Projet de Fin d'Etudes

Réseau de capteurs pour application Smartphone

Elève : Léo MAZIER

Spécialité : IMA

Année 2016/2017

Table des matières

Contexte du projet

De nos jours les téléphones intelligents (ou SmartPhone) sont devenus bien plus que de simples téléphones. Leur utilité est désormais devenue multiple, si bien que les utilisateurs passent désormais plus de temps sur des applications tierces que sur les fonctions principales d'un téléphone.

C'est pourquoi le domaine des applications mobiles est en plein essor récemment et son poids économique devient de plus en plus important. Android et iOS qui s'en partage une très grande majorité des parts de marché sont constamment à la recherche d'innovations.

La pluralité des applications disponibles et le fait qu'une majorité de la population possède désormais un smartphone ou une tablette font que ces derniers sont devenus les outils privilégiés par les utilisateurs dans la plupart des actions du quotidien. Le smartphone est donc devenu le centre névralgique de notre quotidien que ce soit au niveau particulier qu'au niveau professionnel.

De plus les problèmes environnementaux actuels poussent de plus en plus les populations à s'intéresser à leur cadre de vie et des risques environnementaux qui les entourent. Ces risques peuvent être à grande échelle (Tremblement de terre, réchauffement climatiques, risques nucléaires, …) mais sont difficilement vérifiable pour la majorité de la population. Cependant certains paramètres affectant notre vie personnelle et quotidienne peuvent quant à eux être surveillés et leur étude et de plus en plus prisée par une majorité de personne qui veulent savoir s'ils évoluent dans un cadre environnementale sain pour eux et leur entourage.

Cahier des charges

L'application doit permettre à l'utilisateur de visualiser les données de son environnement de façon claire et précise. Il faut que cette application soit intuitive et puisse permettre la lecture de plusieurs données en simultanée.

Pour ce faire l'application se doit de communiquer avec une carte qui servira de réseau afin de gérer différents capteurs environnementaux. Ses capteurs seront alimentés et devront être reliés à une seule et même carte. La communication entre la partie hardware (carte + capteurs) et la partie software (application Android) se fera par liaison série à l'aide d'un câble USB.

La partie Hardware doit être alimentée par le Smartphone afin de faciliter la mobilité du système. De plus sa taille doit rester relativement compact afin d'en faciliter le déplacement.

L'application devra être capable de lire et de traiter les données fournies par les capteurs en temps réel.

Analyse du sujet

Le principe de ce sujet est donc de créer un réseau de capteurs permettant une analyse des données environnementales de l'utilisateur et qui pourront être visibles grâce à une application mobile sur SmartPhone ou sur tablette.
Dans un premier temps il est important de savoir quels types de données nous cherchons à analyser avec nos capteurs tout en restant dans un domaine du personnelle car cette analyse s'intéresse à un public large dont le but est d'analyser rapidement les conditions environnementales autour de soi à l'aide d'un SmartPhone.
Je me suis d'abord penché sur un capteur d'ondes électromagnétiques car c'est un thème qui prend de l'ampleur depuis quelques temps avec l'omniprésence actuelle des réseaux Wifi, des communications Bluetooth aussi bien en terme de diffusion (antennes, box ethernet, …) qu'en terme de réception (TV, téléphones, ordinateurs portables, tablettes, …). En effet désormais l'impact de toutes ses nouvelles technologies est étudié afin de savoir s'il y a un risque ou non sur la santé. Cependant ce thème fut rapidement écarté car il paraît contradictoire de relever des données sur l'impact des réseaux sans fils à l'aide d'un SmartPhone qui est désormais au centre de ce système.
Je me suis alors tourné vers des capteurs permettant la mesure de la qualité de l'air qui est également un sujet récurrent de nos jours à cause des vagues de pollutions.
Puis j'ai décidé de rajouter des analyses complémentaires concernant la température, l'humidité, la pression et au final je me suis retrouvé avec un ensemble assez imposant. Le problème engendré par ces ajouts et que le léger et compact système du début s'est retrouvé bien plus imposant et donc bien moins utile dans le cadre d'une analyse extérieure et brève à l'aide d'un SmartPhone. Deux choix s'offraient alors à moi, soit diminuer le nombre de capteurs soit d'en trouver une utilisation un peu moins mobile sans pour autant le transformer en centrale d'analyse fixe.
J'ai donc décidé de partir sur un ensemble de capteur destiné à une utilisation dans un contexte plus familial et résidentiel. En effet de nos jours on parle beaucoup des mauvaises conditions en extérieur (pollution, …) mais assez peu des conditions au sein même d'une habitation et pourtant ce thème est tout aussi important, de plus se dire que l'air est moins pollué chez soi que dans la rue sur laquelle donne notre fenêtre est saugrenu.
La suite du projet se fera donc en suivant ce principe. Bien que la partie électronique s'en retrouve plus imposante et donc moins utile en déplacement, l'ensemble peut très bien être utilisé dans un contexte un peu moins mobile où le réseau de capteur peut être utilisé afin d'effectuer des analyses rapides des données.

