Garin Lucas
Decelle Lucas
Delise Antoine
Roux Nicolas
Madelon Rémi

Projet POLYTECH PEIP Deuxième année
Tuteur responsable : Didier Donsez

Domotique : Maison Intelligente

2014-2015

Table des matières

Introduction

Dans le cadre du DU Polytech, chaque étudiant doit participer à la réalisation d'un projet. Nous avons choisi entre plusieurs thématiques et avons ensuite former des groupes pour chaque thème.
Nous avons entrepri de travailler sur la domotique avec Didier Donsez comme enseignant responsable.

La domotique regroupe les technologies de l'électronique, de l'automatique, de l'informatique et des télécommunications permettant d'améliorer le confort, la sécurité, la communication et la gestion d'énergie d'une maison, d'un lieu public… Notre tuteur a choisi une approche plutôt pratique autour de ce thème en mettant de côté la ressource documentaire.

En effet, la domotique permet par exemple d'optimiser l'utilisation de l'éclairage, du chauffage afin de réduire notre consommation en énergie. C'est pourquoi Didier Donsez nous a proposé de fabriquer un modèle réduit d'une maison dite « intelligente ».

Cette maquette, à l'échelle d'un Playmobil, permettrait de présenter certaines fonctionnalités de la domotique à travers 3 scénarios : Motion sensor, Free cooling et Fire Alarm. Ces scénarios seront automatisés via des cartes « arduino » exécutant des programmes informatiques.

Didier Donsez nous a proposé de poster la construction de cette maquette sur Internet sous forme de manuel afin que collèges et lycées puissent reproduire cette maison dans le but d'enseigner le langage « arduino ». Cependant, n'importe quelle personne pourrait également reconstruire notre maquette en achetant le matériel décrit dans le manuel.

Nous nous sommes alors lancés dans ce vaste projet, soit la fabrication de la maison et la programmation de nos scénarios.

I) Fabrication de la maison

a) Struture principale

La première étape est la plus importante, tout ce qui suivra dépendra de celle-ci. Elle consiste à créer une maison. Pour cela, nous avons dessiné la structure principale, c'est-à-dire les murs extérieurs et le sol. Nous avons alors utilisé le logiciel Makercase dont le lien est ici : http://www.makercase.com/ .

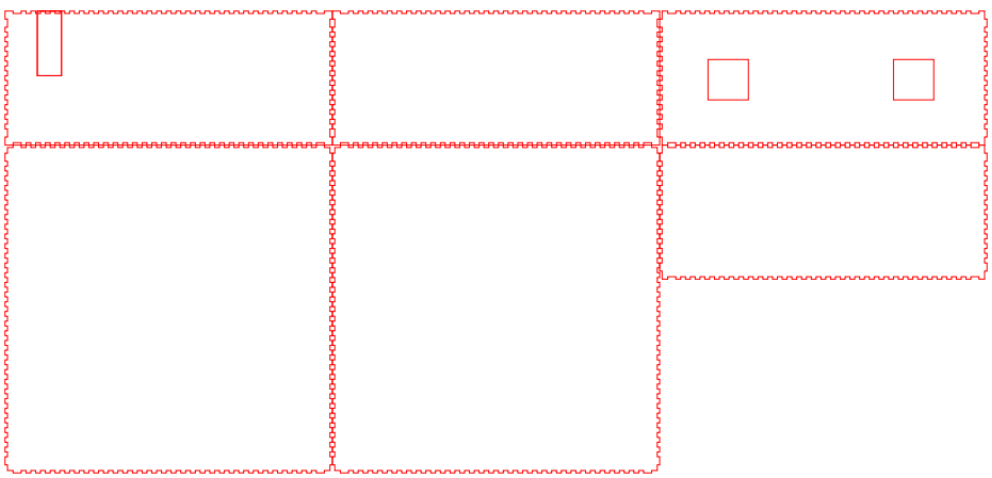
Ce logiciel permet de construire le plan d'un cube ou d'un rectangle dans le but d'être découpé dans un matériau adéquat. Ce logiciel propose 3 façons d'emboiter les murs : Finger, Flat et T-Slot. On utilise le type de fixation « Finger », et de ce fait le cube crée pourra tenir sans colle ni clou.

Le but est de construire une maison adaptée à la taille d'un Playmobil. De ce fait nous avons choisi une dimension de 40cm*40cm avec des murs fait en MBF d'une épaisseur de 3 mm.

Pour une meilleur visualisation et compréhension des différents systèmes présents dans la maison, nous avons effectué une coupe de la maison en ajoutant des traits rouges supplémentaires de biais sur deux murs opposés. Ainsi la hauteur maximum des murs est de 16cm et la hauteur minimum est de 3,5cm. Ensuite, nous avons dessiné une porte d'entrée d'une dimension 8 cm par 3 cm sur un coté perpendiculaire à la coupe.

De plus, nous avons inclue deux fenêtres 5cm par 5cm situées sur le mur du fond, celui qui est le plus grand en hauteur et en largeur après la découpe. Elles sont aussi bien nécessaires pour nos scénarios automatisés que d'un point de vue esthétique.

Nous avons ensuite enregistré les plans en format .svg pour créer la structure de notre maison.

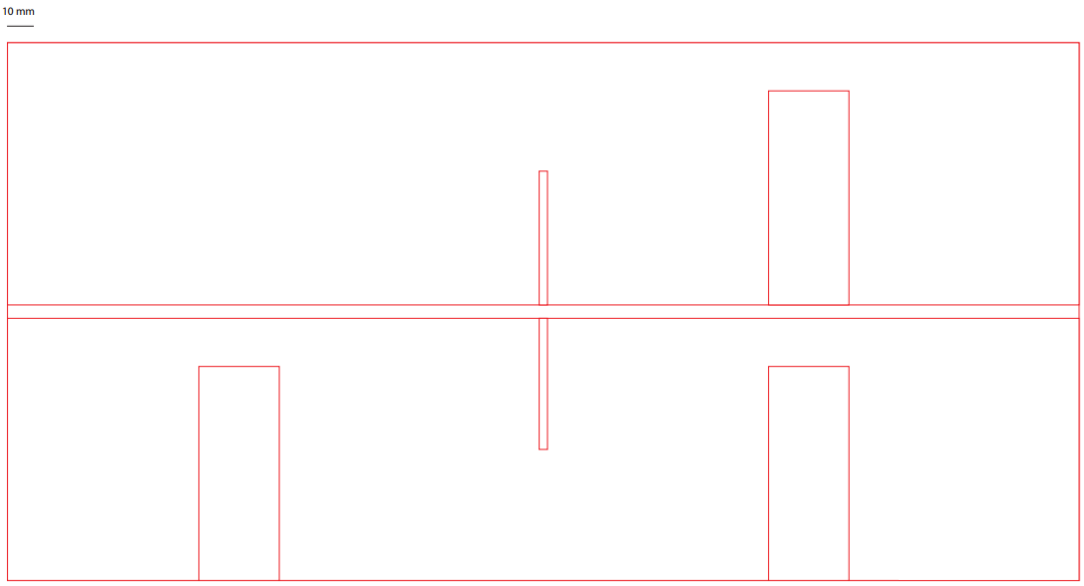


b) Murs intérieurs

Le but de cette étape est de définir la disposition des pièces à l'intérieur de la maison. Ainsi il faut dessiner les plans des murs intérieurs. Il y a 2 logiciels principaux pour concevoir les plans : Inkscape et Adobe Illustrator. Le premier est gratuit mais n'est pas très pratique pour dessiner des plans précis. Le deuxième est payant mais offre de nombreuses fonctionnalités, et est plus adapté à notre besoin. Par chance, Adobe Illustrator met à disposition une version d'essai pendant 30 jours. De ce fait nous avons établi la configuration intérieure de notre maison sur ce logiciel.

Nous avons décidé de séparer la maison en quatre pièces de même taille, soit 20cm*20cm. Ceci nous a permis de mettre en place, et par la suite de présenter, nos quatres scénarios facilement, de manière précise et distincte. Pour cela, nous avons dessiné deux murs perpendiculaires sur Adobe Illustrator. Comme notre maison est censée être utilisée et reconstruite à nouveau, notamment par des collégiens ou des lycéens dans le but de découvrir le potentiel des cartes arduino, nous avons cherché une façon pratique d'ajouter les murs intérieurs dans notre maison. C'est pourquoi, nous avons équipé nos murs de petites fentes, dessinés via Adobe Illustrator, afin de pouvoir les emboîter en forme de croix. Ceci permet de créer une maison en kit-détachable, on peut enlever les murs intérieurs et détacher les murs extérieurs grâce aux encoches « Finger » afin de mettre le tout dans un sac. Nous avons également inclue 3 portes (8cm*3cm) sur les murs intérieurs pour permettre aux « Playmobils » de circuler de pièces en pièces et d'actionner chacun des scénarios individuellement.

Voici les photos des plans:



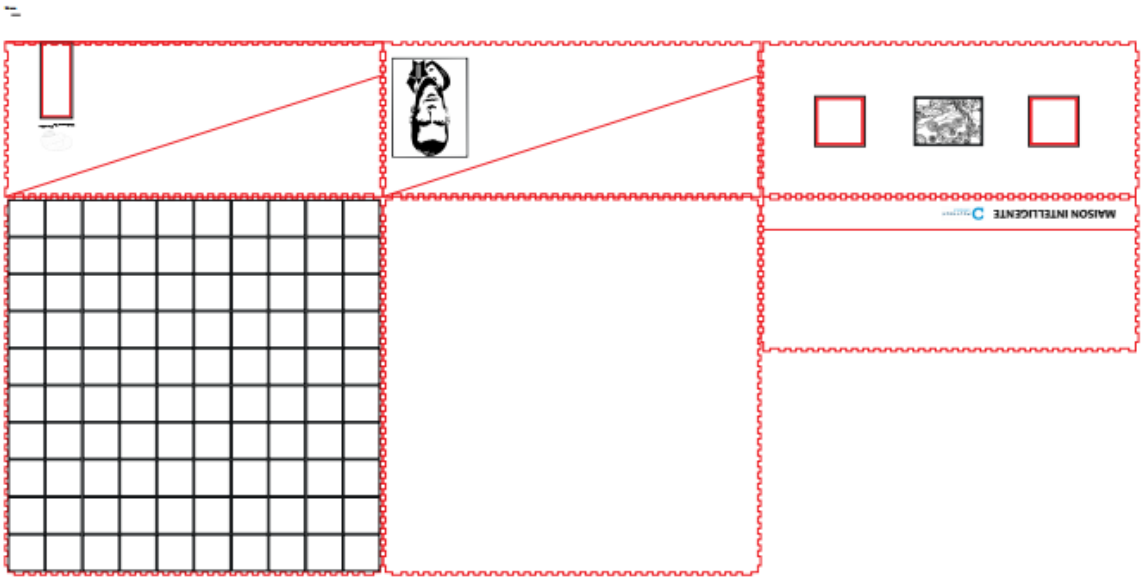
c) Mobiliers

Pour rendre plus réelle notre maison, nous avons décidé d'inclure plusieurs objets du quotidiens, ainsi que du mobilier, propre à une maison. Cette dernière étape de fabrication est majoritairement esthétique mais s'avère utile suivant les scénarios puisque certains d'entre eux nécessite un réveil, un lit, etc.. pour que les scénario s'enclenchent. Nous avons alors récupéré les plans essentiels sur le site http://fablab.ensimag.fr/index.php/PILBI-2013-Team2.

