**Lycée Raymond Queneau**

2018

**Projet final BTS SN**

**La maison du future - Domotique**

**Chauveau Aurélien**

SOMMAIRE

# 1 - Présentation..................................................................................p3

# 2 - Les test unitaire............................................................................p5

## Le thermomètre...............................................................................p5

## L'hygromètre....................................................................................p6

## Capteur de qualité d'air................................................................p8

## Carte SD............................................................................................p10

## Radiateur..........................................................................................p12

## Module RTC.....................................................................................p14

## Compteur Electrique.....................................................................p15

# 3 - Regroupement des composant................................................p17

## Le compteur électrique, la carte SD et le module RTC.........p17

## Le thermomètre, l'hygromètre, capteur de qualité d'air et le radiateur.....................................................................................................p20

## Regroupement final des composant...........................................p21

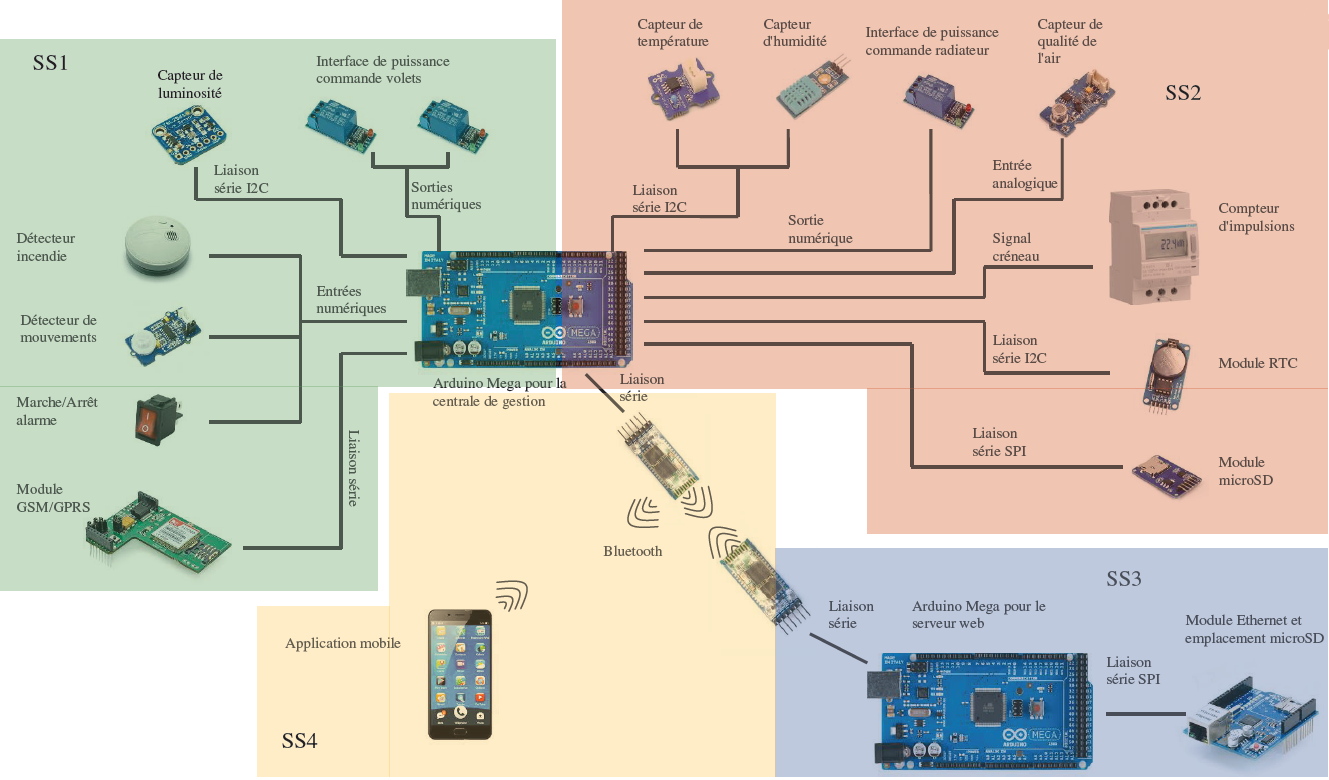
# 4 - Regroupement des différentes partie ..................................p27

## La bibliothèques capteur.h..........................................................p29

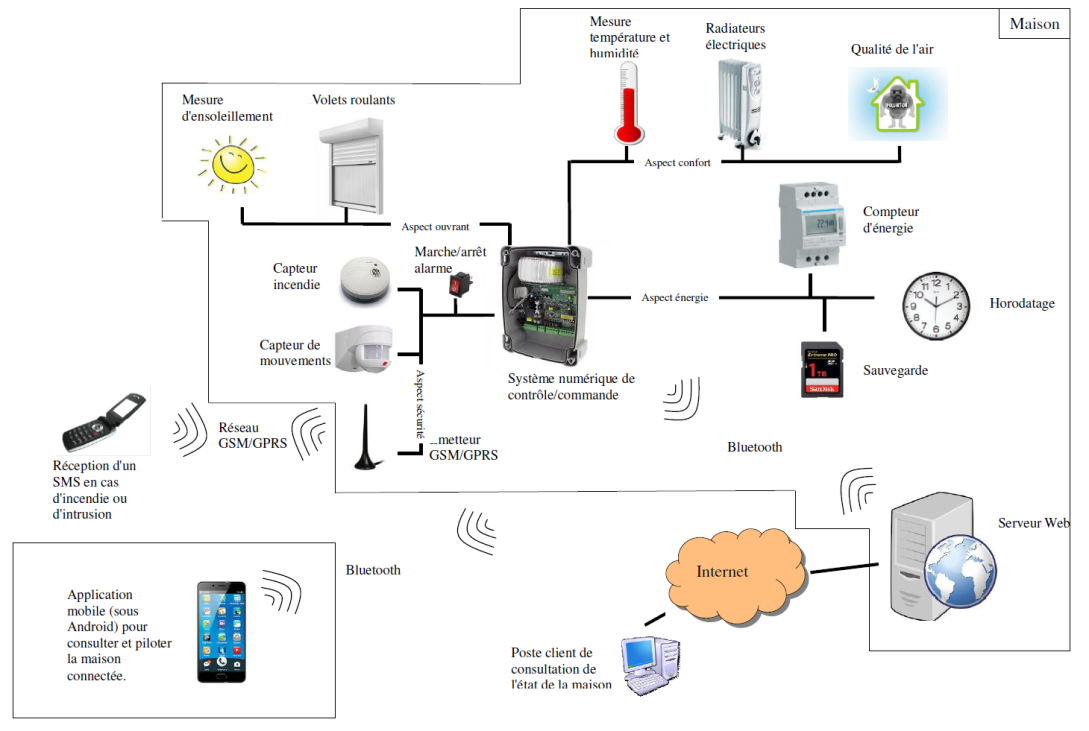
## La bibliothèques gestionMaison.h..............................................p30

## Fonction d'envoie des trames......................................................p31

1 - PRÉSENTATION



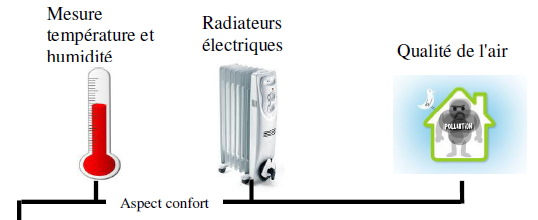
Le projet domotique (la maison du future), consiste à automatiser une maison. Pour cela nous somme 4 étudiants, répartie sur différent aspect du projet.

