

Lycée Raymond Queneau, BTS SN EC.

Compte rendu de groupe

Projet final de BTS

Chauveau Aurélien, Dufresne Nicolas, Girard Tommy,
Guyader Benjamin.



18

Sommaire :

Cahier des charges :	3
Présentation du projet :	3
Expression du besoin :	3
Architecture Matériel :	5
Analyse fonctionnel :	6
Diagramme de cas d'utilisation :	6
Diagramme d'exigence :	7
Diagramme de séquence application :	8
Diagramme de séquence web :	9
Diagramme de séquence sécurité :	10
Organisation de groupe :	11
Matériels :	12
Matériel de l'étudiant 1 :	12
Matériel de l'étudiant 2 :	13
Matériel de l'étudiant 3 :	14
Matériel de l'étudiant 4 :	14
Protocole :	15
Les protocoles d'envoie :	15
Les protocoles de trame :	15
Les protocoles de codage :	15
Journée poster :	16

Cahier des charges :

Présentation du projet :

Ce projet est composé de 4 étudiants et a pour objectif de présenter un prototype de maison connectée qui permettra une interaction sur les quatre aspects suivants:

- L'aspect confort avec une commande des lumières et du chauffage (avec différents capteurs et actionneurs) et une mesure de la qualité de l'air.
- L'aspect énergie avec une mesure en temps réel de la consommation.
- L'aspect sécurité avec la détection incendie et une détection de mouvement.
- L'aspect commande des ouvrants (pour gérer l'ensemble des volets par exemple).

Ce système permettra donc de suivre en direct via un accès internet l'état de la maison (celle-ci disposera donc d'un serveur Web qui regroupera l'ensemble des paramètres de mesure). De plus en cas d'intrusion ou d'incendie, un SMS sera envoyé au propriétaire. Le système sera également accessible en local grâce à une application disponible sur Smartphone et tablette (équipés d'Android) et une connexion Bluetooth.



Expression du besoin :

Les étudiants réaliseront un prototype permettant de répondre au cahier des charges suivant :

- ❖ La partie du système qui gèrera l'aspect confort devra:
 - prendre la mesure de la qualité de l'air.
 - prendre les mesures de températures et d'humidités au sein de la maison.
 - rendre accessible l'ensemble de ces mesures à la fois sur le site web et sur l'application mobile.
 - piloter de façon automatique le chauffage en fonction d'une valeur prédéfinie de la température intérieure (il sera par la suite possible de régler cette température via l'application mobile).
 - permettre via l'application mobile d'allumer ou d'éteindre le chauffage et de commander les lumières.
- ❖ La partie du système qui gèrera l'aspect énergie devra:
 - prendre la mesure de la consommation électrique via un compteur d'énergie.
 - permettre l'horodatage de cette mesure.
 - sauvegarder cette mesure et l'horodatage sur un support amovible de type SDcard afin de conserver la donnée en cas de coupure de courant ou d'arrêt du système d'acquisition.
 - rendre accessible l'ensemble de ces mesures à la fois sur le site web et sur l'application mobile.

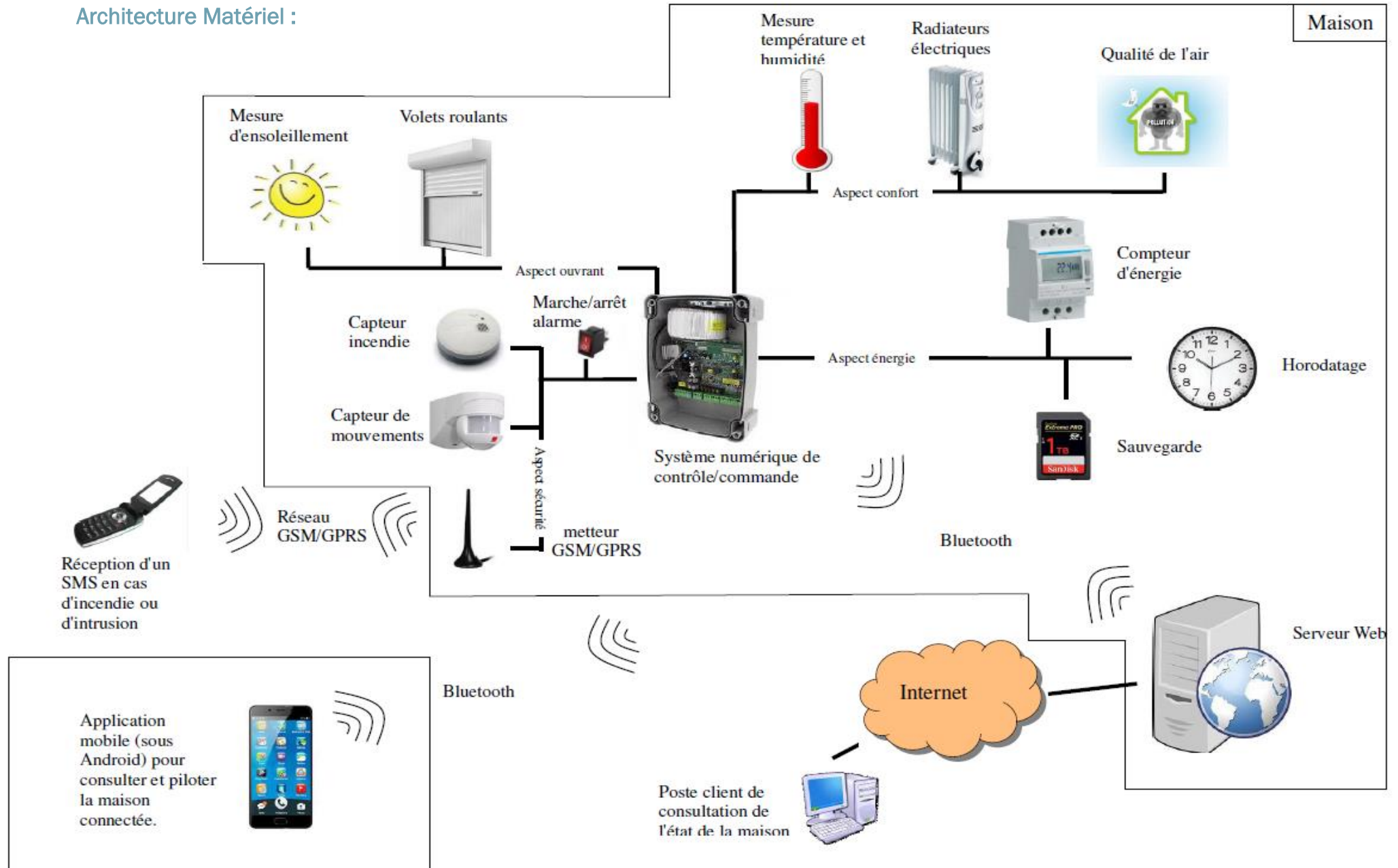
- ❖ La partie du système qui gèrera l'aspect sécurité devra:
 - grâce à un interrupteur, permettre de mettre en service ou non le système d'intrusion.
 - détecter un incendie.
 - en cas d'intrusion, ou d'incendie, envoyer un SMS avec la nature de l'incident à un numéro prédéfini.
- ❖ La partie du système qui gèrera l'aspect commande des ouvrants devra:
 - permettre d'ouvrir ou de fermer un ou plusieurs volets électriques grâce à l'application mobile.
 - fournir l'état des ouvrants à l'application mobile et au site web.
 - prendre la mesure de luminosité extérieure et piloter de façon automatique l'ouverture ou la fermeture de volets.



Le système sera donc composé de trois grandes parties :

- ❖ Une "centrale de gestion" qui regroupera l'ensemble des capteurs et actionneurs présents dans la maison ainsi que le système numérique de contrôle. Elle possèdera également une unité de stockage de type SDcard et un module GSM/GPRS pour l'envoi de SMS. Cette centrale communiquera dans un premier temps avec l'application mobile en Bluetooth. Dans un deuxième temps et de façon similaire, elle communiquera également en Bluetooth avec un deuxième système présent dans la maison qui fera office de "serveur web".
- ❖ Le "serveur web" présent également dans la maison et qui recueillera les informations (état des volets, des radiateurs, température, humidité, qualité de l'air, ensoleillement, etc.) issues de la "centrale de gestion" de façon cyclique (toutes les 3 secondes) afin de les rendre accessibles au monde extérieur via une application web. Le site web permettra donc uniquement une consultation des différents capteurs présents dans la maison. Pour des raisons de sécurité, il ne permettra pas d'agir à distance sur la maison connectée.
- ❖ Une "application mobile" communiquant en Bluetooth avec la centrale de gestion et permettant de recueillir de façon cyclique (toutes les 3 secondes) l'ensemble des informations issues de la "centrale de gestion". Cette application permettra en local (distance de quelques dizaines de mètres par rapport à la centrale de gestion) de piloter une partie de l'installation (demande d'allumage des lumières, commande des volets, mise en service du chauffage, etc.).

Architecture Matériel :



Analyse fonctionnel :

Digramme de cas d'utilisation :

Un cas d'utilisation représente un ensemble d'actions qui sont réalisées par le système et qui produisent un résultat observable pour un acteur particulier.

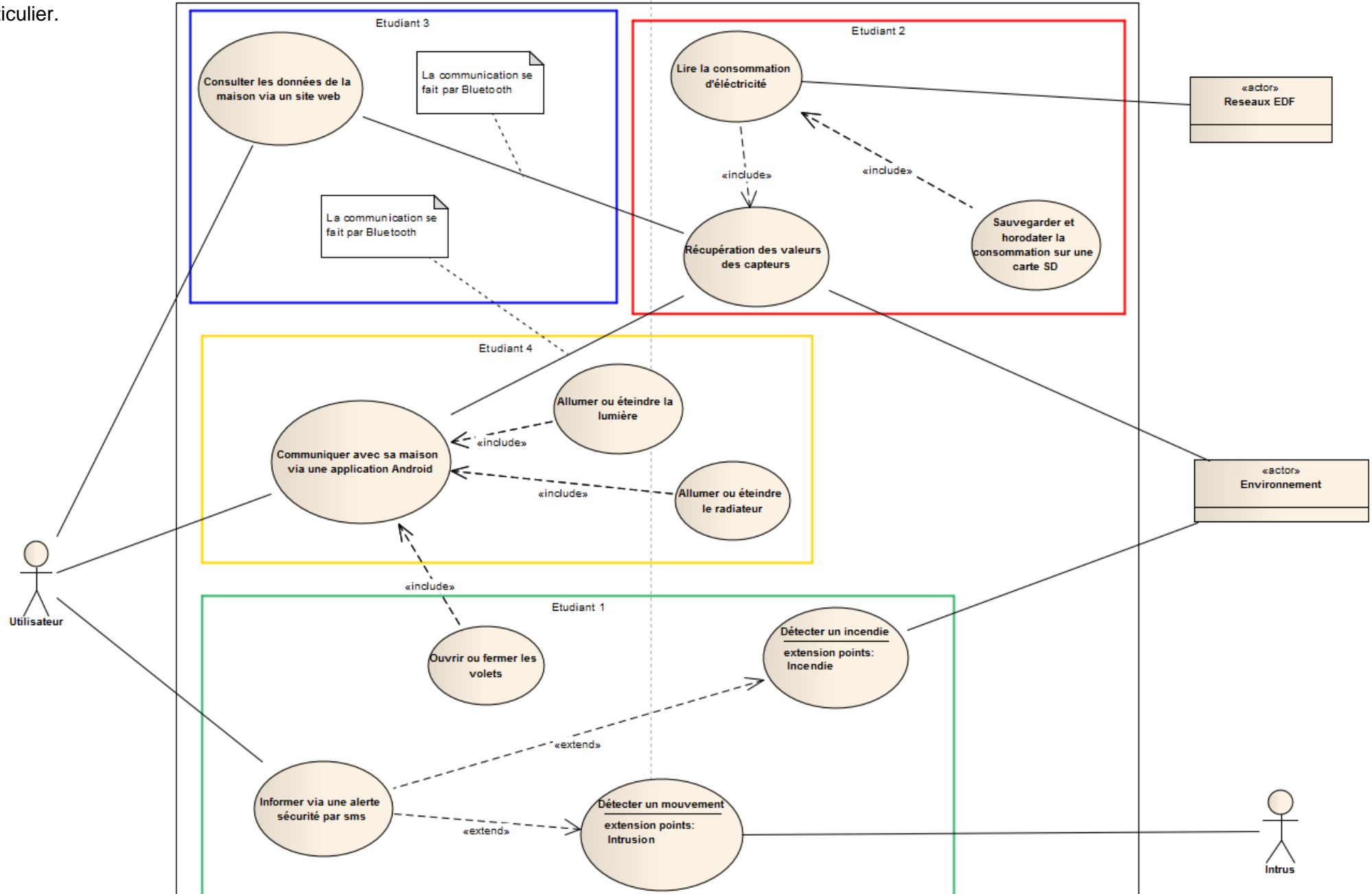


Diagramme d'exigence :

Ce diagramme permet de clarifier les attentes liées au futur système entre le client et le concepteur.