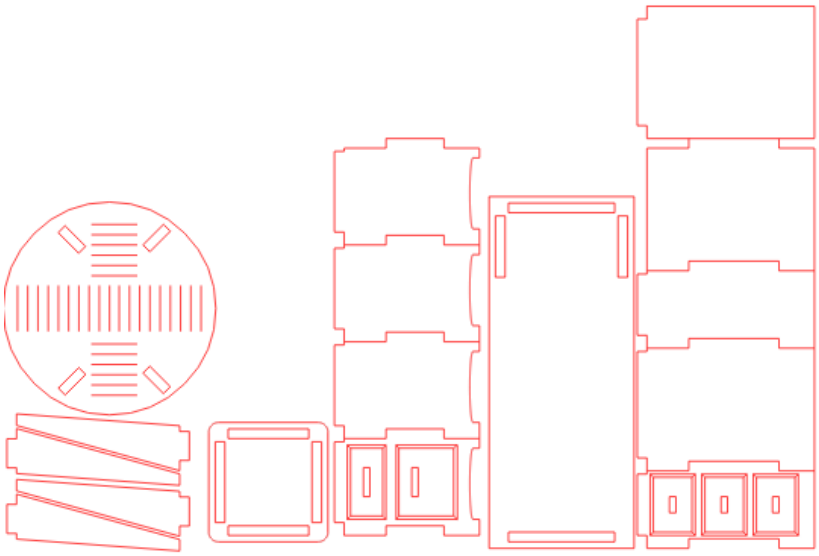
On distingue alors deux types de mobilier, celui en bois, et celui en plastique. Les plans des objets, tel que la table, le bureau etc ... ont été édités sur Adobe Illustrator et peuvent être fabriqué par la découpeuse laser. Les autres plans, comme les WC, le réveil, etc ... ont été réalisés grâce à l'imprimante 3D. Nous avons ajusté la taille des objets grâce aux réglages sur l'imprimante 3D et nous avons lancé leur production.

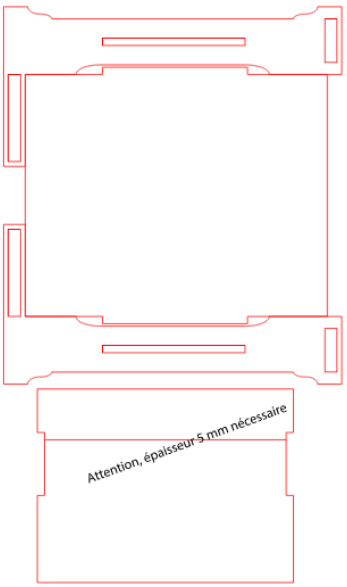
Si quelqu'un veut ajouter des accessoires à notre maison, on trouve énormément de différents objets sur le site https://www.thingiverse.com . L'imprimante 3D permet de matérialiser les plans disponibles sur « Thingiverse », il suffit de modifier la taille avant la production pour rester à l'échelle d'un playmobil.

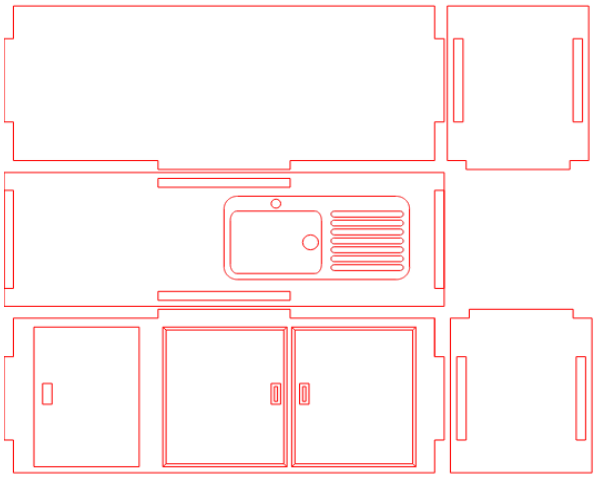
Nous avons ensuite récupérer le plan de notre maison édité sur Makercase et nous l'avons ouvert sur Adobe Illustrator afin d'ajouter n'importe quelle touche personnelle, texte ou photo. Effectivement, la découpeuse laser a aussi la possibilité de graver.
Voici les plans après personnalisation:

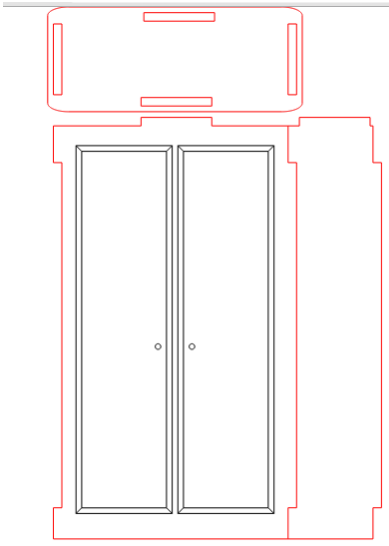


Ainsi nous avons lancé la découpeuse laser pour créer la maison, les meubles et les monter.









/

II) Automatisation de la maison

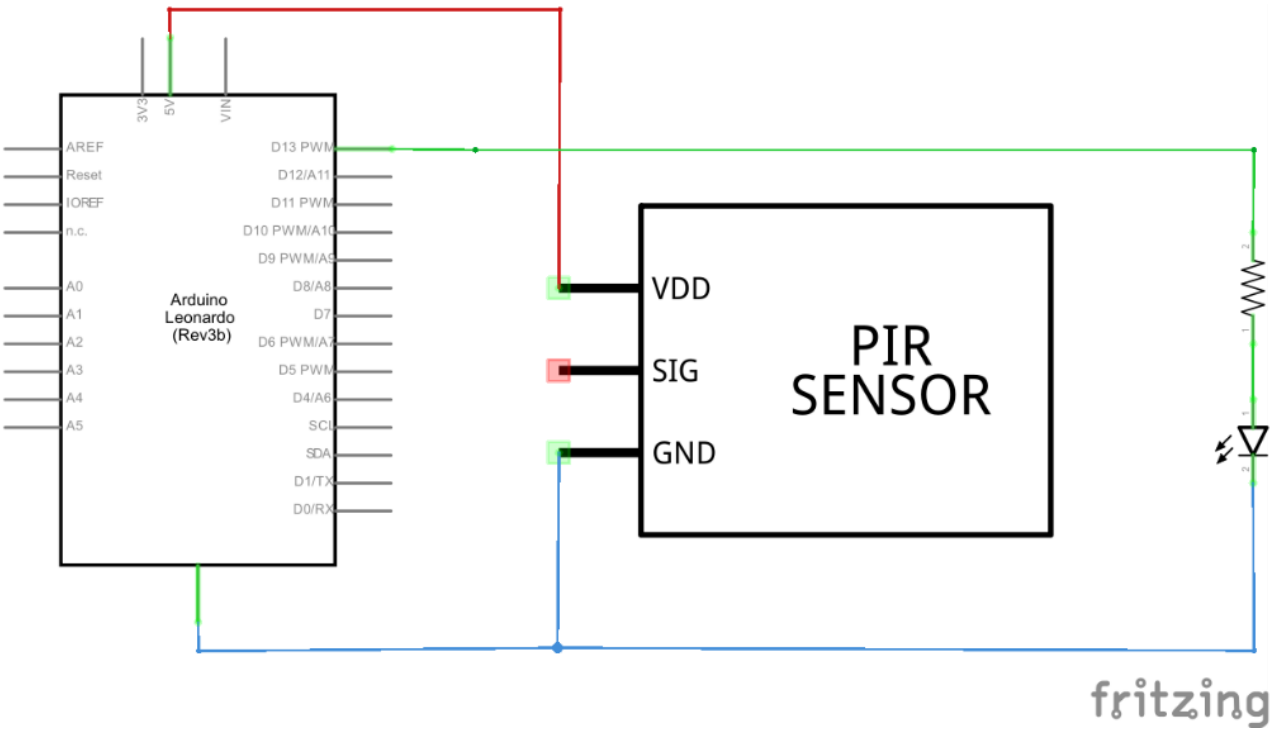
a) Motion Sensor

La gestion de la consommation d'énergie est une tendance actuelle, c'est pourquoi nous avons modélisé un premier scénario qui respecte cette tendance à une échelle réduite. Notre dispositif permet d'allumer automatiquement la lumière en cas de présence dans une pièce mais permet aussi d'éteindre celle-ci en l'absence de mouvement pendant 10 min.

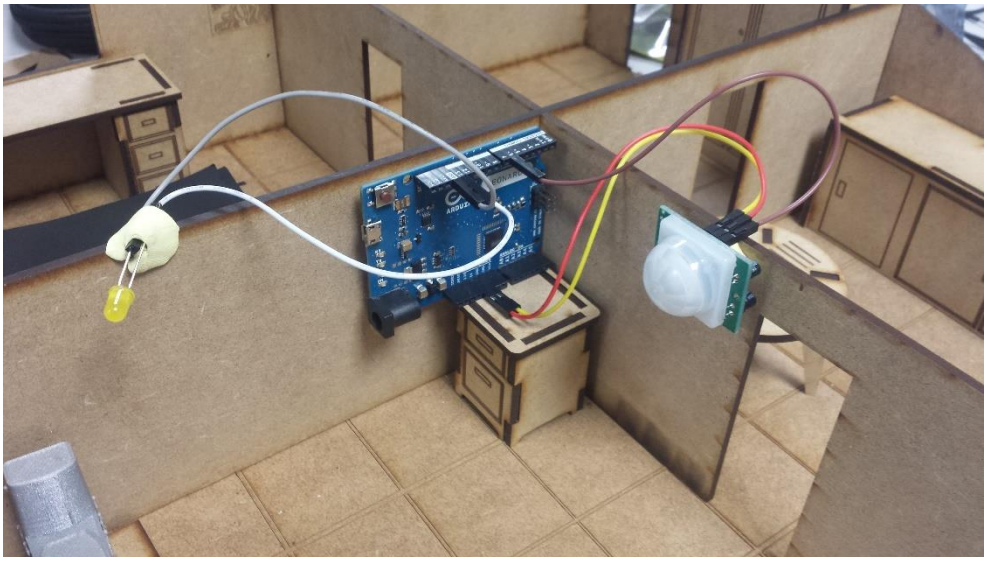
Le matériel que nous avons utilisé est le suivant :

- Un capteur de mouvement
- Une LED (lampe)
- Une résistance 220 ohm
- Une Carte Arduino

Nous avons schématisé les montages électroniques grâce au logiciel Fritzing (http://fritzing.org/home/). Voici le montage du premier scénario « Motion sensor ».