 L'étudiant 1 lui se chargera du contrôle des volets, de la lumière et de la partie sécurité qui comprend un capteur de mouvement et de fumer.

L'étudiant 3 seras charger de créé un serveur web qui affichera les valeurs envoyé par les étudiants 1 et 2.

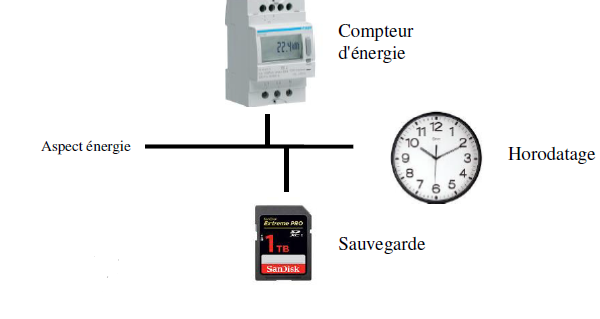
L'étudiant 4 devras créé un application mobil qui permettra de contrôler les actionneurs des étudiants 1 et 2 et d'afficher les valeurs envoyer par les étudiants 1 et 2.

Les étudiants 3 et 4 aurons une partie commune sur la communication bluetooth qui servira à la communication entre la centrale de gestion ( carte arduino ), l'application et le serveur web.

 En tant qu'étudiant 2 je suis charger de l'aspect confort et consommation.

La partie confort du projet consiste à récupérer plusieurs valeurs, utile au propriétaire, telle que : la température, le taux d'humidité ou encore la qualité de l'air. Nous aurons pour cela différents capteurs à choisir et à mettre en fonction afin de réaliser cette collecte des valeurs. Nous devrons également, piloter un relais via une carte arduino. Ce dernier nous permettra par la suite de contrôler et d'automatiser un radiateur. Pour automatiser le radiateur nous devrons récupérer une valeur ( la température ), saisie par l'utilisateur, puis nous devrons égalisé la température de la maison à la valeur récupéré. Une plage d'hystérésis est donc nécessaire, elle seras de 1°C

Chaque valeur récupéré devras être envoyé à intervalle régulier au serveur web et à l'application mobile. La température et l'humidité devront être envoyé toute les 5sec et la qualité de l'aire toute les 10 sec.



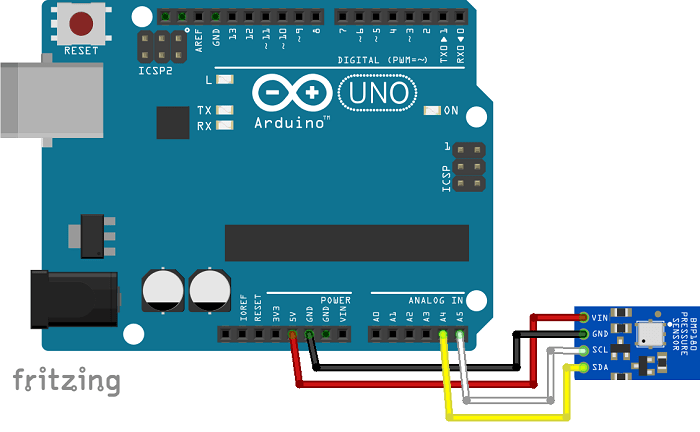
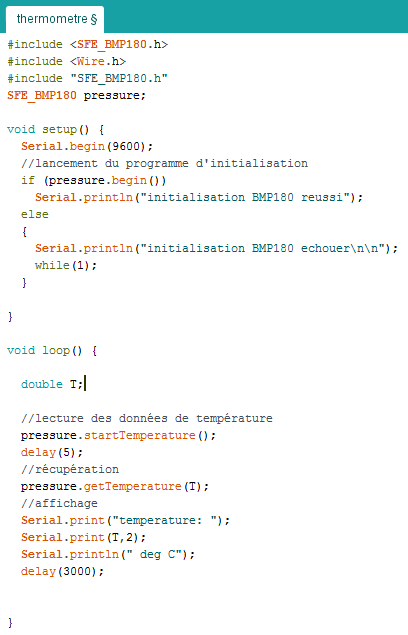
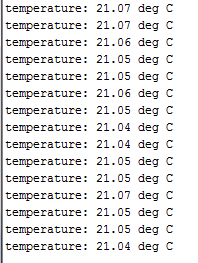
L'aspect consommation d'énergie est aussi à ma charge, pour cela nous avons un compteur d'énergie, une horloge et une carte SD. Tout cela devras nous permettre d'horodater[[1]](#footnote-2) et de sauvegarder sur une caret SD la consommation en watt par heure.

Une partie commune entre l'étudiant 1 et 2, nous charge du regroupe nos codes dans une seul carte arduino et de créé un objet pour stocker le maximum de valeur utile au bon fonctionnement du projet . Le code final devras être lisible, intuitif et donc surtout compréhensible.

Nous devrons réunir aussi tout les étudiants du groupe, afin de nous mettre d'accord sur le protocole de la trame de donnée à envoyer au serveur web et à l'application.

2 - LES TESTS UNITAIRES

# LE THERMOMÈTRE :

 Le capteur de pression BMP180 étant équiper d'un capteur de température, nous l'avons donc utilisé pour notre projet. J'ai par la suite récupéré une bibliothèque sur GITHUB[[2]](#footnote-3) pour commander le capteur. J'ai, ensuite, relier le BMP180 à une carte Arduino, créé un code simple et inclue la bibliothèque à mon code pour réaliser des tests. La communication entre la carte Arduino et notre capteur se fait par I2C.

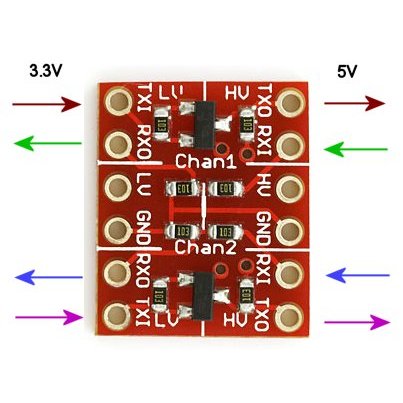
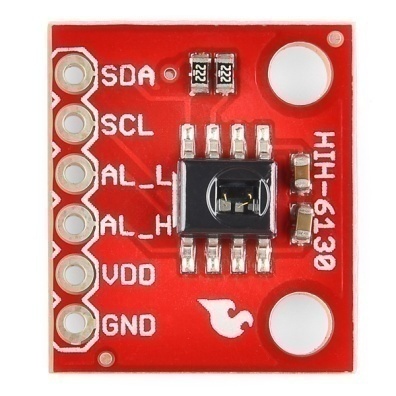
## Valeur obtenue

## Montage

## Code de test unitaire

Le capteur fonctionne bien, il est très sensible à la chaleur qu'on lui transmet lorsqu'on le touche mais à du mal à la dissipé. Il prend très vite 1 degrés en 1 sec jusqu'à 6-7 degrés en 10sec.

