POLYTECH LILLE

Université de Lille

Département IMA5 SC
Année : 2016/2017

Rapport intermédiaire de projet de fin d'études

Sujet : Relai Ethernet LoRa

Cong CHEN
Sonia NDUWAYO

Tuteur ***: Mr Xavier REDON***

Sommaire

1. Contexte

L'actuelle main mise du privé sur les réseaux IP n'est pas forcément une bonne chose pour Internet.
Sur un réseau TCP/IP, l'adresse IP est un identifiant qui permet le routage des données. Cependant, les adresses peuvent être un danger par le fait qu'elles ne changent pas.

Semtech,une entreprise grenobloise, crée en 1960 un modem radio LoRa adapté aux besoins des objets connectés qui offre une technologie de communication à faible consommation d'énergie, une longue portée(pouvant aller jusqu'à 15 km) mais avec un débit faible.

L'internet des objets est un domaine en pleine croissance qui permet entre autres d'établir des interactions entre des utilisateurs et des objets connectables.

Les domaines d'applications sont divers et variés : du contrôle des chauffages dans nos maisons en passant par les smart cities, ou encore le tracking des marchandises,…on y retrouve des dispositifs intelligents tels que des capteurs, des interfaces de communication avec d'autres dispositifs. Aujourd'hui plusieurs équipements embarqués voient le jour, cependant les protocoles de communication n'évoluent pas autant, entraînant donc une forte consommation d'énergie, un besoin énorme en termes de mémoire, réseau,..

2.Objectif

L'objectif du projet est de réaliser un module autonome connectable sur une box Internet permettant de relier deux sites avec une connexion longue distance.

3.Cahier des charges

Le projet se déroulera en deux phases. Nous utiliserons 2 technologies: L'ethernet et la communication par radio fréquences.

Nous réaliserons d'abord un premier prototype avec une architecture construite autour d'un AtMega328P, d'une carte Ethernet (avec un micro-contrôleur Microchip ENC28J60), et d'un modem radio LoRa.

Ensuite, nous réaliserons un deuxième prototype architecturé autour d'un micro-contrôleur 32 bits à processeur ARM Cortex M0 et un modem LoRa inAir9.

Nous avons choisi ce dernier processeur pour :

- le fait qu'un module Ethernet PHY soit intégré; qui opère au niveau de la couche physique du modèle OSI et met en œuvre la partie de la couche physique Ethernet du 1000BASE-T, 100BASE-TX, et les normes 10BASE-T.
- Vitesse du processeur très élevée,
- Présence de plusieurs périphériques série

En ce qui concerne l'environnement de développement,le compilateur MBED permet d'écrire des programmes en C qui peuvent être compilés en ligne.

Nous comptons également

- utiliser le protocole PoE (Power over Ethernet) pour l'alimentation par le port Ethernet,ce qui a comme avantage la facilité de gestion et de maintenance(moins de prises électriques dans une salle machines),

- prévoir un protocole de communication pour la transmission des paquets Ethernet par LoRa.

Comment fonctionne la puce LoRa?

Le SX1276 est une puce permettant des communications sans fil sur des distances allant jusqu'à 15 km, en utilisant la technologie LoRa (Long Range) de Semtech. Cette puce supporte également les modulations FSK haute performance sur les bandes de fréquence ISM (433,868 et 915MHz).

LoRa est un schéma de modulation à spectre étalé qui utilise des impulsions modulées en fréquence linéaire à large bande dont la fréquence augmente ou diminue pendant un certain temps pour coder l'information.

Les avantages de cette approche sont double:

- une augmentation substantielle de la sensibilité du récepteur due au gain de traitement de la technique du spectre étalé
- et une tolérance élevée au désalignement de la fréquence entre le récepteur et l'émetteur.

L'étalement du spectre rend le signal moins sensible aux fluctuations sélectives en fréquence. Le signal est ainsi transmis sur une bande de fréquences beaucoup plus large que la bande de fréquences nécessaire.

4.Travail effectué

Avant de commencer le projet à proprement parler, nous avons d'abord dressé une liste de tous les composants nécessaires pour la réalisation de la carte disponible sur le wiki.
Nous nous focaliserons principalement sur le premier prototype.

