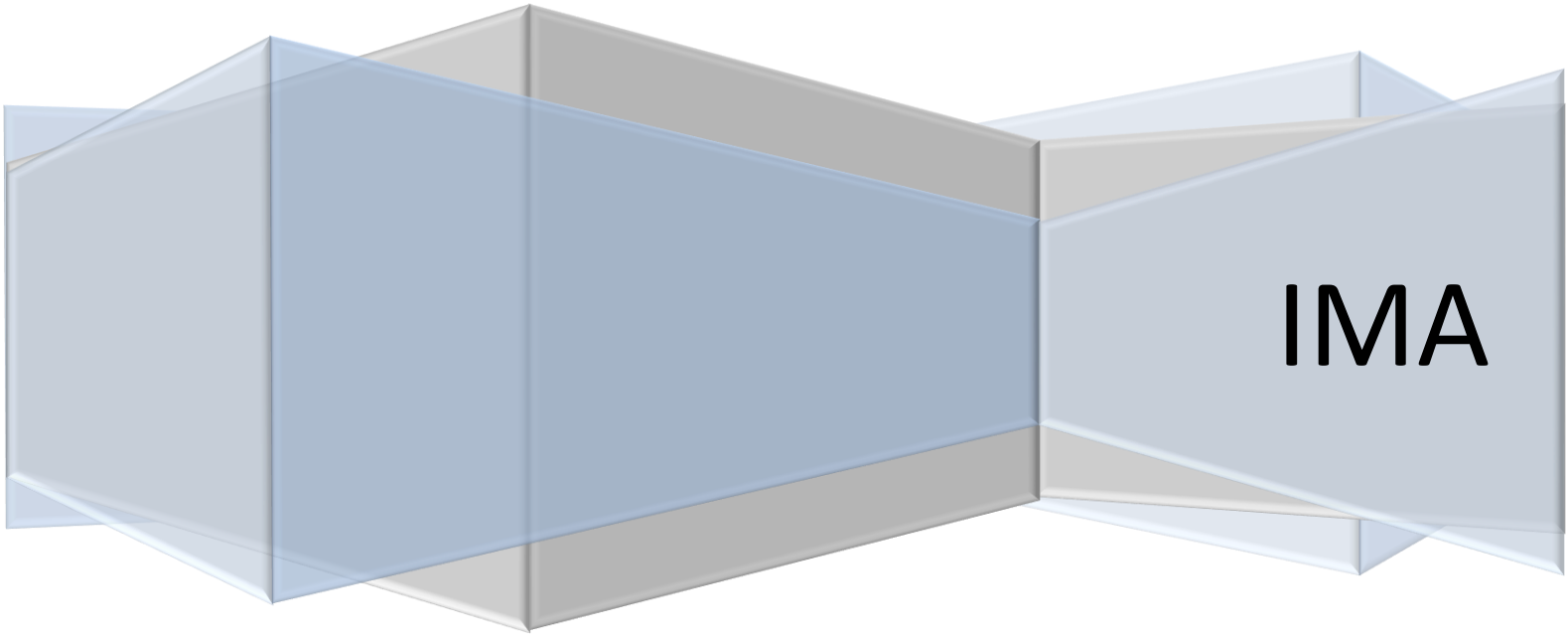
POLYTECH LILLE

Université de Lille 1 SCIENCES ET TECHNOLOGIES

Rapport Projet IMA4

Capteurs enfouis pour vieillissement du béton

Florian Giovannangeli



Encadrants projet : Alexandre BOE - Thomas VANTROYS

Introduction

Dans le cadre de ma 4ème année en Informatique Microélectronique et Automatique à l'école d'ingénieur Polytech Lille, j'ai l'occasion de travailler sur un projet lors du second semestre. Le sujet de ce projet est la réalisation d'un système permettant de contrôler le vieillissement du béton. Le système est composé d'une partie émetteur, enfouie dans le béton, qui mesure des caractéristiques physiques durant la maturation de celui-ci et d'une partie récepteur qui récupère ces données et les traite sur un PC.

Le principal objectif de ce projet est donc de réaliser un système capable de fonctionner en autonomie dans du béton durant une longue période, de mesurer des données et de transmettre ces informations à distance à travers le béton.

Pour ce faire, j'ai utilisé deux modules Adafruit Feather M0 permettant une communication radio en utilisant la technologie LoRa. Le premier module, faisant office d'émetteur, est relié à une carte comprenant les capteurs souhaités, à savoir un capteur d'humidité/température et un accéléromètre. A intervalle régulier, le module sort de son mode veille, récupère les données des capteurs et les envoie par radio. Le second module récepteur, connecté directement à un PC, reçoit les données et les transmet à l'interface de traitement pour les stocker.

Ce rapport présente l'état actuel de mon projet. Vous trouverez dans ce rapport une présentation générale du projet avec le contexte et le cahier des charges. Dans un second temps sont développés les choix et la réalisation du projet avec la partie électronique et informatique. En troisième partie sont présentés les tests effectués et les résultats. Enfin en conclusion sont exposés le bilan du projet et les perspectives d'évolution.

