GAYOUT RICHARD
PHRASAVATH ANTHONY

L3 EEA REL

RAPPORT DE PROJET

Centrale météo Autonome

Encadré par : Thierry Périsse

Sommaire

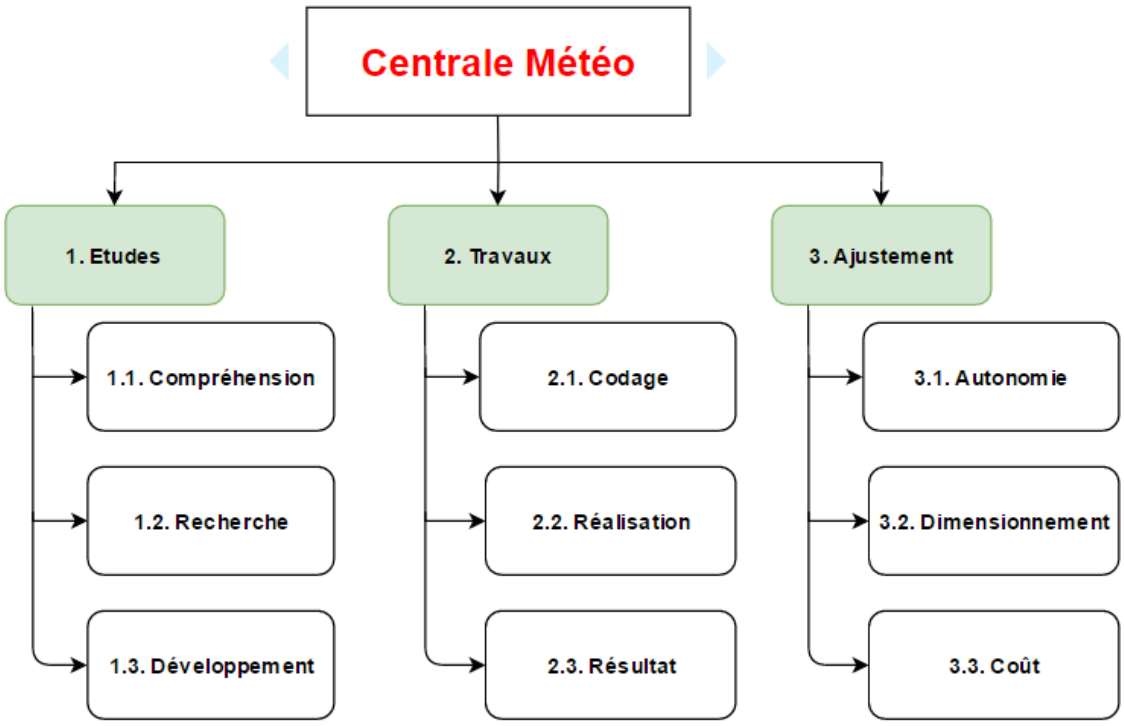
1. Présentation Générale du Projet

1.1) Énonce générale du projet

Pour ce projet en L3 EEA nous avons décidé de travaillé sur la station météo autonome commandé par une carte électronique programmable Arduino. C'est un projet qui nous permet d'améliorer notre travaille en équipe, d'approfondir nos capacité a gérer un projet et qui nous permet d'étendre nos connaissances en électronique et en informatique industriel. Dans ce projet nous avons mis en œuvre une carte Arduino qui communique avec une multitudes de capteurs (lumière, pression, humidité...), nous avons aussi utilisé une antenne longue portée afin de recevoir sur ordinateur les données transmises par la station météo, les données sont alors perçues par le logiciel LABVIEW. Nous avons aussi mis en place une communication par bluetooth, les données reçues sont alors récupérées sur un smartphone. Enfin on a étudié l'autonomie de la station météo, pour cela on a dimensionnée une batterie selon la consommation total du projet, et pour la rendre autonome à l'extérieur la batterie est rechargé par un panneau photovoltaïque.

For this project in L3 EEA we decided to chose the Autonomous Weather Station Project controlled by an Arduino programmable electronic card. It's a project that allows us to improve our teamwork, to improve our ability to manage a project and to extend ou knowledge in electronics and industrial informatics. In this project we have used an Arduino board that communicates with a multitude of sensors (light, pressure, humidity...), also we used a long-range antenna to receive on computer the data transmitted by the weather station, the data are collected by the LABVIEW toftware. Also we have set up a communication by bluetooth, The data received is recovered on a smartphone. Finally, we studied the autonomy of the weather station, we sized a battery according to the total consumption of the project, and to make it autonomous outside the battery is recharged by a photovoltaic panel.

1.2) Organigramme du projet



Le projet ce décompose en 3 parties qui se décomposent eux aussi en 3 sous parties.

- 1ere Partie : Étude

➔ Compréhension : Au début du projet il nous fallait comprendre ce qui nous était demandé et de définir le cahier des charges à ce projet.

➔ Recherche : Pour rendre le projet réalisable il nous fallait rechercher les différents matériaux qui permet d'aboutir à ce que l'on souhaite réaliser.

➔ Développement : Une fois les matériaux identifiés, il nous fallait chercher comment les mettre en œuvre.

- 2eme Partie : Travaux

➔ Codage : Pour utiliser les différents composants on passe par la carte Arduino programmée à partir du logiciel Arduino dans un langage C/C++.

➔ Réalisation : On réalise le branchement des différents matériaux.

➔ Résultat : On test et on relève ensuite les résultats obtenus.

- 3eme Partie : Ajustement

➔ Autonomie : On essai de rendre le projet le moins consommatrice d'énergie possible.

➔ Dimensionnement : On dimensionne ensuite la partie autonomie en fonction de la consommation de la centrale météo.

➔ Coût : On recherche les possibles moyens de rendre le projet le moins coûteux possible.

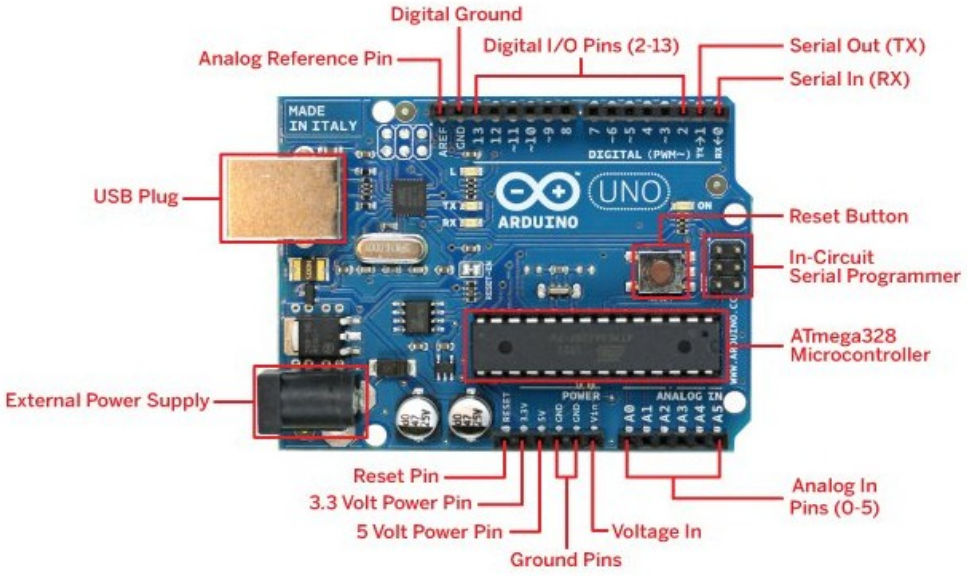
1.3) Matériaux utilisés

Ici la liste des différents matériaux que nous avons utilisé sur ce projet :

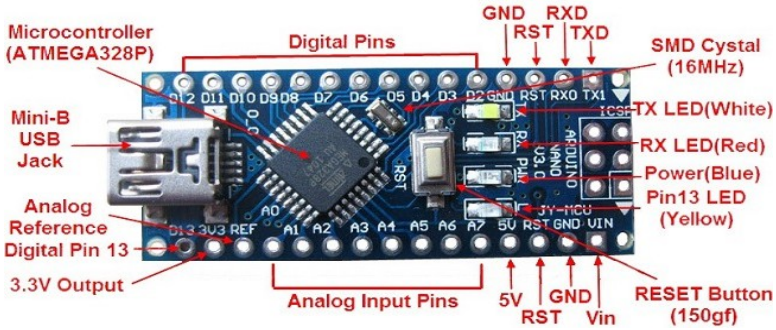
– Carte électronique programmable

L'Arduino est le cœur du système, c'est ce qui va nous permettre de transmettre nos instructions aux différents composants, elle se programme en langage C/C++.

Arduino Uno :



Arduino Nano :

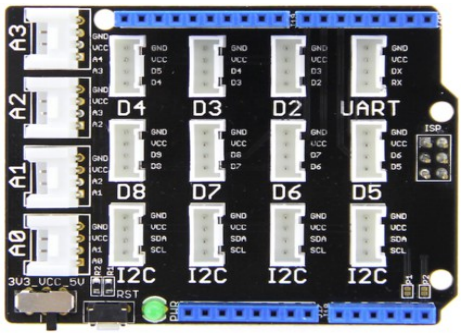


	Mémoire vive	Mémoire flash	Horloge	Digital pin	Analog pin	VCC	Vin
Arduino Uno	2k	32k	16Mhz	14	6	5V	7-12V
Arduino Nano	2k	32k	16Mhz	14	8	5V	7-12V

L'Arduino Uno et Nano ont plus ou moins les mêmes fonctionnalités.

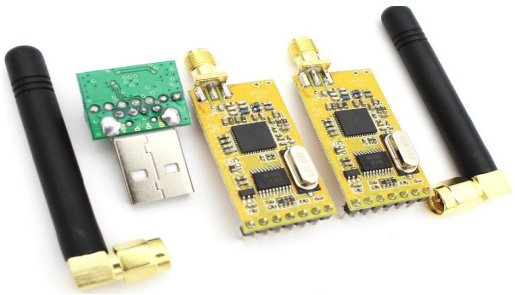
L'Arduino Nano est plus compacte et au lieu d'utiliser l'USB standard pour se connecter à l'ordinateur, il utilise le mini-usb. De plus il y a une diode de protection en série avec l'alimentation provenant de l'USB ce qui fait que le VCC vaut dans ce cas 4,4 ou 4,5V alors qu'alimenté par Vin (entre 8 et 12 V) le VCC fourni par le régulateur fait 5V.

- Module Base Shield



Le module base shield à pour but de connecter aisément d'autres modules à l'arduino. Il va nous permettre de communiquer via le bus I2C avec les différents capteurs et l'écran LCD.

- Antenne APC220

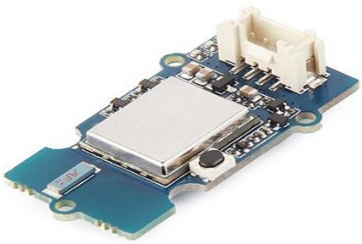


L'antenne APC220 est un moyen de recevoir les données de la station météo en longue portée. Le signal peut être transmise à plusieurs centaine de mètre même si entre le transmetteur et le receveur le signal rencontre des obstacles. Elle s'alimente entre 3,5V et 5,5V et elle a une broche d'activation EN, active lorsque la tension est supérieur à 1,6V ou à vide et en sommeil lorsque la tension est inférieur à 0,5V. L'antenne consomme environ 20mA.

– Module Bluetooth HC-06

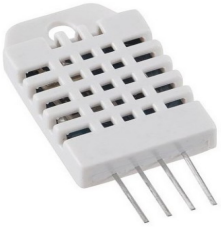
Le module Bluetooth nous permet de recevoir les données de la station météo en courte portée. Le signal transmit n'atteint qu'une dizaine de mètre mais elle nous permet de travailler sur un moyen de communication très utilisé. Ce module s'alimente entre 3,1V et 4,2V et consomme environ 30mA.

– Module WiFi



Le module WiFi est un récepteur émetteur il se raccorde sur un port série UART du module Base Shield. Il intègre le protocole TCP/IP, il permet à notre microcontrôleur d'interagir avec des réseaux WiFi avec seulement quelques lignes de code. Ce module est constituée de trois LED, une LED bleu contrôlé par l'utilisateur, une LED rouge indiquant une connexion actuelle WiFi et une LED verte indiquant l'alimentation du module. Il s'alimente entre 3V et 5V et consomme environ 1mA.