Nous avons incorporé ce scénario dans la maison de cette façon :



Le programme du scénario « Motion Sensor » se trouve en **Annexe**.

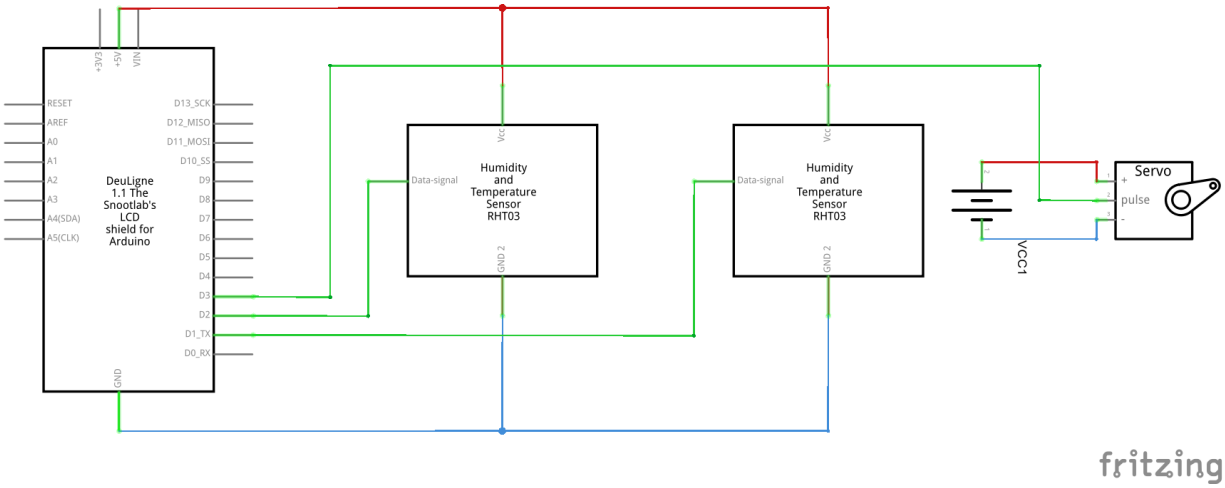
b) Free Cooling

Ce dispositif permet d'améliorer la gestion d'énergie de la maison. En effet la dépense de chauffage ou de climatisation est une part importante du budget énergétique. Ainsi, afin de réduire ce coût, nous avons créé un dispositif prenant en charge l'ouverture et la fermeture des fenêtres en fonction d'une température réglée par l'utilisateur. Il compare la température intérieur et la température extérieure et décide d'ouvrir ou non la fenêtre.

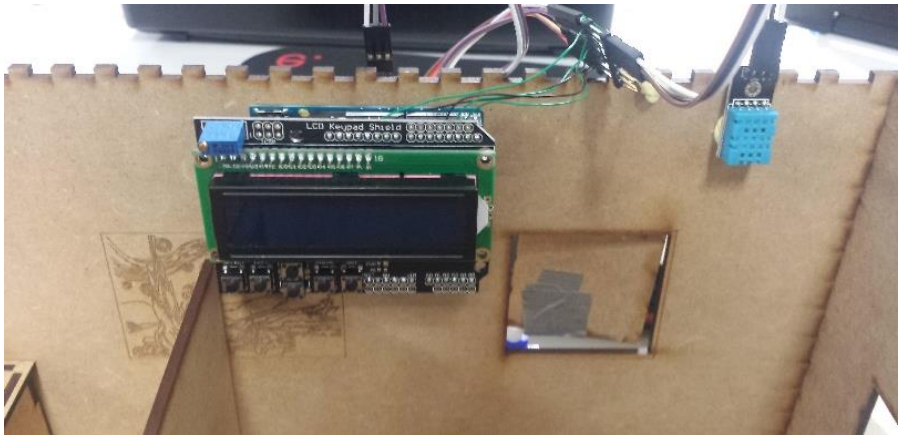
Nous avons utilisé comme matériel :

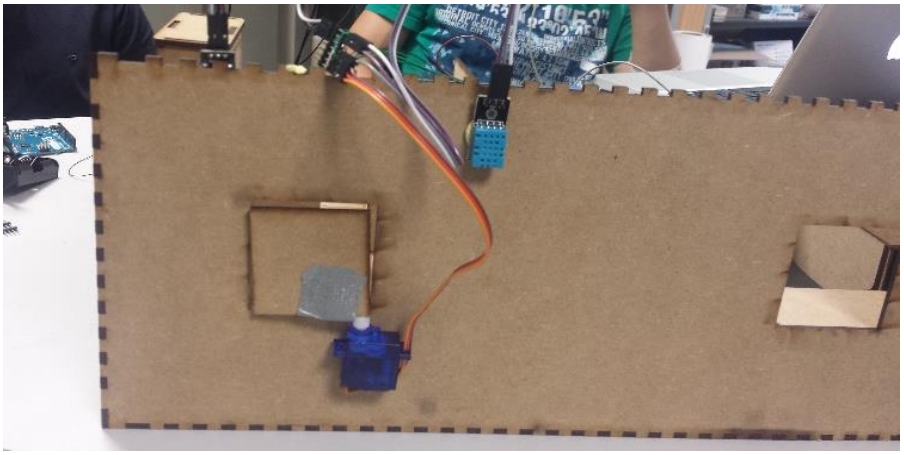
- Deux capteurs DHT 11
- Une shield LSD avec 4 boutons selectors
- Un cerveau moteur
- Des piles ou une batterie
- Une carte arduino

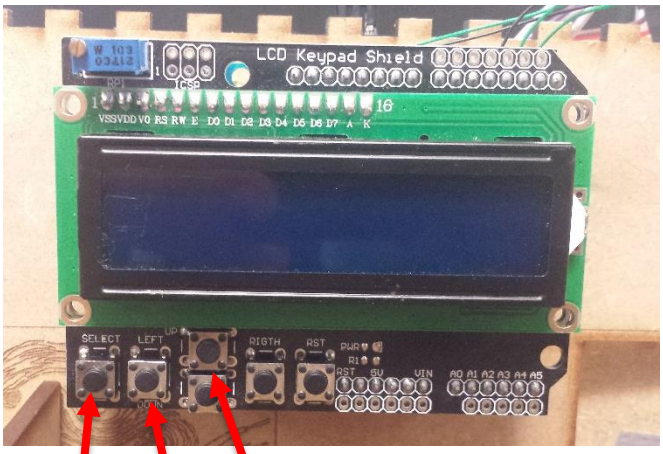
Voici le montage électronique du « Free cooling » sur Fritzing.



Nous avons pris d'autres photos du montage :







Changer la température souhaitée

Ouvrir ou fermer la fenêtre en mode manuel

Passage en mode manuel

Le programme du scénario « Free Cooling » est situé en **Annexe**.

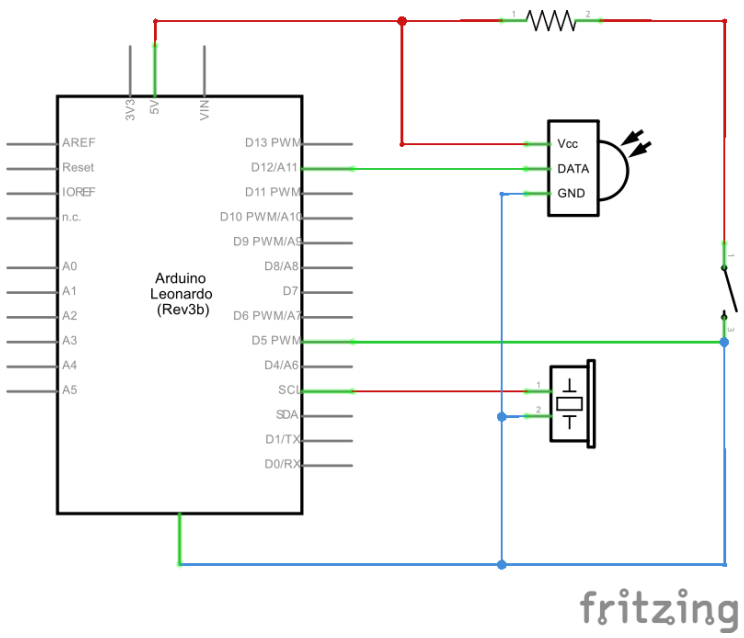
c) Fire Alarm

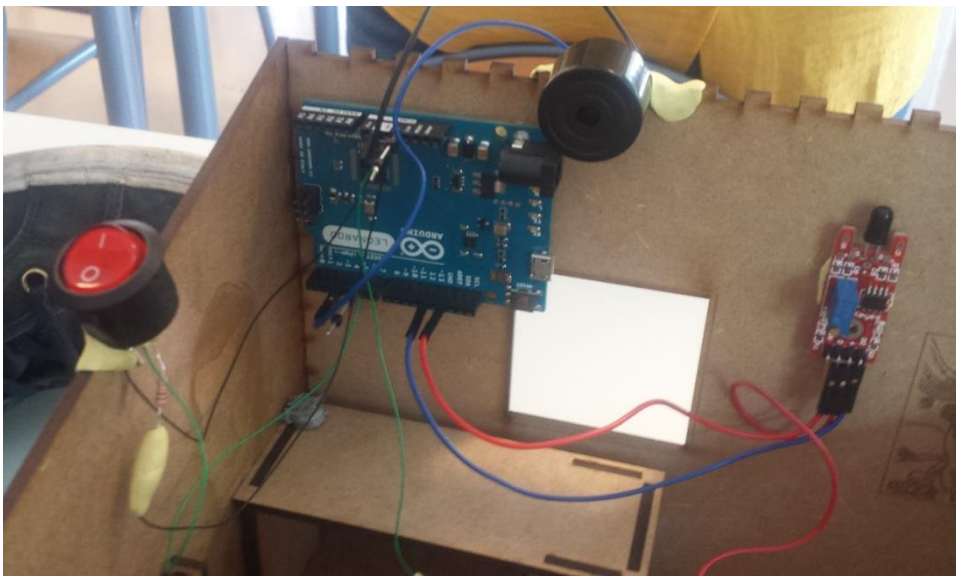
La sécurité est devenue un élément primordial dans le choix d'une maison. Et l'une des plus grande crainte d'accident reste l'incendie. Ainsi nous avons associé différents composants afin de créer un détecteur de flamme. Ce détecteur de flamme déclenche une alarme.