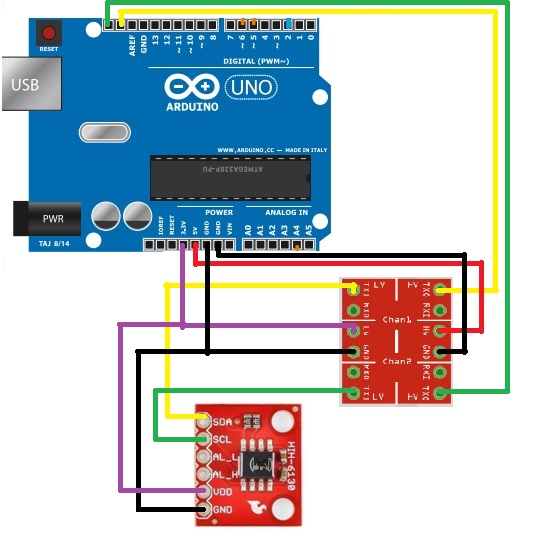
# L'HYGROMÈTRE :

 Un hygromètre est un capteur qui sert à mesurer le taux d''humité dans l'aire. Pour cela nous allons utiliser le capteur HIH6130 qui sert aussi de thermomètre et une bibliothèque trouver sur GITHUB pour gagner du temps. Le HIH6130 mesure le taux d'humidité a +- 5% entre 10% et 90% ainsi que la température entre 5°C et 50°C. Nous allons aussi avoir besoin d'un convertisseur de tension, qui nous servira à ne pas cramer notre capteur d'humidité/température avec le bus I2C qui fonctionne en 5V alors que notre HIH6130 reçois du 3,3V.

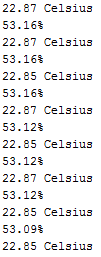
## Montage du système

## HIH6130

## Convertisseur



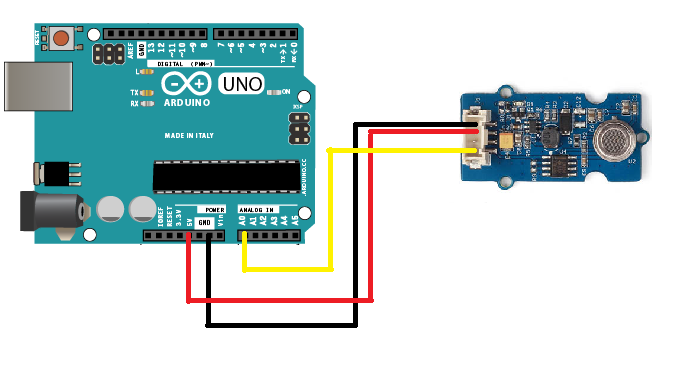
Le convertisseur nous permet de passer de 5V à 3,3V pour passer de la carte Arduino au capteur et vis versa.

 Le code de test unitaire ma permit de vérifiez les valeurs de température du capteur qui sont correct. Nous allons donc plutôt utiliser le HIH6130 pour pouvoir a la fois récupérer la température et le taux d'humidité au lieu d'avoir deux capteurs distinct.

## Valeur obtenue

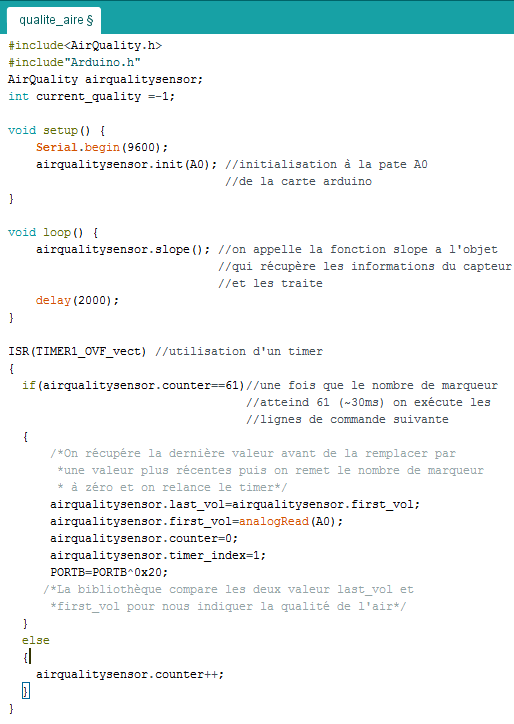
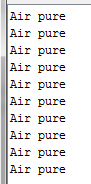
## Code de test unitaire

# CAPTEUR DE QUALITE D'AIR :

 Le capteur de qualité d'aire est branché en analogique a la carte Arduino. On récupère les informations du "Air QualitySensor" avec l'aide d'une bibliothèque ajouter a notre programme. Ce dernier nous renvoie 0 (haute pollution),1,2 ou 3(aire pure).

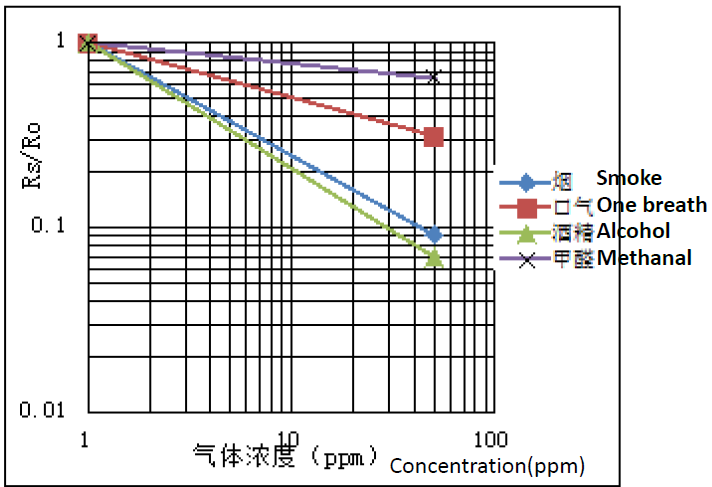
## Capteur de qualité d'air

## Montage du système



## Code de test unitaire

## Valeur obtenue

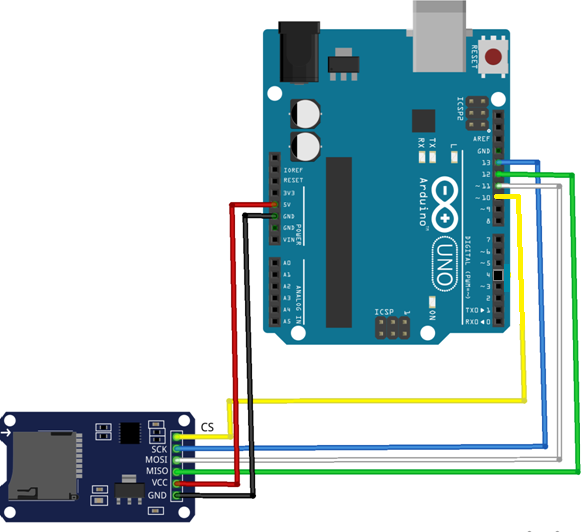


Rs est une résistance, dans un gaz ciblé avec une concentration différente, R0 est la résistance du capteur à l'air propre. Toutes les valeur sont déterminés dans des conditions de test standard.