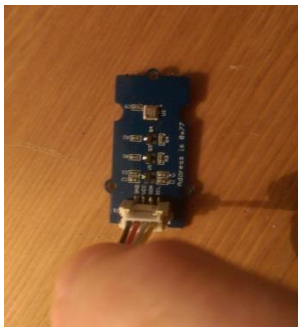
Détails du projet

Ce projet se décompose donc en deux parties bien distinctes. Une concernant le réseau de capteur à proprement parler et qui regroupe le fait de brancher, relier et permettre la communication des différents capteurs.
La deuxième partie quant à elle concerne l'application mobile qui permettra de visualiser les données relevées par les capteurs sur un SmartPhone ou une tablette mobile.

Partie Hardware

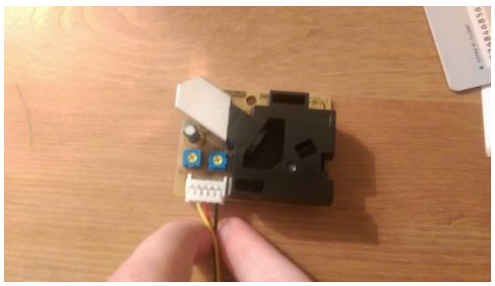
Cette partie repose sur l'utilisation d'une carte Arduino UNO afin de récolter les données de nos capteurs.
Les capteurs utilisés sont des capteurs de la marque Seeed Studio et sont aux nombre de 5 :

_ 1 baromètre permettant le relevé de la pression et de l'altitude :

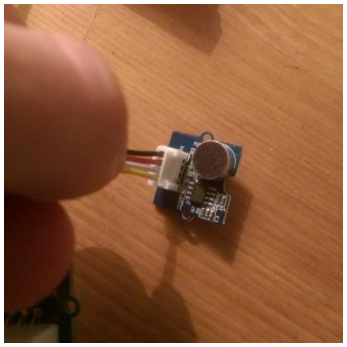


_1 capteur de température et d'humidité :

_1 capteur de qualité de l'air :

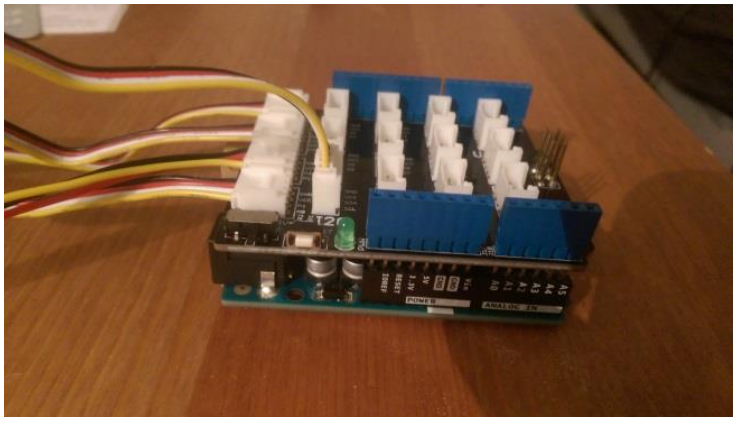


_1 capteur de son qui permet de relevé la puissance du son ambiant :