Le matériel utilisé est le suivant :

- Un détecteur de flammes
- Un buzzer
- Une carde arduino
- Un bouton poussoire

Voici le montage électrique de « Fire Alarm » :





L'alarme peut s'enclencher grâce au programme en **Annexe.**

d) Centralisation des commandes : Interface

Nous avons décidé de développer une interface centralisant les différents « module » permettant à la fois de procéder à des ajouts ou des retraits de module ainsi que d'offrir à l'utilisateur (de la maison) une interface afin d'interagir directement et facilement avec les modules.

Pour cela plusieurs étapes ont été nécessaires :
La première, la conception de l'interface niveau hardware, non nécessaire dans le projet initiale seul les sortis Tx Rx et 2 du server ET des arduino était utilisés, Rx serveur était directement branché à toute les sorties Tx des clients et réciproquement et toutes les sorties 2 était reliés.
Le jour de la présentation arrivant nous nous sommes rendu qu'il nous faudrais encore une petite semaine de travail afin de terminer de développer ce protocole de communication et avons décidé de recommencer à « zéros » (pas exactement vrais) et de repartir sur un autre protocole réalisable dans le peu de temps restant. En effet le problème était que lorsqu'une arduino « n'a rien à dire/envoyer » au serveur elle le fait savoir par le maintien d'un niveau haut, provoquant ainsi la perte des informations des signaux des autres systèmes clients. Il aurait fallu modifier le système de communication série au niveau des librairies système ou encore introduire un ordre afin de demander au système de se taire et d'attendre d'être appelés plutôt que de dire qu'elles n'ont rien à dire (comportement natif à l'arduino). Ce premier protocole avait l'avantage de permettre un nombre illimité (théoriquement) de module installer en même temps et d'être léger en pin client. En effet puisque les clients n'ont pas de sorties qui leur sont réservé spécifiquement sur les sorties serveur, le nombre de module n'a donc aucune importance.
La partie plus compliqué de ce premier protocole était la connexion des modules. En effet le système étant designer pour une maison une coupure de courant peut survenir à tout moment, provoquant une reconnexion de tous les systèmes simultanément. La première étape est de réussir à dissocier les clients, tous les clients dispose de la même librairie pas de nom et aucun moyen de le faire savoir au server (tous les clients parleraient en même temps rendant le signale incompréhensible pour le serveur). Nous avons dans premier temps utiliser une librairie de génération d'UN nombre réellement aléatoire (mais pas réellement uniforme) qui ne nécessite pas de connexion série avec un ordinateur, puisque le système natif de génération de nombre est celui du C qui nécessite un nombre généralement un timestamp (qui est plus ou moins aléatoire sur une ordinateur) mais dans notre cas si elle démarre toute en même temps elles auraient toutes le même nombre.

> Une fois nommé (le nombre est assez grand pur être supposé unique pour une centaine de module ce qui est déjà énorme) le but est de les dissocier, puisque les clients n'ont pas connexion privilégier elles ne savent pas au début quand parler et le serveur ne connais pas encore leur nom pour les appeler.

Le nouveau protocole est plus simple à implémenter et a pu être terminé à temps mais limite le nombre de module, chaque module nécessite 2 pins qui lui sont réservés, donc le nombre de module dépends du nombre de pin serveur disponible de plus 1 transistor et 2 diode sont nécessaire par module.
Les communications se déroulent par la suite selon le schéma suivant :

- Le programme envoi une requête : STC<PPP…> (Size de la requête en bytes, Target de la requête, Command byte qui définit le type de requête, Param nombre de byte indéfini qui sont les paramètres de la requête, sa taille est S-3)
- Le serveur la réceptionne sélectionne le module grâce à Target et traduit la requête en : XSC<PPP…> (X est le type de dispositif qui envoi la requête serveur ou module, était utiliser par la première version du deuxième protocole qui présentais un problème d'écho, le serveur recevait ses propres requêtes)
- Le client réagit à la requête et si nécessaire renvoi une requête de retour du même type qu'il a reçu avec paramètre
- Si la requête est à valeur de retour (ex Refresh qui actualise les valeurs de capteurs sur l'interface) les transmet à l'ordinateur par : STC<PPP…>

De plus la requête connexion est à valeur de retour, le module renvoi un tableau identité :

Le paramètre de la valeur de retour se présente ainsi :

<PTVPTV…> (Paquet size, Type de ‘field’, Value du field en caractère)

exemple :81Fermer signifie que le champs fait 8 octet et demande la création d’une action (un bouton qui envoie une requête) nommé Fermer (lors du clique la requête contient le numéro du bouton dépendant de l’ordre de création, le client appel la fonction stocké dans un tableau de fonction)

‘13’ 2Temperature seras un champ d’affichage de la forme “ Temperature : “

Lors de la marche le serveur envoi des « refresh » régulier aux clients afin de mettre à jour les champs d’affichage et lors de l’appuie sur un bouton dans l’interface graphique le programme appel une fonction du client en passant par le serveur. (à noter que si la connexion, la création d’un nouvel onglet par module dans l’interface et la création des fields dans les onglets et l’appel de fonction client grâce aux bouton de l’interface sont fonctionnel. La commande refresh quant à elle présente encore un bug dont nous n’avons pas eu le temps de résoudre avant la date de présentation.).

```
int HardwareSerial::add_to_buffer(uint8_t *c, uint8_t t)
{
   while(_tx_buffer_tail!=(_tx_buffer_head + 1) % SERIAL_TX_BUFFER_SIZE && t)
   {
      _tx_buffer[_tx_buffer_head] = c[0];
      _tx_buffer_head = (tx_buffer_index_t)(_tx_buffer_head + 1) % SERIAL_TX_BUFFER_SIZE;
      c++;
      t--;
   }
   sbi(*_ucsrb, UDRIE0);
   _written = true;
   return t;
}
```

Ce morceau de programme a été ajouté à la librairie native HardwareSerial.cpp pour permettre d'écrire dans une interruption car c'est impossible nativement.

Nous avons utilisé le matériel suivant :

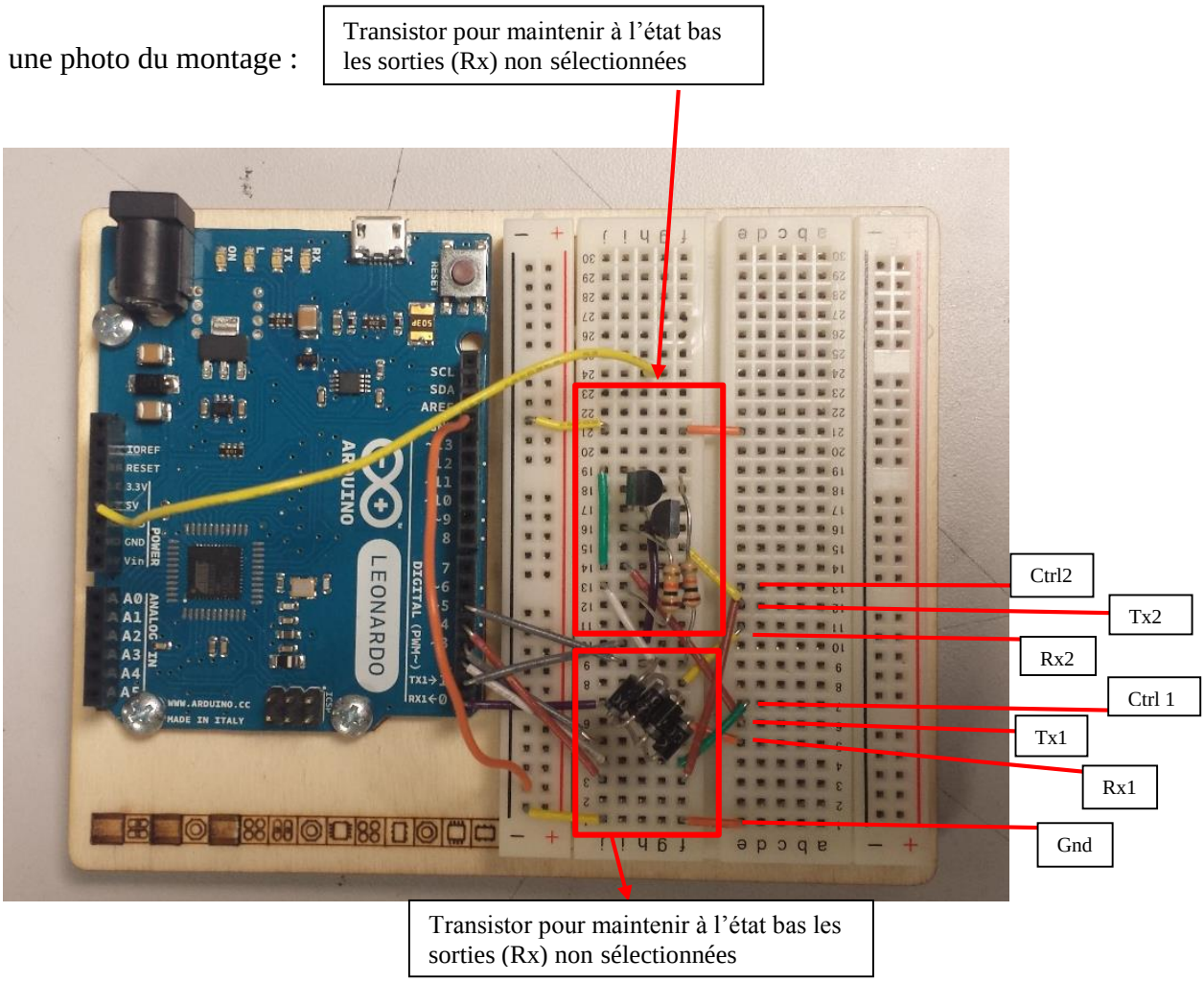
-Carte Arduino

-4 Diodes

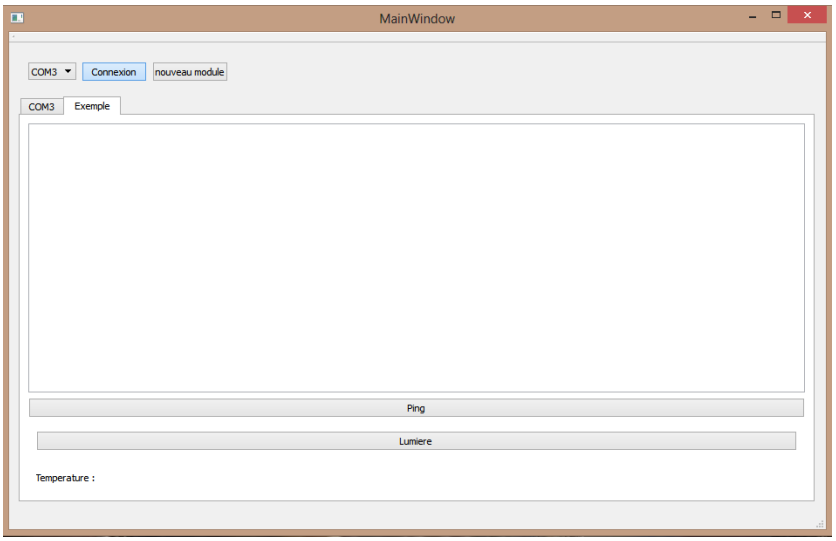
-2 Transistors

-2 Résistances

Voici une photo du montage :