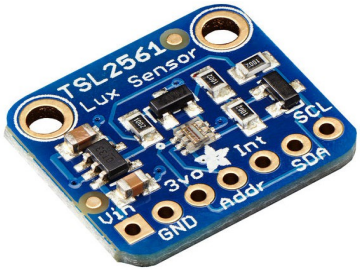
– Capteur de température & humidité DHT22 AM2302



Le capteur de température & humidité DHT22 nous permet de mesurer une température entre -40°C et 80°C et de mesurer le taux d'humidité de 0 à 100%.

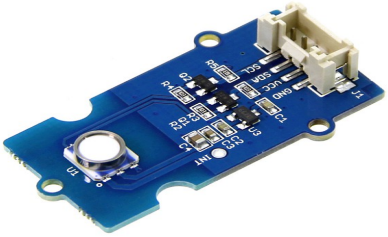
Le capteur ne consomme que quelque centaine de micro ampère et s'alimente entre 3,3V et 6V.

– Capteur d'irradiation TSL2561



Le capteur d'irradiation TSL2561 nous permet de mesurer l'intensité de la lumière entre 0,1 et 40 000 Lux. Le capteur ne consomme que quelque centaine de micro ampère et s'alimente en 3,3V.

– Capteur de pression atmosphérique et altitude BMP280



Le capteur de pression atmosphérique et altitude BMP280 nous permet de mesurer une pression entre 300 et 1100 hectoPascal. Le capteur ne consomme que quelque centaine de micro ampère et s'alimente entre 3,3V et 5V.

– Capteur de pluie LMK-47

Le capteur de pluie LMK-47 protégé contre les oxydations nous permet de détecter la présence de pluie. Le capteur ne consomme que quelque centaine de micro ampère et s'alimente entre 3,3V et 5V.

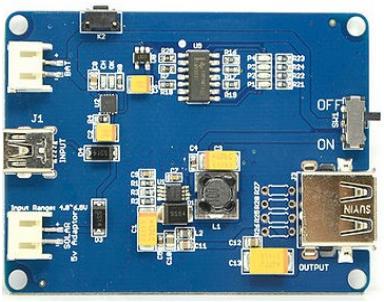
– Ecran LCD

L'écran LCD nous permet d'afficher directement les données reçues par la station météo en local. L'écran s'alimente en +5V et consomme environ 40mA.

– Batterie Lipo Conrad 3,7V 2100mAh

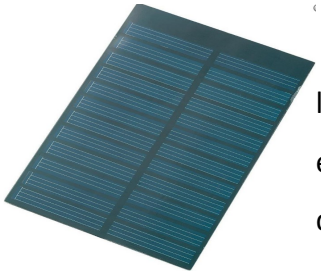
La batterie Lipo Conrad nous permet de rendre autonome la station météo en extérieur sans passer par une alimentation intérieur. La batterie délivre une tension de 3,7V, une capacité de 2100mAh, un courant de décharge max de 42A et un courant de charge max de 4,2A.

– Système de recharge Lipo Rider Pro



La Lipo Rider Pro permet de charger une batterie Lipo 3,7V via un panneau solaire ou une source USB et d'alimenter notre projet de façon autonome avec une tension stabilisée 5V. Ce système est déjà intégralement programmé pour alimenter l'arduino et charger la batterie.

– Panneau solaire polycristallin Conrad 6V 150mA



Le panneau solaire petite mais robuste nous permet de capter l'intensité de la lumière est de la convertir en énergie électrique, cette énergie va alors charger la batterie. Le panneau a une tension nominale de 6V, une puissance de 0,9W et un courant nominal de 150mA.

1.4) Choix technologiques

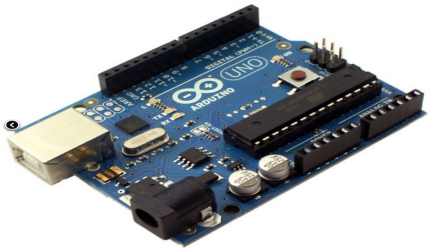
Le choix technologique portant sur une carte en accès libre avec une multitude de composant a été une partie décisif sur la réalisation de notre projet.

- **L'Arduino**

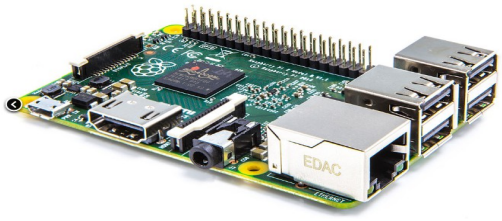
La carte électronique programmable Arduino nous permet d'exécuter les différentes actions qui constitue la station météo, il existe aussi une autre carte pouvant réaliser cette tâche qui était à notre disposition, le Raspberry Pi.

Mais quelle différence y a t'il entre ces deux cartes ?

Arduino



Raspberry Pi



L'Arduino est un mini automate fait pour exécuter du mono-tache. Une fois notre programme chargé, il devient autonome et l'exécute en boucle. Elle est peu coûteuse, très polyvalente, facile à apprendre et conçue pour l'expérimentation électronique et robotique donc idéal pour tout ce qui est pilotage d'appareil électronique.

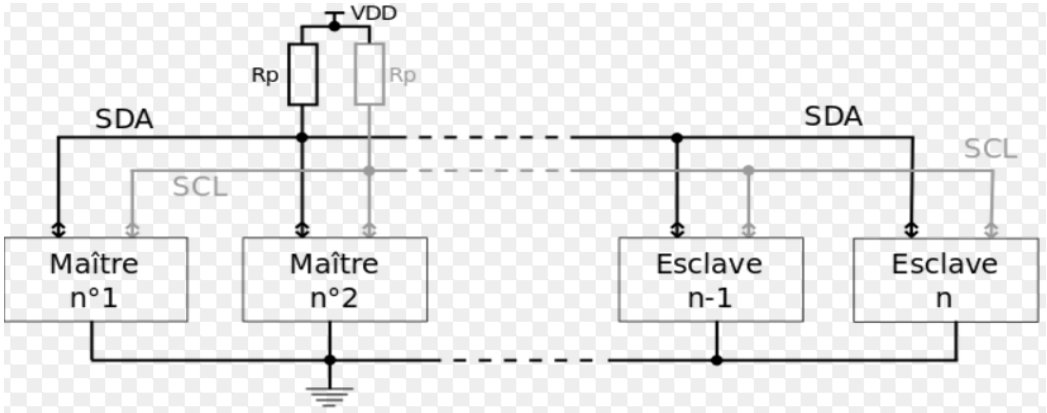
Le Raspberry Pi est un mini PC capable d'exécuter un système d'exploitation, un pilote vidéo, un système audio (entrée et sortie), des ports USB, un système de stockage, donc idéal pour utiliser comme serveur, surveillance vidéo (domotique), passerelle.

Au final dans le cadre de notre projet nous avons choisit l'Arduino car elle convient parfaitement à ce que l'on souhaite réaliser et qu'elle est facile à prendre en main.

- **Protocole I2C**

L'Arduino offre d'origine deux bus de communication I2C et SPI.

Nous utilisons ici le bus I2C.



Ce dernier permet d'établir des connexions synchrones entre plusieurs composants intelligent pour partager des informations via un "bus commun".

C’est un bus deux fils (+ alimentation et masse) :

SDA = Signal de données (datas)

SCL = Signal d'horloge (clock)

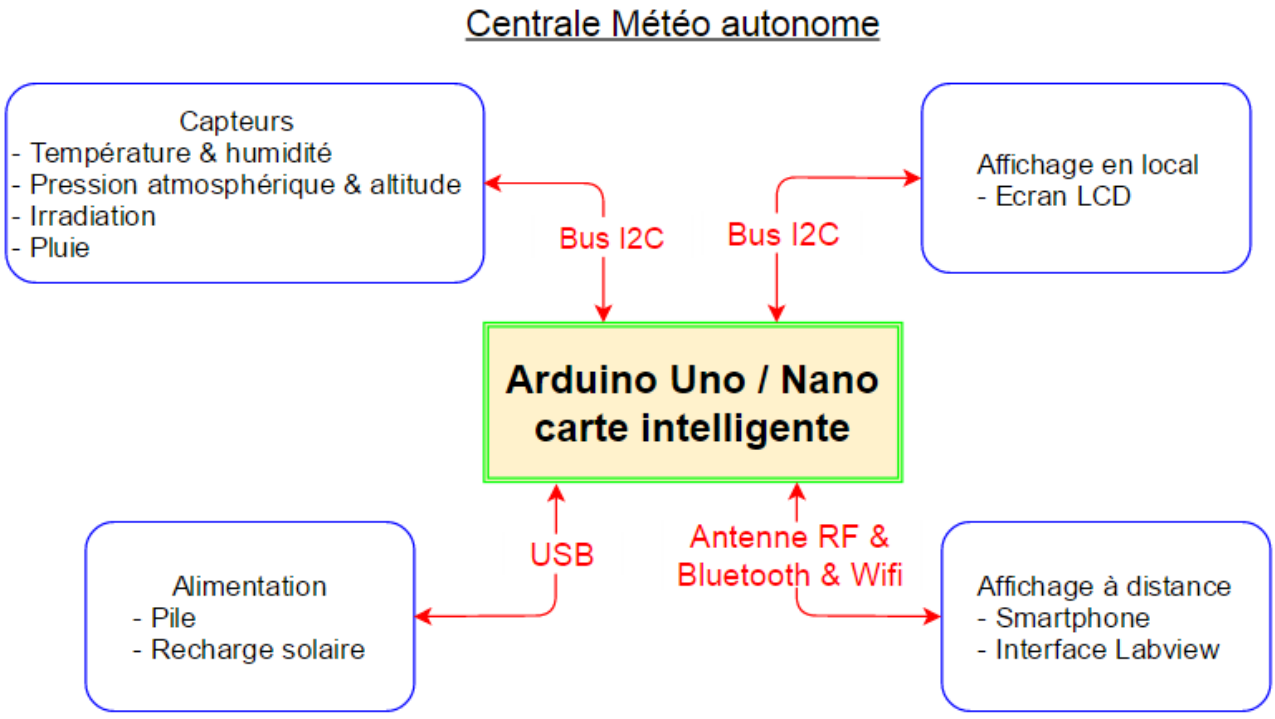
Les données sont transmises en série à une vitesse de 100Kbits/s.

La distance maximale maître-esclave pour un bus I2C est d'environ un mètre et dépend de plusieurs facteurs comme la capacité électrique du bus ou le débit de transmission.

Il y a des milliers de composants qui utilisent une interface I2C, et les cartes de la famille Arduino peuvent toutes les contrôler.

1.5) Schéma fonctionnel du projet

Pour une meilleur lisibilité et une meilleur compréhension du projet et de son architecture, voici un organigramme schématique de chaque sous-ensemble du projet :



Nous avons donc découpé notre projet en 4 grand axes distincts :

- L'acquisition de multiples données provenant de différents capteurs qui communique en bus I2C avec la carte électronique intelligente Arduino.

- L'affichage des informations de la station en local grâce à un écran LCD qui communique en bus I2C avec la carte électronique intelligente Arduino.
- Les différents moyens de communication sans fil, antenne RF, Bluetooth et WiFi et affichage des données sur smartphone ou sur ordinateur via une interface LABVIEW.
- L'autonomie par piles et par rechargement solaire.

