispositifs. La seconde étape a enrichi cette communication en y intégrant des données environnementales dynamiques (température, humidité, luminosité) collectées par des capteurs. Nous avons également appris à structurer efficacement les messages à l’aide de struct  et union, permettant une transmission optimisée en termes de taille et de temps.

La dernière manipulation a constitué la partie la plus complète, puisqu’elle combinait les acquis des manipulations précédentes avec une transmission vers un service cloud, ThingSpeak, via le WiFi. Cela nous a permis de comprendre comment relier des systèmes embarqués locaux à des services distants pour visualiser les données en temps réel.

En somme, ce laboratoire fut particulièrement formateur et utile. Il nous a non seulement permis d’approfondir notre compréhension des communications en mettant en œuvre une solution IoT complète, de la captation jusqu’à la visualisation à distance. Ce type d’architecture ouvre la porte à de nombreuses applications concrètes, tant personnelles qu’industrielles, et constitue une base solide pour de futurs projets technologiques.

# Piste de réflexion

## Capteurs supplémentaires pour enrichir le système

Afin de rendre le système plus pertinent et adapté à des cas d’usage variés, il serait intéressant d’y intégrer d'autres types de capteurs. Par exemple, un capteur de pression atmosphérique (BMP280) pourrait apporter des données environnementales complémentaires. Cet ajout permettrait de mieux contextualiser les conditions de l’environnement surveillé, rendant le système utile dans des scénarios comme la surveillance agricole, la gestion de bâtiment ou le suivi de conditions en milieux industriels.

## Détection d’un capteur défectueux ou hors service

Pour détecter une panne, une perte d’alimentation ou un bug logiciel sur le capteur, le récepteur (ou la passerelle) pourrait vérifier périodiquement si un message a bien été reçu dans une fenêtre de temps donnée. Si aucun paquet n’est reçu après un certain délai (par exemple, 2 ou 3 fois l’intervalle prévu entre les messages), le système pourrait générer une alerte ou un message d’erreur.

Par ailleurs, le système actuel n’envoie pas d’accusé de réception à l’émetteur. Pour corriger cela, une solution simple serait que le récepteur, après avoir reçu un message, renvoie une réponse confirmant la bonne réception du paquet, contenant par exemple l’identifiant du message reçu, voir même renvoyer les valeurs afin d’ajouter une validation sur la qualité de réception. L’émetteur pourrait ainsi vérifier que ses transmissions sont bien arrivées, et éventuellement tenter une nouvelle émission en cas d’absence de réponse dans un délai donné. Ce mécanisme de communication bidirectionnelle améliorerait la fiabilité globale du système.

# Appréciation

En termes d’apprentissage utile, ce laboratoire était extrêmement pertinent. Les compétences acquises dans ce cours sont extrêmement utiles au développement d’intégrations de dispositifs dans un plus grand système. La seule amélioration notable à ce laboratoire qui autrement réponds parfaitement aux attentes serait la gestion du code partagé entre projets. En effet, le fait que du code commun soit nécessaire pour faire communiquer l’envoyeur et le receveur offre l’opportunité aux étudiants d’apprendre à écrire leurs propres librairies pour leurs besoins personnels plutôt que devoir recopier ce code et s’assurer manuellement de sa parité, ce qui peut être la source d’erreurs inattendues.