Contenu

Partie 1 – Présentation du Projet

1. Contexte

Les ouvrages d'art sont généralement réalisés avec du béton. Le processus de maturation du béton est un ensemble de réactions chimiques qui dépendent de nombreux paramètres. Pour être suffisamment robuste, le béton doit sécher dans de bonnes conditions (température et humidité notamment). Le suivi du séchage par la mesure de paramètres comme l'humidité et la température à l'intérieur du béton est donc important afin de pouvoir anticiper et corriger les éventuels problèmes. Concevoir un système qui mesurerait ces paramètres tout au long de la vie du bâtiment serait donc une bonne solution.

Ainsi, l'objectif de ce projet est de confectionner un prototype de capteur capable de mesurer différents paramètres physiques (humidité, température et vibrations) à l'intérieur du béton au cours de sa maturation et de son séchage, et de les transmettre à une station de base. Plusieurs parties sont attendues :

- conception du capteur à base de composants du commerce et sur des standards éprouvés,
- conception d'une interface de récupération des données issues des capteurs,
- tests en situation réelle (coulage de béton avec insertion d'un capteur).

2. Cahier des charges :

a) Contraintes

Du côté capteur, pour que le système puisse être inséré dans le béton et puisse réaliser les fonctions souhaitées, il est nécessaire qu'il soit :

- ✓ capable de transmettre régulièrement des données à travers le béton
- ✓ économe en énergie ; de façon à tenir plusieurs années
- ✓ de petite taille ; pour éviter une fragilisation de la structure en béton
- ✓ étanche ; pour éviter toute dégradation des composants
- ✓ rentable ; car il aura une unique utilisation

Pour être capable de suivre et analyser les caractéristiques mesurées, l'interface de récupération devra, quant à elle :

- ✓ être capable de recevoir en permanence les données envoyées par le(s) capteur(s)
- ✓ transmettre ces informations au PC pour le traitement et le stockage

b) Attentes

De façon à permettre un échange de données à distance, à travers du béton et entre plusieurs futurs modules, la communication se fera par Radio Fréquence (RF). Les deux modules émetteur et récepteur devront donc posséder les mêmes configurations et être capables de se reconnaître.

Dans un premier temps, à l'échelle du prototype, les caractéristiques à mesurer sont la température, l'humidité et les vibrations.

Durant la première phase de séchage, les conditions sont importantes pour garantir la qualité du béton. Pendant les 30 premiers jours après le coulage, les caractéristiques (température, humidité et vibration) seront donc mesurées et transmises toutes les heures. Par la suite, durant la maturation du béton, les données seront récupérées à hauteur d'une mesure par jour jusqu'au 100ème jour de maturation. Après ces 100 jours, les données pourront être transmises tous les mois afin de contrôler d'éventuels problèmes post-maturation. Ce capteur devra être capable d'émettre pendant 5 à 10 ans.

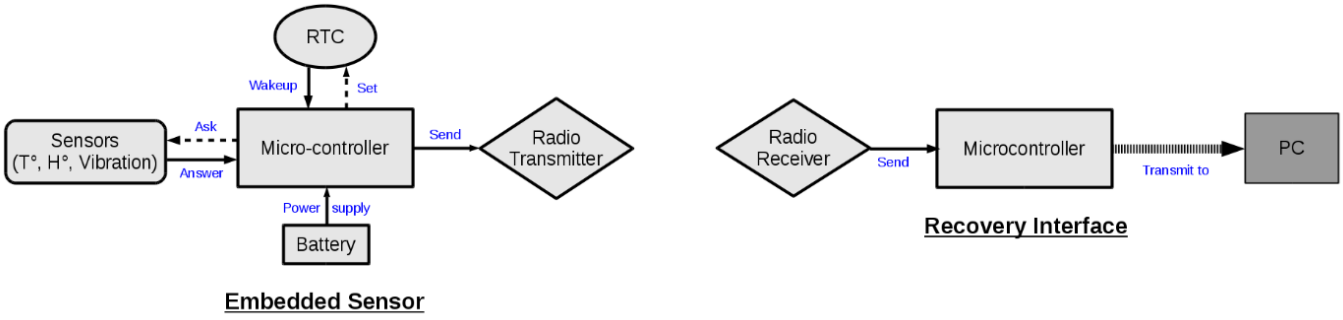


***Image 1** : Schéma de principe du système*

Partie 2 – Présentation du travail effectué

1. Partie électronique

a) Module de contrôle et de communication

Pour réaliser la communication, les deux modules doivent pouvoir transmettre des informations à distance et à travers du béton. La méthode la plus pratique est la communication par Radio Fréquence (RF). Pour ce faire, le choix s'est porté vers les cartes **Adafruit Feather M0 LoRa 433MHz** qui ont l'avantage de contenir déjà la majorité de ce qui nous intéresse. Tout d'abord elles possèdent un microcontrôleur **ATSAMD21G18 ARM** qui propose suffisamment de fonctionnalités pour permettre la commande des capteurs et le traitement des données. Elles sont aussi équipées d'un module radio **RFM9x LoRa** 868/915 MHz, qui est un émetteur/récepteur radio longue distance (Long Range « LoRa »). Ensuite il est possible de les faire fonctionner dans différents modes de fonctionnement (normal, veille...) pouvant être contrôlé par l'Horloge Temps Réel (HTR ou RTC) présente sur la carte et permettant une basse consommation en énergie. Pour finir elles possèdent également un port USART qui sert à programmer le microcontrôleur, rapatrier les données par liaison série vers le PC et connecter une batterie pour l'alimentation.