2. Présentation par étape du partie du projet

2.1) Mise en œuvre de capteur numériques pour la station

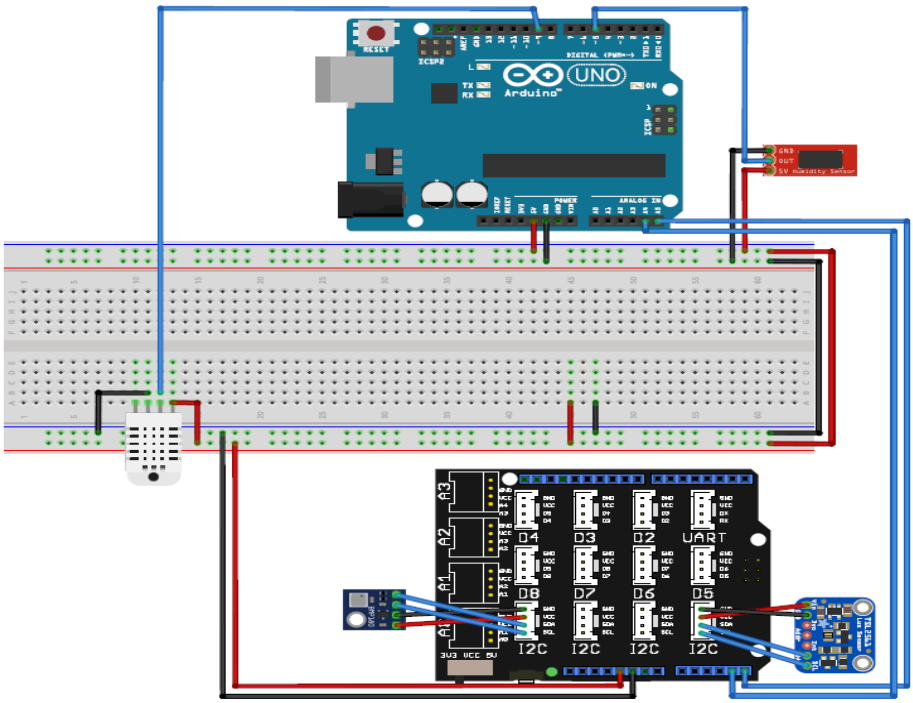
La première étapes de ce projet a était de nous familiariser avec l'utilisation relativement simple de l'arduino. Pour cela nous avons fait des montages élémentaires, basique mais essentiel pour avoir de bonne bases et pouvoir avancer sereinement dans le projet.

Après avoir choisi les principaux composants composants notre station météo vu dans le chapitre un. Nous allons donc implémenter des capteurs qui seront connecté via la liaison I2C a travers le Base Shield et nous visualiserons les informations directement sur le logiciel de programmation de l'Arduino.

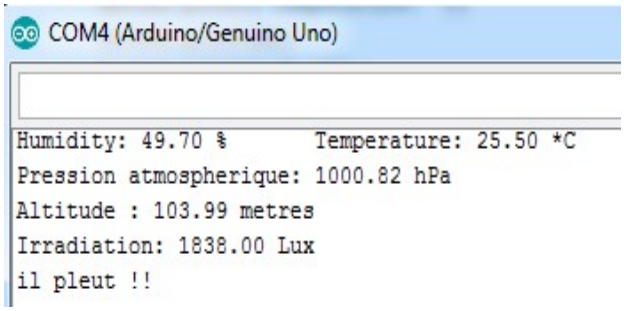
Matériel : Arduino Uno, Base Shield, capteur DHT22 Am2302, capteur TSL2561, capteur BMP280, Capteur LMK-47.

La carte est alimenté directement en USB sur l'ordinateur, les modules sont alimentés en 5V, certain peuvent fonctionner en 3,3V mais pour une raison de performance et de stabilité nous avons préférer garder cette tension standard assurant de bonne performances dans l'ensemble et la différences de consommation est assez faible lors du fonctionnement .

Schéma de câblage :



Résultat obtenue à l'aide du Serial Monitor qui permet de visualiser les informations provenant des divers capteur de la station :



Code Arduino en Annexe 1.

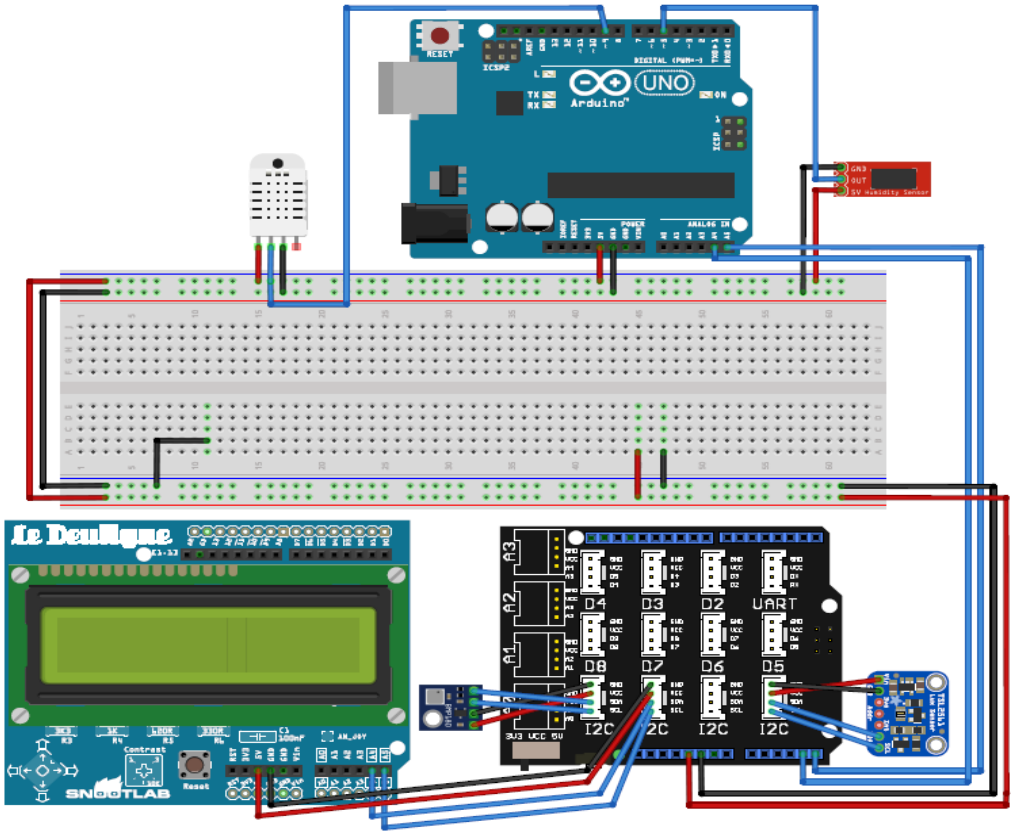
2.2) Affichage des informations de la station en local et externe

Nous allons donc dans cette partie affiché les informations en local (écran LCD) et en externe a l'aide de moyen de communication sans fil.

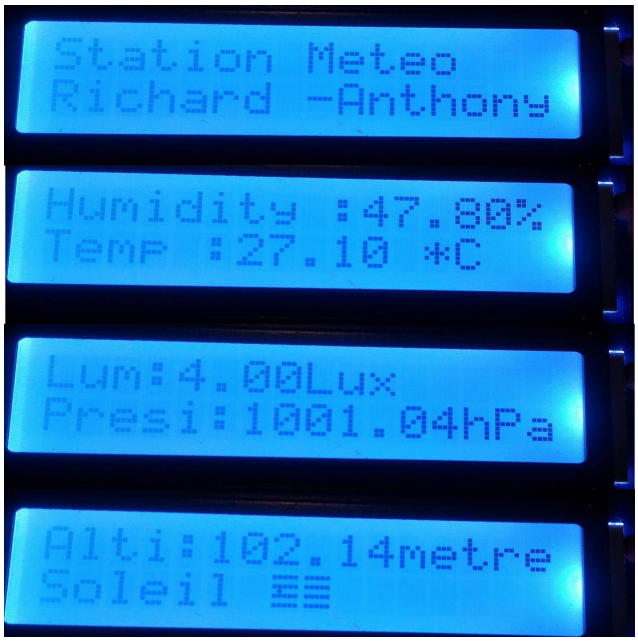
2.2.1) Affichage en local des infos capteurs sur un module LCD

Nous reprenons donc le câblage précédent et on ajoute un écran LCD en I2C pour obtenir les informations en direct sur la station.

Schéma de câblage :



Résultat obtenue sur l'écran LCD qui permet de visualiser les informations provenant des divers capteur de la station en direct :



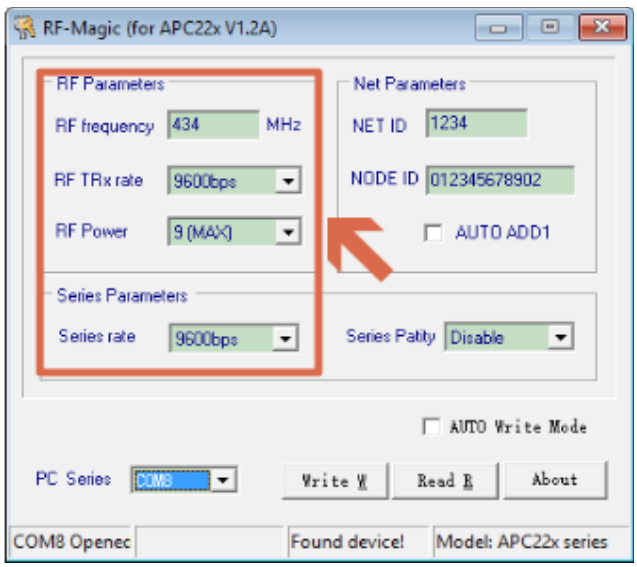
2.2.2) Transfert des infos à distance vers le PC via modules externes (ACP220 ou WIFI ou Bluetooth)

Nous avons étudier 3 moyens (2 fonctionnelle dans notre projet) de communiquer les informations de la station sans fil. Avec cela, nous avons choisi la meilleur solution correspondante a notre cahiers des charges et des besoin du projet mais nous allons détaillé les trois systèmes étudier.

Antenne ACP220 :

Commençons par les antennes ACP 220 de conception chinoise, de bonne facture avec des qualités non négligeable (longue distance 1 km maxi de portée et faible consommation d'énergie) cependant la documentation est presque inexistante et il m'aura fallut pas mal de temps pour bien configurer les deux modules pour les faire communiquer entre-eux. Nous avons un module a connecter sur l'Arduino et un module avec un adaptateur USB pour l'ordinateur.

Un logiciel est fourni ci-dessous pour pouvoir paramétrés les modules mais je suis passé par le paramétrage manuel car après une multitudes d'essai en vain avec le logiciel RF-Magic je n'ai pas réussi a les paramétré correctement.



Il faut donc régler de nombreux paramètres :

-Fréquence RF : ici 434 Mhz bande libre et utilisable car une réglementation d'utilisation existe.

-RF TRx Taux : Taux de transfert ici de 9600 bps (byte par second).

-Puissance RF : La puissance du module (de base a 9 = portée a 1km , abaissé a 5 avec plus ou moins 500-600 métrés de portée).

-Identifiant des modules : il doit être similaire pour que les deux modules communique parfaitement.

Le code de paramétrage est en Annexe 3 (il permet de lire la configuration du module, de venir lui écrire les nouveaux paramétré et enfin les ré-lire voir si l'écriture a bien réussi).

Une fois les modules bien paramétré, nous allons pouvoir envoyer les informations de notre station météo directement vers un logiciel d'affichage des PORT COM de l'ordinateur.