Le programme de l'interface est en **Annexe.**



Interface Graphique

III) Ressenti et Impressions

Nous nous sommes investis dans ce projet de plein gré puisque nous étions satisfaits de la décision de notre tuteur, c'est à dire d'orienter le projet vers une réalisation matérielle plutôt que vers une recherche documentaire au sujet de la domotique.

Le fait d'imaginer, de concevoir et de fabriquer cette maison nécessitent de la prise d'initiative et beaucoup de réflexion. Nous n'avons jamais été réellement confrontés à ce genre de projet avant. Celui-ci nous a permis d'entrevoir le métier d'ingénieur avec le travail et les démarches qui suivent.
Effectivement Didier Donsez nous a proposé son idée et nous étions enjoués et motivé à la réaliser.

Cependant, étant donné notre manque d'expérience dans ce type de projet, nous avons eu beaucoup de difficultés pour savoir par où commencer. Même si notre responsable nous a dicté les tâches à effectuer, nous avons passé énormément de temps à trouver les logiciels et les formats des fichiers adéquats à chaque une de nos étapes de production. De plus, il a fallu comprendre et maîtriser les logiciels que nous utilisions puisqu'ils étaient tous nouveaux pour nous.

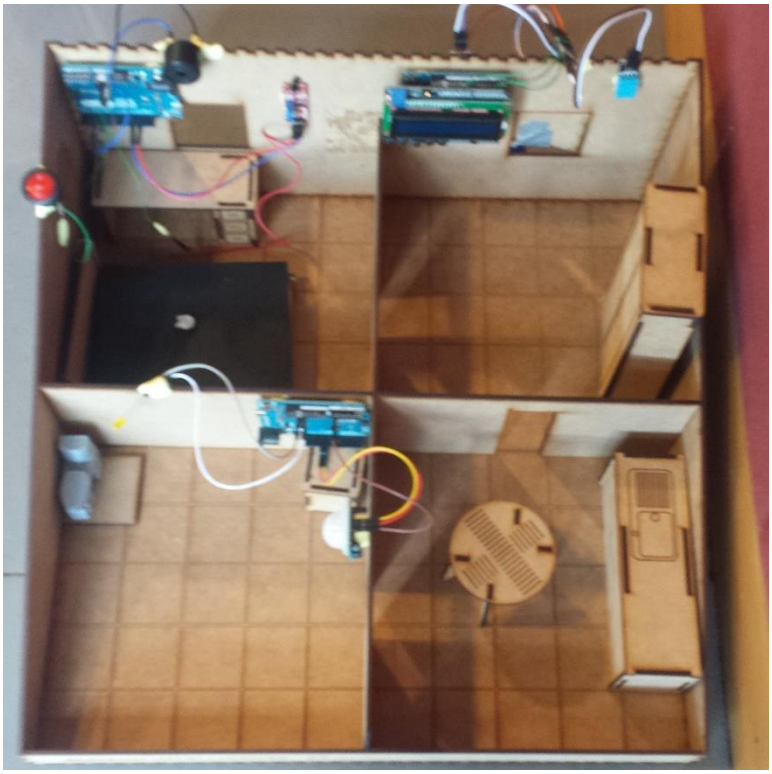
Certes Monsieur Donsez nous a donné beaucoup d'informations élémentaires, mais nous avons été en quelque sorte « lâchés dans la nature ». Ceci nous a réellement permis de nous prendre en main et d'acquérir une maturité essentielle à la gestion de notre projet. En effet, nous avons presque effectué toutes les démarches seuls, en tant « qu'entreprise » chargée de la fabrication d'un nouveau produit. Nous ne sommes absolument pas mécontents de cette façon de procédé, au contraire, cette expérience s'est avérée très enrichissante car nous avons su nous répartir les tâches en fonction des compétences et préférences de chacun pour maximiser notre efficacité.

Cependant, nous avons rencontré beaucoup de difficultés. Il nous a fallu du temps pour récupérer tout le matériel nécessaire, pour comprendre nos erreurs aussi bien au niveau de la programmation, des dessins des plans, ou de l'utilisation des logiciels. Effectivement, il y a beaucoup de subtilités que nous ne connaissions pas et qui nous ont causé des problèmes à plusieurs reprises.

De plus, nous avons dû contacter différentes personnes essentielles à la conception de notre maison. Pour la fabrication de notre maquette, nous avions besoin d'une découpeuse laser et d'une imprimante 3D. Ces types de machines ne sont pas courants, c'est pourquoi Didier Donsez nous a conseillé d'aller au « Fablab » à l'Imag lorsque nous aurons besoin de les utiliser. En revanche, les deux instruments sont tombés en panne, ce qui a considérablement retardé la réalisation de la maison et des meubles qu'elle était censé comporter. Un des dirigeants du « Fablab », Monsieur Maisonnasse nous a envoyé à l' Inria (Montbonnot) pour utiliser leur découpeuse laser. Nous nous sommes rendu sur le lieu de rendez-vous et avons rencontré Monsieur Pinssant, qui nous a consacré une après-midi pour nous aider à la construction de notre maison. Il nous a expliqué comment fonctionne la découpeuse laser et comment arranger et améliorer nos plans.
M. Maisonasse nous a également aidés lors des impressions 3D et surtout en nous donnant son envie à chaque étape que nous entreprenions (lorsqu'il était présent).
Ceci a couté beaucoup de temps à ces deux personnes qui n'étaient en soit pas responsable de nous, ni de notre projet.

La conception de notre maison « intelligente » nous a donné un bon aperçu du métier d'ingénieur. Nous avons appréhendé la démarche, c'est à dire imaginer un produit, réfléchir aux différentes étapes de fabrication, penser au matériel nécessaire, etc... sans oublier les imprévus auxquels nous ne pensons pas et qu'il est impératif de résoudre.

Nous avons acquis une certaine assurance au cours de nos travaux, tant au niveau de nos capacités que de notre adaptabilité. Nous avons su communiquer entre nous et rester concentrés sur notre objectif en essayant de respecter le délai imparti.



Conclusion :

Didier Donsez et Polytech nous ont offert l'opportunité de réaliser notre premier propre produit. Nous avons fabriqué une maison dite « intelligente ». Effectivement, dans le cadre du thème de la domotique, nous avons conçu une maison automatisée à l'échelle d'un playmobil. Elle est capable de gérer l'éclairage en fonction du déplacement des « playmobil », de prévenir en cas d'incendie et de gérer la climatisation de façon écologique. Cependant, les fonctionnalités ne se restreignent pas à celles-ci et d'autres peuvent être ajoutées grâce à un système de centralisation.

A travers le projet nous avons appris à nous connaître plus en profondeur. Nous avons pour la première fois réalisés quelque chose de congrès, un objet. Grâce à cette maquette, nous avons découvert le monde de la pratique. Jusqu'à maintenant nous n'avons étudié que la théorie, et il s'avère qu'il s'agit bien de deux mondes différents.

Nous avons pu constater que certains d'entre nous sont plus dédiés à la réflexion qu'à la pratique et vice versa. Chacun a trouvé sa préférence dans les domaines rencontrés : la programmation, l'électronique, la conception de la maison (design, plans, mobiliers, utilisation de la découpeuse laser et de l'imprimante 3D).

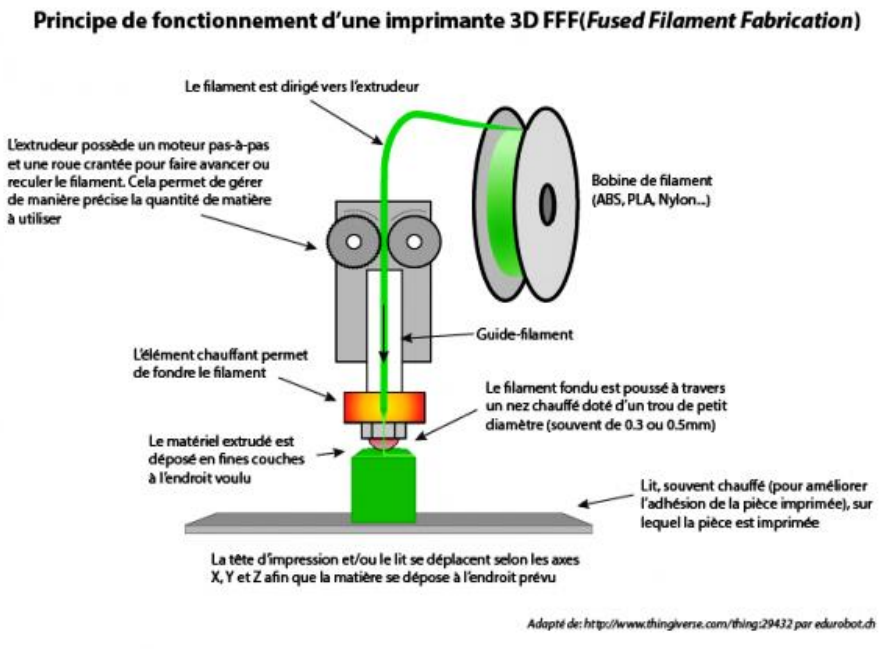
En effet, fabriquer cette maison a nécessité des compétences dans plusieurs secteurs d'activités : l'informatique, l'électronique, les matériaux, le génie civil, même si cela est à une moindre échelle.