4.1. Partie Hardware

4.1.1. Réalisation des librairies des composants

Pour réaliser notre projet, nous avons utilisé le logiciel ***Altium*** pour désigner notre carte en passant par la réalisation des bibliothèques de certains composants. Certaines librairies sont disponibles sur le site d'***Altium*** fournies par le fabricant,pour d'autres composants dont nous ne disposions pas de librairies,nous avons dû les faire sur ***Altium***; tels que :

- régulateur de tension LD1117AV33,
- connecteur SI-50196, quartz ECSS250XM,
- module radio de type RF-LORA-SX1267 …

Pour créer des librairies, nous avons commencé par désigner un schématique du composant, ensuite nous avons fait une empreinte pour chaque composant. Une

liaison entre la schématique et l'empreinte est importante pour faire une correspondance entre les pins.
Voici un exemple à titre d'exemple du modem LoRa :

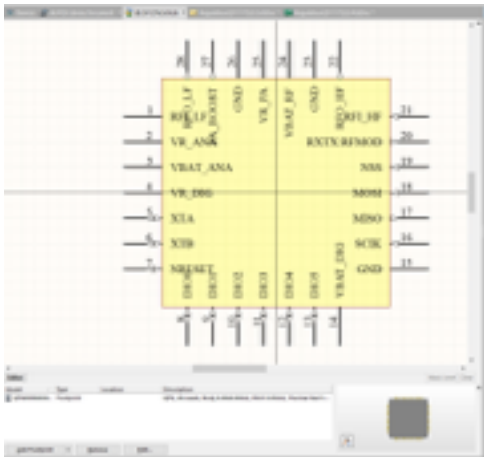


Fig.1 : Schématique du LoRa

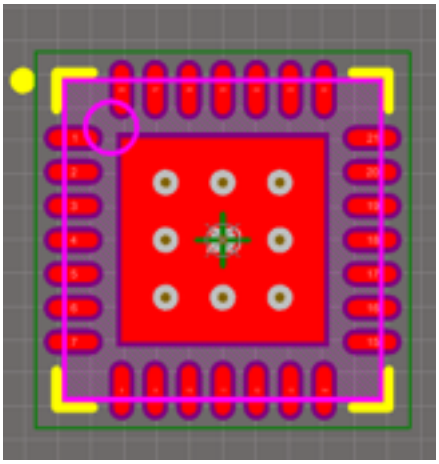


Fig.2 : Empreinte du LoRa

4.1.2. Réalisation des cartes

Après avoir réalisé toutes les librairies des composants, nous avons entamé la réalisation de la carte sur ***Altium Designer***. Comme décrit ci-haut, elle est constituée de 3 parties.

Le module Arduino est l'élément principal de notre carte. Son architecture est autour d'un Atmega 328P et est facilement programmable en C.
Il aura pour fonctions:

- de récupérer les paquets provenant du modem RF ou de la carte Ethernet (communication bidirectionnelle)
- d'effectuer des tests et vérifications(CRC, calcul RSSI,…)
- de transmettre les données à la carte Ethernet ou au modem

Nous avons d'abord défini les besoins nécessaires, entre autres :

- L'alimentation via un connecteur USB
- Un connecteur Ethernet pour l'envoi/ou la réception des paquets Ethernet
- Une antenne pour la partie RF

Pour la partie alimentation, une alimentation extérieure de 9V est prévue avec un régulateur de tension.
Pour la partie modem RF, un connecteur SMA a également été prévu pour accrocher l'antenne.

4.1.3. Principe de fonctionnement

La communication sans fils est gérée par le modem RF LoRa ***inAir9*** opérant sur des bandes de fréquences ISM (« Industrie Sciences et Médical ») autour de 433/868/915Mhz.(L'intérêt de ces modules est que la fréquence de fonctionnement peut être ajustée dans le code). Une alimentation de 3,3V et une antenne sont nécessaires pour son bon fonctionnement. Son architecture est autour du chip *SX1276.*

Exemple d'une trame envoyée par LoRa

Preamble	**Header**	**Payload**

Les différentes parties sont reliées via l'*interface SPI*. Cependant, l'arduino ne disposant que d'une seule interface(donc un chip select pour la sélection d'un slave), une variable sera donc définie dans le code comme un CS afin que l'AtMega328P(master) puisse commander 2 slaves(Ethernet et LoRa).

4.1.4. Réalisation de l'antenne

En RF, il est important d'adapter une ligne de transmission (généralement à 50 ohms) pour minimiser les pertes d'une part, d'autre part pour transmettre le maximum de puissance émise.
Nous avons donc modélisé la ligne à l'aide d'un logiciel AppCAD en essayant d'ajuster la *largeur* et l'*épaisseur* de la ligne en fonction de l'impédance désirée.