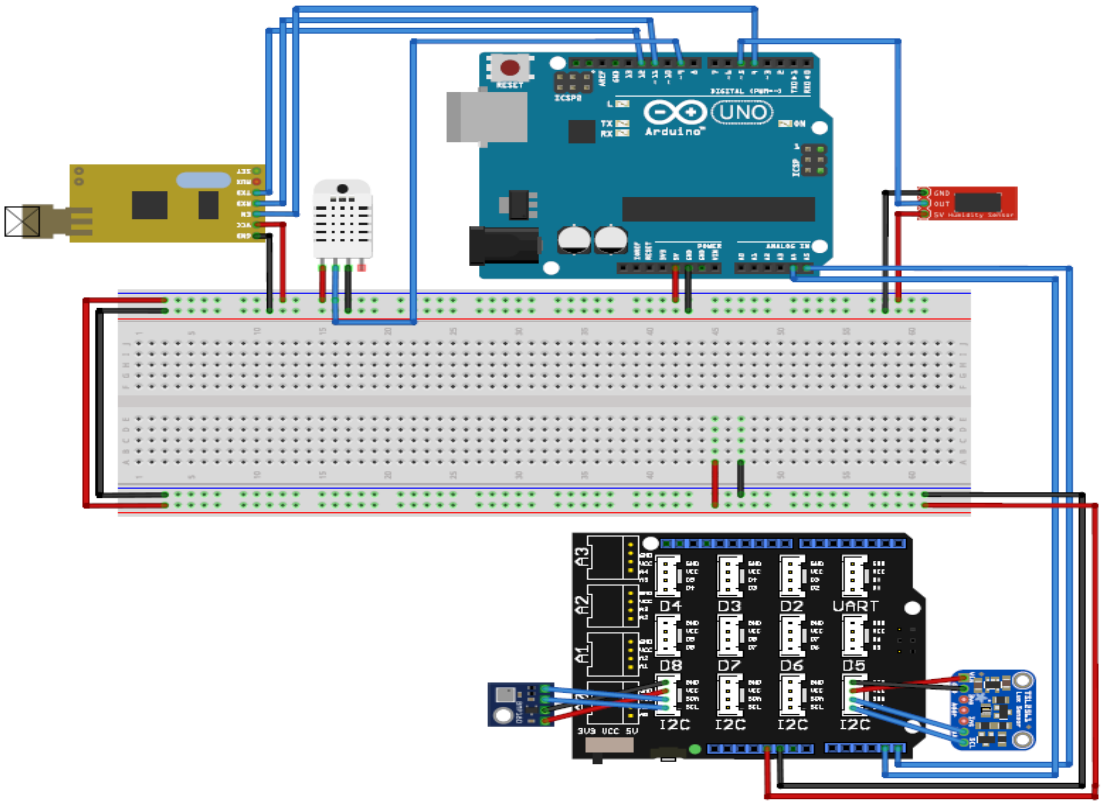
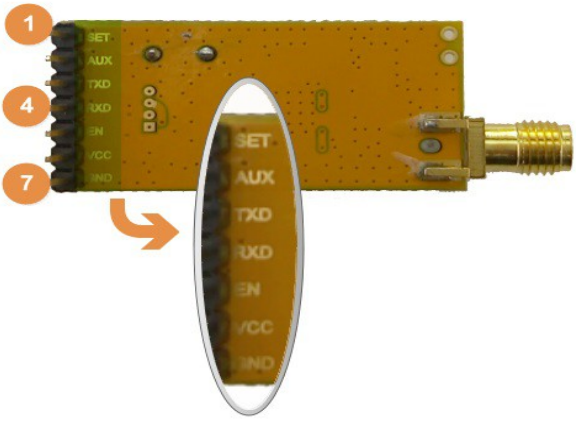


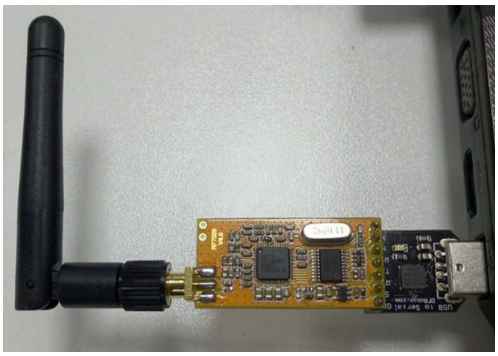
Schéma de câblage :

Le module se connectant a l'arduino est composé de 6 broches, nous en utiliserons 5 pour notre usage.

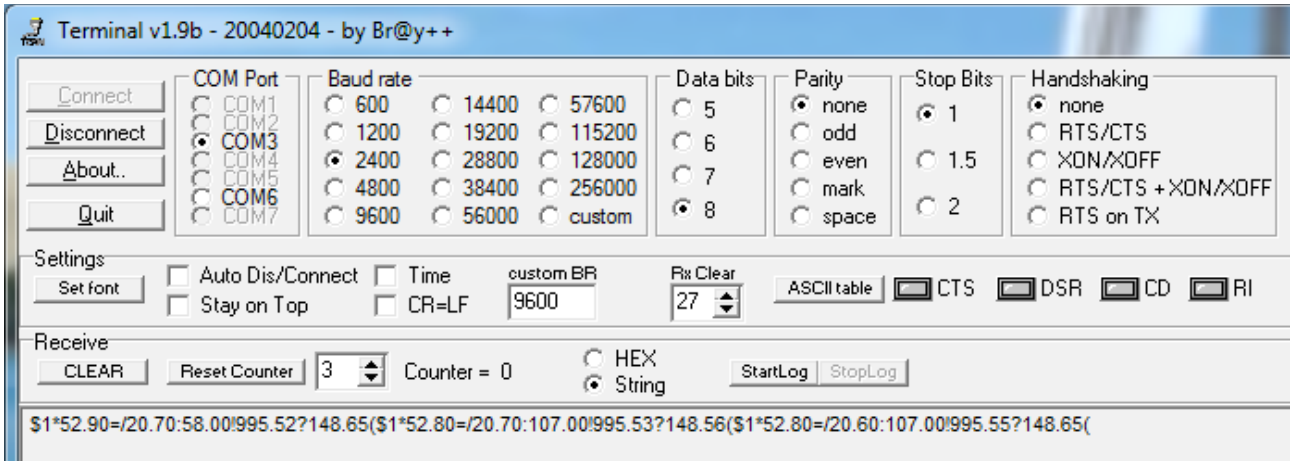
Les broches VCC et GND pour l'alimentation et la masse du module, les broches Tx et Rx pour la communication du module, la broche EN (économie d'énergie) et enfin la broche SET qui sert uniquement lors du paramétrage des modules.



Le module raccordé a l'ordinateur est livré avec un adaptateur USB :



Résultat obtenue sur le logiciel Terminal : On peut voir la trame s'afficher.



On peut voir la trame s'afficher, ce qui montre que la communication fonctionne bien, la trame est a l'état brut car elle est pour une utilisation sur le logiciel Labview.

Code en Annexe 4.

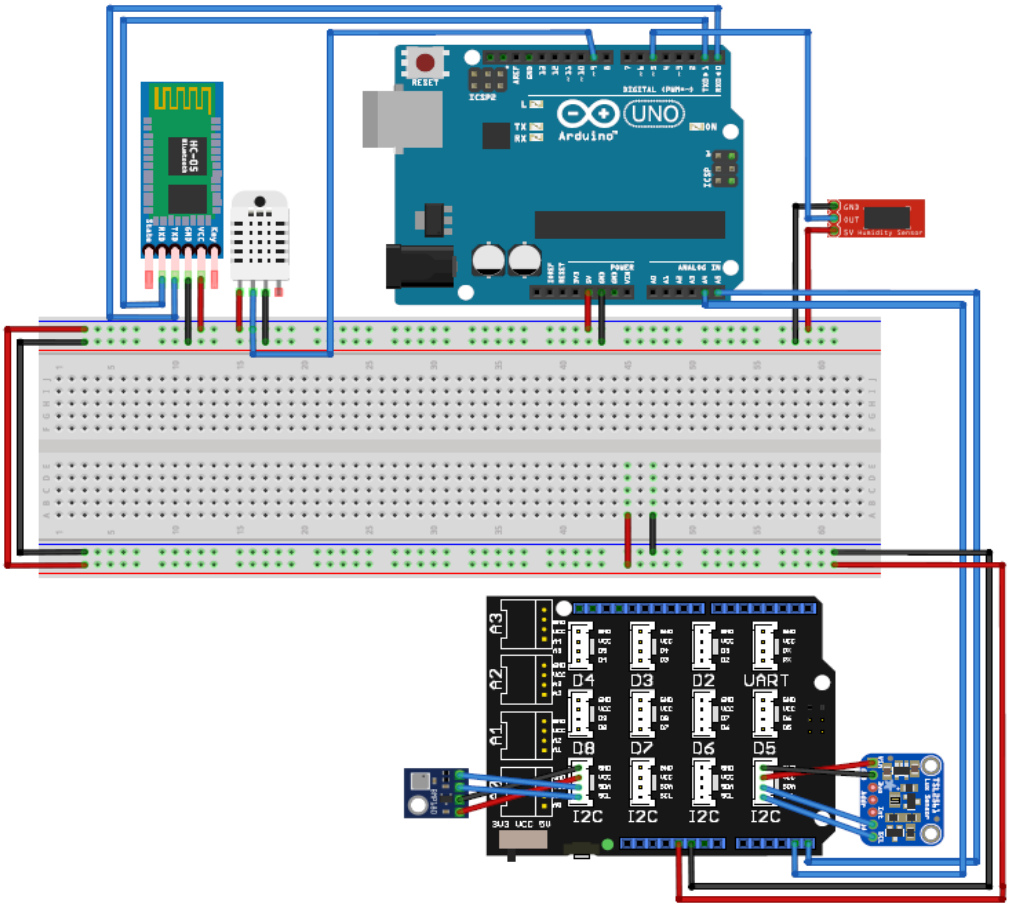
Bluetooth :

Les modules Bluetooth dispose d'une bonne notoriété et sont très répandu, nous avons donc établie une connexion par liaison Bluetooth vers un ordinateur (avec une interface LABVIEW) et une application Bluetooth sur smartphone Android qui seront expliqué dans le chapitre des interfaces.

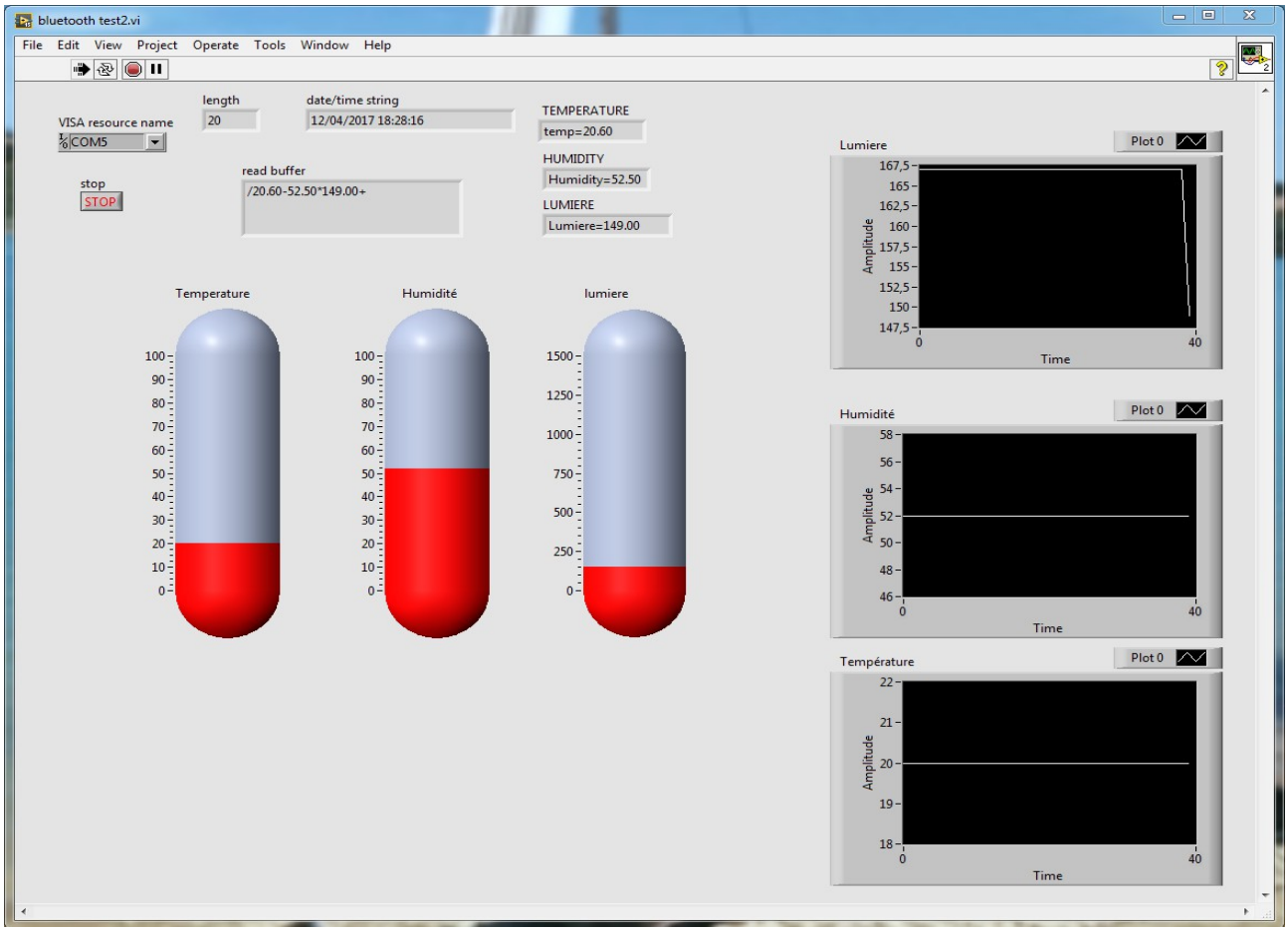
Les modules Bluetooth ne nécessite pas de paramétrage particulier il faut comme tous système communiquant bien les raccordé sur les broches de Tx (Transmetteur) et Rx (Récepteur) associé. Il faut une fois activé, s'appareiller au module nommé HC-06 avec le mot de passe 0000 et la connexion est établie, le transfert de donnée peut commencer.