_1 capteur de rayon UV

Tous ses capteurs sont reliés à un shield venant s'ajouter à la carte Arduino UNO afin de faciliter la connexion entre ces divers éléments. Ce chield se compose de nombreuses entrées sur lesquelles les fiches des capteurs viennent se brancher et sont directement reliées à la carte Arduino UNO à l'aide des connecteurs PIN.



L'ensemble carte plus capteurs sera alimenté à l'aide de la connexion USB avec le SmartPhone qui délivrera une tension de 5V ce qui est nécessaire à son alimentation. Pour ce faire il faudra que la SmartPhone se configure en hôte USB afin que ce soit lui qui délivre l'énergie par le biais de son port USB, contrairement à l'utilisation habituelle où il reçoit l'énergie par se port.

Afin de recevoir les données des capteurs la première étape consiste à définir les différent PIN sur lesquels ces capteurs sont reliés de la sorte :

#define DHTPIN A0
#define DUSTPIN A1
#define SOUNDPIN A2
#define UVPIN A3
…

Par la suite l'ensemble des données relevées par les capteurs sont lues et sont normées en fonction des datasheet associées.

```
{
    sensorValue=analogRead(UVPIN);
    summ=sensorValue+summ;
```

```
pressure = myBarometer.bmp085GetPressure(myBarometer.bmp085ReadUP());//Get the pressure
altitude = myBarometer.calcAltitude(pressure); //Uncompensated caculation - in Meters
atm = pressure / 101325;
```

Ici le relevé de la pression atmosphérique est divisé par la pression atmosphérique normale, soit 1,01325 bar.

Afin de pouvoir connecter la partie Arduino et la partie Android il nous faut utiliser un câble OTG qui permet la jonction entre une prise USB fiche A (« classique ») et une prise USB micro B (SmartPhone). Bien évidemment un câble USB fiche B est nécessaire pour se connecter sur la carte Arduino UNO.

Câble OTG

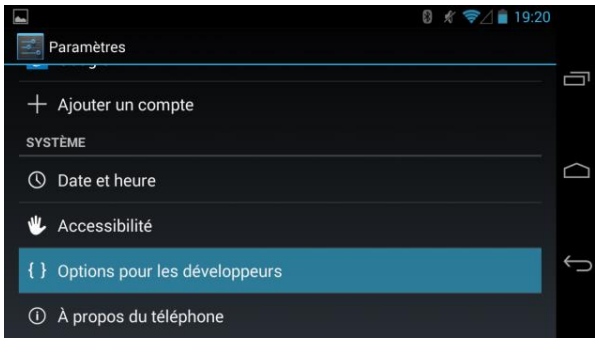
Câble de communication Arduino

Partie Software

Préparation

La partie logicielle sera exclusivement centrée sur L'OS Android car il s'agit du seul système d'exploitation que j'ai à disposition afin de développer l'application mobile.

Pour tester l'application mobile nous allons utiliser un SmartPhone (HTC One M8) directement. Pour cela il est nécessaire de passer le téléphone en « mode développeur » afin de pouvoir développer et déboguer toute sorte d'applications en cours de développement et dont nous sommes à l'origine. Lorsque le « mode développeur » est activé il faut désormais activer le débogage USB.

