 Rapport de Travail – Paul Forsans

Travail effectué :

* Restructurer architecture des fichiers Arduino en une suite de fichiers Header / Cpp (commentés et normés)
* Développement de la fonction GPS, de la fonction Accéléromètre, du stockage dynamique sur l’EEPROM, de la configuration, des presets intégrés à la carte
* Développement de la communication I²C entre la carte principale et la carte connectivité
* Création d’un système de veille pour la carte connectivité
* Créer une ébauche de fichiers mains afin de tester les cartes et les capteurs

1. **Restructurer les fichiers du projet**

Chaque Capteur / Composant a son fichier sous forme **Header .h** / **Source .cpp**

Ce système de librairies arduino fonctionne comme en C++ :

1. Appeler le fichier Header en haut du fichier : « #include ‘’\_\_\_\_.h’’ »
2. Instancier la classe/librairie : « **NomClasse** **nominstance ;** »
3. Utiliser les fonctions de la librairie sous la forme « **nominstance.fonction()** »

Liste des fichiers actuellement créés (NFC et SigFox non testés) :

Accelerometer.h

Accelerometer.cpp

GPS.h

GPS.cpp

EEPROMStorage.cpp

EEPROMStorage.h

NFC.cpp

NFC.h

SigFox.cpp

SigFox.h

Conf.h

Conf.cpp

1. **Développement des fonctions**

(voir commentaires dans les fichiers)

1. **Développement de la fonction I²C entre les cartes**

(pour plus d’infos voir code CarteMain.ino)

Lorsque la carte est réveillée par l’interrupt (voir chapitre **IV**), on ouvre la communication I²C à l’adress Hexadécimale 10.

On envoie ensuite un byte contenant les informations sur l’envoi :

Du MSB au LSB :

Bit 7 : GPS

Bit 6 : Accéléromètre

Bit 5 : BP1

Bit 4 : BP2

Bit 3 : BP3

Bit 2 -> 0 : Non attribués

En fonction de ce que la carte reçoit, elle active puis enregistre les données des capteurs à 1 dans le byte. Dans le cas des BP elle envoie juste bouton **\_** appuyé au port série.

A terme il faudra remplacer l’affichage des valeurs par leur envoi sur SigFox.

1. **Création d’un système de veille pour la carte connectivité**

La bibliothèque AVR (sur laquelle le langage arduino est basé) contient une fonction sleep :

On l’inclut avec « #include ‘’avr/sleep.h’’ »

Lorsqu’il faut endormir la carte il faut choisir le mode de sommeil :

« set\_sleep\_mode(SLEEP\_MODE\_PWR\_DOWN); »

Le mode Power Down est celui qui consomme le moins d’énergie.

Pour réveiller la carte nous utilisons une des PINs Interrupt du µC (d2 ici)

On peut créer une fonction qui s’exécutera même si le processeur est en veille.

Cette fonction se lancera lors d’un changement sur la PIN choisie. Ici nous l’activerons lorsqu’elle sera alimentée (par l’autre carte).

« attachInterrupt(digitalPinToInterrupt(2), wakeUp, RISING); »

(la fonction interrupt s’exécutera lors de la MONTEE du signal.)

ATTENTION : toute variable non locale devant être utilisée dans la fonction wakeUp doit être précédée de volatile :

« volatile bool wokenUp = false; »

1. **Création d’une ébauche de main pour les cartes**

(voir les fichiers CarteMain et CarteCapteurs) ;

Je n’ai pas eu le temps de terminer CarteMain, il faut rajouter l’interrupt et l’envoi du byte pour plusieurs cas.

En cas de problème ou de question n’hésitez pas à me contacter par mail :

[pro.forsans.paul@gmail.com](mailto:pro.forsans.paul@gmail.com)

ou par téléphone :

06.70.72.83.79

PAUL FORSANS