Le module :

Schéma de câblage :



Résultat obtenue sur LABVIEW et l'application Android développée via le logiciel APP Inventor 2 :

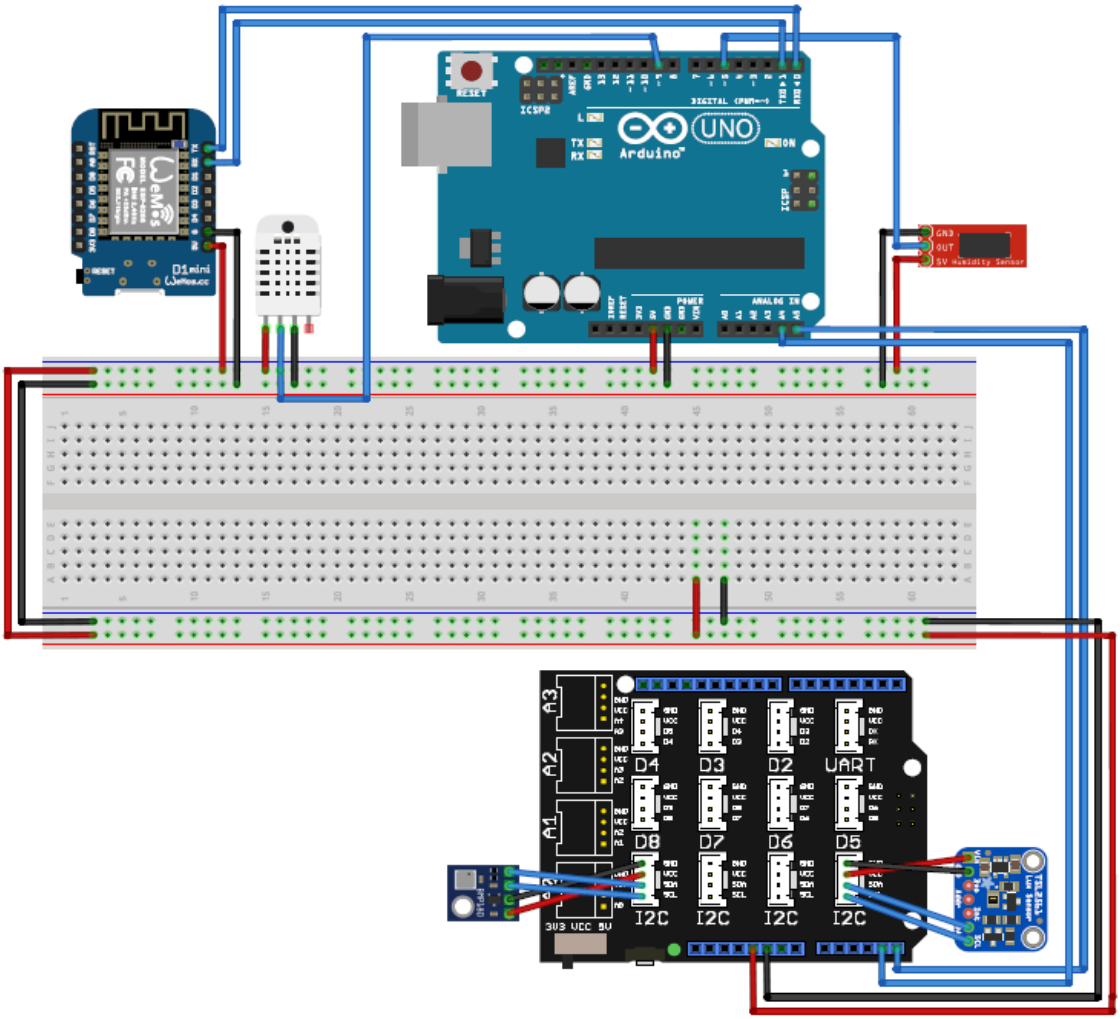


Code Application Android en annexe 5 et Labview en annexe 6.

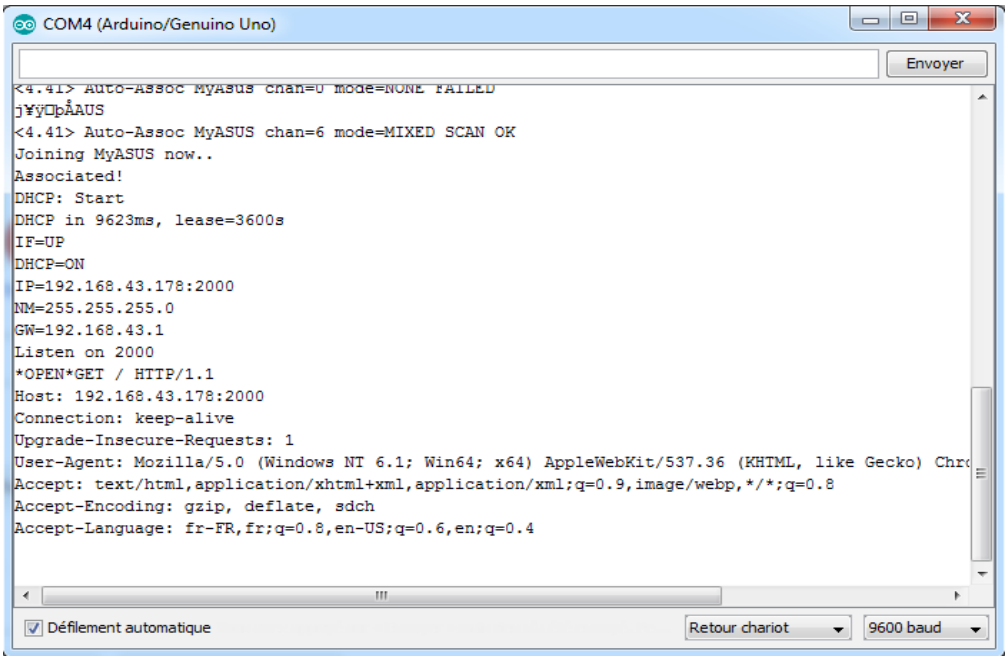
Wi-fi :

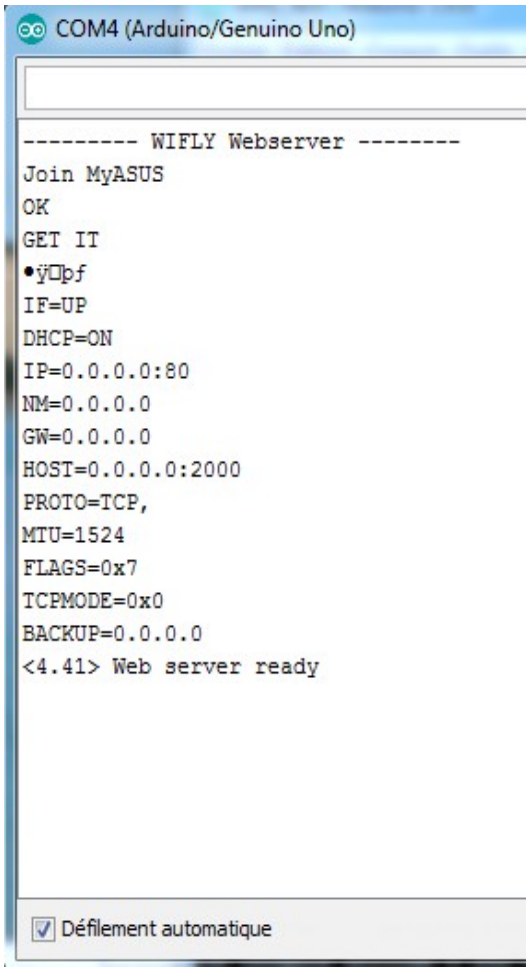
Notre troisième module de communication est le module Wifi Shield comprenant un accès wifi, il permet de se connecter a un point d'accès internet et communiquer avec celui-ci et afficher des information provenant de la station sur une page internet. Cependant après de nombreux test je n'ai réussi qu'a le faire marcher en mode manuelle et non en mode automatique ne permettant pas d'afficher les informations de la station.

Schéma de câblage :



Résultat obtenu en mode manuel qui marche et l'automatique non :





Code en mode manuelle Wi-fi en Annexe 7 et en mode automatique Annexe 8 .

2.2.3) Étude de la portée

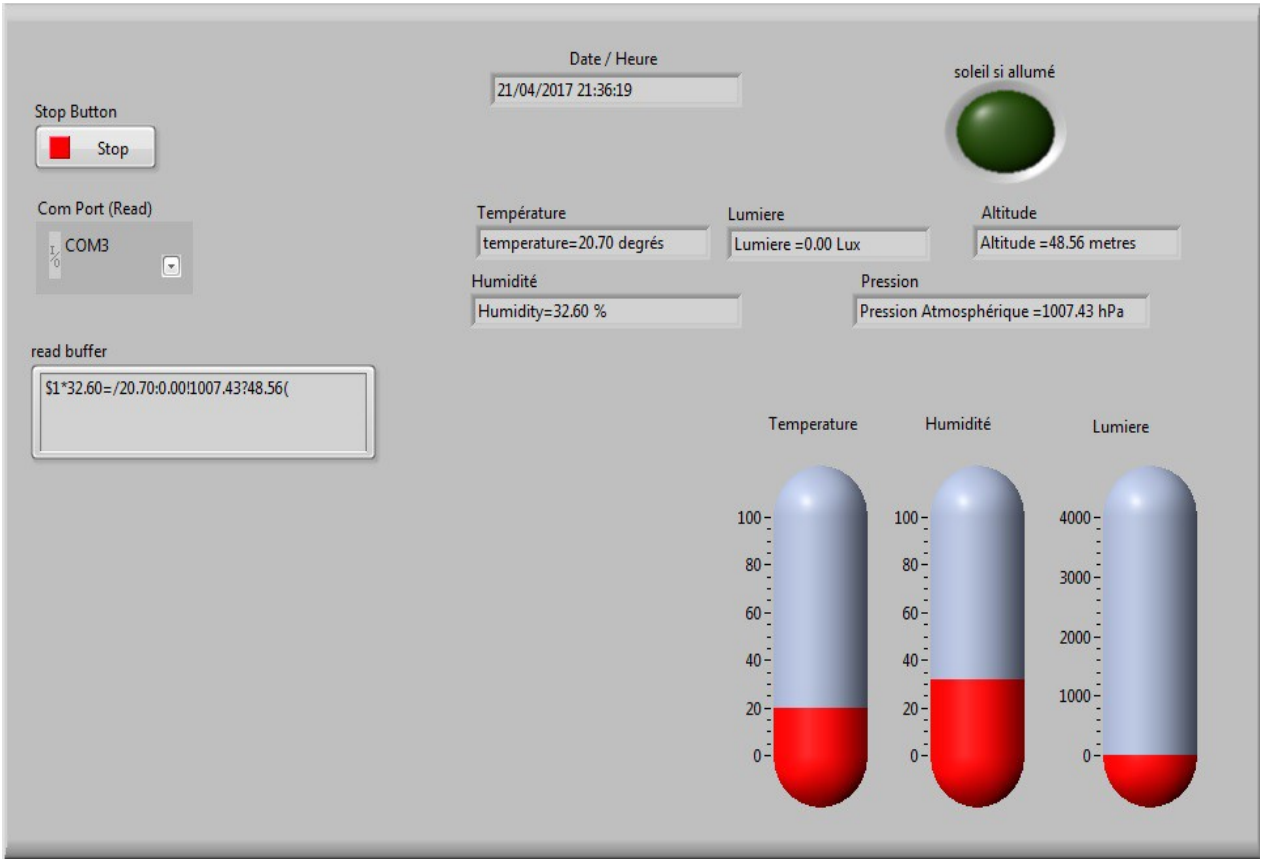
Cette étude nous as permit de définir la meilleur option pour notre station mais aussi en adéquation avec notre cahier des charges. Nous avons déjà une idée des distance de chaque technologie mais nous avons vérifier par nous même pour être sur des relevés .