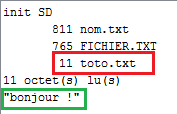
# 

# CARTE SD :

La carte SD nous servira à sauvegarder les données du compteur d'énergie. Pour cela il ma fallut apprendre à lire et a écrire dans une carte SD via une carte Arduino. Un article de Hackable magazine a fait un tutoriel à ce sujet, j'ai donc suivi le tutoriel afin de réaliser mon objectif. Il fallait aussi relier la carte SD à la carte Arduino, nous avons donc pour cela utilisé un "MicroSDcard Adapter".



## Montage du système

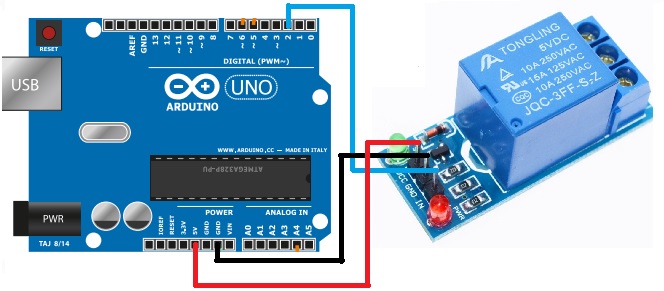


## Code de test unitaire

## Valeur obtenue

**11 toto.txt** : 11 correspond au nombre d'octet dans le fichier nommé **toto.txt**

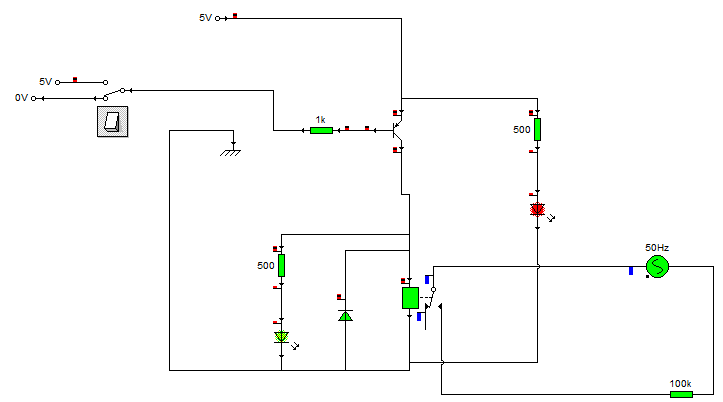
# LE RADIATEURS :

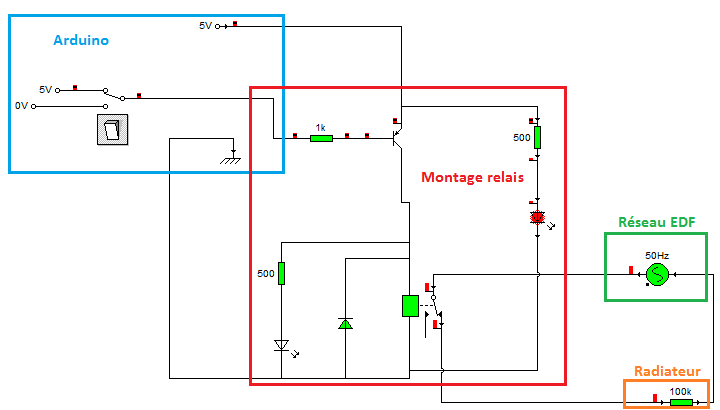
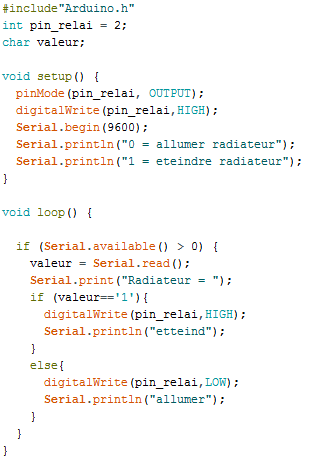
 Pour contrôler le radiateur avec une carte Arduino il nous faut un relais qui nous permettre de contrôler avec le 5V de la carte, le radiateur qui est alimenter en 230V sur le réseaux EDF.

Le schéma si dessus nous montre comment est relier la carte au CMS qui permette de contrôler le relais.

## Montage du système

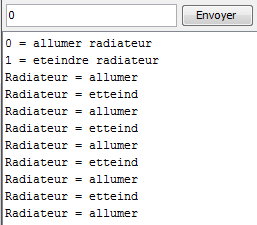
Le radiateur est éteint





## Montage du système

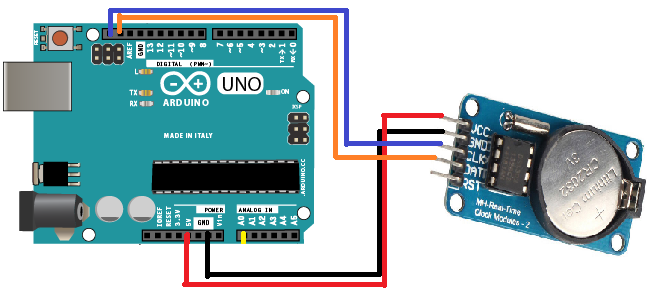
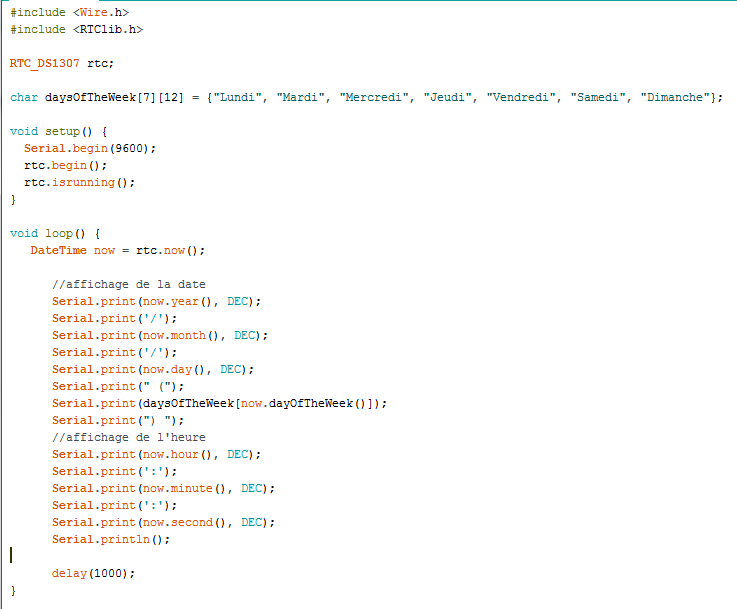
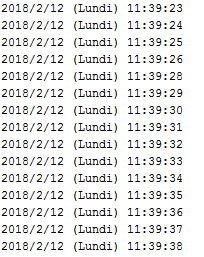
Le radiateur est allumé



## Valeur sur l'interface

## Code de test unitaire

# LE MODULE RTC :

 Le module RTC est un module qui permet de gérer l'heure, même quand le système n'est pas sous tension grâce a une petite pile. Pour le faire fonctionner j'ai fait appel à une bibliothèque sur GitHUB. Avec les méthodes j'ai donc créé un mini programme donnant l’heure qui nous servira pour l'horodatage de donné de consommation.

