Laboratoire 5 – Transmission LoRa

Questionnement :

- Quel serait une méthode alternative de transmission de données sans fil pouvant convenir à un milieu extérieur?
- Comment programmer et gérer des communications sans fil utilisant un protocole de bas niveau?
- Quelle distance peut couvrir une transmission sans fil? Comment mesurer le niveau de signal?

Matériel requis :

- Carte protoTPhys
- Module LoRa (PMOD-RFM95)

Théorie associée :

Dans le domaine de l'IdO (IoT), il est fréquent d'utiliser des entités fonctionnant sur pile et devant être placés dans un environnement distant ou à l'extérieur. La gamme des types de communications se limite souvent au sans fil. Dans cette gamme, diverses technologies sont possibles. Dans le cadre de ce cours, nous travaillerons avec la modulation LoRa, une technique brevetée par l'entreprise Semtech et désormais disponible à bon marché et d'une bonne performance.

Avec LoRa, pas de protocole TCP/IP, donc pas de MQTT ou HTTP. Lors demeure un protocole de niveau 1 : physical. Vous disposez d'un module de télécommunication sans fil, dans la bande ISM (Industrial, Scientific and Medical) centrée sur le 915MHz (Amérique du Nord). Cette bande couvre le spectre de 902MHz à 928MHz.

Le module RFM95 est un « transceiver », un mélange de deux mots : « transmitter » et « receiver », signifiant que le module peut autant transmettre un signal RF qu'en recevoir un. Le module est fabriqué par l'entreprise HopeRF (https://www.hoperf.com/modules/lora/RFM95.html)

Pour plus de détails, consulter les notes de cours et les références fournies.

Partie 1 – La modulation LoRa

Le but de cette première partie est de vous familiariser avec votre module PMOD-RFM95 et la librairie Arduino-LoRa. Vous devez ADAPTER l'exemple de RNT à ce sujet :
https://randomnerdtutorials.com/esp32-lora-rfm95-transceiver-arduino-ide/

Vous devez réaliser deux entités qui communiquent entre eux : l'un est l'émetteur et le second est le receveur. Remarquez combien court sont les messages échangés (en nombre de bytes).

Veuillez respecter méticuleusement la règle d'envoi d'au plus de 1 message par 30 secondes (ou plus).

Pour cette réalisation, vous devez travailler en équipe de 2.

Démontrer à l'enseignant le bon fonctionnement de l'envoi et réception entre 2 protoTPhys.

Partie 2 – transmettre du contenu pertinent

Pour cette partie, vous reprendrez ce que vous avez fait avec la partie 1 mais au lieu d'envoyer un message fixe, vous enverrez un message composé dynamiquement. Comme contenu de ce message, vous utiliserez la lecture des 2 capteurs : DHT11 et le capteur lumineux (LDR).

Le receveur affichera à l'écran OLED les données reçues ainsi que quelques informations supplémentaires, dont :

- Le uptime du dispositif : nombre de secondes depuis sa mise en fonction;
- Le numéro du message : un compteur du nombre de message envoyés et reçus (vous devez compter le nombre de message envoyé ou reçu);
- Une identité du système (1 octet), identifiant unique pour une dispositif : si deux messages se croisent sur la même fréquence, on pourra distinguer le dispositif.

Le délai entre les transmissions sera au minimum de 30 secondes. Il est important de respecter cette règle car il s'agit d'une réglementation sur l'utilisation de la bande de fréquence ISM du 915MHz, par Industrie Canada. En effet, en Amérique du Nord, vous ne pouvez occuper la bande pour plus longtemps que 400ms (dwell time).

⇨ Vous devez **obligatoirement** utiliser une structure de données (*struct* et *union*), **PAS** DE CHAÎNE DE CARACTÈRES (String, char[] ou JSON) EXPLICITES. Vous êtes limités à transmettre le plus court message (nombre d'octets) en le moins de temps possible (TOA - time on air). Vous pouvez valider à l'aide de ce calculateur en ligne : https://iftnt.github.io/lora-air-time/index.html

De plus la configuration des paramètres LoRa du module sera comme suit :

- Bandwidth : **125kHz**
- Coding Rate : **4/5**
- Spreading factor : **7**
- Frequency : (à vous de choisir 1 des 64 canaux disponibles, éviter le canal 1, svp)
- syncWord : à vous de choisir; il s'agit d'une simple méthode pour « camoufler » le message de la transmission.