Du côté récepteur, le module Feather M0 est directement connecté à un ordinateur via un câble USB ce qui est suffisant pour recevoir les données et les transmettre au PC, ou interface de récupération, pour le traitement et le stockage. Le module émetteur est lui connecté à une autre carte de mesure contenant les capteurs, et il est alimenté par batterie via le port USB.

Afin de permettre la communication entre le récepteur et les émetteurs, il est nécessaire de câbler une antenne sous forme de fil sur le pin prévu à cet effet sur chaque module Feather. La datasheet indique la longueur exacte à câbler en fonction de la fréquence du module. Dans notre cas, les antennes mesurent 16,5 cm afin d'avoir une fréquence de 433Mhz.

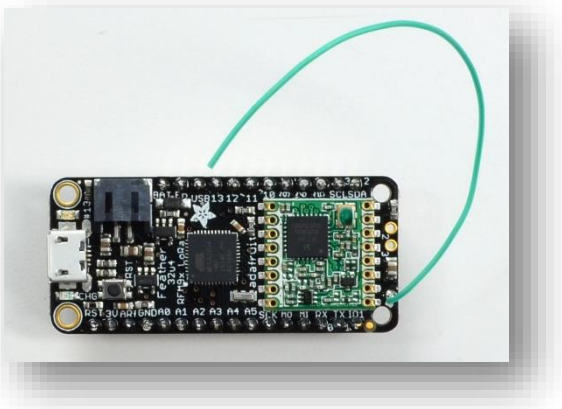


***Image 2** : Module Feather M0 LoRa 433MHz*

b) Carte de mesure

La carte de mesure doit permettre de mesurer la température, l'humidité et les vibrations tout en étant enfouie dans le béton. Pour répondre ainsi au cahier des charges, 2 capteurs ont été choisis : le capteur d'humidité/température HTU21D et l'accéléromètre FXLS8471Q. Ces capteurs sont résistants à de fortes températures et un fort taux d'humidité, ce qui leur permet de ne pas être détériorés lors du coulage du béton et sa maturation par la suite. De plus, ils sont tous les deux programmables en I2C, ce qui facilite leur programmation et leur contrôle par le microcontrôleur. Enfin, ils sont de petite taille et peu gourmands en énergie.

La carte de mesure doit également pouvoir se fixer au module Feather M0. Il a donc été décidé de procéder avec des connecteurs femelles sur la carte de mesure placés symétriquement aux connecteurs mâles du Feather. Malgré le fait que cette solution oblige la carte de mesure à être de la même taille que le Feather prenant ainsi plus de place, cela permet une bonne et solide fixation.

En respectant les fonctions souhaitées et les Datasheets des deux capteurs préalablement étudiées, les schémas du circuit de la carte de mesure sont les suivants :

***Image 3** : Schéma de l'accéléromètre FXLS8471Q*

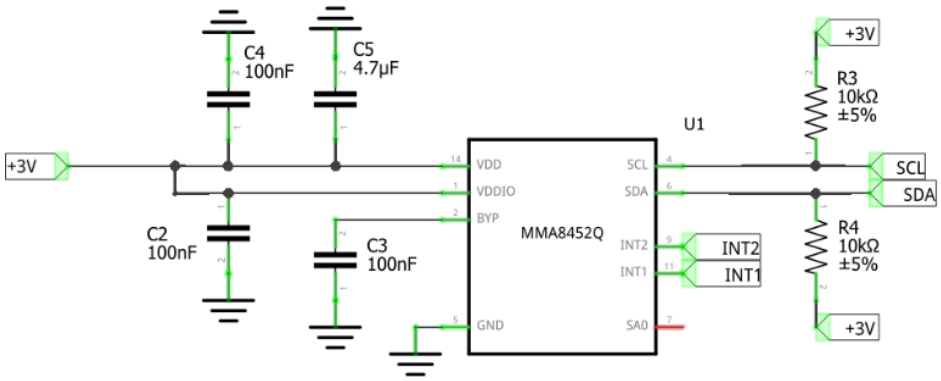
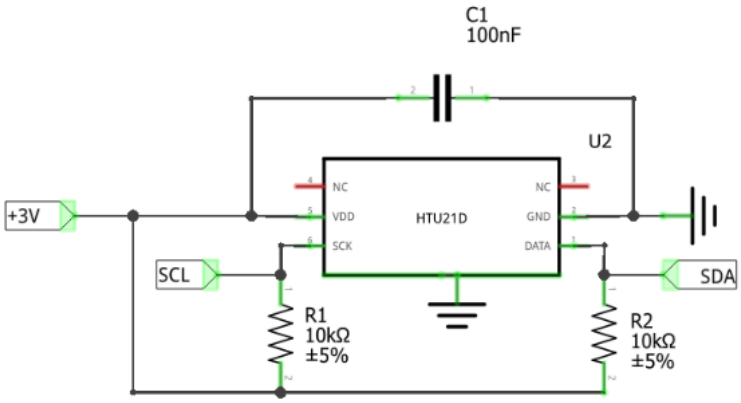