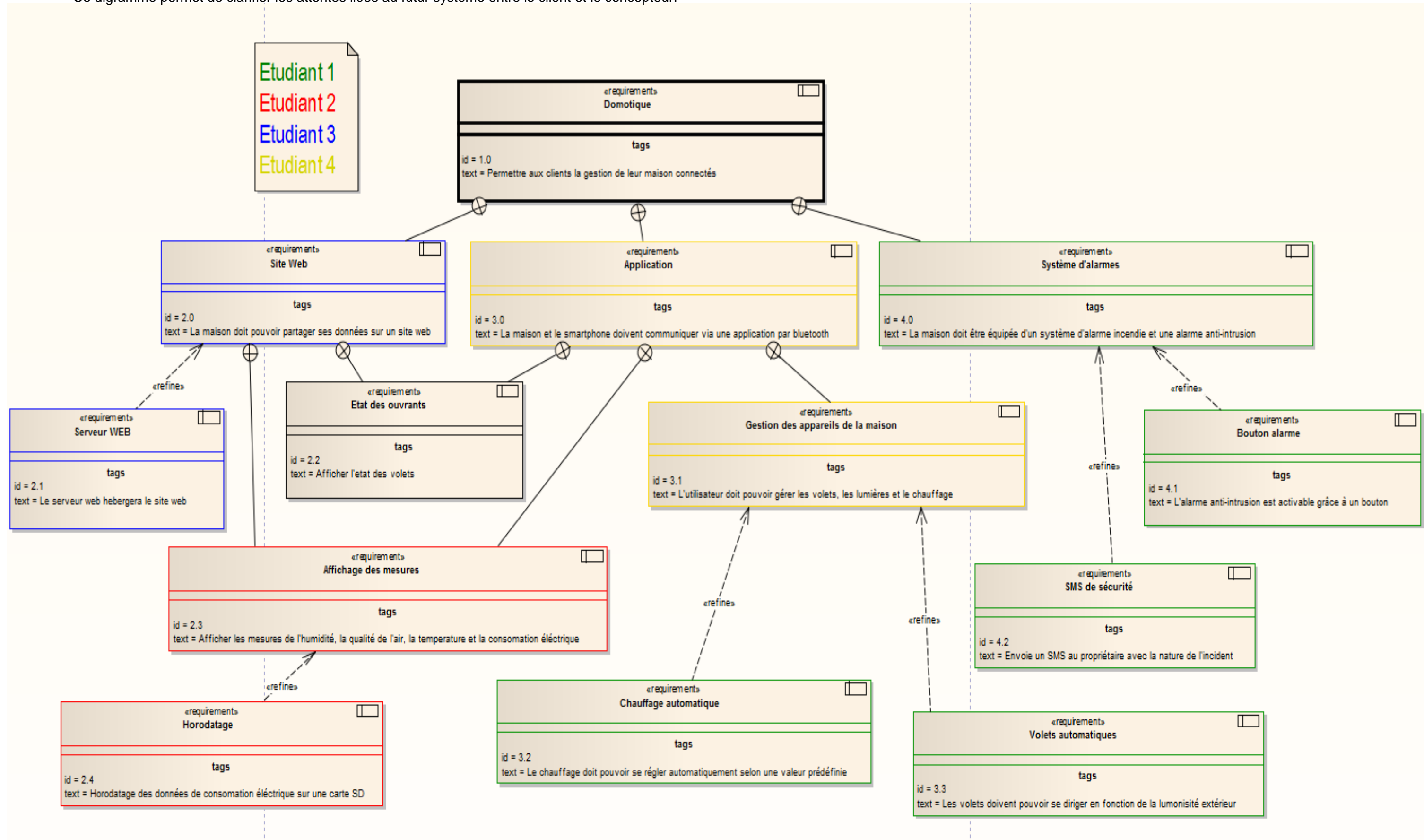


Diagramme de séquence application :

Le digramme de séquence représente les échanges entre les acteurs et le système, et permet également de détailler les échanges d'informations entre différentes parties internes au système.

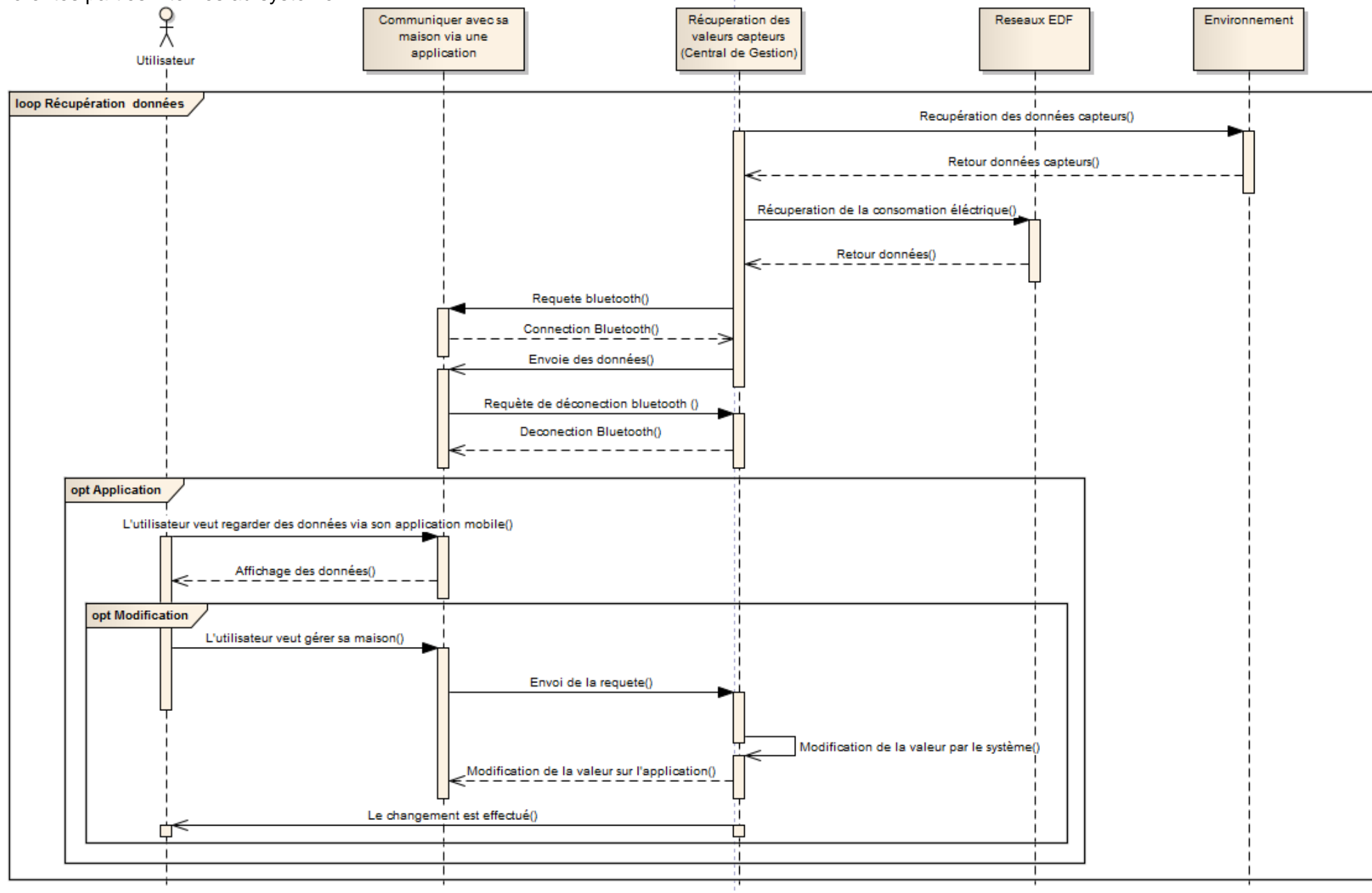


Diagramme de séquence web :

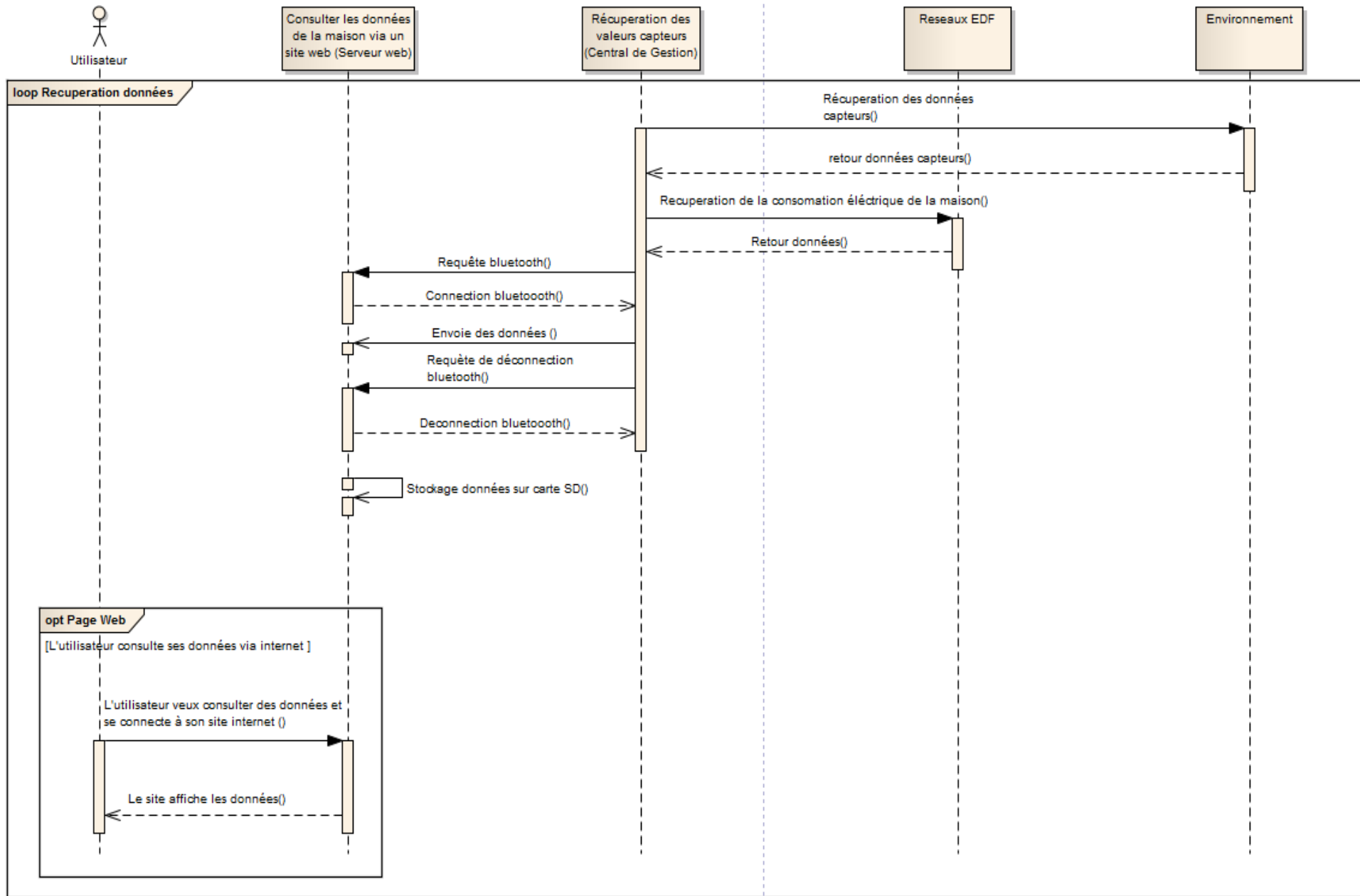
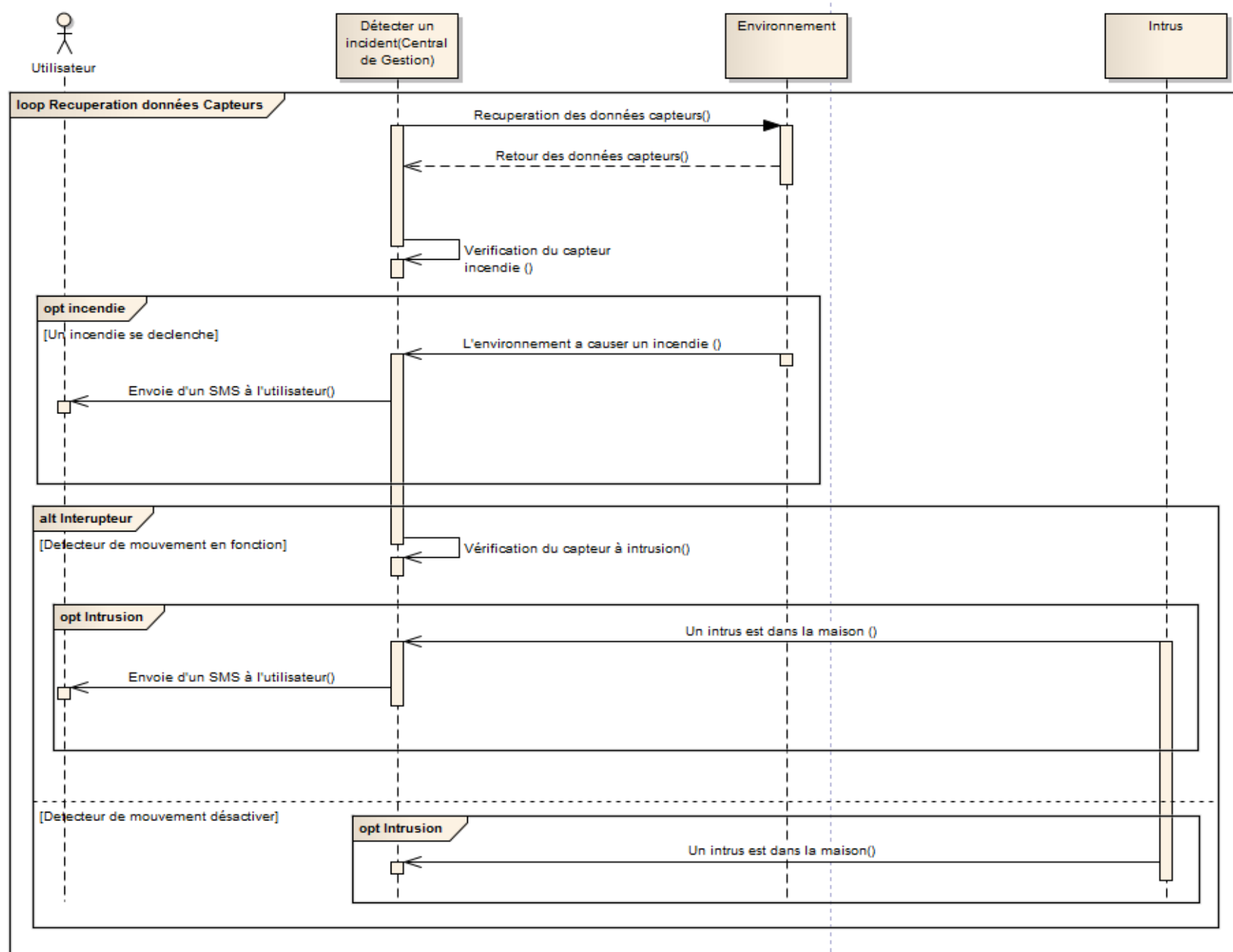
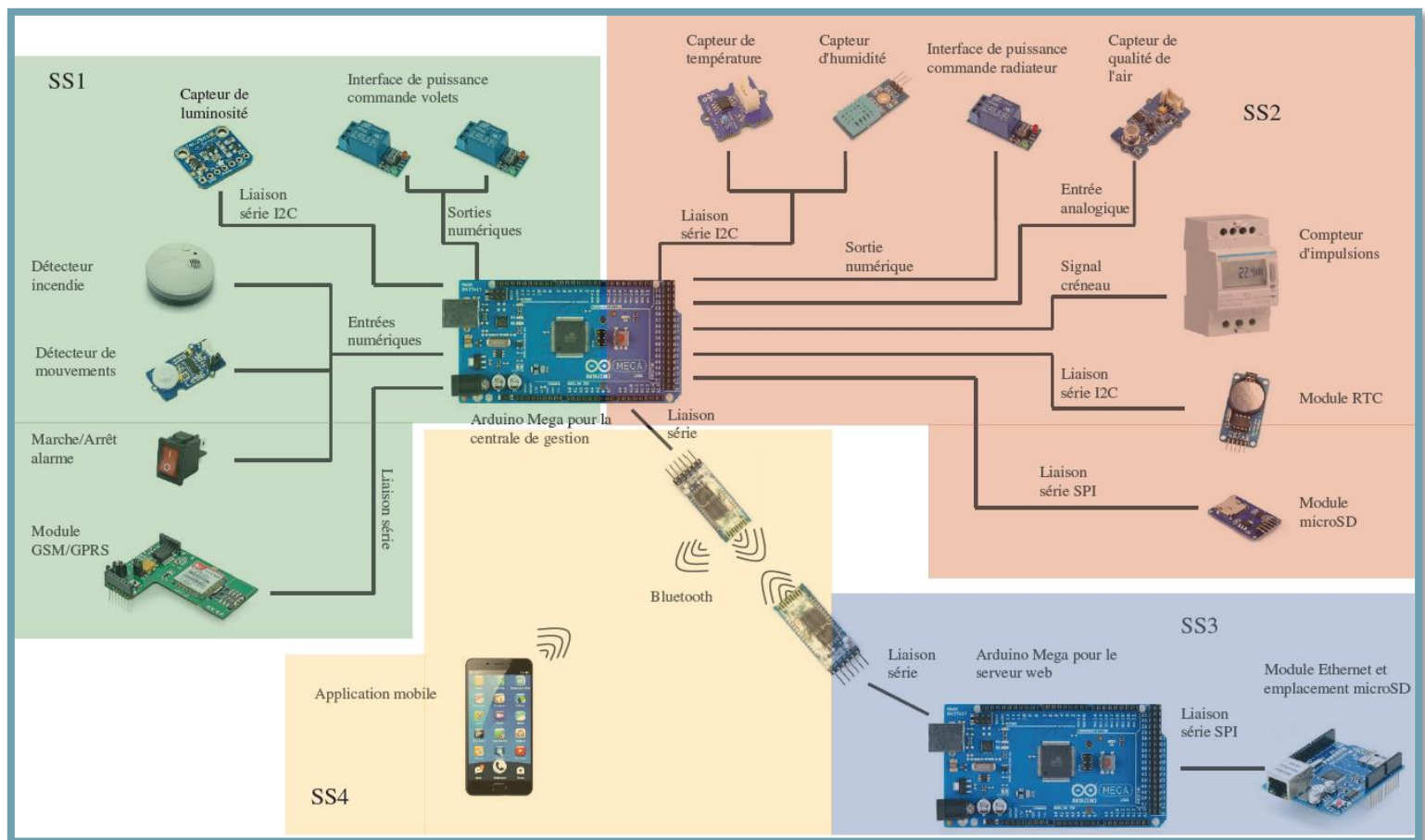