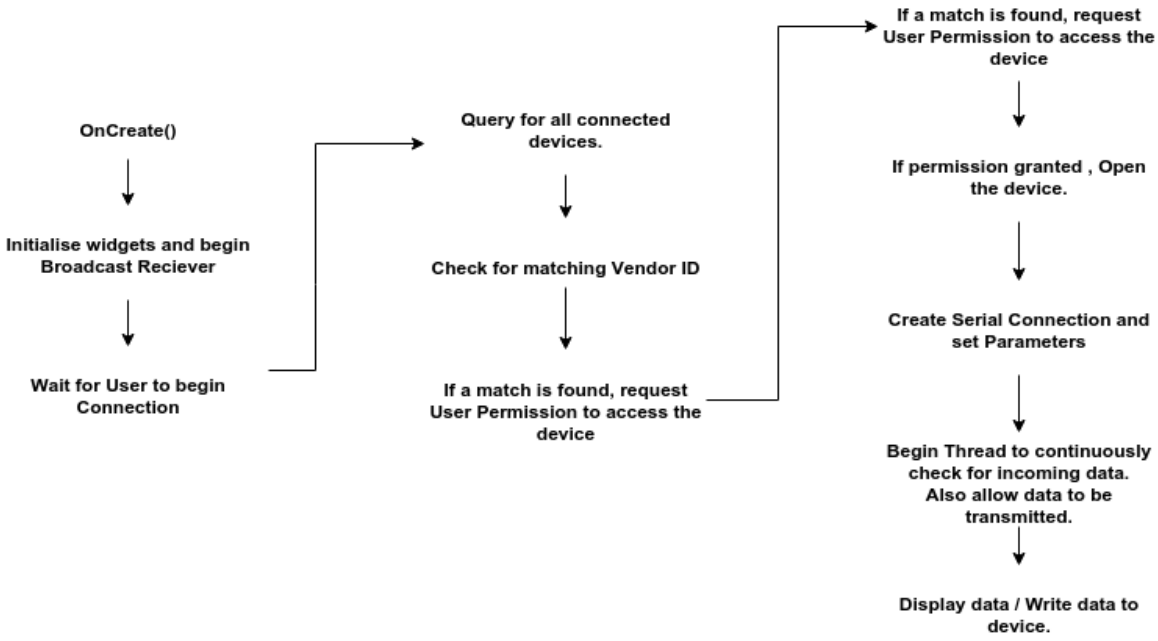


Le développement de l'application mobile sera établi à l'aide du logiciel Android Studio qui est l'un des plus utilisé de par sa facilité de prise en main, son efficacité et sa compatibilité avec n'importe quel système d'exploitation (Linux, Windows, Mac).

La communication USB

Afin de pouvoir communiquer avec la carte Arduino UNO qui est reliée aux capteurs, j'ai décidé d'utiliser une librairie externe « USBSerialLibrary » qui est disponible en open source sur GitHub et qui est régulièrement mise à jour afin d'être compatible avec les appareils les plus récents.

L'architecture concernant la communication entre la carte Arduino UNO et l'application mobile est la suivante :



Dans un premier temps il est très important d'initialiser les différentes classes qui nous seront utiles pour la communication à l'aide du port USB :

```
UsbManager usbManager;
```

Cette classe permet de gérer l'ensemble des données liées au port USB de notre SmartPhone.

```
UsbDevice device;
```

Cette classe nous permet quant à elle de gérer l'état de l'appareil connecté, ici nous n'en avons qu'un seul : la carte Arduino UNO.

```
UsbSerialDevice serialPort;
```

Cette classe permettra la lecture des données sur notre port USB.

```
UsbDeviceConnection connection;
```

Cette classe va nous permettre d'établir une connexion entre la carte Arduino UNO et notre application mobile.

Afin d'éviter de surcharger l'application de données il sera demandé à l'utilisateur de démarrer la connexion à l'aide d'un bouton « Start ». Au clique sur ce bouton, l'application va alors déterminer si des appareils sont connecter ou non au port USB du SmartPhone. Ceci est possible grâce à la ligne de code suivante :