## Valeur obtenue

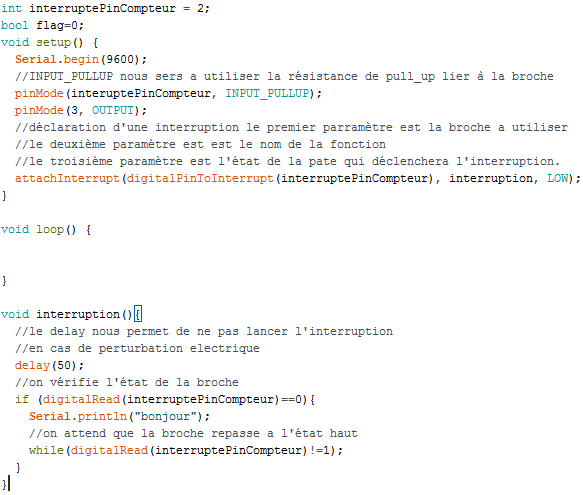
## Code de test unitaire

## Montage du système

# COMPTEUR ELECTRIQUE :

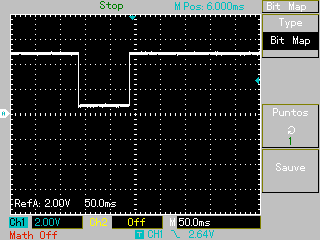
Avec le compteur électrique nous allons pouvoir récupérer le nombre de KWh que le radiateur va consommer. Il délivrera une impulsion tout les 100Wh que nous allons récupérer avec la carte Arduino.

J'ai pour cela utiliser la pin 2 de la carte car elle est la seul pin avec la numéro 3 a posséder une interruption externe. Ensuite nous avons utilisé la résistance de pull-up interne de la pin 2 afin de ne pas avoir à monter le composant sur une carte.

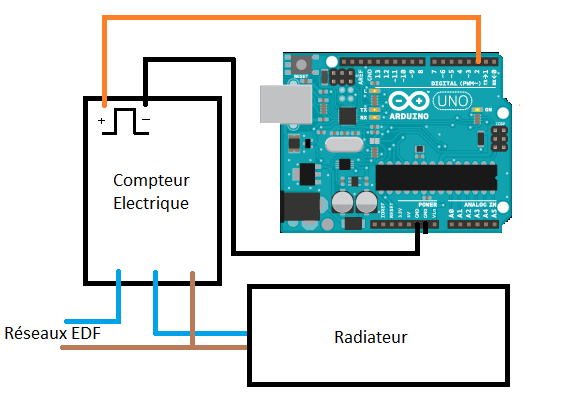


## Code de test unitaire

Le délai est de 50 ms se qui correspond a la moitié de la durée de l'impulsion générer par le compteur en suite on revérifie l'état de la broche. Cette opération nous permet d'éliminer les parasites lié aux perturbations électriques.

Voici à l’oscilloscope, l’impulsion généré par le compteur électrique :

## Montage du système

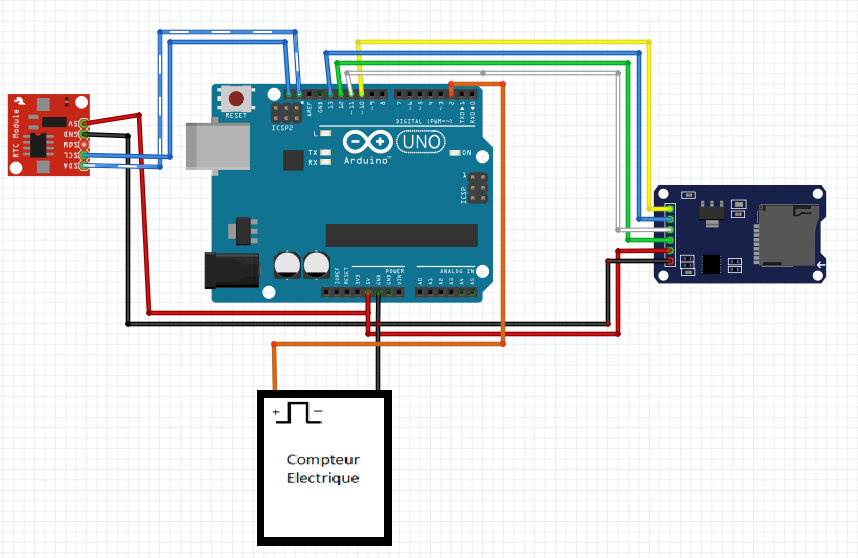


3 - REGROUPEMENT DES COMPOSANTS

Maintenant que tout les teste unitaire son fait nous pouvons regrouper les composant sur une seul carte et les codes dans un seul fichier.

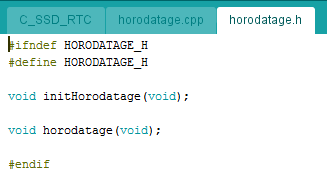
# LA CARTE SSD, LE MODUL RTC ET LE COMPTEUR ÉLECTRIQUE :

## Montage du système



## RTC

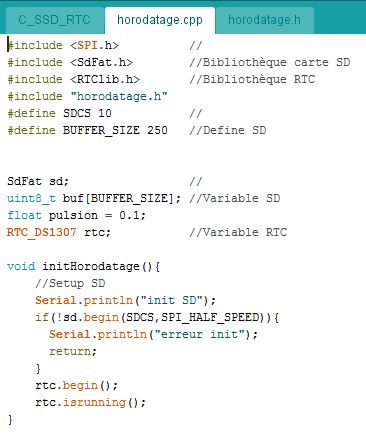
## SSD



## .h du regroupement



## Code principale du regroupement.



Une des utilités d’exécuter le programme d’interruption dans le loop, est que cela évite les conflits d’interruption. Car la bibliothéque qui nous sert à horodater utilise aussi une interruption, donc si nous avions exécuté la fonction horodatage dans le programme d’interruption il y aurais eu conflit d’interruption (une interruption dans une interruption).