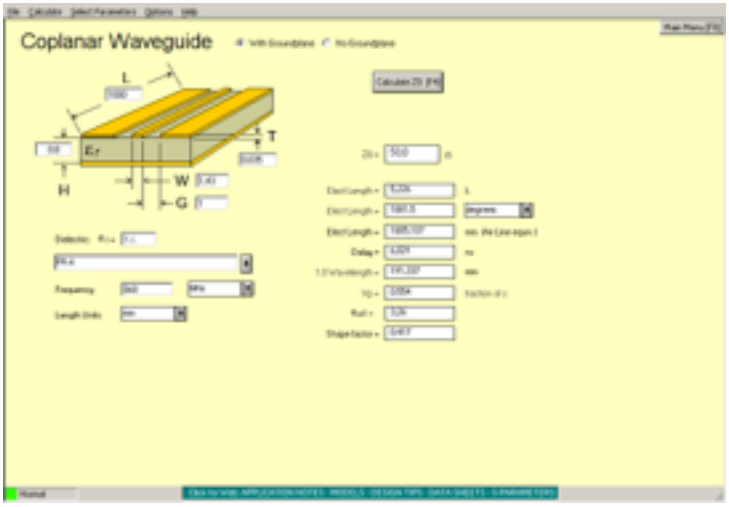


Fig.3 : Simulation ligne de transmission sur AppCAD

NB : A noter que la longueur de la ligne est un paramètre qui a peu d'importance car une ligne adaptée l'est sur toute la ligne.

Il suffira donc de connecter une antenne (soit un fil ou un connecteur SMA + antenne).

4.2. Partie Software

4.2.1. Programmation du modem LoRa

Pour plus de lisibilité, le programme a été fragmenté en 3 grandes parties.

La partie *configuration des **I/Os*** qui contient les fichiers sources pour la configuration des entrées/sorties ainsi que des registres nécessaires.
La partie *SPI* qui contient les fichiers sources des fonctions de lecture et d'écriture sur l'interface série.
La partie *modulationLora* qui contient les fichiers sources des fonctions de réception/émission des paquets reçus par l'antenne ainsi que le calcul de certains facteurs influençant la sensibilité du récepteur radio par exemple (RSSI).

Ensuite,on a créé 2 fichiers qui gèrent le client et le serveur.

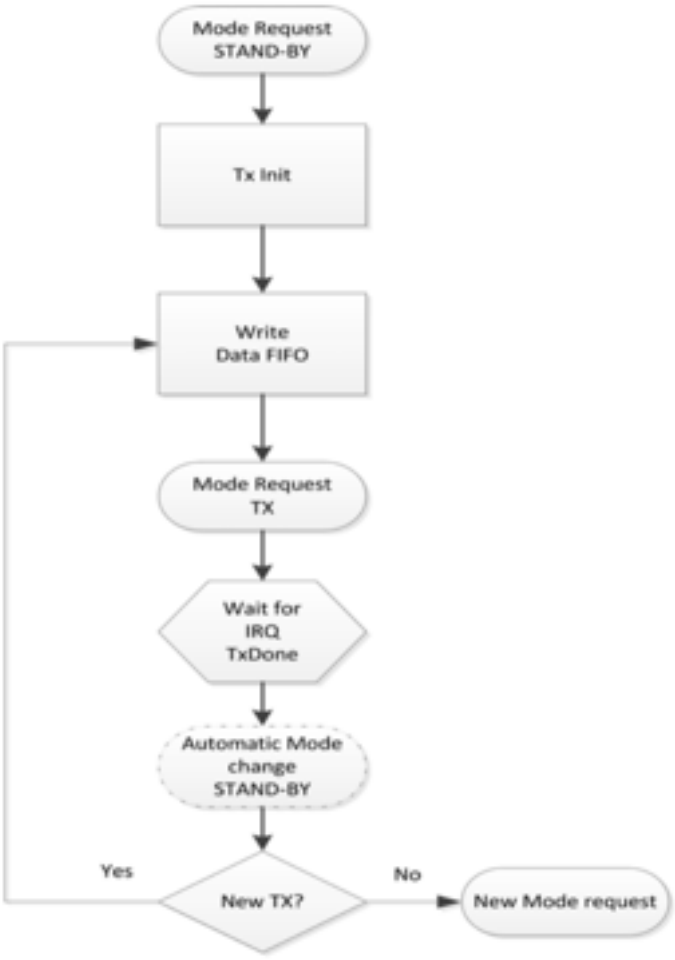
Dans le programme principal, le code configure le client en *mode **RX_continuous*** et le serveur en *mode **TX***. Un message arbitraire est chargé dans le tampon TX et ensuite envoyé après un court délai.