Image 4 : Schéma du capteur HTU21D



Les entrées/sorties **+3V**, **GND**, **SDA** et **SCL** sont connectées aux broches des connecteurs reliées au pin correspondant du module Feather M0. Les entrées **INT1**, **INT2** et **SA0** de l'accéléromètre ont été reliées à des fils flottants permettant ainsi de changer leur valeur (0 ou 1) en cas de besoin durant les tests du prototype.

Le PCB résultant de ce schématique est le suivant :

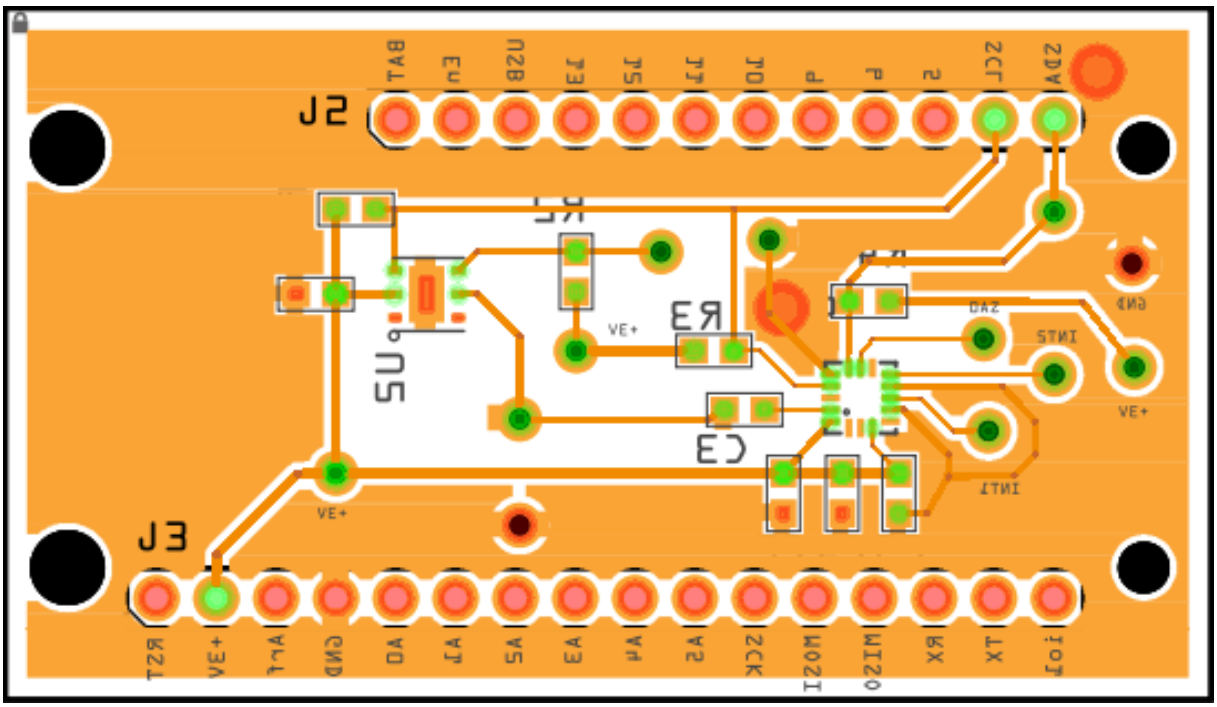


Image 5 : PCB de la carte de mesure (partie inférieure)

Certaines connexions ne sont pas visibles sur le PCB entre les vias car la carte a été imprimée en simple face et les connexions manquantes ont été réalisées à l'aide de fils sur la partie supérieure de la carte.

La carte a ensuite été imprimée au service EEI de l'école et câblée au four avec l'aide de M.Flamen. Une photo du résultat est disponible en page 10 (Image 9).