Distance en mètres	Antenne ACP220	Bluetooth	Wi-fi
	1 km max	10 mètres	20 mètres

Nous relevons que pour la communication en Bluetooth et Wi-fi, il faut privilégié des distance ouverte et non confiné car un mur ou autre chose entre la connexion brouille le signal et donc la réception, concernant les antennes ACP220, 1km en distance ouverte sont atteignable et on peut essaiment traverser de nombreux murs avec une réception bonne.

2.2.4) Prévoir une application LABVIEW qui affichera les données récupérées

Nous avons développé une interfaces Labview pour la réception en Bluetooth et Antenne ACP220 , cependant la version antenne est la plus développée car c'est elle qui sera utilisé pour le projet. Il a fallut comprendre l'ouverture de PORT COM et récupérées les données émise par la station météo et les mettre en formes.



2.3) Mise en œuvre de l'autonomie de la station

2.3.1) Rendre le projet entièrement autonome (fonctionnement sur piles, accus)

Une fois nos choix défini pour la station concernant le matériel (Arduino Uno + base shield + 4 capteur + antenne). Nous avons utilisé une pile 9V pour alimenter notre montage, grâce a l'adaptateur qui se branche directement sur la prise Jack de l'arduino. Celle-ci permet d'alimenter le montage avec une tension entre 7 et 12V et de fournir 5V permanent grâce au régulateur de tension, grâce à cela la station devient autonome mais pour combien de temps ?

2.3.2) Étude de la consommation, de la durée de fonctionnement

Nous avons mesuré la consommation de la station et nous obtenons une consommation de 58 mAh au repos et de 69 mAh en émission, il faut savoir que la station va rester au repos 99% du temps au repos donc la consommation va être de 58 mAh en moyenne global.

Pour démarrer cette partie autonomie, nous avons commencer avec la pile 9V de capacité de 270 mAh, qui donne une autonomie de 4h30 a peu prés, nous somme donc loin de notre objectif du station autonome.

2.3.3) Affiner le projet afin d'avoir une consommation minimale de la Station Météo

Comme nous l'avons vu plus haut, la consommation de la station est très importante et une pile 9V présente une faible autonomie de part de sa petite capacité mais aussi de sa tension élevé qui oblige d'utiliser le régulateur de tension qui a un rendement très faible et donc consomme énormément.

De plus, l'arduino Uno que nous utilisons est très bien car complet en fonctionnalité mais il n'est pas adapté pour une utilisation nomade car il n'a pas une bonne efficacité énergétique.

Dans un premier temps, ont a ajouté un mode sleep (watchdog) a l'arduino qui permet de mettre en veille les capteurs et fonctionnalité inutilisé sur la carte lorsqu'il ne fonctionne pas et activé le mode économie d'énergie de l'antenne ACP220 en la désactivant simplement .

Voici nos mesures :

Mode	Au Repos	En Émission
Arduino Uno seul	34 mAh	//////////////////////////
Arduino Uno + shield + capteur + antenne	58 mAh	69 mAh
Arduino Uno + shield + capteur + antenne + watchdog + eco-antenne	28 mAh	69 mAh

Avec le mode veille watchdog et le mode économie d'énergie de l'antenne, ont arrive a une baisse assez conséquente (autonomie de 9h30 au lieu de 4h30 sur la pile 9V) mais c'est toujours trop important et donc loin de notre objectif.

On a décidé de remplacer l'organe principal , l'arduino Uno par un arduino Nano qui a l'avantage d'avoir que le strict nécessaire pour fonctionner ce qui permet d'avoir un arduino très peu consommateur en énergie face a son frère aîné le Uno.
Notre station va émettre toute les heures pour avoir une information des conditions météo a intervalle régulier.

Voici nos mesures expérimentales :

Mode	Au Repos	En Émission
Arduino Nano seul	11 mAh	//////////////////////////
Arduino Nano + shield + capteur + antenne	37 mAh	49 mAh
Arduino Nano + shield + capteur + antenne + watchdog + eco-antenne	9 mAh	46 mAh

Voici le fonctionnement du Code pour atteindre ces mesures :


```
    digitalWrite(brocheEN, HIGH);          // Antenne active
    delay(1000);

   apc220.print("$");

  if(digitalRead(brocheD5) == LOW)         // capteur de pluie
   { apc220.print("0");
   }
 else if (digitalRead(brocheD5) == HIGH)   // capteur de pluie
   {   apc220.print("1");
   }
    apc220.print("*");                     // Emission de la trame
    apc220.print(h);
    apc220.print("=");
    apc220.print("/");
    apc220.print(t);
    apc220.print(":");
    apc220.print(x);
    apc220.print("!");
    apc220.print(r);
    apc220.print("?");
    apc220.print(y);
    apc220.print("(");

    delay(1000);
    digitalWrite(brocheEN, LOW);          // Antenne inacive
    delay(1000);

    Sleepy::loseSomeTime(10000);          // Mode sleep 10 secondes
```

Avec cette configuration, l'arduino nano plus les mode veilles activés ont arrive a une consommation de seulement 9 mAh pendant 99% du temps, ce qui permet d'avoir une autonomie sur la pile 9V de 30 heures ! , c'est beaucoup mieux mais pas assez pour une station autonome.

2.3.4) Étudier le dimensionnement d'un système de recharge de batterie avec panneau photovoltaïque

Le principe de cet étude est de rendre autonome la station météo en extérieur.
Rappelons les différents équipements constituant la station :

- arduino nano
- base shield arduino
- antenne apc220
- capteur de température/humidité AM2302
- capteur de lumière TSL2561
- capteur de pluie LMK47
- capteur de pression atmosphérique et altitude BMP280

Pour commencer cette étude on devait chercher un moyen de rendre notre projet autonome en extérieur. Après plusieurs recherche sur internet nous avons vu qu'il existé un équipement qui permettrait de faire tourner une carte telle que l'Arduino et de charger en même temps la batterie relié a ce système, cet équipement est la Lipo Rider Pro.

La Lipo Rider Pro est une carte intelligente totalement autonome permettant de charger et d'alimenter notre projet avec une tension stabilisée 5V depuis une batterie au Lithium 3,7V, d'un panneau solaire et/ou une source USB. Ceci est un moyen parfait pour rendre notre projet autonome. Ce système est déjà intégralement programmé pour alimenter l'arduino et charger la batterie via un panneau solaire.

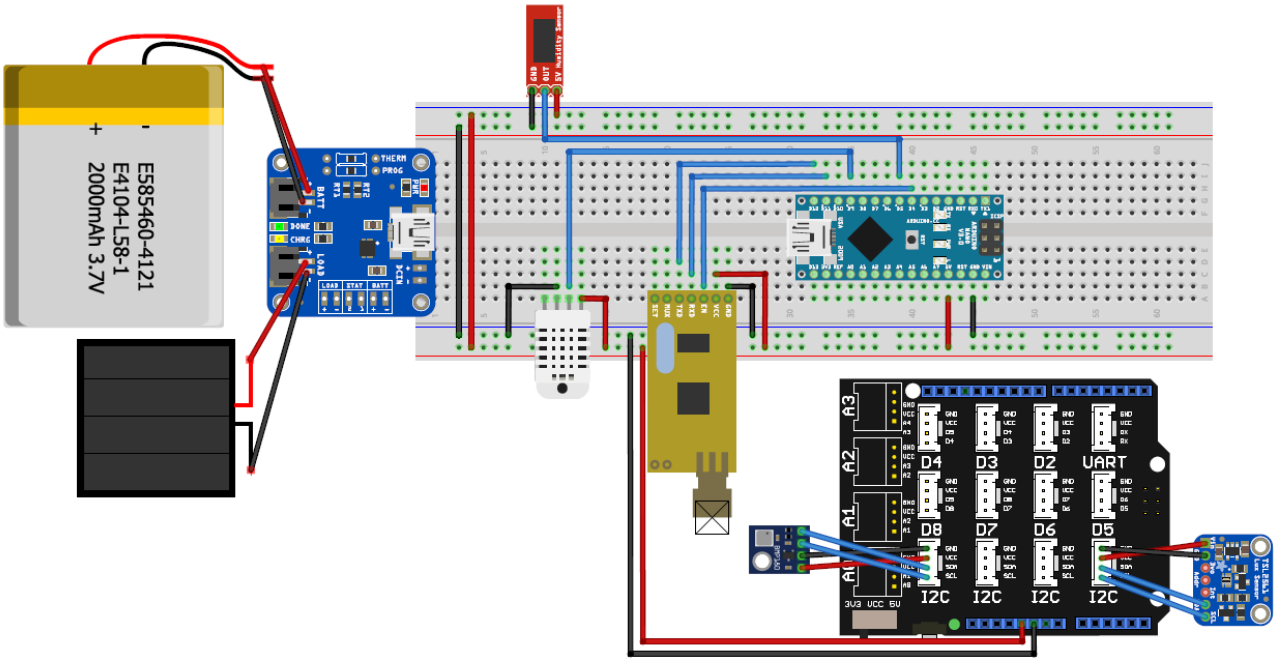
On a ensuite cherché la batterie qui conviendrait à notre station.
On rappelle que sous une alimentation +5V à l'entrée Vin de l'arduino, on arrive à une consommation moyenne de 9mA. On a fini par prendre une accu Lipo CONRAD 3,7V 2100mAh.

Compatible pour notre projet, avec une capacité de 2100mAh donc une autonomie d'environ 230h, un courant de charge max de 4,2A et un courant de décharge max de 42A.

Puis nous avons choisi un panneau solaire adapté pour notre montage, il s'agit d'un panneau CONRAD de dimension 82x120x3mm (largeur, longueur, épaisseur), fournissant 6V et 150mAh ce qui est largement correct pour notre installation .

Grâce a cette solution, la station météo peut tenir théoriquement sur la batterie 230h soit presque 10 jours, ajouté a cela un panneau solaire qui permet de recharger et de faire fonctionner la station, on se retrouve avec une station totalement autonome selon la météo bien sur.