Diagramme de séquence sécurité :



Organisation de groupe :



Le groupe est organisé de la façons suivante :

L'étudiant 1 sera chargé de la gestion des lumières et des volets ainsi que de la sécurité. Cette dernière est équipé d'un détecteur de mouvement, d'un détecteur de fumée et d'un module GSM qui devra envoyer un SMS à l'utilisateur en cas de problème.

L'étudiant 2 sera chargé de la gestion du chauffage, de la gestion de la consommation électrique et de la récupération des informations envoyées par les capteurs. Ses informations devront être envoyées au serveur web et à l'application mobile.

L'étudiant 3 sera chargé de créer un serveur web afin d'y afficher des valeurs utiles à l'utilisateur.

L'étudiant 4 sera chargé de créer une application mobile qui permettra de contrôler les actionneurs ainsi que d'afficher des valeurs utiles à l'utilisateur.

L'étudiant 1 et **L'étudiant 2** devront regrouper leurs codes et ainsi se mettre d'accord pour créer un objet maison dans lequel sera stocké les attributs et les méthodes nécessaires au projet.

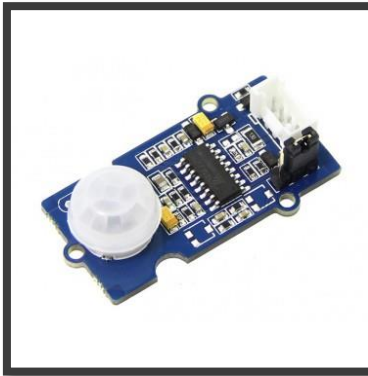
L'étudiant 3 et **L'étudiant 4** devront travailler ensemble pour mettre en marche la communication bluetooth entre l'application mobile et la centrale de gestion ainsi qu'entre le serveur web et la centrale de gestion.

Matériels :

Matériel de l'étudiant 1 :



Détecteur de fumer



Détecteur de mouvement



Relai de contrôle de la lumière



Module GSM



Capteur de lumière



Volet roulant



Ampoule



Bouton d'activation du détecteur de mouvement



Téléphone



Carte arduino



Logiciel arduino

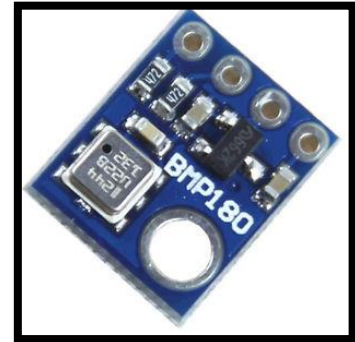
Matériel de l'étudiant 2 :



Capteur de qualité d'air



Capteur d'humidité



Capteur de température



Radiateur



Relai de contrôle de la lumière



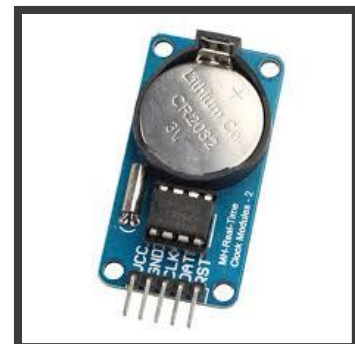
Compteur électrique



Adaptateur SD



Carte SD



Module RTC

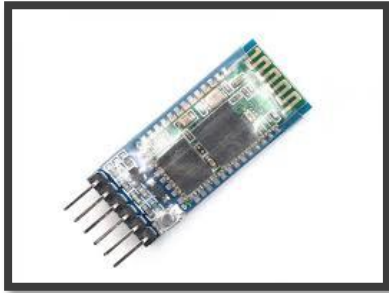


Carte arduino



Logiciel arduino

Matériel de l'étudiant 3 :



Module bluetooth x2



Shield ethernet

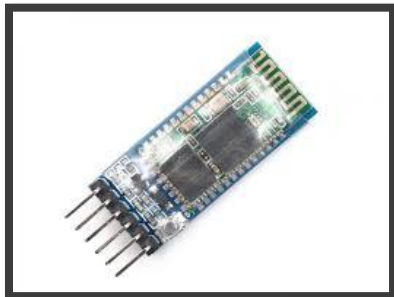


Logiciel arduino



Carte arduino

Matériel de l'étudiant 4 :



Module bluetooth



Tablette



Logiciel arduino



Carte arduino



Mit app inventor 2

Protocole :

Les protocoles d'envoi :

Pour définir les protocoles d'envoies nous nous sommes basé sur le cahier des charges. Ce dernier nous indique que plusieurs valeurs doivent être envoyées à intervalles régulier, les premières valeurs doivent être envoyées toutes les 5 sec et les autres valeurs toutes les 10 sec. Nous envoyons donc une trame toutes les 5 sec avec toutes les valeurs nécessaire actualisé puis toute les 10 sec nous envoyons une trame avec la totalité des valeurs actualisées.

-réaliser la mesure de la température et de l'humidité avec une périodicité de 5 secondes.
-réaliser la mesure de la qualité de l'air avec une périodicité de 10 secondes.

EXTRAIT DU CAHIER DES CHARGES

Les protocoles de trame :

La trame contiendra 14 valeurs séparées de cette manière :

Etat de la lumière / Luminosité / Etat du volet 1 / Etat du volet 2 / Etat du radiateur /
Température / Consommation en Kwh / Taux d'humidité en % / Qualité de l'air/ Jour / Mois /
Année / Heure / Minutes.

Chaque valeur sera séparée par un point d'exclamation. La trame commencera par la première valeur et se terminera par un point d'exclamation :

0!0!0!0!0!0!0!0!0!0!0!0!0!

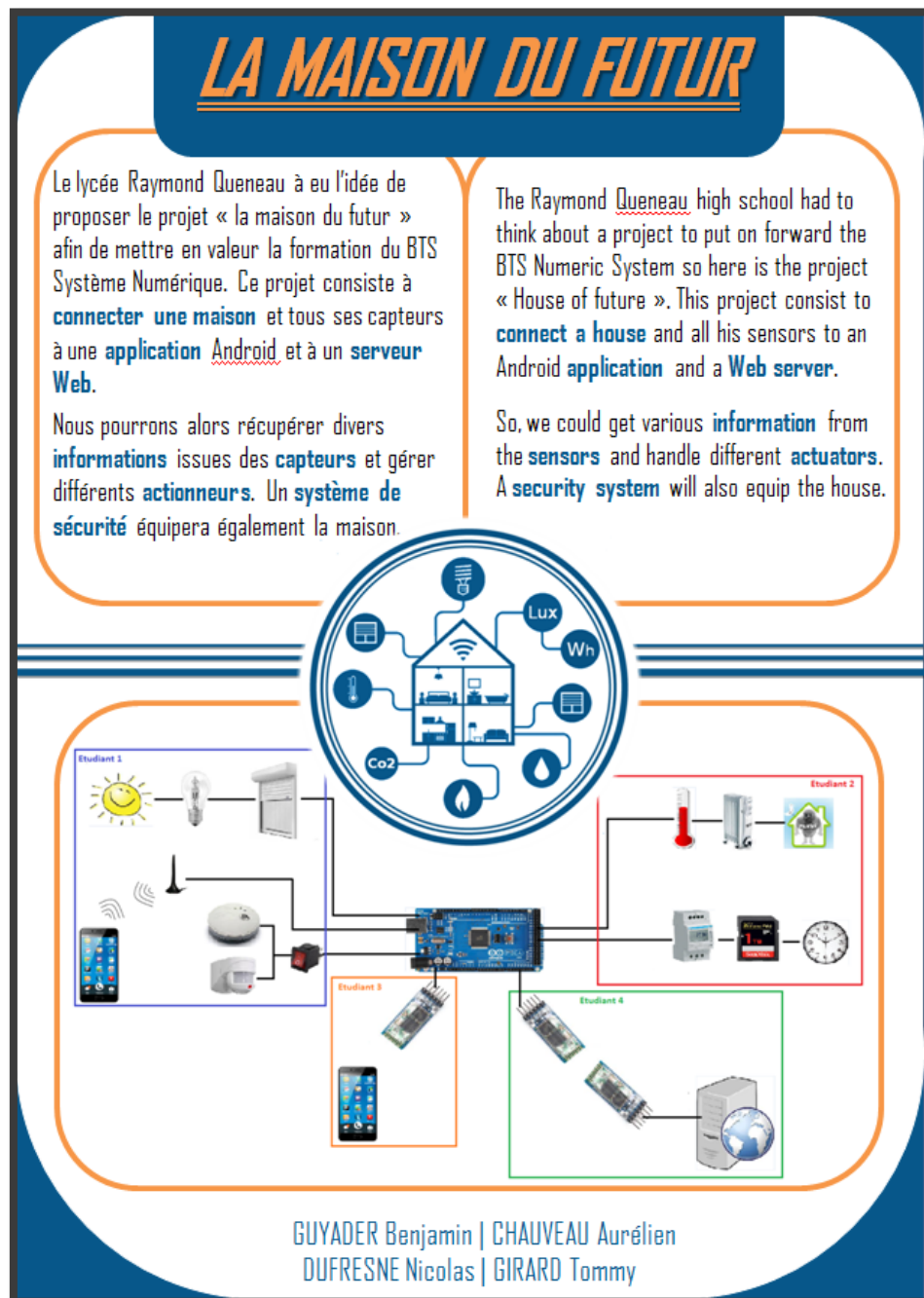
Les protocoles de codage :

Pour pouvoir réunir nos codes de manière efficace nous avons prévue a l'avance que nous allons exécuter toutes nos fonctions dans des mini programme que nous appellerons depuis une bibliothèque commune.

Nous nous somme donc mis aussi d'accord à sur le nom de nos bibliothèque final afin d'y regrouper nos code de manière logique.

Journée poster :

La journée poster est un événement organisée par le lycée Raymond Queneau. Il consiste ,pour les élèves de STI2D et de BTS, à présenter le projet qu'ils ont réalisés tout au long de l'année à des chefs d'entreprises et à des professionnels dans le domaine de la technologie. Pour cela ils doivent réaliser un poster de grande taille sur lequel ils présentent à l'écrit en français et en anglais leur projet.



Poster du projet : Maison du futur

C'est une expérience qui nous permet de nous améliorer à l'oral et de mieux préparer notre soutenance de fin d'année.

Rapport

Projet de BTS

Guyader Benjamin

18

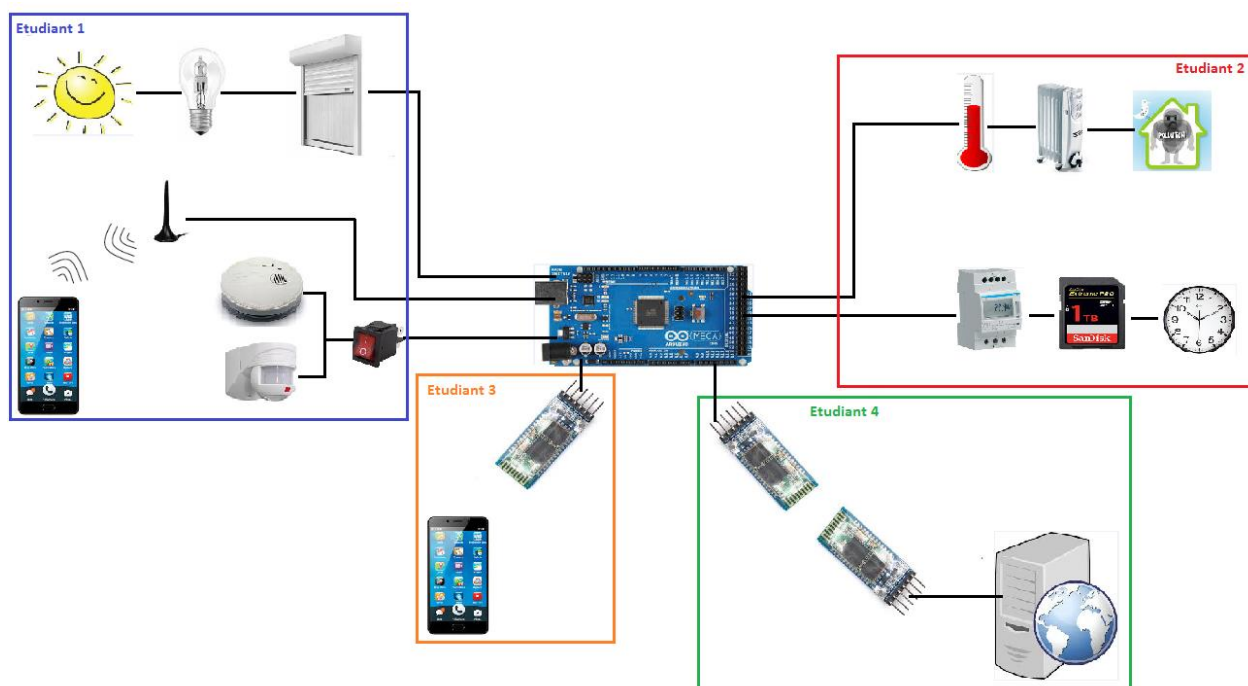
Sommaire

Présentation de ma partie	3
Test unitaire : Volet roulant	4
Test unitaire : capteur de luminosité	7
Test d'intégration : Volet roulant / capteur de luminosité	8
Test unitaire : Lumière	10
Test unitaire : Détecteur de mouvement	12
Test d'intégration : Interrupteur / détecteur de mouvement	14
Test unitaire : Détecteur d'incendie	17
Test unitaire : Module GSM	18
Test d'intégration : Module GSM / Interrupteur / détecteur de mouvement / Détecteur d'incendie	20
Test d'intégration final : Module GSM / Interrupteur / détecteur de mouvement / Détecteur d'incendie / Lumière / Volet roulant / capteur de luminosité	22
Conclusion :	23
Les évolutions possibles :	23
Annexe :	24
Programme principal :	24
capteur.h	27
capteur.cpp	28
controleActionneur.h	31
controleActionneur.cpp	32
horodatageConsomation.h	35
horodatageConsomation.cpp	36
gestionMaison.h	38
gestionMaison.cpp	39
Câblage du système :	42