## .cpp partie 1 du regroupement

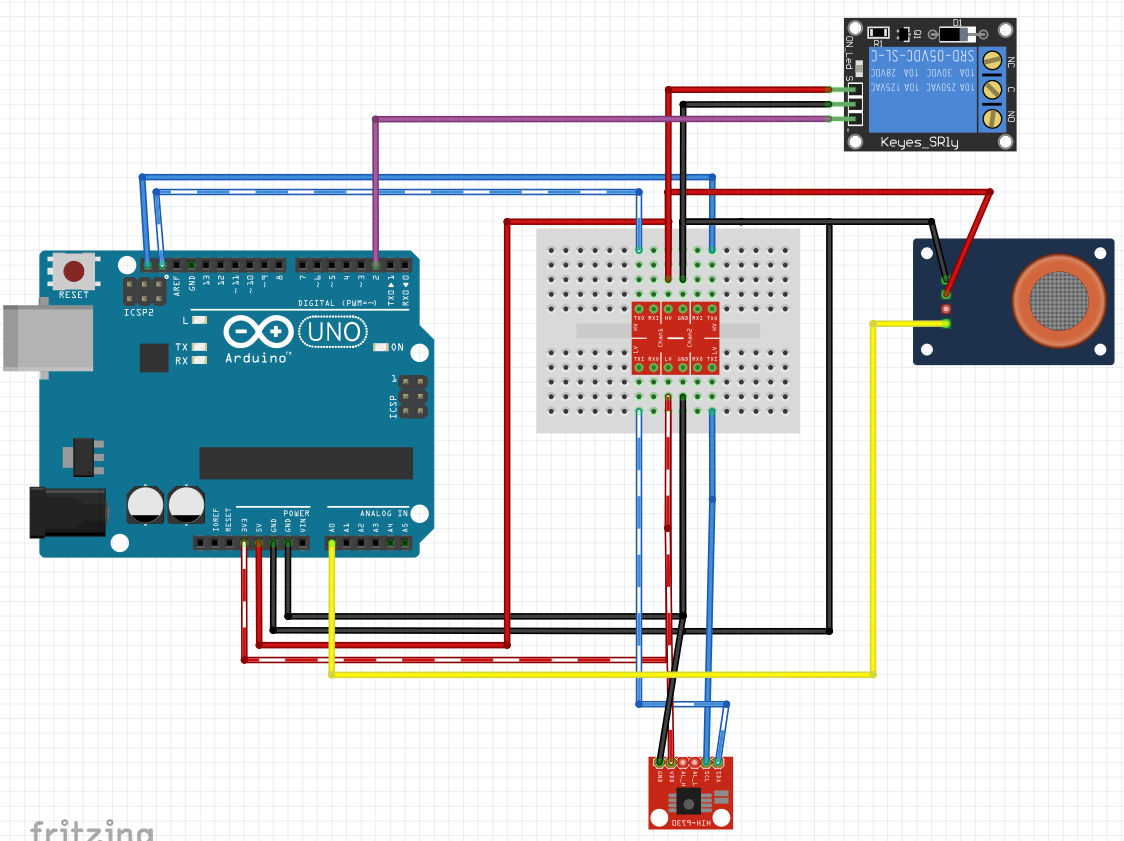
# 

## .cpp partie 2 du regroupement

Dans cette partie nous avons regroupé tout se qui concerne l'horodatage de la consommation électrique. Le compteur crée une impulsion à la quelle on ajoute une date, une heure et une valeur en Watt/h, puis cette trame est sauvegardée sur une carte SD dans un fichier texte (.txt).