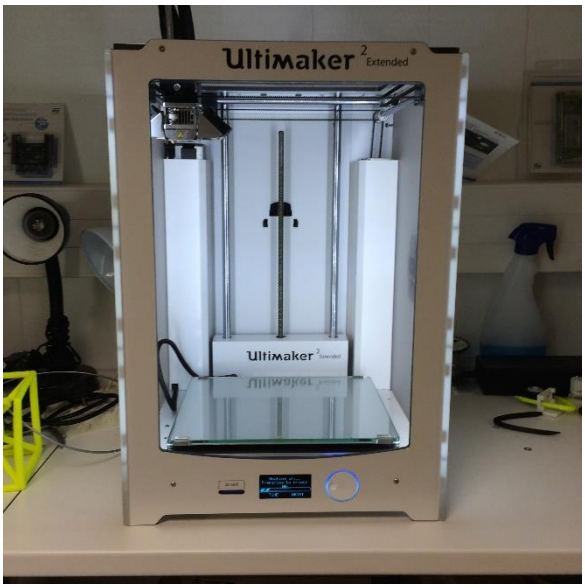
Chacun de nous a su trouver son compte. Nous sommes 4 étudiants provenant du parcours Physique-Mathématiques-Mécaniques, et un étudiant venant de Mathématiques-Informatiques. Nous voulons tous partir dans des domaines différents l'année prochaine et ceci s'est clairement manifesté via notre dévouement et nos compétences personnelles que nous avons mis en pratique afin d'optimiser notre travail d'équipe. Nous avons également renforcé notre personnalité en donnant chacun notre avis et en discutant lorsque nous n'étions pas d'accord.

Ce projet a été vivant, entraînant et motivant pour la suite de nos études. Nous pensons avoir entraperçu une partie de notre future vie active.

Annexes et Programmes

Imprimante 3D



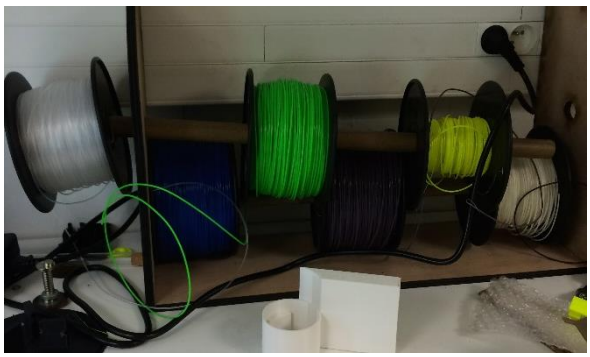


Imprimante 3D Ultimaker 2 Extended

Imprimante 3D Ultimaker 2 Extended

L'impression 3D consiste à superposer des couches de matières avec une imprimante 3D selon des coordonnées transmises par un fichier 3D. Il existe des dizaines de manières permettant d'imprimer ces objets, tels que l'impression par dépôt de matière (FDM ou FFF) dont nous allons parler, la solidification par lumière (stéréolithographie ou SLA), la Polyjet ou le frittage laser. Pour réaliser une impression 3D, l'utilisateur aura besoin d'une imprimante 3D, d'un consommable (filament, poudre …), d'un fichier 3D (format STL ou OBJ), d'un logiciel permettant de transmettre les indications à l'imprimante, et d'un ordinateur. Différentes manières d'exporter les fichiers vers l'imprimante existent : câble USB, Wi-Fi ou carte SD (dans notre cas, nous avons utilisé une carte SD).

Le modèle de l'imprimante que nous avons utilisé (disponible au Fab Lab) est un Ultimaker 2 Extended. Ce modèle fonctionne selon le principe de FDM (acronyme anglais de Fused Deposition Modeling, ou modelage par dépôt de filament en fusion en français). Cette méthode consiste à déposer couche par couche un filament de matière thermoplastique fondu à 200°C. En se superposant, ces couches donnent forme à l'objet. La tête d'impression se déplace selon les coordonnées X, Y, Z transmises par le fichier 3D. L'impression 3D voit maintenant arriver de nouveaux firmaments composites à base de métal (cuivre, bronze…) et même de bois.



Matériaux utilisé pour l'imprimante 3D

Découpeuse Laser

Le découpage au laser est un procédé thermique sans contact qui sert à séparer les matériaux. Le laser est focalisé par une lentille, au travers d'une buse dans laquelle arrive un gaz d'assistance coaxial au faisceau.

Le laser peut réaliser divers travaux de découpe, de la coupe précise au micromètre dans des puces semi-conductrices à une coupe dans une tôle d'acier de 30 millimètres d'épaisseur.

Là où le faisceau laser focalisé rencontre la pièce à usiner, il chauffe le matériau extrêmement fortement, si bien que ce dernier fond ou se vaporise. Une fois que la pièce à usiner est bien infiltrée, la découpe commence. Le faisceau laser se déplace le long des contours et fond le matériau en continu. Souvent, un flux de gaz souffle le produit de la fusion vers le bas, hors de la fente de coupe, qui est à peine plus large que le faisceau laser focalisé.

Pour le perçage, une courte impulsion laser avec une grande puissance volumique fait fondre et se vaporiser le matériau.

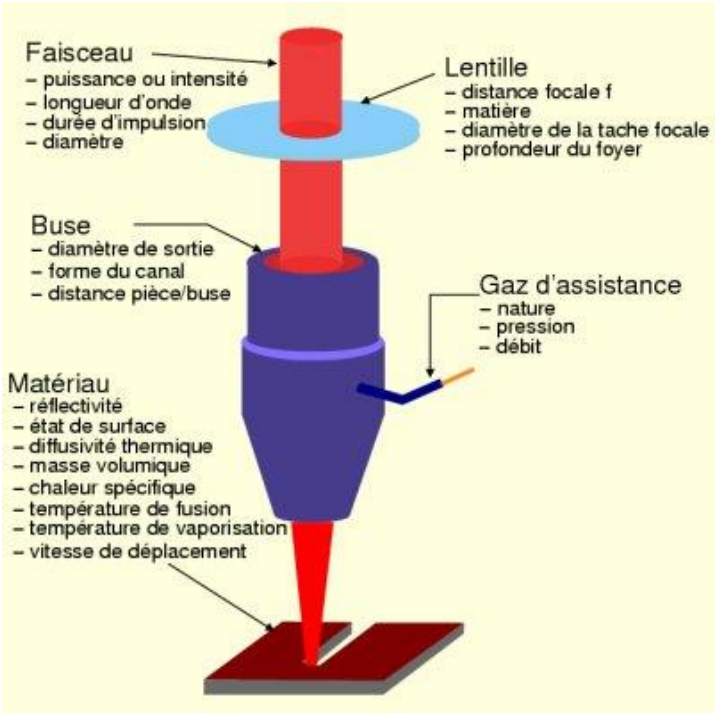


Schéma découpeuse laser

Découpeuse Laser du Fablab

La découpeuse laser peut aussi être utilisée en vue d'effectuer des gravures. Nous en avons réalisé plusieurs pour notre projet (sur les murs, les meubles et le sol). De plus, nous avons pu observer le résultat de la gravure sur du plexiglas, qui est très impressionnant, et pourrait être comparable à une photo.

Programme Motion Sensor

```
////////////////////////////
//VARS

int is_active = false;
int is_move = false;
int count_down = 0;
//the time we give the sensor to calibrate (10-60 secs according to the datasheet)
int calibrationTime = 30;

//the time when the sensor outputs a low impulse
long unsigned int lowIn;

//the amount of milliseconds the sensor has to be low
//before we assume all motion has stopped
long unsigned int pause = 1000;

boolean lockLow = true;
boolean takeLowTime;

int pirPin = 6;    //the digital pin connected to the PIR sensor's output
int ledPin = 13;


////////////////////////////
//SETUP
void setup(){
  Serial.begin(9600);
  pinMode(pirPin, INPUT);
  pinMode(ledPin, OUTPUT);
  digitalWrite(pirPin, LOW);

  //give the sensor some time to calibrate
  Serial.print("calibrating sensor ");
    for(int i = 0; i < calibrationTime; i++){
      Serial.print(".");
      delay(1000);
      }
    Serial.println(" done");
    Serial.println("SENSOR ACTIVE");
    delay(50);
  }

////////////////////////////
//LOOP
void loop(){

    if(digitalRead(pirPin) == HIGH) is_move = true;

    if(count_down > 0 && is_active)
    {
      count_down--;
      delay(50);
    }
    else
    {
      count_down = 200;
      digitalWrite(ledPin, is_move);
      is_active = is_move;
      Serial.print(is_move);
      Serial.print("\n");
      is_move = false;
    }
```