Présentation de ma partie

Dans le projet "La maison du futur", nous pouvons relever trois différentes parties

- Le serveur web (étudiant 4)
- l'application mobile (étudiant 3)
- Les capteurs et actionneurs (étudiants 1 et 2) :

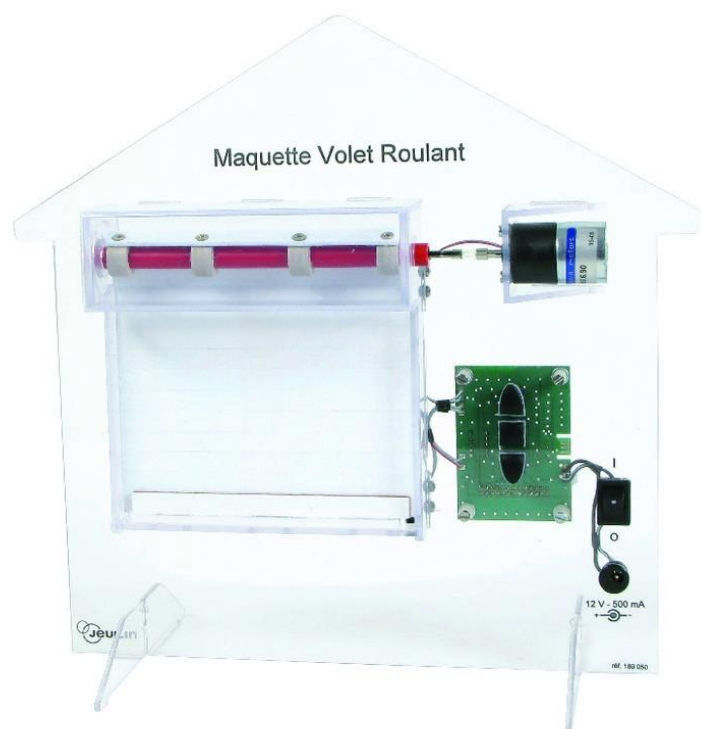


En tant qu'étudiant 1, mon travail se séparait en deux parties :

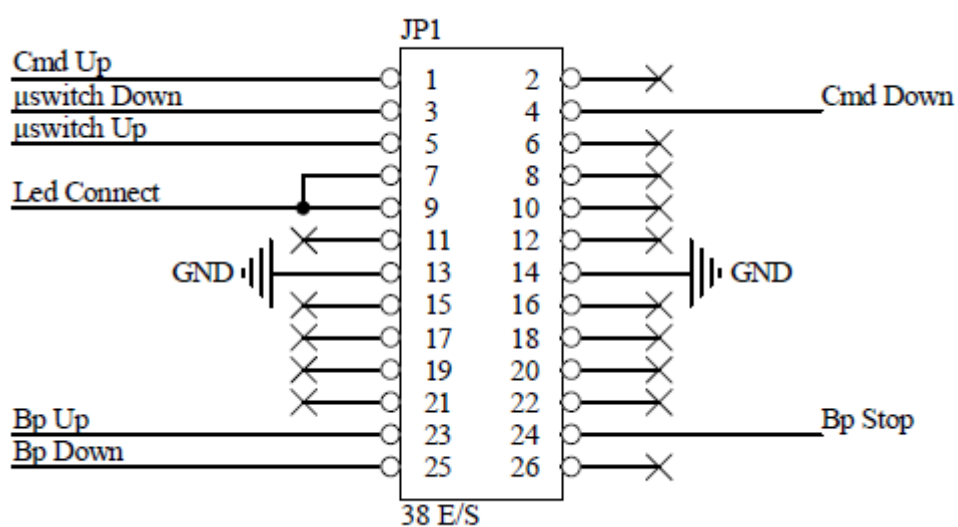
- Pilotage des volets ainsi que de la lumière :
 - Volets Roulants / Capteur de luminosité
 - Ampoule
- Mise en place de la sécurité :
 - Détecteur d'incendie
 - Détecteur de mouvement / Interrupteur
 - Module GSM

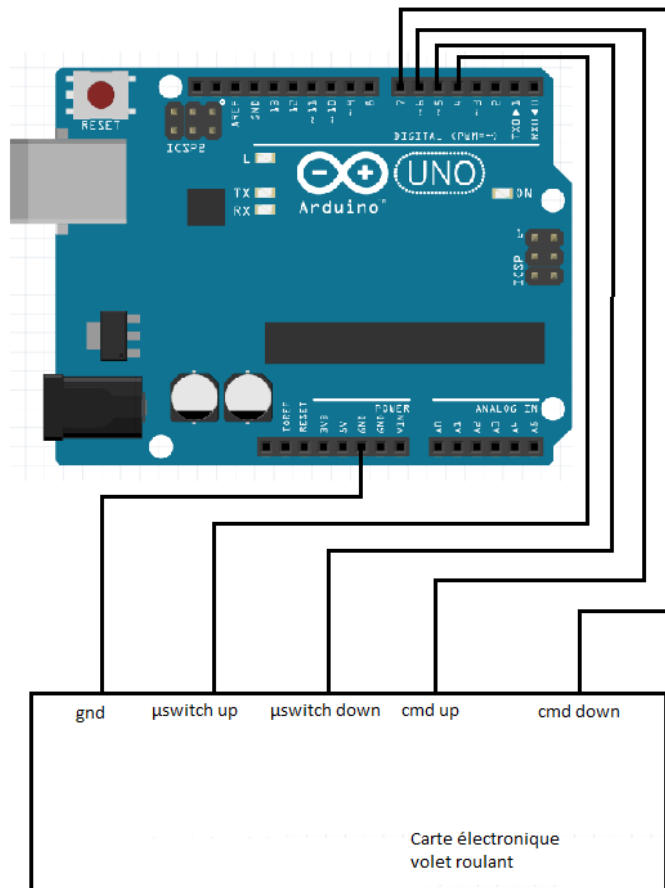
Test unitaire : Volet roulant

De plus en plus de maisons sont équipées de volets roulants, c'est le cas de notre maquette. Il est possible de les piloter via la tablette ou grâce aux boutons présents sur la carte.



Dans un premier temps à l'aide d'un schéma de la carte électronique du volet roulant j'ai recherché les broches utiles pour son usage dans notre projet :





Sur ce schéma j'ai fait apparaître uniquement les broches de la carte que j'utilise dans le projet.

Ces broches sont les suivantes :

- cmdUP : Broche qui permet la montée des volets lorsqu'elle est à un niveau logique HAUT (5V).
- CmdDown : Broche qui permet la descente des volets lorsqu'elle est à un niveau logique HAUT (5V).
- μswitchUP : Cette broche a un niveau BAS lorsque le volet arrive en fin de course vers le haut.
- μswitchDown : Cette broche passe à un niveau BAS lorsque le volet arrive en fin de course vers le bas.
- GND

Pour comprendre le fonctionnement des broches μswitchUP et μswitchDown, j'ai utilisé un voltmètre pour déterminer leurs états en fonction de la position du volet pour pouvoir adapter mon code :

```
while(digitalRead(switchUp)==HIGH){ //tant qu'il n'est pas en fin de course
    digitalWrite(cmdUp,HIGH);        //le volet monte
}
digitalWrite(cmdUp,LOW);              //On arrête la montée
```

J'ai relié toutes ces broches sur celles numériques d'une carte Arduino uno.

Pour vérifier le bon fonctionnement de ces volets, j'ai fait un programme qui permet que, lorsque l'on envoie "1" sur le moniteur série Arduino, le volet s'ouvre et, lorsque l'on envoie "2", le volet se ferme.

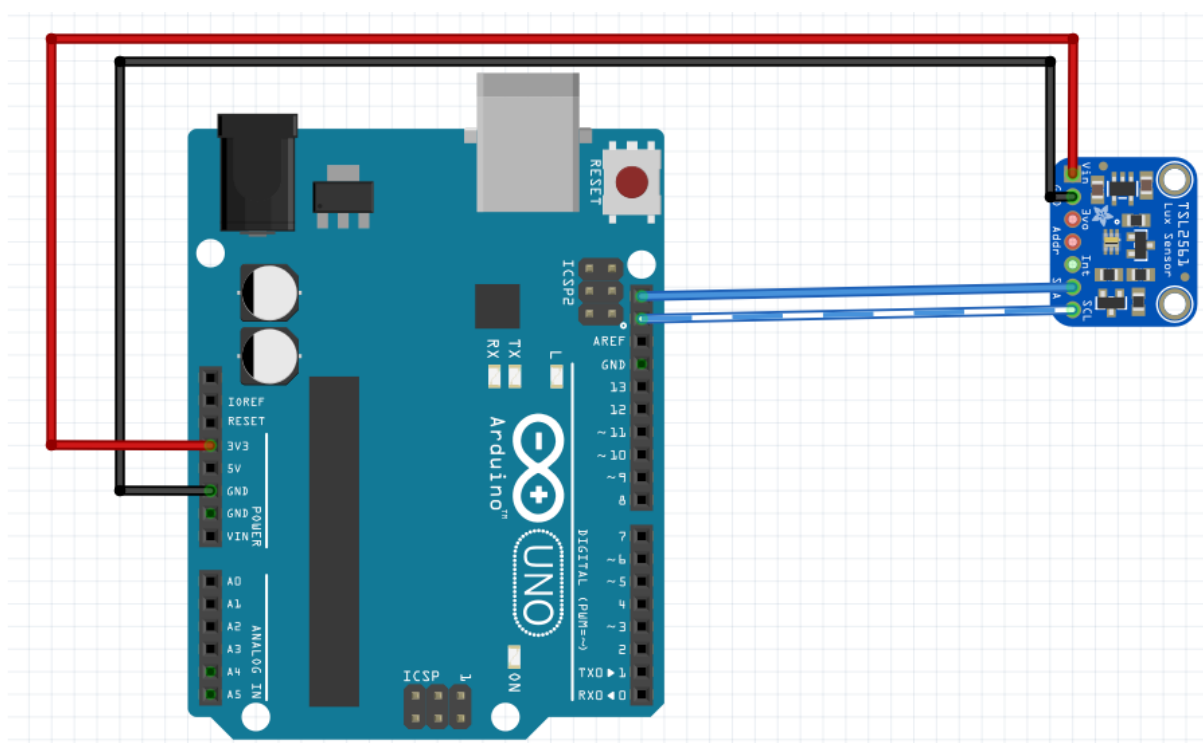
Ce test sur les volets roulants m'a permis de vérifier leurs bons états de marche mais également de comprendre le fonctionnement de la carte électronique.

Test unitaire : capteur de luminosité

Pour faciliter la vie de l'utilisateur, un mode automatique est ajouté aux volets roulants. Ces derniers seront influencés par un capteur de luminosité.



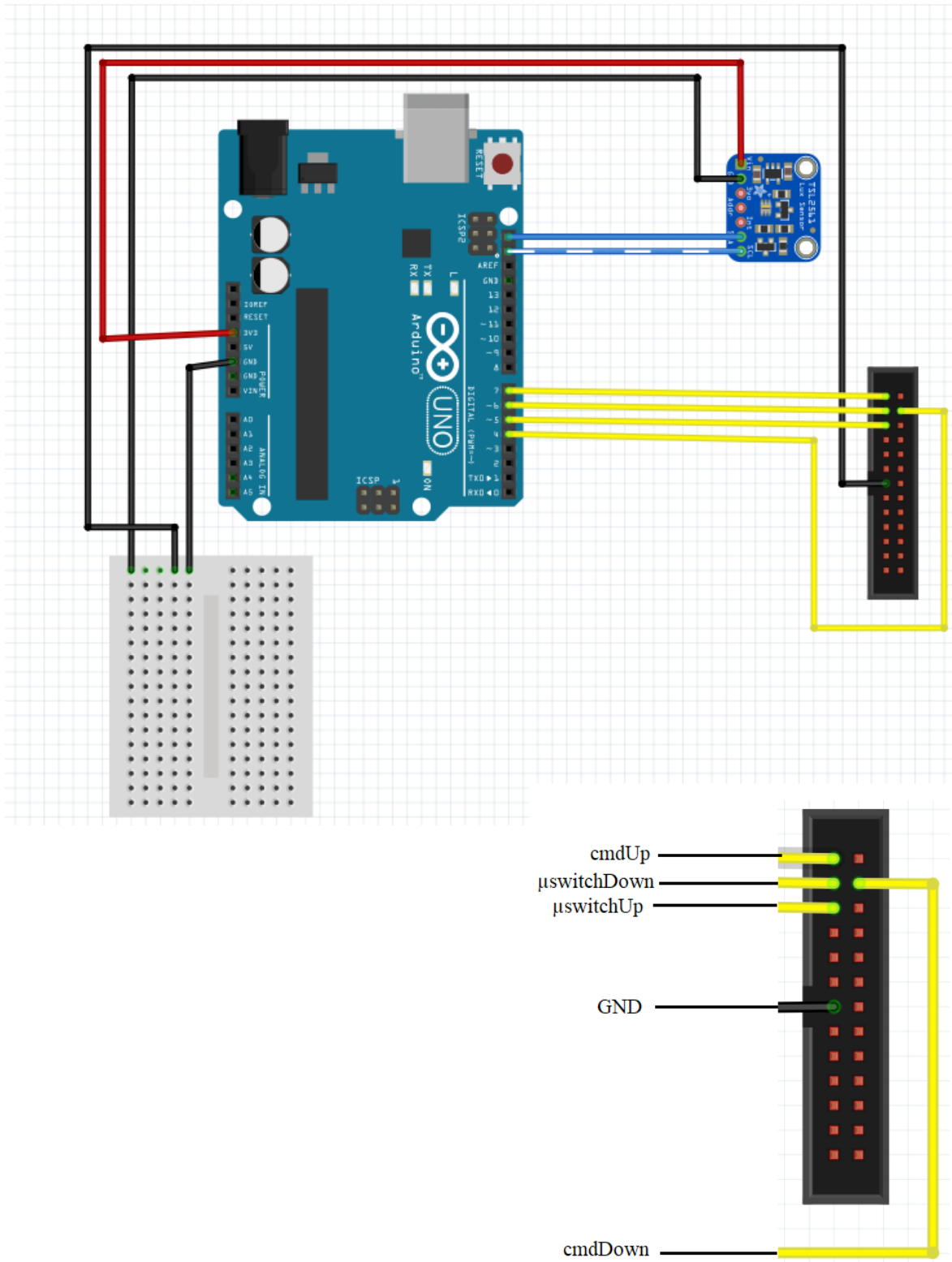
Le capteur TSL2561 a une plage de mesure allant de 0,1 à 40 000 lux, il est également adapté à un positionnement extérieur grâce à sa plage de température qui supporte un grand froid de -30 °C à une température chaude de 80 °C.