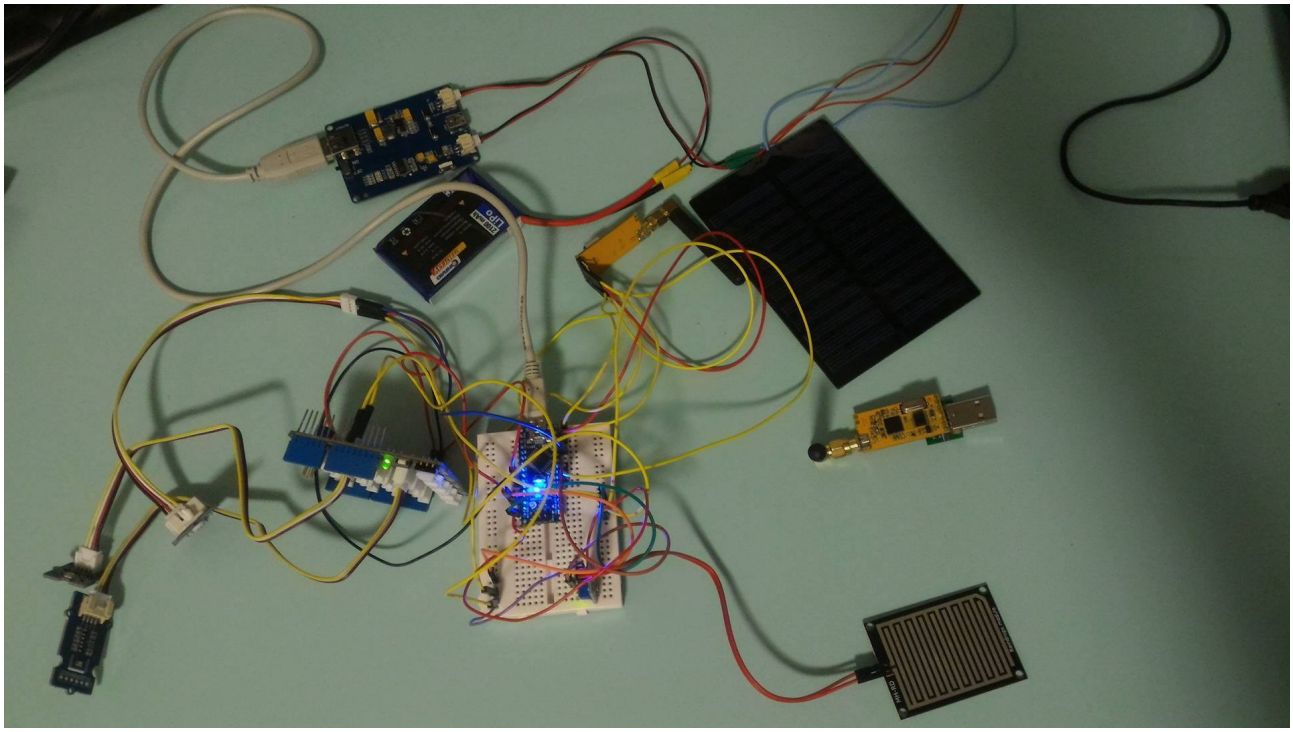
Schéma de câblage :



3. Présentation du projet Fonctionnelle

3.1) Rendu du projet Fini

Voici l'installation final , nous n'avons pas fait de prototype car ce n'était pas demandé .



3.2) Coût du projet

Voici le listing du matériel utilisé avec le prix et donc le coût du projet :

	Prix
Arduino Nano	22,00 €
Module Base Shield	9,70 €
Antenne APC220	55,80 €
Capteur de temp/hum DHT22	16,40 €
Capteur d'irradiation TSL2561	6,80 €
Capteur de pression BMP280	11,00 €
Capteur de pluie LMK47	4,50 €
Lipo Rider Pro	15,50 €
Batterie Lipo Conrad 3,7V 2100mAh	17,00 €
Panneau Solaire Conrad 6V 150mA	25,00 €
Total	183,70 €

3.3) Amélioration possible du projet

Il y a de nombreuse amélioration possibles pour la station météo autonome :

Dans l'interface Labview , il faudrait avoir un fichier qui enregistre les valeurs et que l'Arduino puisse être synchronisé avec l'heure local pour émettre des données que le jour et pas la nuit .
Il faudrait aussi remplacer le capteur de lumière par un Pyromètre pour avoir les données en W/m2 et ajouter un gyromètre pour le vent .
Il faudrait aussi voir pour remplacer l'arduino Nano par un Arduino mini plus performant .

Conclusion

Lors des heures de TP et chez nous, on a pu développer des compétences techniques propres a notre projet, on a su travailler en équipe et on a su gérer les problèmes qu'on a eu par exemple sur l'obtention des données sur Labview et la configuration des modules ACP220.
Dans le projet station météo on a dû approfondir nos connaissances en C/C++ pour le logiciel de programmation arduino, on a dû s'initier au logiciel Labview afin d'afficher les données de la station météo, et on a dû s'initier aussi au développement d'un projet, c'est à dire rechercher les moyens nécessaire à l'aboutissement de ce projet et les mettre en œuvre.
Au final le cahier des charges est totalement respecté, on a su mettre en œuvre les différents capteurs, l'écran LCD, le transfert à distances des informations vers un PC ou un smartphone, et on a su rendre totalement autonome la station en extérieur via un accu et un panneau photovoltaïque en partant sur une étude de consommation minimal de la station.
La pratique de toutes ces connaissances nous a beaucoup appris et c'est grâce à ce genre de projet qu'on peut exercer nos savoir dans un environnement de travail qu'on peut trouver dans une entreprise.

Annexes

Annexe 1 : Code Arduino Mise en route de capteur

```
#include "DHT.h"
#include <SoftwareSerial.h>
#include <Wire.h>
#include <Digital_Light_TSL2561.h>
#include "Seeed_BMP280.h"

#define DHTPIN A2
#define DHTTYPE DHT22   // DHT 22  (AM2302)
#define brocheD5 5

DHT dht(DHTPIN, DHTTYPE);
BMP280 bmp280;

void setup()
{
   Wire.begin();
   Serial.begin(9600);
   Serial.println("Information de la station :");
   TSL2561.init();
   dht.begin();
   pinMode(brocheD5, INPUT);
    bmp280.init();
}

void loop()
{
   float h = dht.readHumidity();
   float t = dht.readTemperature();
   float pressure;
   float pressurehPa;
   pressure = (bmp280.getPressure());
   float r=(bmp280.getPressure()*0.01);
   float y=bmp280.calcAltitude(pressure);
   float n=TSL2561.readVisibleLux();

      Serial.print("Humidity: ");
      Serial.print(h);
      Serial.print(" %\t");
      Serial.print("Temperature: ");
      Serial.print(t);
      Serial.println(" *C");
       Serial.print("Pression atmospherique: ");
      Serial.print(r);
      Serial.println(" hPa");
```

```
        Serial.print("Altitude : ");
       Serial.print(y);
       Serial.println(" metres");
       Serial.print("Irradiation: ");
       Serial.print(n);
       Serial.println(" Lux");

     if(digitalRead(brocheD5) == LOW)
{
   Serial.println("il pleut !! ");
 }
else if (digitalRead(brocheD5) == HIGH)
{
   Serial.println("Soleil ");
   }
   delay(3000);
}
```

Annexe 2 : Code Arduino capteur + écran LCD

```
#include "DHT.h"
#include <SoftwareSerial.h>
#include <Wire.h>
#include <Digital_Light_TSL2561.h>
#include "Seeed_BMP280.h"
#include "rgb_lcd.h"

rgb_lcd lcd;
#define DHTPIN A2     // what pin we're connected to

#define DHTTYPE DHT22   // DHT 22  (AM2302)
#define brocheD5 5

DHT dht(DHTPIN, DHTTYPE);
BMP280 bmp280;

void setup()
{
   Wire.begin();
   Serial.begin(9600);
   Serial.println("Information de la station :");
   TSL2561.init();
   dht.begin();
   pinMode(brocheD5, INPUT);
   bmp280.init();
   lcd.begin(16, 2);
}
```

```
void loop()
{
   float h = dht.readHumidity();
   float t = dht.readTemperature();
   float pressure;
   float pressurehPa;
   pressure = (bmp280.getPressure());
   float r=(bmp280.getPressure()*0.01);
   float y=bmp280.calcAltitude(pressure);
   float n=TSL2561.readVisibleLux();

      Serial.print("Humidity: ");
      Serial.print(h);
      Serial.print(" %\t");
      Serial.print("Temperature: ");
      Serial.print(t);
      Serial.println(" *C");
      Serial.print("Pression atmospherique: ");
      Serial.print(r);
      Serial.println(" hPa");
      Serial.print("Altitude : ");
      Serial.print(y);
      Serial.println(" metres");
      Serial.print("Irradiation: ");
      Serial.print(n);
      Serial.println(" Lux");

    if(digitalRead(brocheD5) == LOW)
{
      Serial.println("il pleut !! ");
}
    else if (digitalRead(brocheD5) == HIGH)
{
      Serial.println("Soleil ");
}
          lcd.setCursor(0,0); //On commence à écrire en haut à gauche
          lcd.print("Station Meteo"); //On écrit "Statio météo"
          lcd.setCursor(0,1); //On commence à écrire en haut à gauche
          lcd.print("Richard -Anthony");
          delay(3000); //On attend 3s
          lcd.clear(); //On efface le lcd

          lcd.setCursor(0,0); //On commence à écrire en haut à gauche
          lcd.print("Humidity :");
          lcd.print(h);
          lcd.print("%");
          lcd.setCursor(0,1); //On se met sur la ligne du bas à gauche
          lcd.print("Temp :");
          lcd.print(t);
          lcd.print(" *C");
```

```
            delay(3000); //On attend 3s
            lcd.clear(); //On efface le lcd

            lcd.setCursor(0,0); //On commence à écrire en haut à gauche
            lcd.print("Lum:");
            lcd.print(n);
            lcd.print("Lux");
            lcd.setCursor(0,1); //On commence à écrire en haut à gauche
            lcd.print("Presi:");
            lcd.print(r);
            lcd.print("hPa");
            delay(3000); //On attend 3s
            lcd.clear(); //On efface le lcd

            lcd.setCursor(0,0); //On commence à écrire en haut à gauche
            lcd.print("Alti:");
            lcd.print(y);
            lcd.print("metres");
            lcd.setCursor(0,1); //On se met sur la ligne du bas à gauche
           if(digitalRead(brocheD5) == LOW)
{
      lcd.println("il pleut !! ");
}
    else if (digitalRead(brocheD5) == HIGH)
 {
      lcd.println("Soleil ");
}
          delay(3000); //On attend 3s
          lcd.clear(); //On efface le lcd
}
```

Annexe 3 : paramétrage module

```
#include <SoftwareSerial.h>

#define brocheRX 12
#define brocheTX 11
#define brocheSET 8

SoftwareSerial apc220(brocheRX,brocheTX);


void printspeed(unsigned char data) {
  switch(data) {
    case '0':
    Serial.println("1200 bps") ; break;
    case '1':
    Serial.println("2400 bps") ; break;
```

```
    case '2':
    Serial.println("4800 bps") ; break;
    case '3':
    Serial.println("9600 bps") ; break;
    case '4':
    Serial.println("19200 bps") ; break;
    case '5':
    Serial.println("38400 bps") ; break;
    case '6':
    Serial.println("57600 bps") ; break;
    default:
    Serial.println("???? bps");
  }
}


void setup() {

  String chaine ;

  pinMode(brocheSET, OUTPUT);
  apc220.begin(9600);
  delay(10);
  apc220.println("WR 43400 2 5 1 0");
  delay(10);

  pinMode(brocheSET, LOW);
  delay(50);

  pinMode(brocheSET, HIGH);
  delay(10);

  if(apc220.available()) {
    apc220.readStringUntil('\n');
  }

  Serial.begin(115200);

 digitalWrite(brocheSET, LOW);
 delay(10);

 apc220.println("RD");
 delay(10);

 if(apc220.available()) {
     chaine = apc220.readStringUntil('\r');
 }

 Serial.print("Configuration : ");
 Serial.println(chaine);
```

```
 Serial.print("Vitesse radio: ");
 printspeed(chaine.charAt(12));

 Serial.print("Vitesse UART: ");
 printspeed(chaine.charAt(16));

 Serial.print("Parite: ");
 switch(chaine.charAt(18)) {
  case '0':
    Serial.println("aucune"); break;
  case '1':
    Serial.println("impaire"); break;
  case '2':
    Serial.println("pair"); break;
   default:
   Serial.println("????"); break;
  }
 }


void loop()
  {
  }
```