Faites l'essai d'aller placer l'émetteur à l'extérieur de la classe (tentez même un autre étage au cégep) et de constater la portée de la modulation LoRa. Si vous êtes assez curieux, tenter l'expérience à l'extérieur… préparez-vous en conséquence.

Présentez à l'enseignant le bon fonctionnement de votre application, émetteur-receveur.

Partie 3 – passerelle à Thingspeak

Dans cette dernière partie, vous allez concevoir un système de communication de type passerelle. Une passerelle (Gateway en anglais) permet, entre autres, d'adapter la transmission de données d'un type à un autre. Dans le cas qui nous concerne, vous allez modifier le receveur afin qu'il relaie l'information au service infonuagique Thingspeak (voir labo 3), via le WiFi. Le receveur doit donc écouter, d'une part, le message reçu de son antenne LoRa et le transformer afin que les données soient acheminées au service Thingspeak.

Pour utiliser le WiFi, celui-ci devra être initialisé lors du démarrage de l'application. Aussi, vous utiliserez le WiFi nommé « MQTT », publié sur le réseau du CÉGEP (aucun mot de passe n'est requis). Pour vous y connecter, vous devrez utiliser une adresse MAC que votre enseignant vous fournira.

Présentez à l'enseignant le résultat de votre application.

En guise de réflexion, d'analyse et pour aller plus loin

À cette étape vous avez en main un système complet qui 1) capte des données et les transmet sur un lien radiofréquence (RF) à modulation LoRa. 2) un système receveur localisé près d'un accès WiFi qui relaie l'information à un service infonuagique.

Imaginez ce que pourrait être une application utile de ce système. Donnez un exemple et décrivez cette application et son utilité.

Citez quelques exemples d'autres éléments de captation qui pourraient être intéressant d'inclure du côté capteur afin de rendre le système encore plus intéressant/pertinent.

Comment votre système détecterait le cas où le système capteur rendait l'âme, n'avait plus de pile ou rencontrerai une dérive logicielle imprévue (un bug)… et n'émettait plus?

Si vous analysez un peu plus ce système, vous pourrez remarquer que le système capteur n'est jamais informé de la réception de la transmission. **Proposez un principe qui remédierait à ce problème**.

Connaissez-vous la distance qui peut séparer le système capteur et la passerelle (transmission LoRa)? Prenez un moment pour faire des essais et mesurer la distance, tout en notant les obstacles (édifices, arbres, autres) dans le chemin de la transmission. Comment vous y prendrez-vous (outils, données)?

Remise

Pour la remise du rapport de laboratoire, PAR ÉQUIPE de 2 :

- Fournir les codes logiciels complètes de chacune des **parties 2 et 3**, séparément, bien identifiées, dans leur format original (fichiers .ino), incluant une mise en forme à l'aide du modèle (entête, section, commentaires)
- **Dans un document Word**, page de présentation, dates des manipulations, veuillez développer/expliquer le déroulement général de votre code de la **partie 3**, bien identifiée, justifiez ce que font les parties pertinentes de votre code.
- Joindre une image ainsi que le lien public d'un graphique de la partie 3.

En guise de conclusion :

- Élaborer une réponse sur les questions soulevées à la section « En guise de réflexion, d'analyse et pour aller plus loin »
- Votre appréciation personnelle de ce laboratoire, ce que vous avez appris (ou non),

Annexe 1 – structures de données en C++

(Consulter les notes de cours pour en apprendre sur l'utilisation des structures de données en langage C++)

Annexe 2 – librairies LoRa et Stream

En ce qui concerne la librairie LoRa, il y a une petite subtilité du langage C++ qui mérite une attention.

La librairie LoRa applique les principes d'héritage et de polymorphisme, propres aux langages orientés objets dont le C++ fait partie. En effet, la librairie LoRa emprunte des formes pré-définies de la grande librairie Stream. Stream défini des standards d'interfaces d'entrée et de sorties de flux de données, peu importe l'interface physique utilisée.
Par analogie, on peut dire qu'un fabriquant de tuyaux de ciment met à votre disposition un jeu de tuyaux tout en évitant de vous vendre un aqueduc, un réseau d'égouts, ou d'éléments décoratifs de terrain.