La communication avec ce capteur se fait en I2C, donc à l'aide de 3 fils : le SDA, le SCL et le GND. Pour le fonctionnement de ce capteur, 4 fils étaient donc nécessaires. Pour ma part, c'est le seul module qui doit être alimenté en 3,3 V.

Pour vérifier son bon fonctionnement, j'ai récupéré une bibliothèque sur GITHUB et j'ai fait un programme qui permet simplement de récupérer la luminosité en lux chaque seconde.

Test d'intégration : Volet roulant / capteur de luminosité



Ce regroupement a pour but de tester le volet roulant lorsqu'il est en mode automatique, c'est à dire que lorsqu'on passe au-dessus d'un certain seuil de luminosité (35 lux pour le test) le volet s'ouvre :

```
if(capteurLuminosite()>35.00){           //si la luminosité est supérieure à 35 lux
    while(digitalRead(switchUp)==HIGH){ //tant que le volet n'est pas en fin de course
        digitalWrite(cmdUp,HIGH);       //le volet monte
    }
    digitalWrite(cmdUp,LOW);
}
```

Et à l'inverse si on descend en dessous d'un certain seuil (10 lux pour le test), le volet se ferme :

```
if(capteurLuminosite()<10.00){           //si la luminosité est inférieure à 10 lux
    while(digitalRead(switchDown)==HIGH){ //tant que le volet n'est pas en fin de course
        digitalWrite(cmdDown,HIGH);       //le volet descend
    }
    digitalWrite(cmdDown,LOW);
}
```

Test unitaire : Lumière

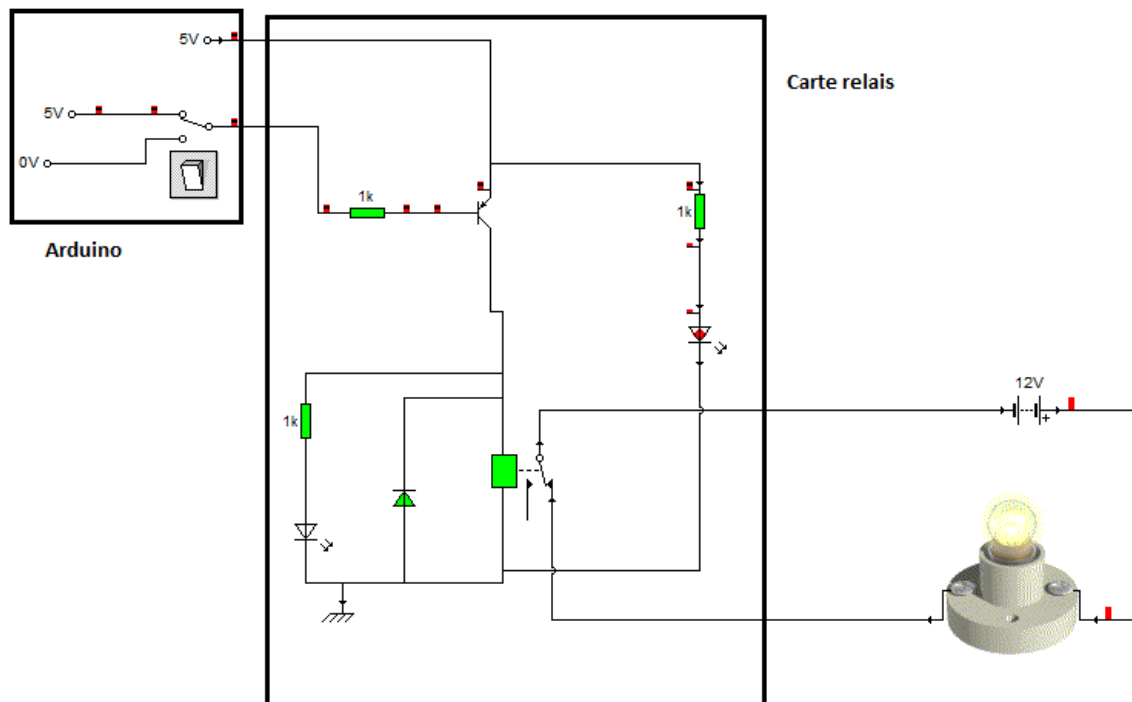
L'application mobile ne pilote pas uniquement les volets roulants, deux boutons sont affectés à la gestion de la lumière.



Il n'est pas possible de piloter une lumière avec une carte Arduino directement. Pour pouvoir le faire, il faut utiliser un relais qui va nous permettre de dissocier la partie puissance de la partie commande.



Le schéma ci-dessous montre la composition du relais ainsi que son branchement avec la lumière et le microcontrôleur.



Pour que la lumière soit allumée

- On doit avoir du courant dans la bobine
- Le transistor doit être passant
- On doit donc appliquer « 1 » sur la sortie de la Arduino

Test unitaire : Détecteur de mouvement

Lorsque l'utilisateur n'est pas chez lui, il doit être averti en cas d'intrusion. C'est pour cela que le système de sécurité est muni d'un détecteur de mouvement.

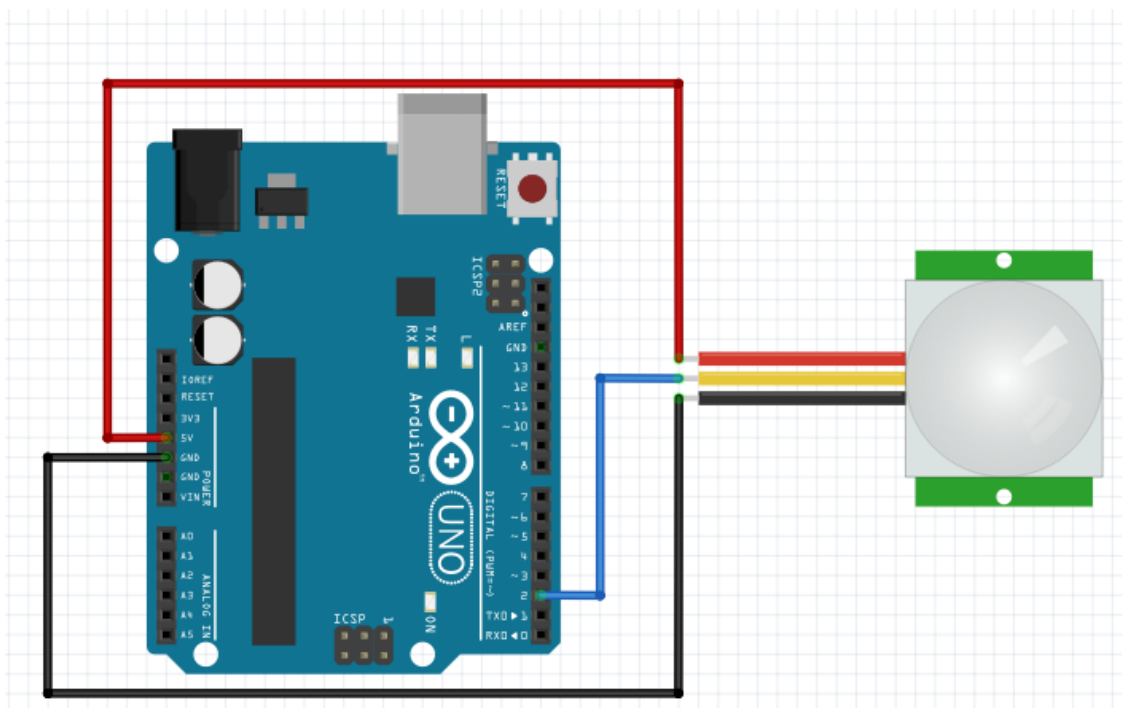


Ce détecteur est constitué d'un capteur à infrarouge détectant les mouvements d'une personne ou d'un animal. Il a un angle de détection de 120° et sa distance de détection va jusqu'à 6 mètres.

Trois broches le font fonctionner :

- Une alimentation de 5V qui permet un branchement facile sur une carte Arduino.
- Une broche de donnée que l'on pourra brancher en entrée numérique sur une carte Arduino.
- GND

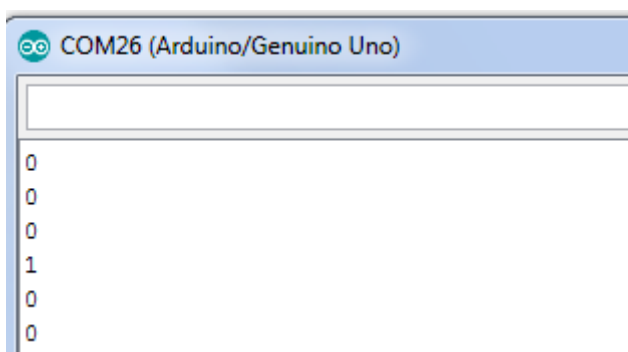
Pour tester le bon fonctionnement de ce capteur j'ai d'abord effectué le branchement sur une carte ArduinoUno.



J'ai ensuite créé une fonction retournant un booléen permettant de récupérer l'état de la broche de donnée.

```
boolean mouvementDetecte()  
{  
    int valeurCapteur = digitalRead(CAPTEUR_MOUVEMENT);    //on affecte à valeurCapteur l'état de la broche de donnée  
    if(valeurCapteur == HIGH)  
    {  
        return true;  
    }  
    else  
    {  
        return false;  
    }  
}
```

J'ai affiché sur le moniteur série la valeur de retour de la fonction :



Lorsqu'un mouvement est détecté, la fonction renvoie bien "true".

Test d'intégration : Interrupteur / détecteur de mouvement

L'utilisateur doit évidemment pouvoir désactiver le détecteur lorsqu'il est chez lui, c'est pour cela que l'on ajoute un interrupteur. Pour ce projet, un simple bouton fait office d'interrupteur :



C'est un bouton poussoir momentané, lorsqu'il est pressé il émet un signal « HIGH » et lorsqu'il ne l'est pas « LOW ». Pour pouvoir l'utiliser dans notre projet, il y a une particularité dans le code, il faut utiliser une interruption.

```
attachInterrupt(digitalPinToInterrupt(interruptBouton), interruptionBouton, RISING);
```