Annexe 4 : Arduino + antenne full

```
#include <SoftwareSerial.h>
#include "DHT.h"
#include <Wire.h>
#include <Digital_Light_TSL2561.h>
#include "Seeed_BMP280.h"
#include <JeeLib.h> // Low power functions library


#define DHTPIN A2
#define DHTTYPE DHT22
#define brocheRX 12
#define brocheTX 11
#define brocheSET 8
#define brocheEN 4
#define brocheD5 5



DHT dht(DHTPIN, DHTTYPE);
ISR(WDT_vect) { Sleepy::watchdogEvent(); } // Setup the watchdog
BMP280 bmp280;
SoftwareSerial apc220(brocheRX,brocheTX);
```

```
void setup() {

   Wire.begin();       //Initialisation du bus I2C
   dht.begin();        //Initialisation du capteur Température / humidité
   TSL2561.init();
   Serial.begin(115200);
   pinMode(brocheEN, OUTPUT);
   pinMode(brocheD5, INPUT);
   apc220.begin(2400);
   bmp280.init();
   pinMode(brocheSET, OUTPUT);digitalWrite(brocheSET, HIGH);

}

void loop() {



   float h = dht.readHumidity();     // déclaration de h = valeur
   float t = dht.readTemperature();
   float x= TSL2561.readVisibleLux();

   float pressure;
   float pressurehPa;
   pressure = (bmp280.getPressure());
   float r=(bmp280.getPressure()*0.01);
   float y=bmp280.calcAltitude(pressure);


   digitalWrite(brocheEN, HIGH);
   delay(1000);

  apc220.print("$");

 if(digitalRead(brocheD5) == LOW)
  { apc220.print("0");
  }
else if (digitalRead(brocheD5) == HIGH)
  {   apc220.print("1");
  }
   apc220.print("*");
   apc220.print(h);
   apc220.print("=");
   apc220.print("/");
   apc220.print(t);
   apc220.print(":");
   apc220.print(x);
   apc220.print("!");
   apc220.print(r);
   apc220.print("?");
```

```
    apc220.print(y);
    apc220.print("(");

    delay(1000);
    digitalWrite(brocheEN, LOW);
    delay(1000);

    Sleepy::loseSomeTime(10000);

}
```

Annexe 5 : Code appli android

```
#include <Wire.h>                 //Ajout de la bibliothèque i2c pour utiliser le lcd / capteur
#include <Digital_Light_TSL2561.h> //Ajout de la bibliothèque du Capteur de lumiére
#include "DHT.h"                  //Ajout de la bibliothèque du Capteur Temperature / humidité

#define DHTPIN A2       // Pin connecté
#define DHTTYPE DHT22   // DHT 22  (AM2302)

DHT dht(DHTPIN, DHTTYPE);

void setup() {


   Wire.begin();       //Initialisation du bus I2C

   dht.begin();        //Initialisation du capteur Température / humidité

   Serial.begin(9600); //Initialisation du bluetooth

   TSL2561.init();     //Initialisation du capteur de lumiére
}


void loop() {

    float h = dht.readHumidity();     // déclaration de h = valeur
    float t = dht.readTemperature();  // déclaration de t = valeur
    float z = TSL2561.readVisibleLux(); // déclaration de z = valeur
    float y = ( (  (  TSL2561.readVisibleLux() )*0.11 ) /1 );  // déclaration de y = valeur

  /*Serial.print("The Light value is: ");
  Serial.println(TSL2561.readVisibleLux());
  delay(2000);*/

  Serial.print(t);
  Serial.print(",");
```

```
  Serial.print(h);
  Serial.print(",");
  Serial.print(z);
  Serial.print(",");
  Serial.print(y);
  Serial.print(",");
  delay(1000);

}
```

Annexe 6 : Code labview Bluetooth

```
#include <Wire.h>                 //Ajout de la bibliothèque i2c pour utiliser le lcd / capteur
#include <Digital_Light_TSL2561.h> //Ajout de la bibliothèque du Capteur de lumiére
#include "DHT.h"                  //Ajout de la bibliothèque du Capteur Temperature / humidité

#define DHTPIN A2       // Pin connecté
#define DHTTYPE DHT22   // DHT 22  (AM2302)

DHT dht(DHTPIN, DHTTYPE);

void setup() {


   Wire.begin();       //Initialisation du bus I2C

   dht.begin();        //Initialisation du capteur Température / humidité

   Serial.begin(9600); //Initialisation du bluetooth

   TSL2561.init();     //Initialisation du capteur de lumiére
}


void loop() {

    float h = dht.readHumidity();     // déclaration de h = valeur
    float t = dht.readTemperature();  // déclaration de t = valeur
    float z = TSL2561.readVisibleLux(); // déclaration de z = valeur
  // float y = ( (  (  TSL2561.readVisibleLux() )*0.11 ) /1 );  // déclaration de y = valeur

  /*Serial.print("The Light value is: ");
  Serial.println(TSL2561.readVisibleLux());
  */
  Serial.print("/");
  Serial.print(t);
  Serial.print("-");
  Serial.print(h);
```

```
  Serial.print("*");
  Serial.print(z);
  Serial.print("+");

}
```

Annexe 7 : Wifi manu

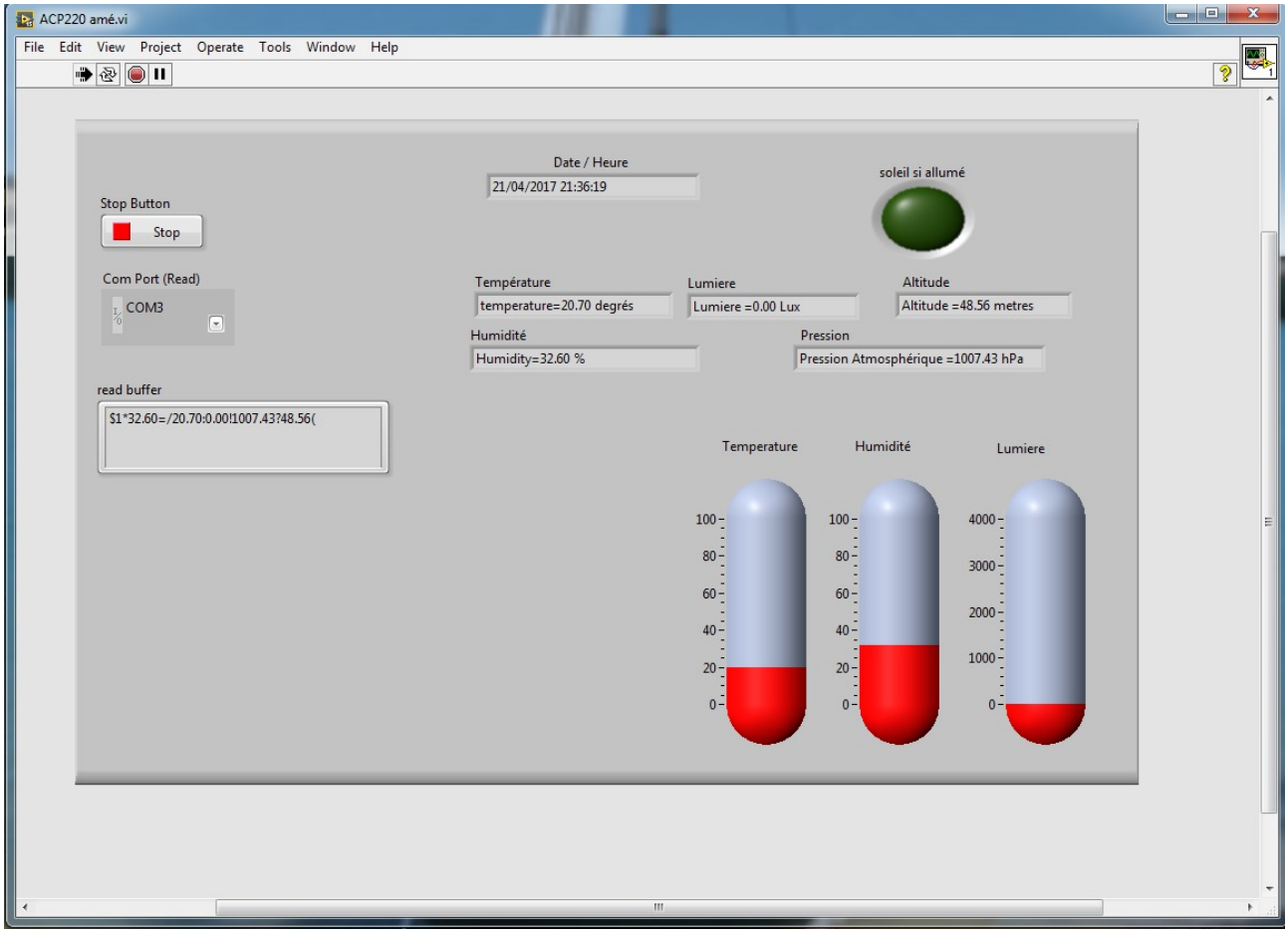
```
#include <Arduino.h>
  #include <SoftwareSerial.h>
  #include "WiFly.h"

  // set up a new serial port.
  SoftwareSerial uart(2, 3); // create a serial connection to the WiFi shield TX and RX pins.
  WiFly wifly(&uart); // create a WiFly library object using the serial connection to the WiFi shield we
created above.

  void setup()
  {
  uart.begin(9600); // start the serial connection to the shield
  Serial.begin(9600); // start the Arduino serial monitor window connection
  delay(3000); // wait 3 second to allow the serial/uart object to start
  }

  void loop()
  {

  while (wifly.available())  // if there is data available from the shield
    {
    Serial.write(wifly.read()); // display the data in the Serial monitor window.
    }

  while (Serial.available()) // if we typed a command
    {
    wifly.write(Serial.read()); // send the command to the WiFi shield.
    }

 }
```

Annexe 8 : Code wifi auto

```
#include <SoftwareSerial.h>
#include "WiFly.h"

#define SSID      "MyASUS"
#define KEY       "richardga5"
// check your access point's security mode, mine was WPA20-PSK
// if yours is different you'll need to change the AUTH constant, see the file WiFly.h for avalable
security codes
#define AUTH      WIFLY_AUTH_WPA2_PSK

int flag = 0;

// Pins' connection
// Arduino       WiFly
//  2    <---->    TX
//  3    <---->    RX

SoftwareSerial wiflyUart(2, 3); // create a WiFi shield serial object
WiFly wifly(&wiflyUart); // pass the wifi siheld serial object to the WiFly class

void setup()
{
  wiflyUart.begin(9600); // start wifi shield uart port

  Serial.begin(9600); // start the arduino serial port
  Serial.println("--------- WIFLY Webserver --------");

  // wait for initilization of wifly
  delay(1000);

  wifly.reset(); // reset the shield
  delay(1000);
  //set WiFly params

  wifly.sendCommand("set ip local 80\r"); // set the local comm port to 80
  delay(100);

  wifly.sendCommand("set comm remote 0\r"); // do not send a default string when a connection opens
  delay(100);

  wifly.sendCommand("set comm open *OPEN*\r"); // set the string that the wifi shield will output
when a connection is opened
  delay(100);

  Serial.println("Join " SSID );
  if (wifly.join(SSID, KEY, AUTH)) {
    Serial.println("OK");
  } else {
```

```
    Serial.println("Failed");
  }
  Serial.println("GET IT");
  wifly.sendCommand("get ip\r");
  char c;

  while (wifly.receive((uint8_t *)&c, 1, 300) > 0) { // print the response from the get ip command
    Serial.print((char)c);
  }

  Serial.println("Web server ready");

}

void loop()
{

  if(wifly.available())
  { // the wifi shield has data available

    if(wiflyUart.find("*OPEN*")) // see if the data available is from an open connection by looking for
the *OPEN* string
    {
      Serial.println("New Browser Request!");
      delay(1000); // delay enough time for the browser to complete sending its HTTP request string

      // send HTTP header
      wiflyUart.println("HTTP/1.1 200 OK");
      wiflyUart.println("Content-Type: text/html; charset=UTF-8");
      wiflyUart.println("Content-Length: 244"); // length of HTML code
      wiflyUart.println("Connection: close");
      wiflyUart.println();

      // send webpage's HTML code
      wiflyUart.print("<html>");
      wiflyUart.print("<head>");
      wiflyUart.print("<title>My WiFI Shield Webpage</title>");
      wiflyUart.print("</head>");
      wiflyUart.print("<body>");
      wiflyUart.print("<h1>Hello World!</h1>");
      wiflyUart.print("<h3>This website is served from my WiFi module</h3>");
      wiflyUart.print("<a href=\"http://yahoo.com\">Yahoo!</a> <a
href=\"http://google.com\">Google</a>");
      wiflyUart.print("<br/><button>My Button</button>");
      wiflyUart.print("</body>");
      wiflyUart.print("</html>");

    }
  }
}
```

Annexe labview :



Bibliographie

Internet : http://thierryperisse.free.fr/?page_id=224
« Doc centrale météo »
« Arduino LCD »
« Connecter votre Arduino en Bluetooth »
« Radio APC220 Arduino »
« Lipo Rider Pro autonomie solaire »