Pour en revenir à notre librairie, nous avons la librairie LoRa qui reprend des fonctions définies par la librairie (de bas niveau) Stream. Dans un certain sens, Stream offre des fonctions comme available(), readBytes(), read(), etc mais ne dit pas que ces fonctions doivent être appliquées à une interface série ou à un module LoRa, elle demeure générique.

À titre d'exemple, pensons à une interface sérielle, employant la librairie Serial. À partir d'une interface série, il est possible de connaître si une donnée y est disponible (available()), de lire un seul octet à la fois (read()), de lire l'interface jusqu'à ce qu'un caractère y soit détecté (readStringUntil()), etc. Toutes ces fonctions proviennent de Stream, et non pas de la classe Serial elle-même. Mais parce que Serial impose Stream, elle rend spécifiques des fonctions génériques.

La librairie LoRa, qui communique avec le module RFM95, voit ce dernier tout comme on verrait une interface série: des octets y entrant et en sortant. Au final, l'objet LoRa de la librairie LoRa.h vous offre l'accès à PLUS de méthodes que ce que vous trouveriez en parcourant le contenu du LoRa.h.

Et voici comment repérer cette subtilité (au cas où vous reverriez ceci dans une future vie). Dans LoRa.h, on trouve:

```
32
33  class LoRaClass : public Stream {
34  public:
35    LoRaClass();
36
```

Ce qui signifie que la LoRaClass (dont est issu l'objet LoRa) étend ses possibilités en employant les méthodes de Stream.

Et donc pour lire plus efficacement la réception de message (une structure = une série d'octets), on aura recours à la méthode **readBytes***([conteneur_octets],[nb_octets]*) de l'objet **LoRa**.

Pour en savoir plus sur Stream:
https://reference.arduino.cc/reference/en/language/functions/communication/stream/

Annexe 4 – Adresses MAC pour le WiFi et config du ESP32

L'antenne SSID nommée MQTT du CEGEP autorise seulement certaines adresses MAC très spécifiques. Pour utiliser le ESP32 en mode WiFi, il sera requis de (re)configurer l'adresse MAC du module WiFi. Voici, à droite, un bloc de 17 adresses.

168	**A0**	0x24:0x6f:0x28:0x01:0x01:0x01:0xA0
169	**A1**	0x24:0x6f:0x28:0x01:0x01:0x01:0xA1
170	**A2**	0x24:0x6f:0x28:0x01:0x01:0x01:0xA2
171	**A3**	0x24:0x6f:0x28:0x01:0x01:0x01:0xA3
172	**A4**	0x24:0x6f:0x28:0x01:0x01:0x01:0xA4
173	**A5**	0x24:0x6f:0x28:0x01:0x01:0x01:0xA5
174	**A6**	0x24:0x6f:0x28:0x01:0x01:0x01:0xA6
175	**A7**	0x24:0x6f:0x28:0x01:0x01:0x01:0xA7
176	**A8**	0x24:0x6f:0x28:0x01:0x01:0x01:0xA8
177	**A9**	0x24:0x6f:0x28:0x01:0x01:0x01:0xA9
178	**AA**	0x24:0x6f:0x28:0x01:0x01:0x01:0xAA
179	**AB**	0x24:0x6f:0x28:0x01:0x01:0x01:0xAB
180	**AC**	0x24:0x6f:0x28:0x01:0x01:0x01:0xAC
181	**AD**	0x24:0x6f:0x28:0x01:0x01:0x01:0xAD
182	**AE**	0x24:0x6f:0x28:0x01:0x01:0x01:0xAE
183	**AF**	0x24:0x6f:0x28:0x01:0x01:0x01:0xAF
184	**B0**	0x24:0x6f:0x28:0x01:0x01:0x01:0xB0

La configuration du module ESP32 doit contenir ce qui suit :

a) Inclusion d'une librairie pour accéder aux fonctions de modification :

```
#include <esp_wifi.h>
```

b) La déclaration d'un tableau de 6 octets contenant l'adresse MAC :

```
byte mac[] = {
  0x24, 0x6f, 0x28, 0x01, 0x01, 0xFE
};
```

c) Et les instructions suivantes de configuration :

```
void initWiFi() {
  WiFi.mode(WIFI_STA);
  esp_wifi_set_mac(WIFI_IF_STA, &mac[0]);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  //Serial.println(WiFi.localIP());
  nbReconnectWifi++;
}
```

L'ordre des 3 instructions indiquées en gras est importante.