2. Partie informatique

a) IDE Arduino

La programmation des 2 modules émetteur/récepteur se fait à l'aide de l'**IDE Arduino** de version 1.6.4 ou plus. Il est nécessaire dans un premier temps de configurer l'IDE pour qu'il puisse reconnaître et utiliser les cartes Adafruit Feather M0. La configuration se fait dans les préférences (*Fichier > Préférences*) où il faut ajouter dans le gestionnaire de cartes supplémentaires la librairie qui contient le package **Adafruit SAMD Boards** permettant d'utiliser les Feather M0 :

https://adafruit.github.io/arduino-board-index/package_adafruit_index.json

Une fois la configuration faite, la carte Adafruit Feather M0 est disponible dans les outils de programmation et les programmes peuvent être téléversés dans les modules par USB.

b) Programmation des capteurs

Sur l'IDE Arduino, de nombreuses librairies sont disponibles afin de programmer et d'utiliser les différentes fonctionnalités de la carte, ainsi que les composants supplémentaires tels que les capteurs utilisés. Pour l'utilisation de la carte de mesure, il faut donc intégrer 2 librairies :

- SparkFun_HTU21D_Breakout_Arduino_Library-master
  - SparkFunHTU21D.cpp
  - SparkFunHTU21D.h
- FXLS8471Q-master-P
  - FXLS8471Q-P.ccp
  - FXLS8471Q-P.h

La librairie du HTU21D est disponible sur Internet. Celle de l'accéléromètre FXLS8471Q en revanche était codée pour une connexion SPI, or mes deux capteurs sont en I2C, je n'ai donc pas pu l'utiliser. Je m'en suis tout de même servi, ainsi qu'avec d'autres exemples trouvés et surtout à l'aide la Datasheet, pour réécrire ma propre librairie permettant cette fois une configuration et une utilisation des fonctions en I2C. Ces programmes contiennent la déclaration et la configuration de tous les registres, variables et fonctions permettant de faire fonctionner les capteurs. J'ai donc pu ensuite sélectionner les fonctions souhaitées et les intégrer à mon programme d'Emission dans l'IDE Arduino.

Dans un premier, ne sachant pas comment exploiter efficacement les mesures de vibration fournies par l'accéléromètre, seules les sensibilités en X, Y et Z seront mesurées et transmises.

c) Programmation de la RTC

Pour répondre au cahier des charges et réaliser un module économe en énergie envoyant des données à intervalles bien définis, il est nécessaire d'utiliser la **RTC** (ou Horloge Temps Réel) présente sur la carte Feather M0. En effet, la RTC est une horloge interne à basse consommation qui permet de connaître le temps et la date après la mise en route du module. Pour ce faire, j'ai utilisé la librairie RTCZero-master, initialement prévue pour Arduino UNO mais qui fonctionne sur les Feather.

Cette librairie permet dans un premier temps de régler l'heure et la date de la RTC lors de la mise en route, puis de l'utiliser pour régler des alarmes. Une autre possibilité donnée par cette librairie, et qui nous intéresse tout particulièrement, est de mettre tout le module en mode standby et de le réveiller grâce à une interruption générée par les alarmes. Ce mode standby permet ainsi de réaliser une importante économie d'énergie.

Les différentes étapes d'envoi des données définies dans le cahier des charges sont ensuite gérées à l'aide d'un compteur de jours qui configure la prochaine alarme en fonction du cas dans lequel on se trouve.

```
198     /* CHOICE OF CASE FOR STANDBY */
199     if (nbDay >= 100){       // Step 3 (>100 days) = Data once a month
200       alarmHour = 8 ;
201       alarmMonth += 1 % 12 ;
202     }
203     if (nbDay >= 30 && nbDay < 100){  // Step 2 (>30 days) = Data once a day
204       alarmDay += 1 % 30 ;
205       alarmHour = 8 ;
206       nbDay += 1;
207     }
208     else      // Step 1 (Start -> 30 days) = Data once per hour
209     {
210       alarmHour += 1 ;
211       alarmHour = alarmHour % 24; // checks for roll over 24 hours and corrects
212       nbHour +=1 ;
213     }
214
215     // Calculation of the number of days from the beginning
216     if (nbHour == 24){
217       nbHour = 0 ;
218       nbDay += 1 ;
219     }
```

***Image 6** : Réglage de l'alarme en fonction des étapes*

d) Réalisation de la trame de données

L'échange de données entre les différents Feather M0 émetteur et le récepteur se fait grâce à des fonctions définies dans la librairie nommée **RadioHead**, et plus précisément les programmes *RH_RF95.c* et *RH_RF95.h* spécifiques aux modules utilisés. Dans le cas de ce projet, la trame qui est envoyée par radio est la suivante :

HEADER	Temp	Humid	SensX	SensY	SensZ	CRC

Le Header et le CRC sont définis dans la librairie et sont gérés automatiquement par l'émetteur et le récepteur, il n'est donc pas nécessaire de les manipuler ou les modifier dans l'IDE Arduino. En revanche la trame des données est à définir selon ses besoins. Les fonctions de RadioHead qui gèrent l'envoi et la réception de cette trame sont définies pour transmettre des entiers non signés 8bits, or les valeurs reçues des capteurs sont des réels. Il est donc nécessaire de les convertir sans risquer de perdre une partie de ces valeurs. Pour cela j'ai utilisé des pointeurs et des casts pour les convertir, puis des fonctions **memcpy** qui permettent de copier les variables bit par bit dans un buffer.