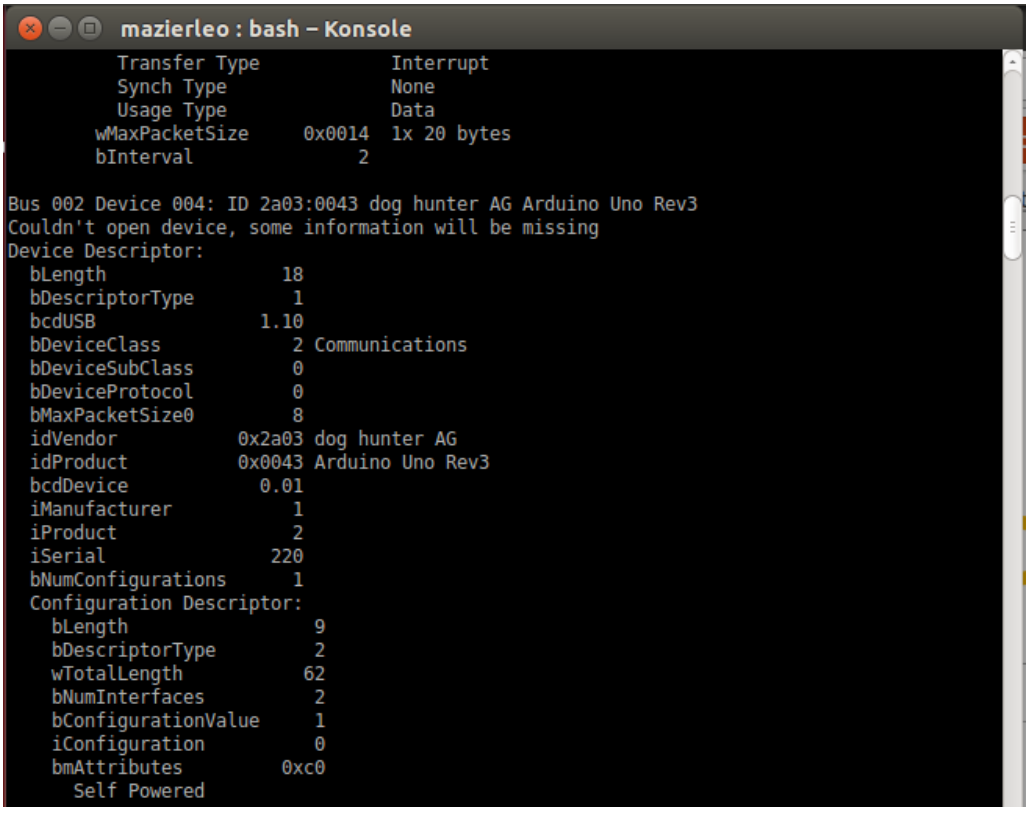
```
public void onClickStart(View view) {
            HashMap usbDevices = usbManager.getDeviceList();
            …
            }
```

La classe « usbManager » nous permet d'avoir accès aux informations concernant le port USB de notre SmartPhone et on peut constater ci-dessus que l'ensemble des appareils connectés à notre SmartPhone seront stockés dans « usbDevices ».

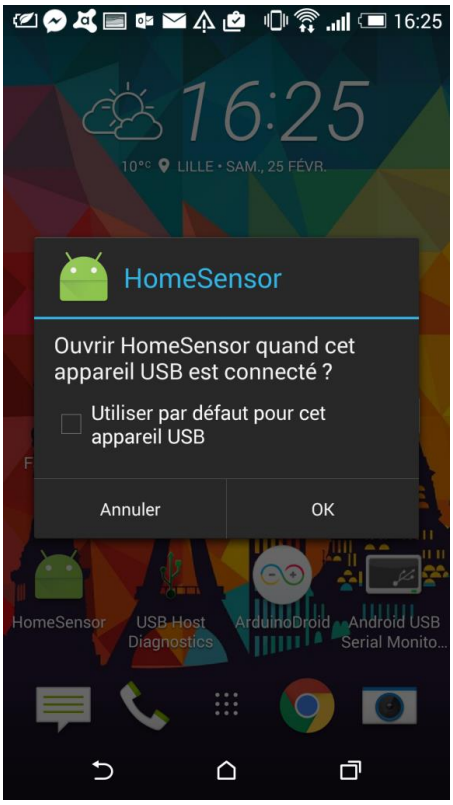
On vérifie alors qu'il y a bien au moins 1 appareil connecté au port USB, dans notre cas il y en a exactement 1, il s'agit de l'Arduino UNO. Cependant afin d'établir la connexion avec cette dernière il faut vérifier que l'objet connecté est bien celui attendu. Pour cela on vérifie le Vendor ID (VID) de notre appareil connecté.

```
int deviceVID = device.getVendorId();
if (deviceVID == 0x2A03)
```