Les contenus du registre et du tampon *RX* sont émis en continu, de sorte que toutes les modifications apportées à la transmission des données doivent être vues là.

Un Makefile général a été écrit en C. L'utilitaire *make* d' Unix permet de compiler les fichiers sources séparément et donc de réduire le temps de compilation.

4.2.2. Machine d 'états

Fig.5 : Séquence des tâches en mode transmission



Description

Les registres statiques de configuration ne peuvent être accédés qu'en mode veille.

- La pile FIFO en mode LoRa ne peut être rempli qu'en mode veille.
- La transmission des données s'effectue par envoi d'une demande de mode TX.
- Une fois la demande effectuée, l'interruption TxDone est levée et le modem de remet en mode veille.
- Après la transmission, la radio se remet en mode veille en attente d'une demande Tx suivante.

Remplissage de la pile FIFO des données de transmission LoRa. Pour écrire des paquets de données dans la pile FIFO, l'utilisateur doit:

1. Régler le registre ***FifoPtrAddr*** sur ***FifoTxPtrBase***.
2. Ecrire des octets ***PayloadLength*** dans ***RegFifo***.

Fig.6: Séquence des tâches en mode réception.

Description

En mode réception continue, le modem balaye le canal en continu pour trouver un préambule. Chaque fois qu'un préambule est détecté, le modem l'envoie et se met en mode attente du prochain préambule.

Si la longueur du préambule dépasse la valeur anticipée définie par les registres *RegPreambleMsb* et *RegPreambleLsb*, le préambule sera supprimé et la recherche d'un préambule redémarre.

Il est également important de noter que les octets démodulés sont écrits dans la mémoire tampon de données dans l'ordre reçu. Le premier octet d'un nouveau paquet est écrit juste après le dernier octet du paquet précédent.

En mode continu, la séquence de traitement de paquets reçus est donnée ci-dessous :

1. En mode veille, on sélectionne le mode ***RXcont*.**
2. Lors de la réception d'un en-tête CRC valide, l'interruption ***RxDone*** est levée. Le mode reste en mode ***RXcont*** en attente d'un prochain paquet RX LoRa.
3. L'indicateur PayloadCrcError doit être vérifié pour l'intégrité des paquets.
4. Si le paquet a été reçu correctement, le tampon de données FIFO peut être lu.
5. Le processus de réception (étapes 2 à 4) peut être répété.

En mode continu, les informations d'état sont disponibles uniquement pour le dernier paquet reçu, càd que les registres correspondants doivent être lus avant le prochain ***RxDone***.

4.2.3. Programmation de la carte Ethernet

Nous avons écrit deux codes : un pour l'émetteur et un pour le récepteur.

Le code de l'émetteur permet de récupérer l'adresse ***MAC*** du récepteur et l'adresse ***MAC*** de l'émetteur. L'émetteur renvoie des données toutes les secondes(1s). Quant au programme pour le récepteur (*voir annexe*), il permet à ce dernier de recevoir des données qui sont envoyées par l'émetteur.

Un petit test entre ces deux cartes Ethernet réalisé sur le ***minicom -os*** a été concluant(*voir photo*).

Fig.6 : Tests des cartes Ethernet

La carte émettrice et la carte réceptrice sont connectées par un câble Ethernet.

Pour ce qui est de l'envoi des paquets Ethernet, l'algorithme ci dessous apporte une solution.

Dans une boucle, il va réaliser quatre étapes.

*Si un paquet est reçu par le ENC28j60, il est recopié dans le tableau *Ethernet*.
*Si un paquet est présent dans le tableau *Ethernet*, l'octet courant est envoyé sur le port série.
*Si un octet arrive sur le port série(liaison série/LoRa), il est ensuite stocké dans le tableau *série*.
*Enfin si le paquet dans le tableau *série* est complet, il est ensuite envoyé sur Ethernet.

Il sera donc testé sur le deuxième prototype.

5. Tests

Les commandes inAir9 ne nous étant pas encore parvenues, nous avons donc décidé de faire des tests basiques avec les ***Feather LoRa M0 433Mhz*** sur la liaison RX/TX afin de voir si les données envoyées étaient bien reçues par l'autre modem et ensuite retransmises via la liaison *SPI* sur la carte Ethernet.