Programme Free Cooling

```
#include <LiquidCrystal.h>
#include <DHT.h>
#include <Servo.h>

#define DHTTYPE DHT11 //Si vous utiliser le DHT 11
#define p_int 1
#define p_ext 2
int t_int = 0;
int t_ext = 0;
int t_cible = 0;
unsigned long data_time = 0;
boolean manuel = 0;
boolean ouvert = 0;
unsigned long temps = 0;

// select the pins used on the LCD panel
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
DHT d_int(p_int, DHTTYPE); //On initialise le capteur DHT
DHT d_ext(p_ext, DHTTYPE); //On initialise le capteur DHT
Servo fen;

// define some values used by the panel and buttons
int lcd_key     = 0;
int adc_key_in  = 0;
#define btnRIGHT  0
#define btnUP     1
#define btnDOWN   2
#define btnLEFT   3
#define btnSELECT 4
#define btnNONE   5

// read the buttons
int read_LCD_buttons()
{
 adc_key_in = analogRead(0);      // read the value from the sensor
 // my buttons when read are centered at these valies: 0, 144, 329, 504, 741
 // we add approx 50 to those values and check to see if we are close
 if (adc_key_in > 1000) return btnNONE; // We make this the 1st option for speed
reasons since it will be the most likely result
 // For V1.1 us this threshold
 if (adc_key_in < 50)   return btnRIGHT;
 if (adc_key_in < 250)  return btnUP;
 if (adc_key_in < 450)  return btnDOWN;
 if (adc_key_in < 650)  return btnLEFT;
 if (adc_key_in < 850)  return btnSELECT;

 // For V1.0 comment the other threshold and use the one below:
/*
 if (adc_key_in < 50)   return btnRIGHT;
 if (adc_key_in < 195)  return btnUP;
 if (adc_key_in < 380)  return btnDOWN;
 if (adc_key_in < 555)  return btnLEFT;
 if (adc_key_in < 790)  return btnSELECT;
*/


 return btnNONE;  // when all others fail, return this...
}
```

```
void setup()
{
  fen.attach(3);
  fen.write(5);
 lcd.begin(16, 2);
 lcd.setCursor(0,0);
 //lcd.print("Welcome");
}

void loop()
{
  int ttemp = d_int.readTemperature(); //fltHumidity = dht.readHumidity();
  if(!isnan(ttemp)) t_int = ttemp;
  ttemp = d_ext.readTemperature(); //fltHumidity = dht.readHumidity();
  if(!isnan(ttemp)) t_ext = ttemp;

  lcd_key = read_LCD_buttons();  // read the buttons

  if(lcd_key == btnSELECT) manuel = 1 - manuel;
  if(manuel)
  {
    if(lcd_key == btnLEFT) ouvert = 1 - ouvert;
    lcd.setCursor(0,0);
    lcd.print("Commande manuel ");

    lcd.setCursor(0,1);
    if(ouvert)
      {lcd.print("          Ouvert"); fen.write(90);}
    else
      {lcd.print("          Fermer"); fen.write(5);}
  }
  else
  {
    if(lcd_key == btnUP)
    {
      t_cible++;
      temps = millis();
    }
    if(lcd_key == btnDOWN)
    {
      t_cible--;
      temps = millis();
    }

    if(temps<millis()-2000)
    {
      if(t_cible<t_int && t_ext<t_int) fen.write(90);
      else if(t_cible>t_int && t_ext>t_int) fen.write(90);
      else  fen.write(5);


      lcd.setCursor(0,0);
      lcd.print("Int. : ");
      if(t_int<10) lcd.print("0");
      lcd.print(t_int);
      lcd.print(" C     ");

      lcd.setCursor(0,1);
      lcd.print("Ext. : ");
      if(t_ext<10)
        lcd.print("0");
      lcd.print(t_ext);
      lcd.print(" C     ");
    }
```

```
    else
    {
      lcd.setCursor(0,0);
      lcd.print("Temp. cible :   ");
      lcd.setCursor(0,1);
      lcd.print("           ");
      if(t_cible<10)
        lcd.print("0");
      lcd.print(t_cible);
      lcd.print(" C ");
    }
  }
  if(lcd_key != btnNONE)
    delay(300);
  else
    delay(10);
 /*lcd.setCursor(9,1);            // move cursor to second line "1" and 9 spaces over
 lcd.print(millis()/1000);      */// display seconds elapsed since power-up

/*
 lcd.setCursor(0,1);            // move to the begining of the second line

 switch (lcd_key)               // depending on which button was pushed, we perform
an action
 {
   case btnRIGHT:
     {
     lcd.print("RIGHT ");
     break;
     }
   case btnLEFT:
     {
     lcd.print("LEFT   ");
     break;
     }
   case btnUP:
     {
     lcd.print("UP    ");
     break;
     }
   case btnDOWN:
     {
     lcd.print("DOWN  ");
     break;
     }
   case btnSELECT:
     {
     lcd.print("SELECT");
     break;
     }
     case btnNONE:
     {
     lcd.print("NONE  ");
     break;
     }
 }*/

}
}
}
```

Programme Fire Alarm

```
#define p_buz 3
#define p_sensor 12

long ring = 0;
int last = 0;

void setup() {Serial.begin(9600);
  pinMode(p_buz, OUTPUT);
  pinMode(p_sensor, INPUT);
  pinMode(5, INPUT);
  last = digitalRead(5);
}

void loop() {
  if(digitalRead(p_sensor))
  {
    tone(p_buz, 5, 10000);
  }
  if(digitalRead(5)!=last)
  {
    last = digitalRead(5);
    noTone(p_buz);
  }
}
```

Programme de l'Interface

Server

```
#define PROT_PING 11
#define PROT_SET_MOD 10
#define PROT_FRESH_CMD 19

uint8_t nb_m = 0;
uint8_t curr_nb_m = 1;
long last_fresh = 0;


uint8_t PROT_buffer[64];


void setup() {
  Serial.begin(9600);
  Serial1.begin(9600);
  Serial1.flush();
  for (int i = 2; i < 10; i++)
    pinMode(i, OUTPUT);

  for (int i = 1; i < 5; i++)
    digitalWrite(2 * i + 1, HIGH);
}

void serialEventRun()
{
  if (!Serial.available())
    return;
  PROT_getbytesp();

  // SXTCCCC...  S:size X:command T:target C:param
  uint8_t PROT_S = PROT_buffer[0];
  uint8_t PROT_X = PROT_buffer[1];
  uint8_t PROT_T = PROT_buffer[2];

  PROT_exec(PROT_S, PROT_X, PROT_T);
}


uint8_t PROT_obuffer[] = {55, 3, 10};
/*void loop() {
  Serial1.write("abc");
delay(500);
  while(Serial1.available())Serial.write(Serial1.read());
}*/
void loop()
{/*
  if (nb_m)
  {
    if (!curr_nb_m && last_fresh < millis() + 1000)
      delay(300); // Pour ne pas saturer les clients par des refresh
    refresh(curr_nb_m);
    curr_nb_m++;
    if (curr_nb_m > nb_m)
```

```
    {
      curr_nb_m = 1;
      last_fresh = millis();
    }
  }
  else
    delay(100);
    */
    delay(1000);
}

void refresh(uint8_t target)
{ /*uint8_t temp[] = {'a','b'};

  PROT_prepare(19, 5, target, temp);
  PROT_transmit();*/
  //Serial.write(temp,temp[0]);/*
   PROT_open(target);
   PROT_request(55, 19, 3, 0);
   PROT_send(target);
   PROT_getbytes1p();
   PROT_close(target);
   PROT_prepare(PROT_buffer[2], PROT_buffer[1], target, PROT_buffer + 3);
   Serial.write(PROT_buffer,PROT_buffer[1]);//PROT_transmit();
}

void PROT_set_mod(char target)
{
  nb_m = target;
  PROT_open(target);
  Serial1.add_to_buffer(PROT_obuffer, 3);
  PROT_send(target);
  PROT_getbytes1p();
  PROT_close(target);
  PROT_prepare(PROT_buffer[2], PROT_buffer[1], target, PROT_buffer + 3);
  PROT_transmit();
}

void PROT_ping(char target)
{
  PROT_open(target);
  //Serial1.add_to_buffer(PROT_obuffer, 3);
  PROT_request(55, 11, 3, 0);
  uint16_t m = millis();
  PROT_send(target);
  PROT_getbytes1p();
  m -= millis();
  m = -m;
  uint8_t temp[2] = {m / 256, m % 256};
  PROT_close(target);
  PROT_prepare(11, 5, target, temp);
  PROT_transmit();
}

void PROT_exec(uint8_t csize, uint8_t command, uint8_t target)
{
  switch (command) {
    case PROT_PING:
      PROT_ping(target);
```

```
      break;
    case PROT_SET_MOD:
      PROT_set_mod(target);
      break;
    case 27:
      PROT_act(target);
      break;
      /*default:
        Serial.print("Euc"); // Error unknow command
        Serial.print(command);*/
  }
}

void PROT_act(char target)
{
  PROT_open(target);
   PROT_request(55, 27, 4, PROT_buffer+3);
   PROT_send(target);
   PROT_close(target);
}


void PROT_getbytesp()
{
  uint8_t i = 1;
  while (!Serial.available())delay(5);
  PROT_buffer[0] = Serial.read();
  while (PROT_buffer[0] > i)
  {
    if (Serial.available())
    {
      PROT_buffer[i] = Serial.read();
      i++;
    }
    else
      delay(1);
  }
}


void PROT_open(char target)
{
  digitalWrite(target*2 + 1,LOW);
  delay(100);
}
void PROT_close(char target)
{
  delay(100);
  digitalWrite(target * 2 + 1, HIGH);
}

void PROT_send(char target)
{
  Serial1.flush();
  delay(100);
  digitalWrite(target*2,LOW);
  digitalWrite(target*2,HIGH);
  delay(100);
}
```

```
void PROT_getbytes1p()
{ //19 26
  bool b = true;
  while (Serial1.peek() != uint8_t(56))
    if (Serial1.available())
      PROT_buffer[0] = Serial1.read();
  //while(!Serial.find(&t,1))delay(50);
  while (b)
  {
    uint8_t i = 0;
    b = false;
    while (i < 3)
      if (Serial1.available())
      {
        PROT_buffer[i] = Serial1.read();
        i++;
      }
      else
        delay(10);
    while (PROT_buffer[1] > i)
    {
      if (Serial1.available())
      {
        PROT_buffer[i] = Serial1.read();
        i++;
      }
      else
        delay(1);
    }
    //if(PROT_buffer[0] != 55) b=true;
  }
}

void PROT_prepare(const uint8_t command, const uint8_t c_size, const uint8_t
target, const uint8_t *param)
{
  uint8_t i;
  PROT_buffer[0] = c_size;
  PROT_buffer[1] = command;
  PROT_buffer[2] = target;
  for (i = 3; i < c_size; i++)
    PROT_buffer[i] = param[i - 3];
}

void PROT_transmit()
{
  Serial.write(PROT_buffer, PROT_buffer[0]);
}

void PROT_request(const uint8_t sender, const uint8_t command, const uint8_t
c_size, const uint8_t *param)
{
  Serial1.write(sender);
  Serial1.write(c_size);
  Serial1.write(command);
  Serial1.write(param, c_size - 2);
}
```