Le VID peut être obtenu en effectuant la ligne de commande : « lsusb –v » dans un terminal Linux.



Afin de laisser une part d'autorité à l'utilisateur on demande alors à ce dernier s'il souhaite confirmer la connexion avec l'appareil connecter à l'aide d'une fenêtre apparaissant sur son SmartPhone.



Si la demande de connexion est validée, cette dernière est alors créée et configurée. La communication peut alors débutée.

La configuration de la connexion est la suivante :

```
if (serialPort.open()) { //Set Serial Connection Parameters.
    setUiEnabled(true);
    serialPort.setBaudRate(9600);
    serialPort.setDataBits(UsbSerialInterface.DATA_BITS_8);
    serialPort.setStopBits(UsbSerialInterface.STOP_BITS_1);
    serialPort.setParity(UsbSerialInterface.PARITY_NONE);
    serialPort.setFlowControl(UsbSerialInterface.FLOW_CONTROL_OFF);
    serialPort.read(mCallback);
    tvAppend(textView,"Serial Connection Opened!\n");
```

On vérifie alors que la connexion est bien ouverte et on configure cette connexion à l'aide de la classe « serialPort ». On procède ici à une configuration classique car notre système n'a pas de demande particulière. La première ligne nous permet activer l'interface Homme-Machine sur notre application mobile et la dernière permet de faire apparaître la phrase indiquée afin que l'utilisateur vérifie que la communication à l'aide du port USB est prête.

De plus on procède un appel à mCallback afin d'afficher les données retournées par notre Arduino comme nous allons le voir plus bas.

Il est également nécessaire de configurer la communication par USB dans le Manifest de l'application.

…

```
<uses-feature android:name="android.hardware.usb.host"/>
```

…

Cette ligne nous permet d'autoriser la connexion en tant que hôte USB de notre application et doit être placé dans l'entête du manifest afin qu'elle soit disponible pour l'ensemble de l'application.

De plus il faut également préciser dans l'activité concernée la présence d'un appareil connecté afin que la connexion puisse avoir lieu. Sans ça la connexion entre le port USB et l'application mobile ne sera pas créée.

```
<activity android:name=".UsbActivity">
<intent-filter>
    <action android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
/>
</intent-filter>

<meta-data
    android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
    android:resource="@xml/device_filter" />
</activity>
```

Affichage des données

Une fois que notre communication via le port USB est correctement configurée et fonctionnelle il nous faut alors afficher les données de nos capteurs.

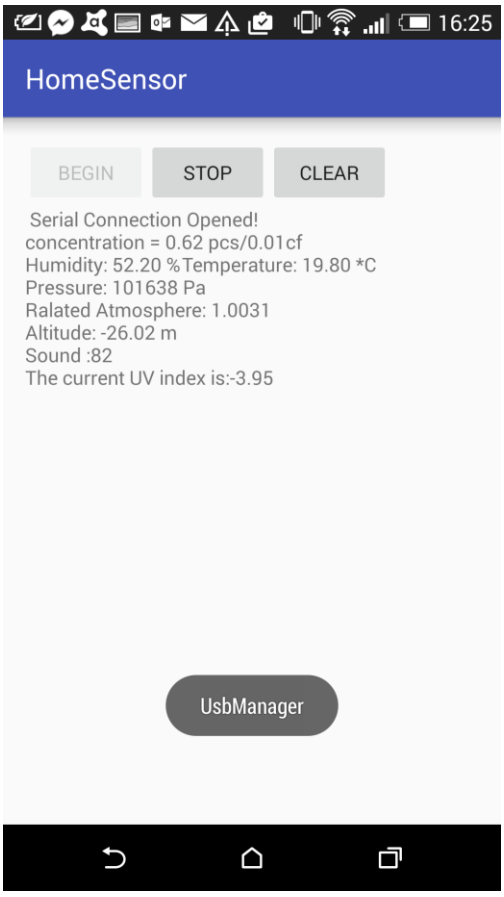
Pour cela nous utilisons un « CallBack » qui va afficher les données à chaque fois qu'il les reçoit :

```
UsbSerialInterface.UsbReadCallback mCallback = new
UsbSerialInterface.UsbReadCallback() {
    @Override
    public void onReceivedData(byte[] arg0) {
    …
    tvAppend(textView, data);
    …
}
```