# LE TERMOMÉTRE, l'HYGROMÉTRE, LE CAPTEUR DE QUALITE D'AIR ET LE RELAIS :

## Montage du système



## RELAIS

LE RELAIS

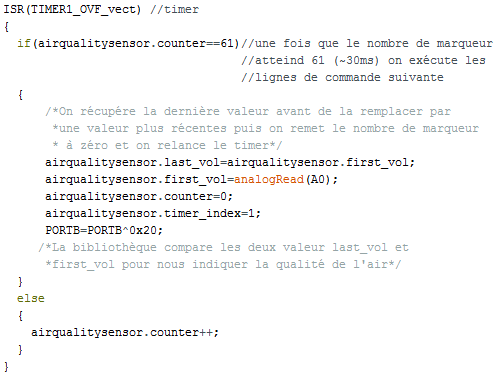
## HYGROMÉTRE

## TERMOMÉTRE

## QUALITE D'AIR

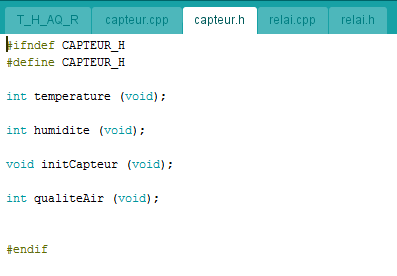


## Code principale du regroupement.



## capteur.cpp partie 2 du regroupement

## capteur.cpp partie 1 du regroupement



## capteur.h du regroupement



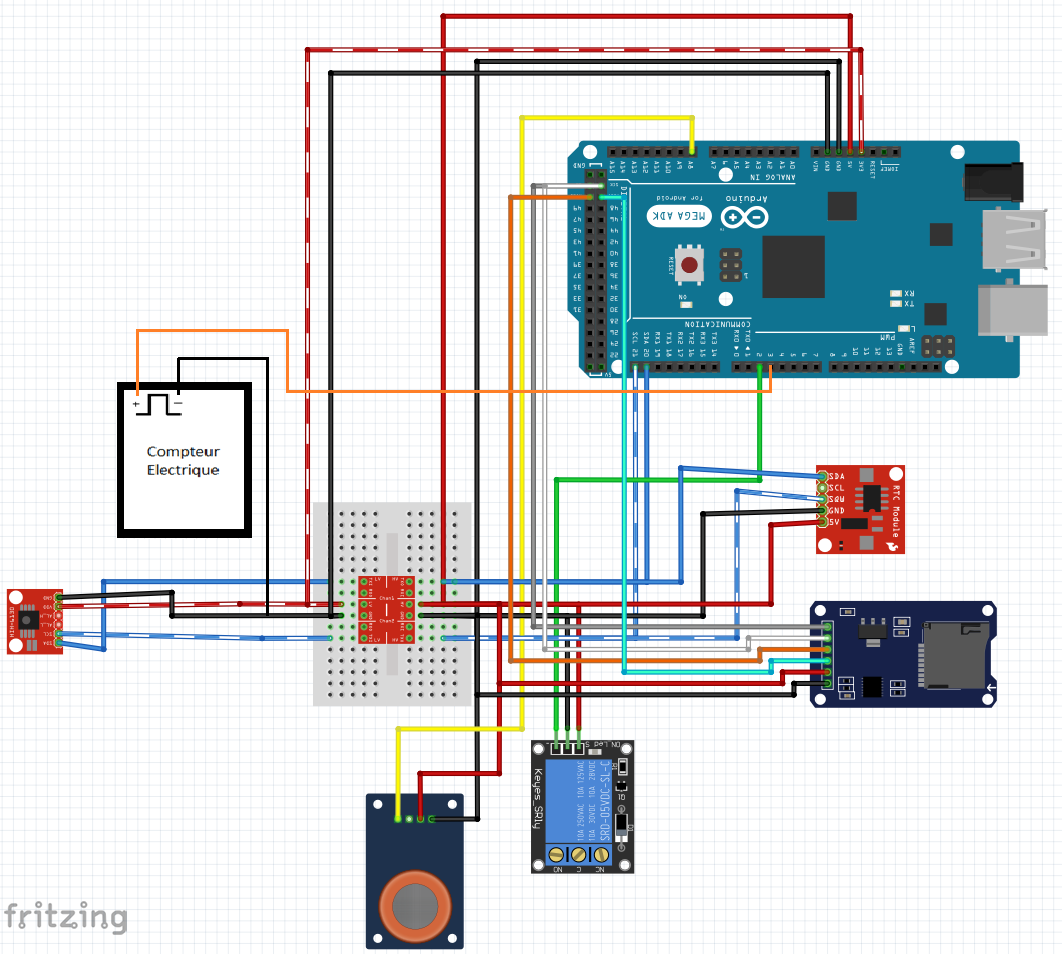
## capteur.cpp du regroupement



## capteur.h du regroupement

Ici nous avons regroupé les capteurs, et actionneurs de la partie confort. On y récupère la température(°C), le taux d’humidité (%), la qualité de l'aire et on contrôle le relai qui agira sur le radiateur.

# REGROUPEMENT FINAL DES COMPOSANTS :

 Une fois la première partie de regroupement réalisée, nous pouvons à présent regrouper la totalité de cette partie en un seul montage et un seul programme. Nous pourrons ensuite regroupée les codes de la partie de l'étudiant 1 avec notre partie.

## Montage du système

## QUALITE D'AIR

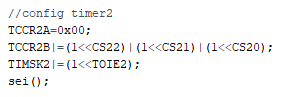
## RELAIS

## RTC

Lors du branchement du module SD nous avons changer de **pin** pour la communication SPI car elle varie en fonction de la carte que nous utilisons.

**Arduino UNO** : MOSI = 11 / MISO = 12 / SCK = 13 / SS = 10

**Arduino MEGA** : MOSI = 51 / MISO = 50 / SCK = 52 / SS = 53



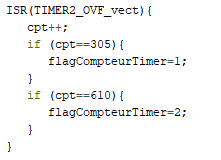
Ici nous paramétrons un timer qui nous permettra de récupérer les valeurs et états de nos capteurs et actionneurs.

**TCCR2A** nous indique que le timer est en mode normale

**TCCR2B** est le coefficient diviseur de la fréquence du CPU

**TIMSK2** nous indique pour que le timer entre en interruption à chaque débordement (Overflow)

**sei() :** Active les interruption.

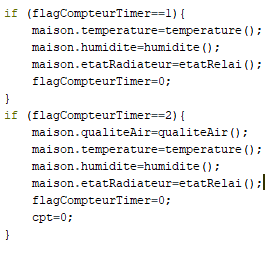
****

Ici notre timer entre en interruption 61 fois

par seconde. Se qui correspond à la lever des drapeaux.

Le premier drapeau se lève toute les 305 débordement

du timer donc toute les 5 sec. Quant au deuxième

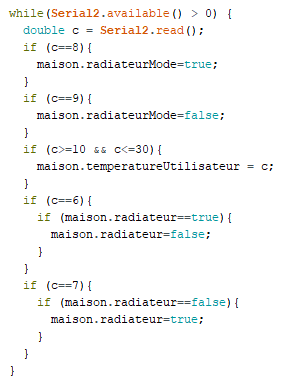
drapeau il se lève toute les 10 seconde.

Nous voyons maintenant que les

données sont stockées dans l’objet maison

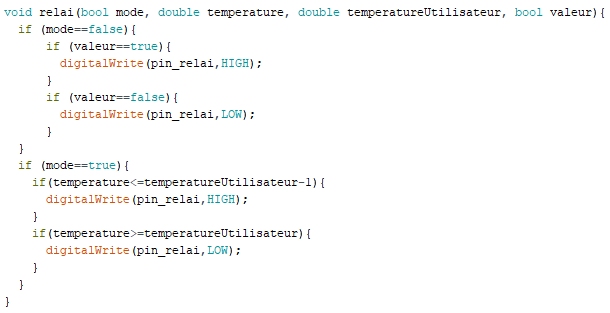
toutes les 5 et 10 secondes.

Les drapeaux sont remis a zéro ainsi que le compteur, à la fin du programme, lancer par le deuxième drapeau.

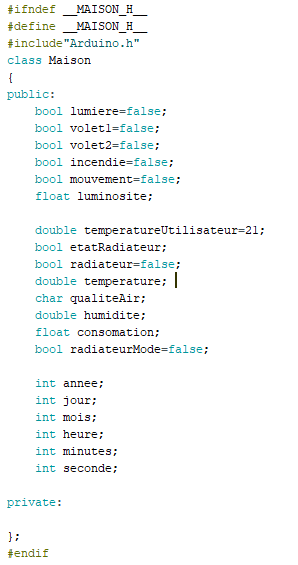


Ce morceau du code nous permet de récupérer les données envoyer par la tablette. C’est donné sont des valeurs qui corresponde à une requête soumise par l’utilisateur. Par exemple si la valeur envoyée est 8 nous passons en mode manuelle, si la valeur est 9 nous passons en mode automatique. De cette manière nous pouvons contrôler le mode de fonctionnement du radiateur.

Maison.temperatureUtilisateur prend la valeur choisie par l’utilisateur pour sont chauffage. Ainsi lorsque le chauffage sera en mode automatique le chauffage restera à la température souhaitée.



Ici nous avons le code qui permet de contrôler le chauffage en automatique et en manuelle. On peut remarquer qu’il y a une plage d’hystérésis de 1 degrés qui nous permet de ne pas casser notre relai.