Du côté récepteur, il est donc nécessaire d'effectuer la même méthode dans le sens inverse afin de récupérer les différents bits des variables et ainsi pouvoir reformer la valeur réelle exacte.

```
/* READING AND SENDING DATA */
float Temp = myHumidity.readTemperature() ;
unsigned char* pT = (unsigned char*)&Temp ;

float Humd = myHumidity.readHumidity() ;
unsigned char* pH = (unsigned char*)&Humd ;

float X;
unsigned char* pX = (unsigned char*)&X ;
float Y;
unsigned char* pY = (unsigned char*)&Y ;
float Z;
unsigned char* pZ = (unsigned char*)&Z ;
myAccel.ReadAccel(&X, &Y, &Z);

uint8_t RadioPacket[20];    // Packet to send


memcpy(RadioPacket,pT,sizeof(Temp));                    //Copy 4 byt
memcpy(RadioPacket+sizeof(Temp),pH,sizeof(Humd));       //Copy 4 byt
memcpy(RadioPacket+sizeof(Temp)+sizeof(Humd),pX,sizeof(X));
memcpy(RadioPacket+sizeof(Temp)+sizeof(Humd)+sizeof(X),pY,sizeof(Y))
memcpy(RadioPacket+sizeof(Temp)+sizeof(Humd)+sizeof(X)+sizeof(Y),pZ,

rf95.send((uint8_t *)&RadioPacket, sizeof(RadioPacket)); // Sending

rf95.waitPacketSent();  // Data sent
delay(10);
```

***Image 7** : Lecture et envoi des données*

```
if (rf95.available())
{
  // Should be a message for us now
  uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
  uint8_t len = sizeof(buf);

  if (rf95.recv(buf, &len))
  {
    digitalWrite(LED, HIGH);
    n++ ;

    delay(10);

    RH_RF95::printBuffer("Received: ", buf, len);

    memcpy(T.pT,buf,4);      // Copy 4 first bytes for Temp
    memcpy(H.pH,buf+4, 4);   // Copy 4 next bytes for Humid
    memcpy(X.pX,buf+8, 4);   // Copy 4 next bytes for X
    memcpy(Y.pY,buf+12, 4);  // Copy 4 next bytes for Y
    memcpy(Z.pZ,buf+16, 4);  // Copy 4 next bytes for Z
```

***Image 8** : Réception des données*

e) Interface de récupération des données

Du côté du récepteur, le module connecté directement au PC reçoit les données de l'émetteur et peut les imprimer sur le port Série. L'IDE Arduino propose d'ailleurs une interface Série permettant d'afficher toutes les données reçues. Or les données ainsi affichées ne sont pas enregistrées et il suffit de fermer l'IDE pour les perdre rendant leur exploitation impossible.

Une première solution consiste à utiliser un autre logiciel de lecture du port Série, tel que **PuTTY**, qui donne la possibilité d'enregistrer toutes les informations reçues dans un fichier .log (équivalent d'un fichier texte). Cette solution fonctionne parfaitement mais elle rend difficile l'exploitation des données en partant d'un fichier texte.

Une autre solution trouvée pour résoudre ce problème est l'utilisation d'une macro Tableur (Excel ou LibreOffice) qui est connectée au port Série et qui, selon une typologie bien définie, copie les valeurs reçues dans le tableur. J'ai pu ainsi trouver un exemple de ces macros appelé **OpenDaqCalc** sur **LibreOffice**. En modifiant cette macro pour l'adapter aux besoins du projet et en intégrant la typologie dans mon programme Arduino récepteur, il est alors possible d'enregistrer toutes les données et de les exploiter avec des graphiques par exemple. De plus une fonction de cette macro permet d'imprimer l'heure et la date de l'ordinateur sur lequel elle se trouve au moment de la réception des données permettant, en cas de besoin, de retrouver une mesure à un moment donné.

Un exemple du tableur ainsi obtenu est disponible dans la prochaine partie (Image 10).

Partie 3 – Tests et résultats

1. Tests hors béton

Les premiers tests du système ont été effectués hors béton dans la salle de projet en salle E311 de l'école. Le module émetteur est branché sur une batterie LogiLink et le module de récupération des données est connecté directement à l'ordinateur et transmet les informations reçues à la macro LibreOffice. Afin d'avoir des données plus rapidement, l'intervalle d'envoi des données ne suit pas les différentes étapes prévues dans le cahier des charges, mais a été changé à un envoi par minute.

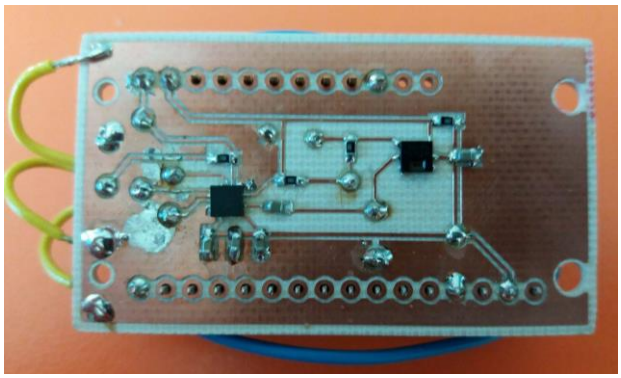


***Image 9** : Carte de mesure câblée et en test*

Les résultats obtenus sont cohérents. La température obtenue reste aux alentours de 28°C, température de la salle, et l'humidité autour de 34%. Des tests en soufflant sur les capteurs ont été effectués et démontrent des changements effectifs de la température. Concernant l'accéléromètre, les sensibilités, ou positions X Y et Z, exprimées en mg/LSB sont proportionnelles aux valeurs fournies dans la Datasheet et varient lors d'un déplacement du module.