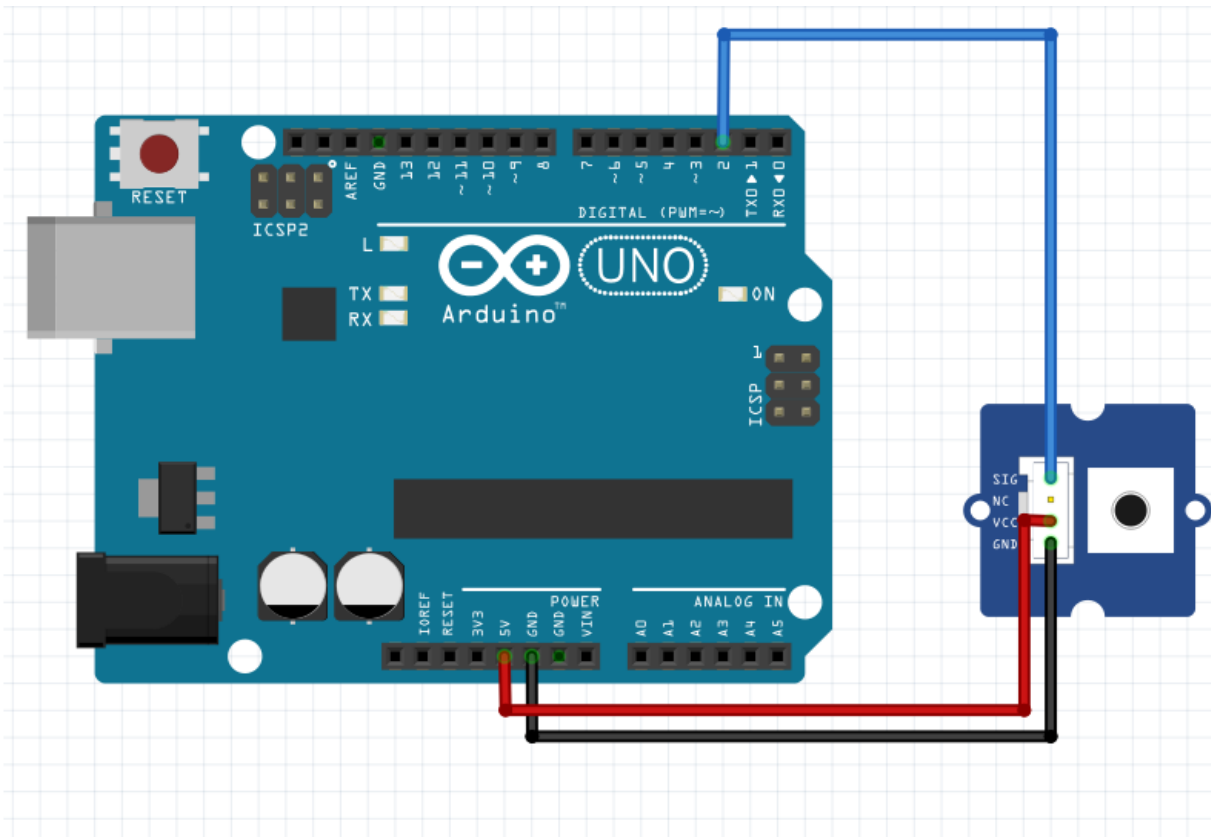
Broche d'interruption

Nom de
l'interruption

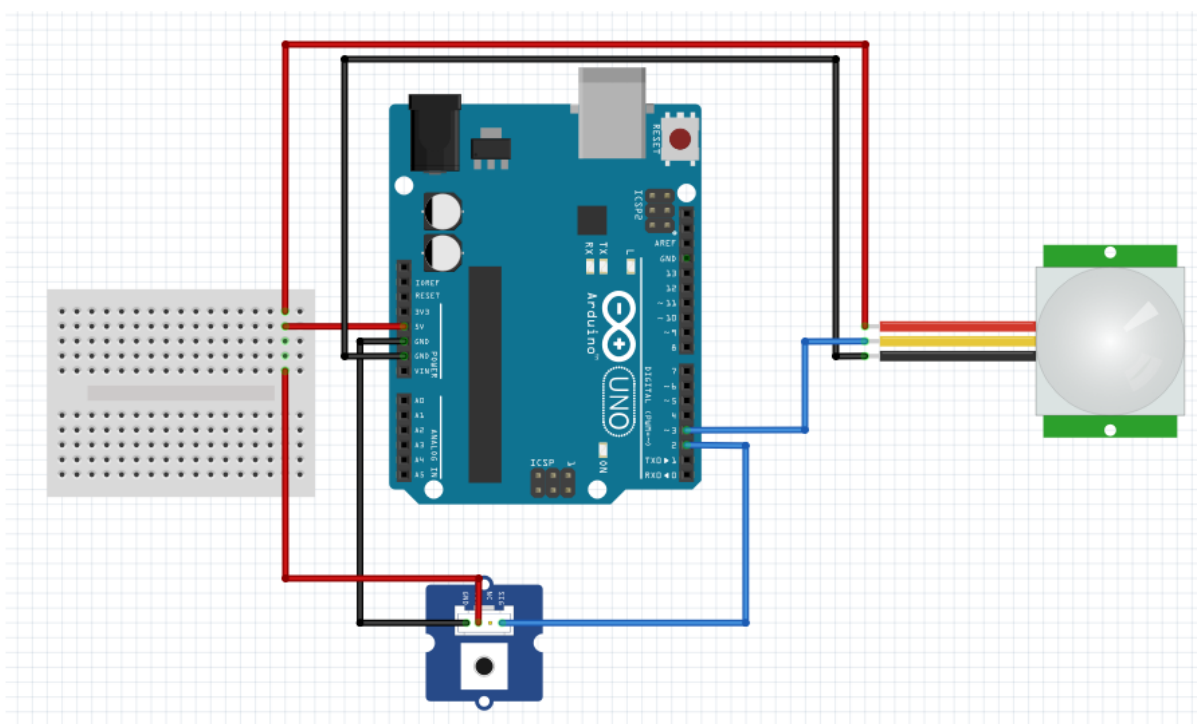
On entre dans l'interruption quand
la broche passe de LOW à HIGH

```
void interruptionBouton(){ //programme d'interruption du bouton
  if (boutonAllumer==true){ //Si le bouton est en "position ON"
    boutonAllumer=false; //On le met en position "position OFF"
    Serial.println("allumer"); //affichage sur le moniteur série de l'état du bouton
  }
  else{ //Si le bouton est en position OFF
    boutonAllumer=true; //On le met en "position ON"
    Serial.println("eteint");
  }
}
```

Pour utiliser une interruption on ne peut pas se mettre sur n'importe quelle broche. Sur une carte UNO, uniquement deux permettent d'utiliser l'interruption : la 2 et la 3.



J'ai regroupé le code de l'interrupteur avec celui du détecteur de mouvement puis j'ai effectué le branchement :

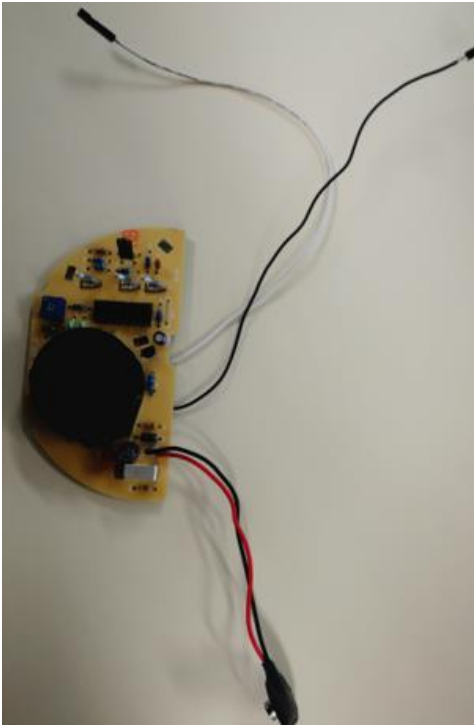


Lorsque l'on appuie sur le bouton on active ou on désactive l'alarme en ajoutant simplement une condition devant le message de détection :

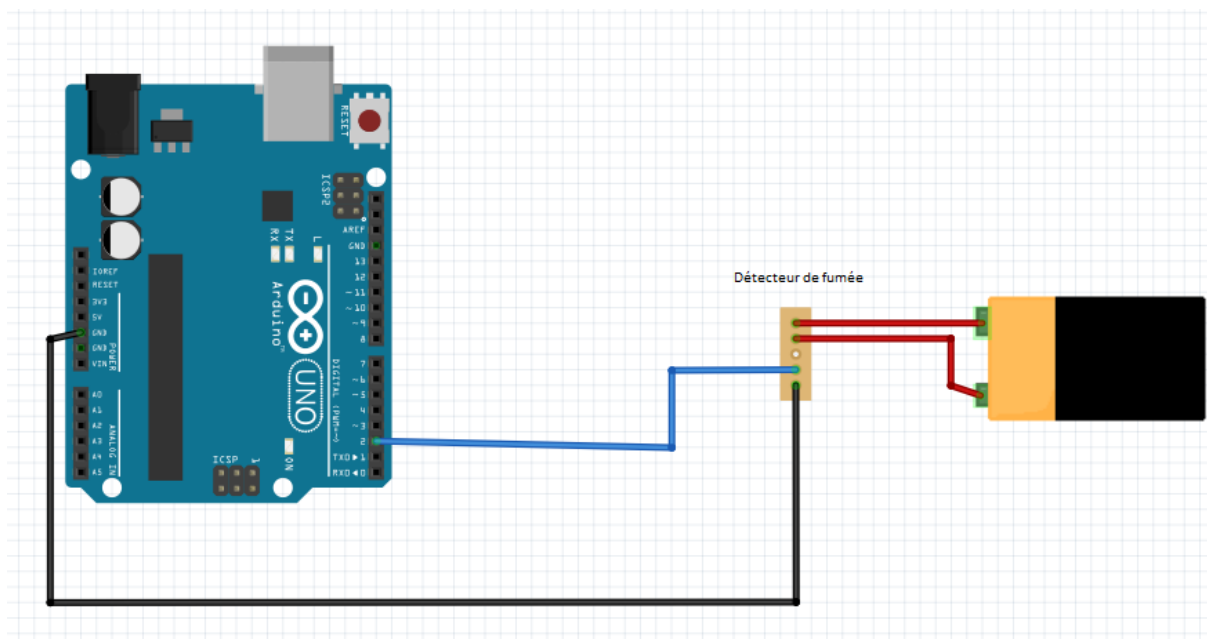
```
if(boutonAllumer==true){  
    if(mouvementDetecte()== true){  
        Serial.println("Mouvement detecte !");  
    }  
}
```


Test unitaire : Détecteur d'incendie

Le système de sécurité comprend également un détecteur de fumée qui est constamment en marche contrairement au détecteur de mouvement.



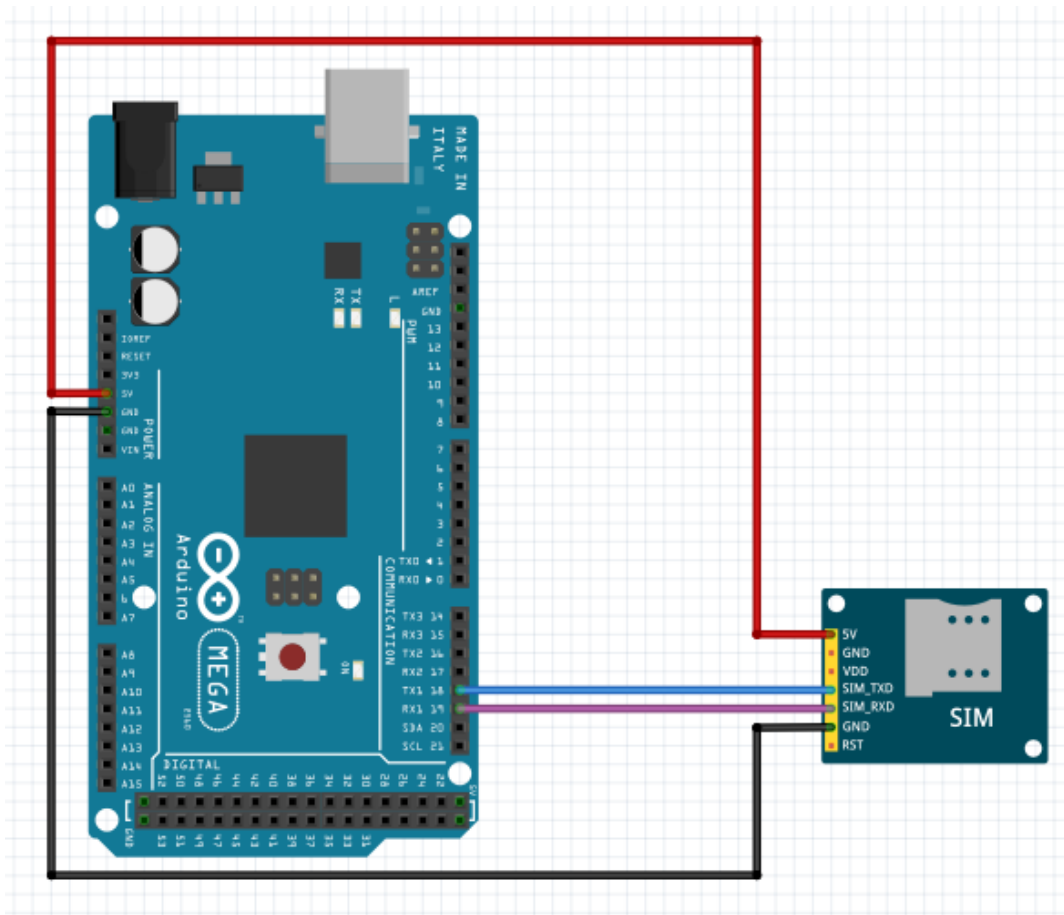
Le principe du fonctionnement de ce capteur est basé sur une transmission infrarouge entre une diode et une photodiode. Si de la fumée passe entre l'émetteur et le récepteur, cela va créer de l'opacité qui va atténuer le signal reçu par la photodiode.



Test unitaire : Module GSM

Le module GSM permet d'envoyer un message à l'utilisateur en cas de détection d'incendie ou d'intrusion.

Avant de pouvoir tester son fonctionnement, nous avons acheté une carte SIM prépayée chez Lebara Mobile qui convient bien pour son utilisation dans le projet.



J'ai dans un premier temps envoyé différentes commandes à l'aide du moniteur série Arduino.

J'ai vérifié la présence de réseau avec la commande AT+CREG?.

Avant de préparer un SMS il y a trois commandes AT à rentrer :

- AT+CMGF=1 Pour passer en mode texte.
- AT+CSCA="+33660003000" L'adresse du serveur qui, dans notre cas, est celle de Bouygues.
- AT+CMGS="numéro" Numéro du destinataire.

Il faut ensuite écrire le message et terminer par un "CTRL+Z". Or, sur le moniteur série, on ne peut pas.

Pour pouvoir envoyer un message, il nous faut donc un code qui rentre automatiquement les commandes AT nécessaires :

```
int sendATcommand(char* ATcommand, char* expected_answer, unsigned int timeout){

    int x=0, answer=0;
    char response[100];
    unsigned long previous;

    // Initialisation de la chaine de caractère (string).
    memset(response, '\0', 100);

    delay(50);

    // Initialisation du tampon d'entrée (input buffer).
    while( Serial1.available() > 0) Serial1.read();

    // Envoi des commandes AT
    Serial1.println(ATcommand);

    x = 0;
    previous = millis();

    // Cette boucle attend la réponse du module GSM.

    do{
        // Cette commande vérifie s'il y a des données disponibles dans le tampon.
        //Ces données sont comparées avec la réponse attendue.
        if(Serial1.available() != 0){
            response[x] = Serial1.read();
            x++;
            // Comparaison des données
            if (strstr(response, expected_answer) != NULL)
            {
                answer = 1;
            }
        }
        // Attente d'une réponse.
    }while((answer == 0) && ((millis() - previous) < timeout));

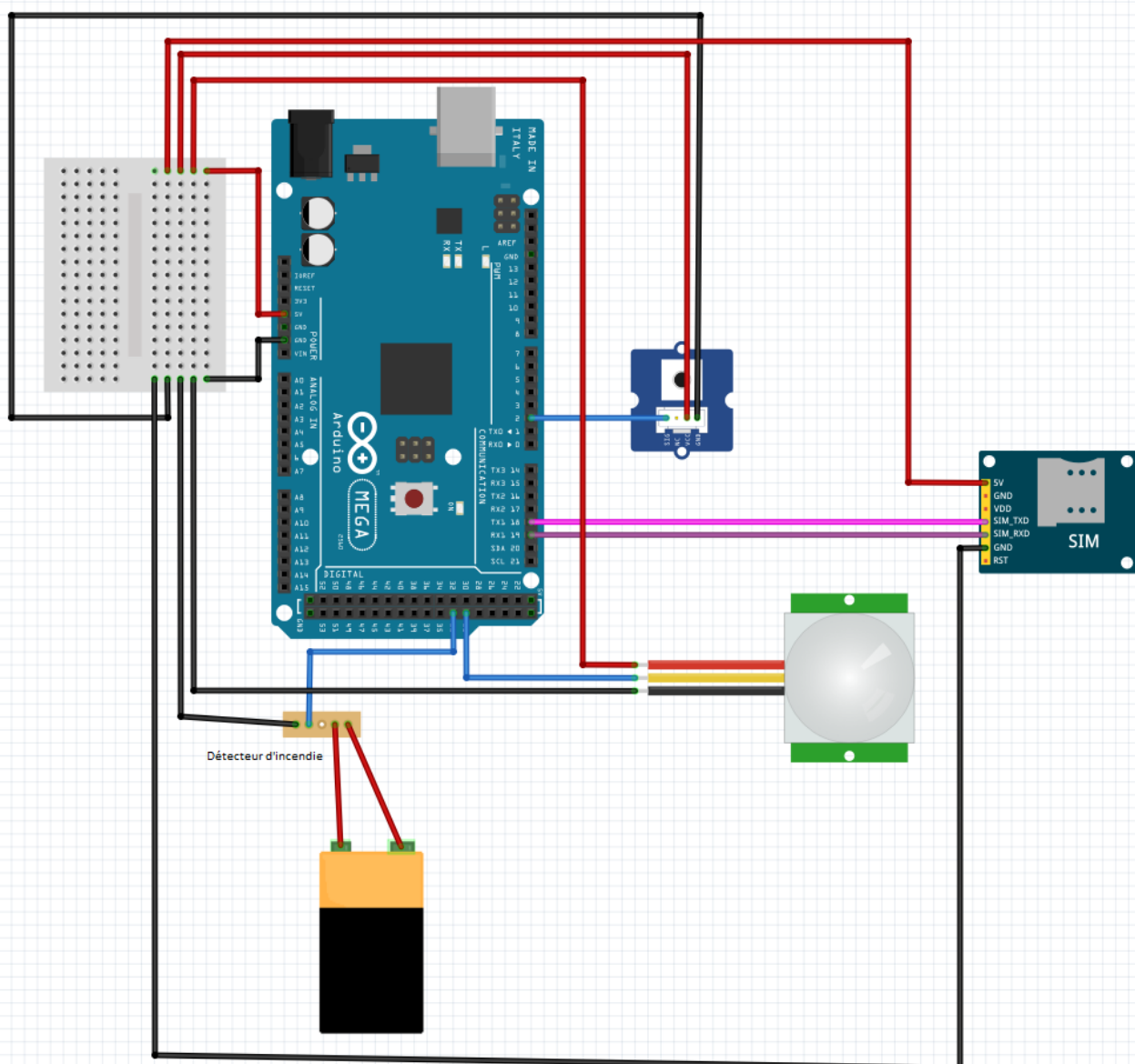
    //Serial.println(response); //Cette ligne permet de debuguer le programme en cas de problème !
    return answer;
}

void envoieSMS(bool incendie, bool mouvement){
    // Activation du mode texte pour les SMS.
    sendATcommand("AT+CMGF=1", "OK", 1000);

    sprintf(aux_string, "AT+CMGS=\"%s\"", phone_number);
    // Envoi du numéro de téléphone au module GSM.
    sendATcommand(aux_string, ">", 2000);
    Serial1.println("un mouvement est detecte !"); //On écrit le message
    Serial1.write(0x1A); //On fini par CTRL+Z
}
```