Client Lib

```
#define Q(x) #x
#define QUOTE(x) Q(x)

#define FILE_ARG truck
//#include QUOTE(FILE_ARG)

/*#define PROT_pr0(sn1,nn1) "#sn1#nn1"
#define PROT_pr1(sn1,nn1,sn2,nn2) "#sn1#nn1#sn2#nn2"
#define PROT_pr2(sn1,nn1,sn2,nn2,sn3,nn3) "#sn1#nn1#sn2#nn2#sn3#nn3"*/
#define PROT_pr3(sn1,nn1,sn2,nn2,sn3,nn3,sn4,nn4) Q(sn1) nn1 Q(sn2) nn2
Q(sn3) nn3 Q(sn4) nn4

#define PROT_NOM Q(Nom_module)
#define PROT_NOM_SIZE 10

#ifndef PROT_PR1_N
  #define PROT_PR1_N
  #define PROT_PR1_S 0
#endif
#ifndef PROT_PR2_N
  #define PROT_PR2_N
  #define PROT_PR2_S 0
#endif
#ifndef PROT_PR3_N
  #define PROT_PR3_N
  #define PROT_PR3_S 0
#endif

uint8_t* PROT_vals[3];
uint8_t PROT_vals_size[] = {0,0,0};
//(void (*ptr)())[3];
//void* PROT_fct[3];
void (*PROT_fct[3])();



#define PROT_PING 11
#define PROT_PRES 10
#define PROT_FRESH_CMD 19

uint8_t testo[] = "24Â°C";



void PROT_init() {
  Serial1.begin(9600);
  pinMode(13, OUTPUT);
  PROT_vals[0] = testo;
  PROT_vals_size[0] = 4;
  attachInterrupt(1, PROT_call, RISING);
}

void PROT_call()
{
  uint8_t PROT_buffer[32];
  //STXCCCC... S:sender T:size X:command C: param
  if (!PROT_getbytes1p(PROT_buffer))
```

```
  {
    PROT_prepare(56, 100, 3, 0, PROT_buffer); //,&head
    Serial1.add_to_buffer(PROT_buffer, PROT_buffer[1]);
    return;
  }

  uint8_t PROT_S = PROT_buffer[0];
  uint8_t PROT_T = PROT_buffer[1];
  uint8_t PROT_X = PROT_buffer[2];


  PROT_exec(PROT_X, PROT_T, PROT_buffer);
}

void PROT_exec(char command, char csize, uint8_t *PROT_buffer)
{
  switch (command) {
    case PROT_PING: PROT_pres(PROT_buffer);
      PROT_ping(PROT_buffer);
      break;
    case PROT_PRES:
      PROT_pres(PROT_buffer);
      break;
    case PROT_FRESH_CMD:
      PROT_fresh(PROT_buffer);
      break;
      case 27:
      PROT_act(PROT_buffer);
      break;
      /*default:
        char cmd[] = " Euc";
        cmd[0] = command;
        Serial1.add_to_buffer(cmd, 4); // Error unknow command*/
  }
}

void PROT_act(uint8_t *PROT_buffer)
{
  //(PROT_fct[PROT_buffer[3]])();
  //PROT_fct = &PROT_act;
  (PROT_fct[PROT_buffer[3]])();
}

void PROT_ping(uint8_t *PROT_buffer)
{
  PROT_prepare(56, 11, 3, 0, PROT_buffer);
  Serial1.add_to_buffer(PROT_buffer, PROT_buffer[1]);
}

void PROT_pres(uint8_t *PROT_buffer)
{

  uint8_t temp[] =
PROT_pr3(PROT_NOM_SIZE,PROT_NOM,PROT_PR1_S,PROT_PR1_N,PROT_PR2_S,PROT_PR2_N,P
ROT_PR3_S,PROT_PR3_N);

  PROT_prepare(56, 10, PROT_NOM_SIZE + 3 + PROT_PR2_S + PROT_PR3_S +
PROT_PR1_S + 3, temp, PROT_buffer); //,&head
  Serial1.add_to_buffer(PROT_buffer, PROT_buffer[1]);
```

```
}

#define PROT_FRESH "24.5Â°C"
#define PROT_FRESH_SIZE 6

void PROT_fresh(uint8_t *PROT_buffer)
{
  uint8_t temp[] = "azert";

  PROT_prepare(56, 10, 8, temp, PROT_buffer); //,&head
  Serial1.add_to_buffer(PROT_buffer, PROT_buffer[1]);
  /*int i = 0;
  PROT_buffer[0] = 56;
  PROT_buffer[1] = 3 + PROT_vals_size[0] + PROT_vals_size[1] +
PROT_vals_size[2];
  PROT_buffer[2] = 19;
  Serial1.add_to_buffer(PROT_buffer, 3);
  while(i<3 && PROT_vals_size[i])
  {
    Serial1.add_to_buffer(PROT_vals[i], PROT_vals_size[i]);
    i++;
  }
  */
}

bool PROT_getbytes1p(uint8_t *PROT_buffer)
{
  uint8_t i = 0;
  while (i < 3)
    if (Serial1.available())
    {
      PROT_buffer[i] = Serial1.read();
      i++;
    }
    else
      return false;
  while (PROT_buffer[1] > i)
  {
    if (Serial1.available())
    {
      PROT_buffer[i] = Serial1.read();
      i++;
    }
    else
      return false;
  }
  return true;
}

void PROT_prepare(const uint8_t sender, const uint8_t command, const uint8_t
c_size, const uint8_t *param, uint8_t *PROT_buffer)
{
  uint8_t i;
  PROT_buffer[0] = sender;
  PROT_buffer[1] = c_size;
  PROT_buffer[2] = command;
  for (i = 3; i < c_size; i++)
    PROT_buffer[i] = param[i - 3];
}
```

Exemple

```
#define PROT_NOM Q(Nom_module)
#define PROT_NOM_SIZE 10

#define PROT_PR1_N Q(light)
#define PROT_PR1_S 5

#define PROT_PR2_N Q(exbutton)
#define PROT_PR2_S 8

void ex_act()
{
  digitalWrite(13,HIGH);
}

void setup()
{
      pinMode(13, OUTPUT);
      digitalWrite(13,LOW);
      PROT_init();
         PROT_fct[0] = & ex_act;
}

void loop()
{
  delay(1000);
}
```

Bibliographie

http://www.makercase.com/

http://air.imag.fr/index.php/Main_Page

http://air.imag.fr/index.php/Arduino

http://air.imag.fr/index.php/PIR_Motion_Sensor

http://air.imag.fr/index.php/DHT11/DHT21/DHT22_etc._Temperature_%26_Humidity_sensors

http://air.imag.fr/index.php/135038_Arduino_Flame_Detection_Sensor_Module

http://air.imag.fr/index.php/Gas_Sensors

http://air.imag.fr/index.php/HC-SR04_Ultrasonic_Sensor_Distance_Measuring_Module

http://air.imag.fr/index.php/NFC_Shield_for_Arduino

http://air.imag.fr/index.php/NRF24L01_2.4GHz_Wireless_Arduino_Transceiver

http://air.imag.fr/index.php/ESP8266

http://fablab.ensimag.fr/index.php/PILBI-2013-Team2

https://github.com

http://www.arduino.cc/

www.adobe.com/fr/products/illustrator.html

https://inkscape.org/fr/

https://www.blender.org

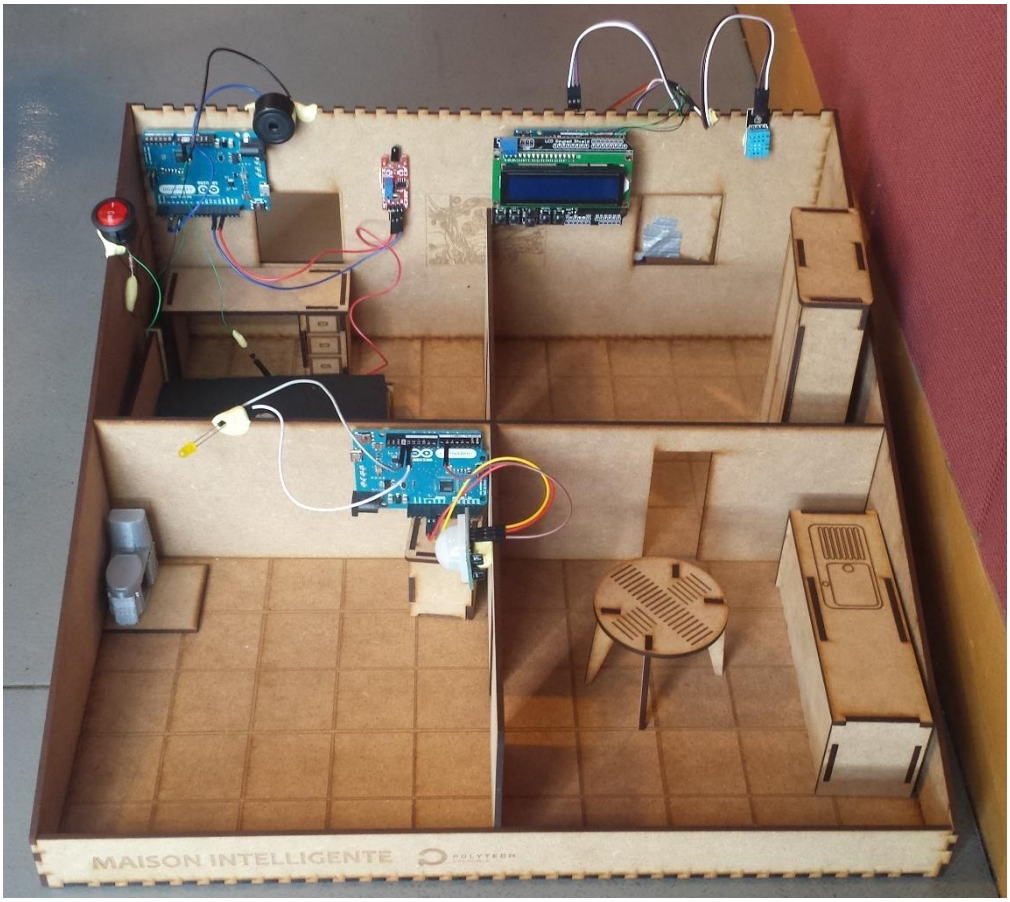
www.sketchup.com/fr

Remerciements

Nous remercions Monsieur Didier Donsez pour avoir été à l'origine de ce projet ainsi que pour l'aide et les conseils qu'il nous a apporté tout au long de celui-ci.

Nous tenons à remercier également, toute l'équipe du Fablab, et en particulier Jérome Maisonasse et Amr Alzouhri-Alyafi, pour avoir répondu à toutes nos questions et pour nous avoir encadrés lors de l'utilisation des différentes machines.

Pour finir nous remercions Monsieur Rémi Pinssant pour nous avoir accordé du temps à la réalisation de notre projet.



Etudiants : ***DU PEIP seconde année***

Decelle Lucas

Garin Lucas

Delise Antoine

Roux Nicolas

Madelon Rémi

Tuteur enseignant : Donsez Didier

E-Mail : didier.donsez@gmail.com

Thème : Domotique

Titre : Maison Intelligente

Résumé :

Dans le cadre du DU Polytech nous avons dû fabriquer une maison automatisée à l'échelle d'un Playmobil. Ce projet se résume en deux grandes étapes ; la réalisation de la maquette et la programmation des scénarios automatisés. Après avoir dessiné les plans de la maison, nous l'avons monté et nous avons adapté à cette maquette 3 scénarios automatisés illustrant le principe de domotique: un détecteur de présence, une alarme incendie et un système de ventilation.

Puis, dans le but d'une utilisation ou d'une reproduction de cette maison intelligente par d'autrui, nous avons décidé de centraliser nos scénarios afin de permettre l'ajout d'un ou plusieurs montages supplémentaires. Ceci permet à chacun de personnaliser sa maison en l'automatisant à son envie et à l'infini, toute en la gardant simple à monter et à démonter.