Num	Date	Time	Temp	Humid	sensX	sensY	sensZ
1	15/06/2017	14:20:05	29,02	34,27	0,08	0,93	0,34
2	15/06/2017	14:20:45	29,08	32,86	0,07	0,93	0,35
3	15/06/2017	14:21:45	29,06	34,96	0,06	0,93	0,37
4	15/06/2017	14:22:50	29	33,03	0,05	0,92	0,37
5	15/06/2017	14:23:45	28,84	32,99	0,05	0,92	0,37
6	15/06/2017	14:24:45	28,77	32,79	0,06	0,92	0,37
7	15/06/2017	14:25:45	28,77	32,8	0,04	-0,96	0,44
8	15/06/2017	14:26:45	28,69	33,26	0,04	-0,95	0,43
9	15/06/2017	14:27:45	28,57	33,34	0,04	-0,95	0,43
10	15/06/2017	14:28:45	28,56	33,57	0,04	-0,96	0,43
11	15/06/2017	14:29:45	28,54	33,29	0,04	-0,96	0,43
12	15/06/2017	14:30:45	28,49	33,32	0,04	-0,96	0,43
13	15/06/2017	14:31:45	28,38	33,18	0,04	-0,96	0,44
14	15/06/2017	14:32:45	28,35	33,6	0,04	-0,96	0,43
15	15/06/2017	14:33:45	28,41	34	0,04	-0,96	0,43
16	15/06/2017	14:34:45	28,42	34,15	0,04	-0,96	0,43
17	15/06/2017	14:35:45	28,39	34,43	0,04	-0,96	0,43
18	15/06/2017	14:36:45	28,38	34,79	0,04	-0,96	0,43
19	15/06/2017	14:37:45	28,4	34,61	0,04	-0,96	0,43
20	15/06/2017	14:38:45	28,31	34,79	0,04	-0,96	0,44
21	15/06/2017	14:39:45	28,34	34,82	0,04	-0,96	0,43
22	15/06/2017	14:40:45	28,34	35,2	0,04	-0,96	0,43
23	15/06/2017	14:41:45	28,44	34,76	0,04	-0,96	0,43
24	15/06/2017	14:42:45	28,44	34,7	0,04	-0,96	0,43
25	15/06/2017	14:43:45	28,47	34,73	0,04	-0,96	0,43
26	15/06/2017	14:44:45	28,5	35,07	0,04	-0,96	0,43
27	15/06/2017	14:45:45	28,57	34,87	0,04	-0,96	0,43
28	15/06/2017	14:46:45	28,59	34,46	0,04	-0,96	0,43
29	15/06/2017	14:47:45	28,49	34,58	0,04	-0,96	0,43
30	15/06/2017	14:48:45	28,53	34,48	0,04	-0,95	0,43
31	15/06/2017	14:49:45	28,53	34,18	0,04	-0,95	0,43
32	15/06/2017	14:50:45	28,45	34,51	0,04	-0,96	0,43
33	15/06/2017	14:51:45	28,54	34,31	0,04	-0,96	0,43
34	15/06/2017	14:52:45	28,56	34,37	0,04	-0,96	0,44
35	15/06/2017	14:53:45	28,59	37,85	0,04	-0,95	0,45

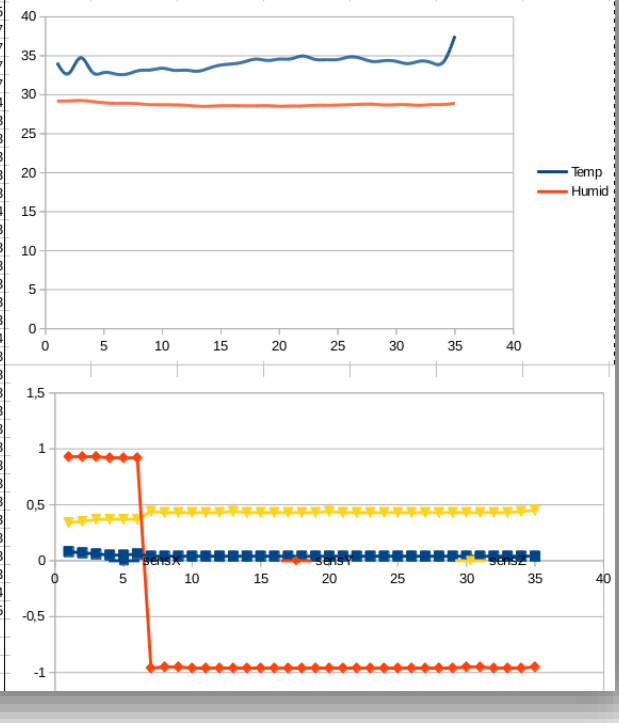


***Image 10** : Interface de récupération et traitement des données*

2. Tests situation réelle

La carte a ensuite été testée en situation réelle dans du béton. La boîte de protection étanche du boîtier d'ayant pas encore été créée, une protection en polystyrène a été réalisée puis entourée de béton avec l'aide de M.Parent dans le hall du Génie Civile.