Test d'intégration : Module GSM / Interrupteur / détecteur de mouvement / Détecteur d'incendie

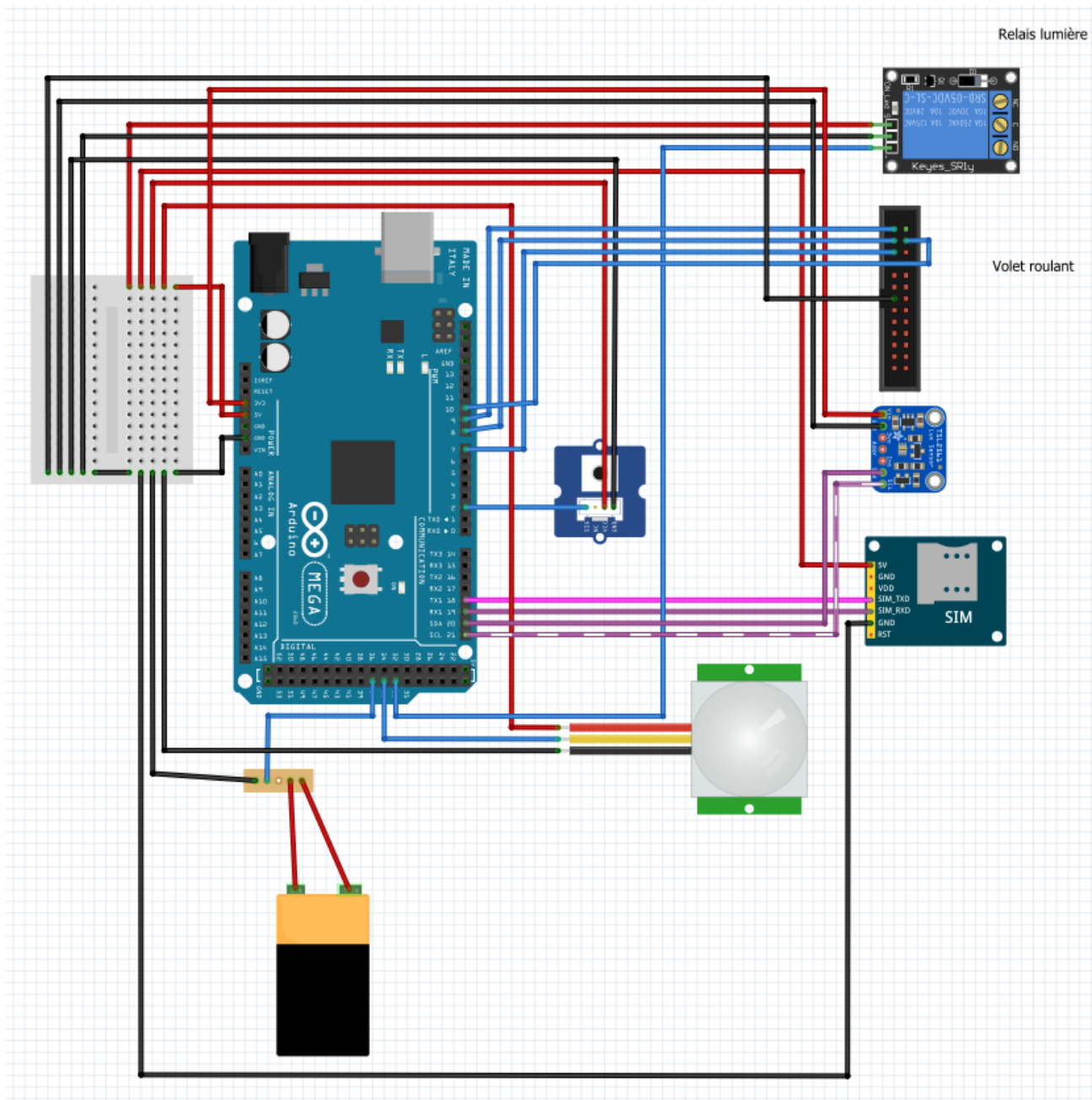
Dans ce regroupement on va retrouver tous les composants qui constituent la partie sécurité. Le module GSM permet d'envoyer un SMS dans le cas où le détecteur de mouvement (s'il est en marche selon l'état du bouton) ou le détecteur d'incendie prévient un danger.



```
if(alarme==true){                                     //Si le bouton est en état "marche"
  if(mouvementDetecte()==true){                       //On vérifie si un mouvement est détecté
    Serial1.println("un mouvement est detecte !");    //Si oui, on envoie un message à l'utilisateur
    Serial1.write(0x1A);                             //"CTRL+Z nécessaire à la fin d'un message en commande AT
  }
}
if(fumeeDetecte()==true){                             // on vérifie si un incendie est détecté
  Serial1.println("un incendie est detecte !");
  Serial1.write(0x1A);
}
```

Le problème de ce programme est que l'on reçoit trop de messages en peu de temps. Pour éviter ce "spam", lors du regroupement on a décidé de rajouter une condition avec un booléen qui change d'état en même temps que l'on envoie la trame toutes les 10 secondes.

Test d'intégration final : Module GSM / Interrupteur / détecteur de mouvement / Détecteur d'incendie / Lumière / Volet roulant / capteur de luminosité



Conclusion :

Durant ce projet, le travail en équipe était primordial. Pour chaque capteur, chaque actionneur, il fallait obligatoirement communiquer avec l'étudiant qui s'occupait de l'application mobile. Nous pouvions tester le fonctionnement de sa partie avec la mienne à chaque test unitaire.

L'organisation des fichiers, l'utilisation de GITHUB sont des éléments qui ont permis de finaliser le projet dans les temps.

Les évolutions possibles :

Pour la partie sécurité, des modifications sur l'activation/désactivation de l'alarme peuvent être effectuées :

- Remplacer l'interrupteur par un clavier matriciel.
- Isoler le module GSM pour éviter les IEM qui parfois perturbent l'envoi d'un SMS.

Annexe :

Programme principal :

```
#include "capteur.h"
#include "controleActionneur.h"
#include "horodatageConsomation.h"
#include "gestionMaison.h"
#include <avr/io.h>
#include <avr/interrupt.h>

Maison maison;

void initTimer2 (void);

unsigned short interruptePinCompteur = 3; //Variable Compteur
unsigned short flagCompteurEnergie=1;
unsigned short flagCompteurTimer=0;
volatile unsigned int cpt=0;
unsigned short interruptBouton=2;

void setup(){
    initSerial();
    initTimer2();
    initActionneur();
    initHorodatageConsomation();
    initCapteur();
    //Setup Compteur d'energie interruption
    pinMode(interruptePinCompteur, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(interruptePinCompteur), interruptionCompteur, LOW);
    //setup bouton alarme interruption
    pinMode(interruptBouton, INPUT);
    attachInterrupt(digitalPinToInterrupt(interruptBouton), interruptionBouton, RISING);
}

void loop(){
    /*****Le drapeau 1 du Timer1 se "lève" toute les 5 sec*****/
```



```

if (flagCompteurTimer==1){
    maison.temperature=capteurTemperature(); //récupération de la température dans un attribue
    maison.humidite=capteurHumidite();
    maison.etatRadiateur=etatThermostat();
    maison.emissionTrame(); //émission de la trame pour le serveur WEB et la tablette
    flagCompteurTimer=0; //remise du drapeau 1 a 0 "on baisse le drapeau".
}

/*****Le drapeau 2 du Timer1 se "lève" toute les 10 sec*****/

if (flagCompteurTimer==2){
    envoiSMS(maison.incendie, maison.mouvement); //émission du sms d'alerte si alerte
    maison.qualiteAir=capteurQualiteAir();
    maison.temperature=capteurTemperature();
    maison.humidite=capteurHumidite();
    maison.etatRadiateur=etatThermostat();
    maison.luminosite=capteurLuminosite();
    maison.incendie = false; //on remet les valeurs des attribue d'alerte a l'état passif
    maison.mouvement = false;
    maison.emissionTrame(); //émission de la trame pour le serveur WEB et la tablette
    flagCompteurTimer=0; //remise du drapeau 2 a 0 "on baisse le drapeau".
}

maison.lectureTablette(); //on récupère les valeurs valeurs envoyer par la tablette

/*****Partie sécurité*****/

if (maison.incendie == false){
    maison.incendie=capteurIncendie();
}

if (maison.alarme==true){
    if (maison.mouvement == false){
        maison.mouvement=capteurMouvement();
    }
}

/*****Controle des actionneur en fonction des commande de l'utilisateur*****/

```

```
controleThermostat(maison.radiateurMode, maison.temperature, maison.temperatureUtilisateur,
maison.radiateur);
```

```
maison.volet1=controleVolet1(maison.volet1Etat, maison.voletMode);
```

```
maison.volet2=controleVolet2(maison.volet2Etat, maison.voletMode);
```

```
maison.lumiere=controleLumiere(maison.lumiereEtat);
```

```
/******Le drapeau 1 de l'interruption compteur******/
```

```
if (flagCompteurEnergie==1){
```

```
    maison.consomation=consomation();
```

```
    maison.annee=annee();
```

```
    maison.mois=mois();
```

```
    maison.jour=jour();
```

```
    maison.heure=heure();
```

```
    maison.minutes=minutes();
```

```
    maison.seconde=seconde();
```

```
    horodatage(maison.consomation);
```

```
    flagCompteurEnergie=0;
```

```
}
```

```
}
```

```
/*******/
```

```
void interruptionBouton(){
```

```
    if (maison.alarme==true){
```

```
        maison.alarme=false;
```

```
        Serial.println("eteint");
```

```
}
```

```
else{
```

```
    maison.alarme=true;
```

```
    Serial.println("allume");
```

```
}
```

```
}
```

```
void interruptionCompteur(){
```

```
    delay(50);
```

```
    if (digitalRead(interruptePinCompteur)==0){
```

```
    flagCompteurEnergie=1;
    while(digitalRead(interruptePinCompteur)!=1);
}
}

void initTimer2 (){
    //config timer2
    TCCR2A=0x00;
    TCCR2B |=(1<<CS22)|(1<<CS21)|(1<<CS20);
    TIMSK2 |=(1<<TOIE2);
    sei();
}

ISR(TIMER2_OVF_vect){
    cpt++;
    if (cpt==305){
        flagCompteurTimer=1;
    }
    if (cpt==610){
        flagCompteurTimer=2;
        cpt=0;
    }
}
```

capteur.h

```
#ifndef CAPTEUR_H
#define CAPTEUR_H

void initCapteur (void);
double capteurTemperature (void);
double capteurHumidite (void);
int capteurQualiteAir (void);
float capteurLuminosite(void);
bool capteurIncendie(void);
bool capteurMouvement(void);
void envoieSMS(bool, bool);
int sendATcommand (char*, char*, unsigned int);

#endif
```

capteur.cpp

```
#include "capteur.h"
#include "Arduino.h"
#include <HIH6130.h>
#include <AirQuality.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2561_U.h>
#define CAPTEUR_INCENDIE 35
#define CAPTEUR_MOUVEMENT 36
```

```
Adafruit_TSL2561_Unified tsl = Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT, 12345);
```

```
HIH6130 rht(0x27);
```

```
AirQuality airqualitysensor;
int current_quality = -1;
```

```
int answer;
char aux_string[30];
char phone_number[] = "+33648977501";
```

```
void initCapteur(){
  Serial.println("initialisation des capteurs...");
  rht.begin();
  pinMode(CAPTEUR_INCENDIE, INPUT);
  pinMode(CAPTEUR_MOUVEMENT, INPUT);
  airqualitysensor.init(A0);
  delay(5000);
  Serial.println("Capteurs initialises.");
}
```

```
/******Fonction Capteur : Temperature, Humidité, Qualité d'air, Luminosité,
Incendie, Mouvement******/
```

```
double capteurTemperature(){
  rht.readRHT();
  return rht.temperature;
}
```

```
double capteurHumidite(){
  rht.readRHT();
  return rht.humidity;
}
```

```
int capteurQualiteAir(){
```

```

current_quality=airqualitysensor.slope();
return current_quality;
}

```

```

float capteurLuminosite(void){
    tsl.setGain(TSL2561_GAIN_16X);

```

```

    sensors_event_t event;
    tsl.getEvent(&event);

```

```

    return event.light;
}

```

```

bool capteurIncendie(void){
    int fumeeDetecte = digitalRead(CAPTEUR_INCENDIE);
    if(fumeeDetecte == HIGH){
        return true;
    }
    else{
        return false;
    }
}

```

```

bool capteurMouvement(void){
    int mouvementDetecte = digitalRead(CAPTEUR_MOUVEMENT);
    if(mouvementDetecte == HIGH){
        return true;
    }
    else{
        return false;
    }
}

```

```

/*****Fonction d'envoi du SMS
d'alerte*****/

```

```

void envoieSMS(bool incendie, bool mouvement){

```

```

    //while( (sendATcommand("AT+CREG?", "+CREG: 0,1", 500) || sendATcommand("AT+CREG?", "+CREG: 0,5", 500))
    == 0 );

```

```

    // Activation du mode texte pour les SMS.