L'affichage se fait alors à la façon d'un terminal et de manière continue bien qu'une légère temporisation fut mise en place dans l'envoi de données sur la partie Arduino.

```
        lowpulseoccupancy = 0;
        starttime = millis();
      }
    }
```

Ainsi l'interface ne se retrouve pas surchargée et l'utilisateur garde une lecture aisée des données relevées.



Conclusion

Au cours de ce projet je fus amené à gérer et configurer une communication USB pour une application mobile ce qui fut très intéressant et qui souleva de nombreuses réflexion afin d'aboutir à un résultat convenable.

Ce projet fut essentiellement orienté vers le système d'exploitation Android uniquement ce qui peut être considéré comme un manque d'ouverture envers les différentes utilisations possibles mais qui me permet de travailler d'avantage sur cet OS afin de faciliter mon stage de fin d'étude qui portera justement sur l'établissement d'une application sous Android.

Le sujet soulevé par ce projet est très intéressant car de nos jours la question du bien-être et de la santé est un thème régulièrement apporté notamment quand celui-ci porte sur nos conditions de vie liées à notre environnement. En effet nous faisons désormais de plus en plus attention aux paramètres environnementaux nous entourant afin de savoir si nous évoluons dans des conditions adéquates à notre bien-être.

Concernant le projet en lui-même, bien que l'affichage reste relativement basique, ce système permet donc de relever les données en temps réel des différents capteurs que nous avons intégrés à notre carte Arduino UNO. Il serait intéressant d'y ajouter un design plus agréable pour l'utilisateur en terme d'application mobile ainsi qu'en terme matériel. En effet un travail plus approfondi en terme d'esthétique concernant l'application Android rendrait cette dernière plus aboutie et permettrai une utilisation pour le grand public. De plus il semble nécessaire de créer un boitier afin de contenir l'ensemble carte Arduino + capteurs afin d'obtenir un objet réellement compact et uni ce qui ferait avancé le projet à un stade plus avancé de développement, ou tout du moins plus proche d'une éventuelle ouverture au public.

Au cours des dernières semaines je me suis également penché sur le fait de pouvoir faire communiquer ce système à l'aide des technologies Wifi ou Bluetooth afin de passer outre le câble de liaison USB. Paradoxalement cela engendrai une régression de la mobilité du réseau de capteurs car la partie hardware nécessiterai une alimentation dont la source serait autre que la liaison USB actuelle. Il pourrait alors être intéressant de mettre en place un système de batterie afin de répondre à ce besoin tout en créant une fonction permettant de visualiser le niveau actuel de batterie ou tout du moins notifiant un niveau faible afin d'éviter d'en tomber à cours.

Dans le cadre où cette application fut créée dans un contexte résidentielle, l'aspect ethernet fut étudié car il pourrait être intéressant d'avoir un accès aux donnés sur l'ensemble d'un réseau local. Notamment le site web « Plotly » est une première approche de ce qui pourrait être faisable. En effet ce site web permet une visualisation directe et gratuite de données. Il faudrait alors configurées nos données afin qu'elles soient envoyées par Wifi ou câble ethernet à la base de données de plotly liée au compte utilisateur.