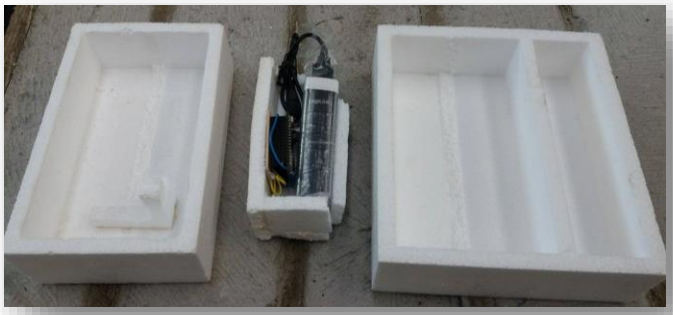


***Image 11** : Moules et boitier de protection en polystyrène*

***Image 12** : Coulage du béton dans les moules*

Le test en situation réelle a cependant échoué. En effet, après coulage du béton, l'émetteur a transmis une température allant dans les centaines de degrés et une très forte humidité. Une fois le béton séché, nous avons constaté que la carte de mesure s'était oxydée et avait grillée. Malgré la protection en polystyrène qui était censée protéger le circuit, nous pensons que la forte humidité lors du coulage combinée aux vibrations permettant d'homogénéiser le béton a permis à une partie de celui-ci trop liquide de s'engouffrer dans les fines cavités de la protection. Ceci a entrainé une réaction avec les circuits en cuivre ce qui a fait griller le circuit.

L'incident vient donc de la boite de protection pas assez hermétique lors du coulage. Le problème est alors de trouver une solution pour réaliser une boite suffisamment protectrice pour éviter un contact du béton, tout en permettant à l'humidité de passer pour être mesurée.

***Image 13** : Résultat du test en situation réelle (carte de mesure oxydée)*

Conclusion et perspectives

Ce projet a donc permis de réaliser le prototype d'un système électronique qui pourrait permettre de contrôler le vieillissement du béton. Ce système autonome en énergie serait capable de fournir des informations importantes sur les différentes étapes de maturation du béton permettant, sur du moyen et long terme, de déceler d'éventuels problèmes de coulage du béton et de pouvoir y remédier à temps. Cela conduirait ainsi, dans les secteurs du Génie Civil, à l'amélioration de la qualité ainsi que de la sécurité des ouvrages en béton.

A l'heure actuelle, les tests réalisés et présentés dans ce rapport ont permis de mesurer des valeurs de température, humidité et position, de les transmettre par radio à une interface de récupération de données et de les exploiter dans un fichier de type tableur. Cependant ces tests ont été réalisés hors béton, et ceux en situation réelle n'ont pas donné de résultats. De nouveaux tests doivent donc être réalisés et le système requiert plusieurs améliorations.

Dans un premier temps, afin de protéger le circuit mais d'éviter un coulage du béton en plusieurs parties comme c'est le cas dans ce projet, un **boitier** ayant des propriétés quasi-hermétiques tout en n'empêchant pas les mesures de température et d'humidité devra être mis au point. C'est notamment le problème qui est survenu dans nos tests en situation réelle.

Dans un second temps, au niveau de la **taille** qui devrait poser problème dans la solidité du béton sur un ouvrage plus grand. Il serait donc nécessaire de trouver des moyens de diminution de la taille, notamment au niveau du Feather M0 et du système de batterie très imposants. Une solution de carte faite entièrement à la main possédant un microcontrôleur et une communication Radio pourrait remplacer le Feather côté émetteur. Concernant l'**alimentation**, un autre système pourrait être mis en place, par exemple avec des piles boutons, malgré leur faible longévité, ou une alimentation par des ondes radios, comme l'utilisent les standards **ZigBee**, mais qui peut être insuffisante pour alimenter les capteurs.

D'un point de vue plus global, des tests à **plusieurs modules** émetteurs devront être réalisés de façon à optimiser le contrôle de vieillissement du béton. En effet, dans le cadre de ce projet les caractéristiques physiques ne devait être mesurées que dans une petite parcelle bétonnée. A l'échelle d'un mur ou même d'un bâtiment, des mesures à plusieurs endroits seraient sûrement nécessaires pour être convenablement exploitées. Une synchronisation de la réception des données devra donc être mise en place afin d'éviter une perte d'informations, ainsi qu'un contrôle renforcé pour éviter d'éventuelles interférences provenant d'appareils utilisant la même fréquence radio.