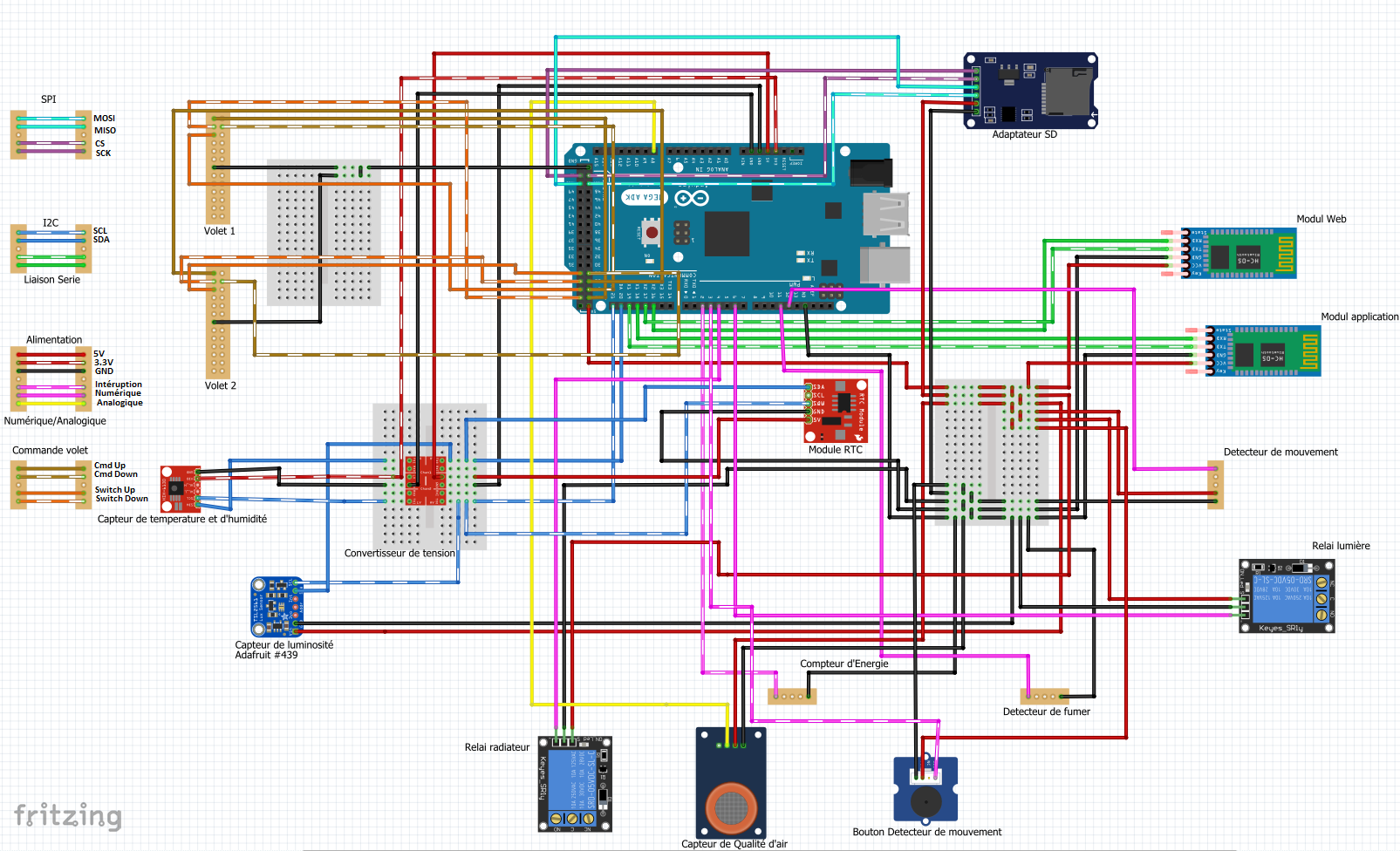


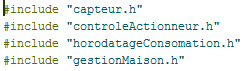
Voici la première version de l’objets maison. Il sera amélioré par la suite avec l’arriver du code de l’étudiant 1.

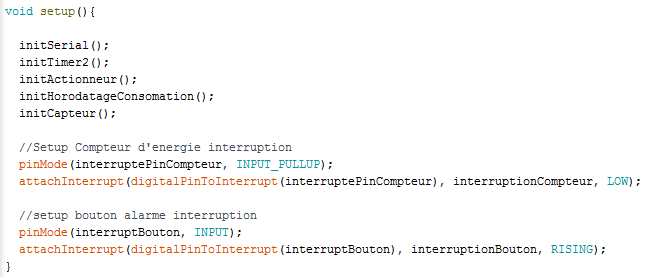
Dans cet objet nous avons toutes les variables liées à un capteur ou à un actionneur (sont état, les valeurs qu’il renvoie, son mode de fonctionnement, etc.).

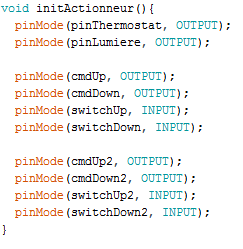
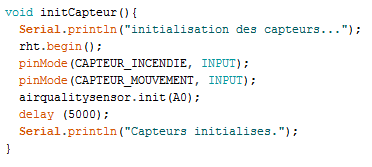
L’heure et la date son aussi sauvegarder séparément pour pouvoir par la suite les envoyer un par un à la tablette ou au serveur Web.

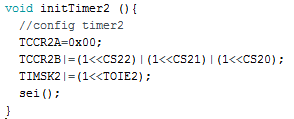
4 - REGROUPEMENT DES DIFFÉRENTES PARTIE.

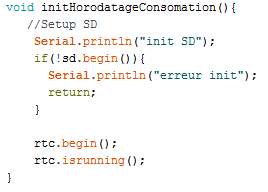


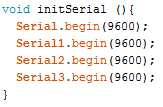
 Dans cette partie nous verrons le travail que réalisé lors du regroupement de toutes nos partie. Dans une premier temps nous avons réorganiser le code, nous avons créé de nouvelles bibliothèques plus intuitives et plus claires.

C'est nouvelle bibliothèque on leurs propre fonction initialisation ceux qui rend le void setup() beaucoup plus intuitif.







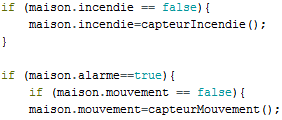
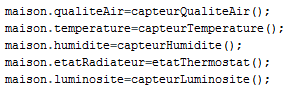


# LA BIBLIOTHÈQUE CAPTEUR.H :

Dans cette bibliothèque nous gérons tout les capteurs ( humidité, luminosité, température, qualité d'aire, mouvement et incendie), ainsi que l'envoie de SMS créé par l'étudiant 1 .



Chaque capteur renvoie une valeur qui est récupérer dans l'objet maison.



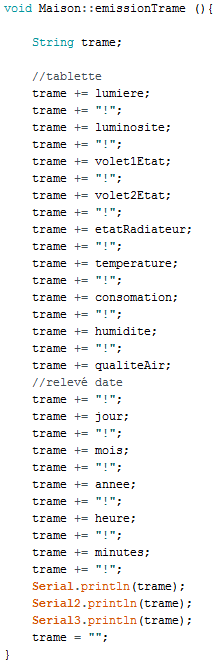
# LA BIBLIOTHÈQUE GESTIONMAISON.H :



Voici l'objet maison finalisé, on y stocke toute valeur utilisable ainsi que deux méthodes dont nous allons nous servir pour communiquer avec la tablette et pour envoyer nos donnés au serveur web.

Ceci est une première ébauche de notre objet, il est nettement améliorable, surtout au niveau des règles d'encapsulation.

# FONCTION D'ENVOIE DE LA TRAME :



Dans cette fonction nous ajoutons les valeurs récupérer des différent capteur les une après les autres en les séparant par un point d'exclamation.

Cette opération nous génère une trame que nous allons envoyer.

trameC.png

État de la lumière 0 = éteint et 1 = allumé

Luminosité en LUX

État du volet n°1 0 = fermer et 1 = ouvert

État du volet n°1 0 = fermer et 1 = ouvert

État du radiateur 0 = éteint et 1 = allumé

Température en °C

Consommation en KwH

Taux d'humidité en %

Qualité de l'aire de 0 à 3

Heure et date : jour ! moi ! année ! heure ! minutes !

1. **horodater :** Ajouter de manière automatique la date et l'heure à un document. [↑](#footnote-ref-2)
2. **GITHUB :** Service web d'hébergement et de gestion de développement de logiciels. [↑](#footnote-ref-3)