```

```

    sendATcommand("AT+CMGF=1", "OK", 1000);

```

```

    sprintf(aux_string, "AT+CMGS=\"%s\"", phone_number);

```

```

    // Envoi du numéro de téléphone au module GSM.

```

```

    sendATcommand(aux_string, ">", 2000);

```

```
if(mouvement==true){
  Serial.println("un mouvement est detecte !");
  Serial1.println("un mouvement est detecte !");
  Serial1.write(0x1A);
}
if(incendie==true){
  Serial1.println("un incendie est detecte !");
  Serial1.write(0x1A);
}
}

/*****Fonction Timer et Commande
AT*****/
```

```
int sendATcommand(char* ATcommand, char* expected_answer, unsigned int timeout){
```

```
  int x=0, answer=0;
  char response[100];
  unsigned long previous;
```

```
  // Initialisation de la chaine de caractère (string).
  memset(response, '\0', 100);
```

```
  delay(50);
```

```
  // Initialisation du tampon d'entrée (input buffer).
  while( Serial1.available() > 0) Serial1.read();
```

```
  // Envoi des commandes AT
  Serial1.println(ATcommand);
```

```
  x = 0;
  previous = millis();
```

```
  // Cette boucle attend la réponse du module GSM.
```

```
  do{
    // Cette commande vérifie s'il y a des données disponibles dans le tampon.
    //Ces données sont comparées avec la réponse attendue.
    if(Serial1.available() != 0){
      response[x] = Serial1.read();
```

```

    x++;
    // Comparaison des données
    if (strstr(response, expected_answer) != NULL)
    {
        answer = 1;
    }
}
// Attente d'une réponse.
}while((answer == 0) && ((millis() - previous) < timeout));

//Serial.println(response); //Cette ligne permet de debuguer le programme en cas de problème !
return answer;
}

ISR(TIMER1_OVF_vect) //timer
{
    if(airqualitysensor.counter==61)//une fois que le nombre de marqueur
        //atteind 61 (~30ms) on exécute les
        //lignes de commande suivante
    {
        /*On récupère la dernière valeur avant de la remplacer par
        *une valeur plus récentes puis on remet le nombre de marqueur
        * à zéro et on relance le timer*/
        airqualitysensor.last_vol=airqualitysensor.first_vol;
        airqualitysensor.first_vol=analogRead(A0);
        airqualitysensor.counter=0;
        airqualitysensor.timer_index=1;
        PORTB=PORTB^0x20;
        /*La bibliothèque compare les deux valeur last_vol et
        *first_vol pour nous indiquer la qualité de l'air*/
    }
    else
    {
        airqualitysensor.counter++;
    }
}

```

controleActionneur.h

```

#ifndef CONTROLEACTIONNEUR_H
#define CONTROLEACTIONNEUR_H

```

```

void initActionneur(void);
void controleThermostat (bool, double, double, bool);
bool controleLumiere(int);
bool controleVolet1(bool, bool);
bool controleVolet2(bool, bool);

```

```
bool etatThermostat (void);
#endif
```

controleActionneur.cpp

```
#include "controleActionneur.h"
#include "Arduino.h"
#include "capteur.h"
```

```
int cmdUp = 24;
int cmdDown = 25;
int switchUp = 26;
int switchDown = 27;
```

```
int cmdUp2 = 28;
int cmdDown2 = 29;
int switchUp2 = 30;
int switchDown2 = 31;
```

```
int pinThermostat = 4;
int pinLumiere = 33;
char valeurEtat;
```

```
void initActionneur(){
    pinMode(pinThermostat, OUTPUT);
    pinMode(pinLumiere, OUTPUT);
```

```
    pinMode(cmdUp, OUTPUT);
    pinMode(cmdDown, OUTPUT);
    pinMode(switchUp, INPUT);
    pinMode(switchDown, INPUT);
```

```
    pinMode(cmdUp2, OUTPUT);
    pinMode(cmdDown2, OUTPUT);
    pinMode(switchUp2, INPUT);
    pinMode(switchDown2, INPUT);
}
```

```
/******Fonction Controle des actionneurs : Thermostat, Lumiere,
Volet******/
```

```
void controleThermostat(bool mode, double temperature, double temperatureUtilisateur, bool valeurEtat){
    if (mode==false){
        if (valeurEtat==true){
            digitalWrite(pinThermostat,HIGH);
        }
        if (valeurEtat==false){
```



```
    digitalWrite(pinThermostat,LOW);
  }
}
if (mode==true){
  if(temperature<=temperatureUtilisateur-1){
    digitalWrite(pinThermostat,HIGH);
  }
  if(temperature>=temperatureUtilisateur){
    digitalWrite(pinThermostat,LOW);
  }
}
}
```

```
bool controleLumiere(int positionLumiere){
  bool etat=false;
  if(positionLumiere==1){
    digitalWrite(pinLumiere, HIGH);
    etat=true;
  }
}
```

```
if(positionLumiere==0){
  digitalWrite(pinLumiere, LOW);
  etat=false;
}
```

```
return etat;
}
```

```
bool controleVolet1(bool sens, bool mode){
```

```
  bool positionVolet;
  if(mode==true){
    if(sens==true){          //ouvrir
      while(digitalRead(switchUp)==HIGH){
        digitalWrite(cmdUp,HIGH);
      }
      digitalWrite(cmdUp,LOW);
      positionVolet = true;
    }
  }
```

```
  if(sens==false){          //fermer
    while(digitalRead(switchDown)==HIGH){
      digitalWrite(cmdDown,HIGH);
    }
    digitalWrite(cmdDown,LOW);
    positionVolet = false;
  }
```

```

    }
}

if(mode==false){
    if(capteurLuminosite(>35.00){
        while(digitalRead(switchUp)==HIGH){
            digitalWrite(cmdUp,HIGH);
        }
        digitalWrite(cmdUp,LOW);
        positionVolet = true;
    }
    if(capteurLuminosite(<10.00){
        while(digitalRead(switchDown)==HIGH){
            digitalWrite(cmdDown,HIGH);
        }
        digitalWrite(cmdDown,LOW);
        positionVolet = false;
    }
}
return positionVolet;
}

```

```

bool controleVolet2(bool sens, bool mode){

```

```

    bool positionVolet;
    if(mode==true){
        if(sens==true){                //ouvrir
            while(digitalRead(switchUp2)==HIGH){
                digitalWrite(cmdUp2,HIGH);
            }
            digitalWrite(cmdUp2,LOW);
            positionVolet = true;
        }

        if(sens==false){              //fermer
            while(digitalRead(switchDown2)==HIGH){
                digitalWrite(cmdDown2,HIGH);
            }
            digitalWrite(cmdDown2,LOW);
            positionVolet = false;
        }
    }
}

```

```

if(mode==false){
    if(capteurLuminosite(>35.00){

```

```

    while(digitalRead(switchUp2)==HIGH){
        digitalWrite(cmdUp2,HIGH);
    }
    digitalWrite(cmdUp2,LOW);
    positionVolet = true;
}
if(capteurLuminosite())<10.00){
    while(digitalRead(switchDown2)==HIGH){
        digitalWrite(cmdDown2,HIGH);
    }
    digitalWrite(cmdDown2,LOW);
    positionVolet = false;
}
}
return positionVolet;
}

/*****Récupération de l'état du thermostat :
allumer/eteint*****/

```

```

bool etatThermostat(){
    bool val = digitalRead(pinThermostat);
    return val;
}

```

horodatageConsomation.h

```

#ifndef HORODATAGECONSOMATION_H
#define HORODATAGECONSOMATION_H

```

```

void initHorodatageConsomation(void);
void horodatage(float);
void initSD(void);
void initRTC(void);
float consomation(void);
int annee(void);
int jour(void);
int mois(void);
int heure(void);
int minutes(void);
int seconde(void);

```

```

#endif

```

horodatageConsomation.cpp

```

#include <SPI.h>      //
#include <SdFat.h>    //Bibliothèque carte SD
#include <RTCLib.h>   //Bibliothèque RTC
#include "horodatageConsomation.h"
#include "gestionMaison.h"

/*Le module SD fonctionne en liaison SPI et les pin SPI sont différentes en fonction de la carte
carte Arduino utilisée. cf : https://www.arduino.cc/en/Reference/SPI */
#define BUFFER_SIZE 250 //définition de la taille du buffer.

SdFat sd;           //
uint8_t buf[BUFFER_SIZE]; //Variable SD
float pulsion = 0;
RTC_DS1307 rtc;     //Variable RTC

void initHorodatageConsomation(){
    //Setup SD
    Serial.println("init SD");
    if(!sd.begin()){
        Serial.println("erreur init");
        return;
    }
    rtc.begin();
    rtc.isrunning();
}

/*****Fonction qui horodate la consommation d'énergie et sauvegarde le tout sur une carte
SD*****/

void horodatage(float consommation){

    SdFile fichier;

    pulsion=pulsion+0.1;
    //écriture dans le fichier txt Compteur_Elec dans la SD
    if(!fichier.open(&sd, "Compteur_bis_bis_bis.txt", O_RDWR|O_TRUNC|O_AT_END)){
        Serial.println("Erreur");
        return;
    }
    fichier.print(consommation);
    fichier.print("KWh ");
    DateTime now = rtc.now();
    //affichage de la date
    fichier.print(now.year());
    fichier.print('/');

```

```

    fichier.print(now.month());
    fichier.print('/');
    fichier.print(now.day());
    fichier.print(" ");
    //affichage de l'heure
    fichier.print(now.hour());
    fichier.print(':');
    fichier.print(now.minute());
    fichier.print(':');
    fichier.println(now.second());
    fichier.close();

    //sd.ls("/", LS_SIZE|LS_R);

    //lecture du contenu du fichier txt dans la SD
    if(!fichier.open(&sd, "Compteur_bis_bis_bis.txt", O_READ)){
        Serial.println("erreur");
        return;
    }
    fichier.read(buf, sizeof(buf));
    String myString = String ((char *)buf);
    myString.trim();
    Serial.println(myString);
    fichier.close();
}

/*****Fonction de comptage de la consommation*****/

float consommation(){
    pulsion=pulsion+0.1;
    return pulsion;
}

/*****Fonctions de récupération de la date et de
l'heure*****/

int annee(){
    DateTime now = rtc.now();
    return now.year();
}

int mois(){
    DateTime now = rtc.now();
    return now.month();
}

```

```
int jour(){
    DateTime now = rtc.now();
    return now.day();
}
```

```
int heure(){
    DateTime now = rtc.now();
    return now.hour();
}
```

```
int minutes(){
    DateTime now = rtc.now();
    return now.minute();
}
```

```
int seconde(){
    DateTime now = rtc.now();
    return now.second();
}
```

gestionMaison.h

```
#ifndef __MAISON_H__
#define __MAISON_H__
class Maison
{
public:
    bool lumiere=false;
    bool volet1=false;
    bool volet2=false;
    float luminosite;
    bool alarme=false;
    bool incendie;
    bool mouvement;

    double temperatureUtilisateur=21;
    bool radiateur=false;
    double temperature;
    int qualiteAir;
    double humidite;
    float consommation;

    bool etatRadiateur;
    bool lumiereEtat=false; //_false=0=eteind|true=1=allumer
    bool volet1Etat=false; //_false=0=eteind|true=1=allumer
```

```
bool volet2Etat=false; //_false=0=eteind|true=1=allumer
bool voletMode=true; //_true=0=mode manuel|false=1=mode automatique
bool radiateurMode=false; //_false=0=mode manuel|true=1=mode automatique
```

```
int annee;
int jour;
int mois;
int heure;
int minutes;
int seconde;
```

```
void emissionTrame (void);
void lectureTablette (void);
```

```
private:
};
void initSerial (void);
```

```
#endif
```

gestionMaison.cpp

```
#include "gestionmaison.h"
#include "Arduino.h"
```

```
void initSerial (){
    Serial.begin(9600);
    Serial1.begin(9600);
    Serial2.begin(9600);
    Serial3.begin(9600);
}
```

```
/******Fonction de récupération des donnée de la
tablette*****/
```

```
void Maison::lectureTablette (){
```

```
    while(Serial2.available() > 0) {
        double c = Serial2.read();
        if (c==0){ //lumiere eteinte
            if (lumiere==true){
                lumiereEtat=false;
            }
        }
        if (c==1){ //lumiere allumer
            if (lumiere==false){
                lumiereEtat=true;
            }
        }
    }
}
```

```
}  
}  
if (c==2){ //volet1 fermer  
    if (volet1Etat==true){  
        volet1Etat=false;  
    }  
}  
if (c==3){ //volet1 ouvrir  
    if (volet1Etat==false){  
        volet1Etat=true;  
    }  
}  
if (c==4){ //volet2 fermer  
    if (volet2Etat==true){  
        volet2Etat=false;  
    }  
}  
if (c==5){ //volet2 ouvrir  
    if (volet2Etat==false){  
        volet2Etat=true;  
    }  
}  
if (c==6){ //chauffage etteind  
    if (radiateur==true){  
        radiateur=false;  
    }  
}  
if (c==7){ //chauffage allumer  
    if (radiateur==false){  
        radiateur=true;  
    }  
}  
if (c==8){ //chauffage manuel  
    radiateurMode=true;  
}  
if (c==9){ //chauffage auto  
    radiateurMode=false;  
}  
if (c>=10 && c<=30){  
    temperatureUtilisateur = c;  
}  
if (c==31){ //volet manuel  
    voletMode=true;  
}  
if (c==32){ //volet auto  
    voletMode=false;
```



```

    }
}
}

/*****Fonction d'emission des valeurs au serveur Web et a la
Tablette*****/

```

```
void Maison::emissionTrame (){
```

```
    String trame;
```

```
    //tablette
```

```
    trame += lumiere;
```

```
    trame += "!";
```

```
    trame += luminosite;
```

```
    trame += "!";
```

```
    trame += volet1Etat;
```

```
    trame += "!";
```

```
    trame += volet2Etat;
```

```
    trame += "!";
```

```
    trame += etatRadiateur;
```

```
    trame += "!";
```

```
    trame += temperature;
```

```
    trame += "!";
```

```
    trame += consomation;
```

```
    trame += "!";
```

```
    trame += humidite;
```

```
    trame += "!";
```

```
    trame += qualiteAir;
```

```
    //relevé date
```

```
    trame += "!";
```

```
    trame += jour;
```

```
    trame += "!";
```

```
    trame += mois;
```

```
    trame += "!";
```

```
    trame += annee;
```

```
    trame += "!";
```

```
    trame += heure;
```

```
    trame += "!";
```

```
    trame += minutes;
```

```
    trame += "!";
```

```
    Serial.println(trame);
```

```
    Serial2.println(trame);
```

```
    Serial3.println(trame);
```

```
    trame = "";
```

```
}
```

Câblage du système :