6. Travail restant

Dans un premier temps, nous allons d'abord achever le travail commencé càd imprimer, réaliser une série de tests, souder et programmer la carte conçue.

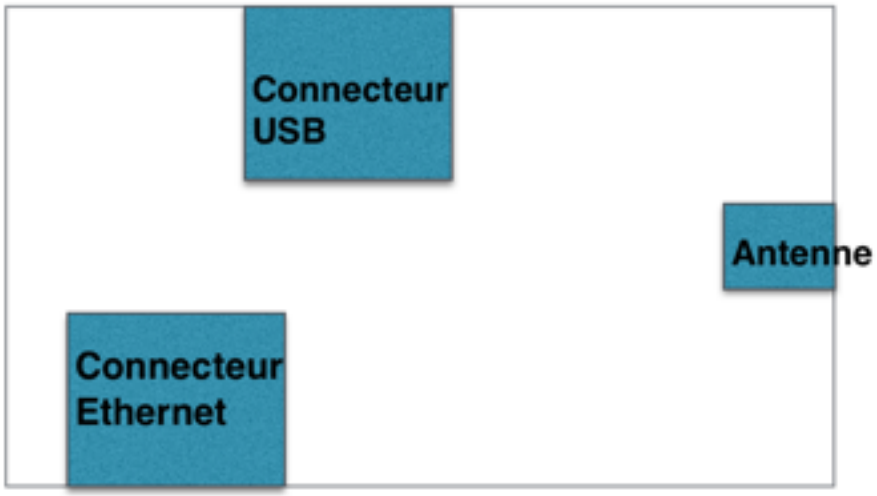
Dans un second temps, nous nous concentrerons sur le 2ème prototype.
Nous disposons déjà des boards ***mbed LPC1768*** avec un Ethernet PHY intégré.
Nous utiliserons les modules inAir9 *de chez Modtronix* comme module RF.
Le protocole PoE sera donc testé sur le 2ème prototype. Les paquets Ethernet proviendront des commutateurs disponibles en E306.

Annexes

1. Code de la carte Ethernet (récepteur)

```
#include <stdio.h>
#include <string.h>
#include <avr/io.h>
#include <util/delay.h>
#include “serial.h”
#include “enc28j60.h”
#include “spi.h”

#define MAX_PACKET          128
#define MAC_SIZE            6
#define TYPE_SIZE           2

unsigned char mac_sender[ ]={0x00,0x10,0x10,0x10,0x01};
unsigned char mac_receiver[ ]={0x00,0x10,0x10,0x10,0x02};
unsigned char packet_type[ ]={0x11,0x11};
unsigned char packet[MAX_PACKET];


int main(void)
{

        init_printf();
        spi_init();
        enc28j60Init(mac_receiver);

        unsigned char version=enc28j60getrev();
        printf(“revision: %d\n”,version);

        while(1){
                int size=enc28j60PacketReceive(MAX_PACKET,packet);
                printf(“Packet received size = %d\n”,size);
                int I,col=0;
                for(i=*MAX_SIZE+TYPE_SIZE; i<size;i++){
                        printf(“%02x”,packet[i]);
                        col++;
                        if(col==16) { printf(“\n”);col=0}
        }

        if(col>0) printf(“\n”);
        _delay_ms(1000);
        }

return 0;
}
```

2. Vue générale de la carte imprimée



3.Planning du travail restant

Dès le mois de Janvier 2017

A	B	C	D
	Tache 1	Tache 2	Tache 3
semaine 1	Finalisation de la carte	Impression de la carte pôle EEI	Programmation du MBED LPC1768
semaine 2	Soudure de la carte	Tests et correction de la carte imprimée	Programmation du MBED LPC1768
semaine 3	Programmation de la carte imprimée	Programmation de la carte imprimée	Réalisation du test avec MBED branché sur un commutateur
semaine 4	Réalisation du test avec MBED branché sur un commutateur	Programmation du MBED LPC1768-inAir9	Programmation du MBED LPC1768-inAir9
semaine 5	Réalisation du test MBED-inAir9	Réalisation du test MBED-inAir9	Réalisation du test MBED-inAir9
semaine 6	Test en milieu urbain(longue distance)MBED-inAir9	Test en milieu urbain(longue distance)MBED-inAir9	Test en milieu urbain(longue distance)MBED-inAir9
semaine 7	Recueil avis des professeurs à propos des résultats des tests	Améliorations éventuelles	
semaine 8	Rapport et la soutenance	Rapport et la soutenance	Rapport et la soutenance