Projet d'Electronique avec Arduino – PeiP2

Année scolaire 2018-2019

Smart'House
Maquette de maison connectée

Etudiants : Alexandre MAZURIER & Sébastien MICHEL

Encadrants : Pascal MASSON, Nassim ABDERRAHMANE

Remerciements

Nous tenons à remercier avant tout M. MASSON, enseignant et responsable des projets d'électronique avec Arduino pour avoir organisé un tel programme qui nous permet de découvrir une approche plus pratique de l'électronique et d'acquérir des compétences de travail en binôme. De plus, nous voulions le remercier pour sa disponibilité pendant et en dehors des séances, lorsque que nous avions besoin de conseils ou d'informations. Il convient aussi de remercier M. ABDERRAHMANE pour les conseils utiles et bienveillants qu'il a pu nous apporter durant la première partie des séances de projet.

Sommaire

I. Introduction & objectif du projet

Avec le développement des nouvelles technologies et d'Internet, la domotique s'invite aujourd'hui dans de plus en plus de foyers. Allant du simple contrôle d'une lumière aux assistants vocaux de maison les plus complexes, la domotique a de plus en plus d'applications. Ce domaine nous a semblé très intéressant et tout à fait compatible avec un projet Arduino communiquant. Nous avons dans ce projet exploré les différentes possibilités qu'offre la domotique. Ce projet nous donne, à échelle réduite, une idée des enjeux de la domotique.

L'objectif de notre projet est de réaliser, à l'aide d'un Arduino, une maquette de maison connectée. Chaque module de la maquette se contrôle directement depuis une application Android dédiée, et ce, même si l'on se trouve à l'autre bout du monde. Ainsi, l'utilisateur peut choisir, depuis son Smartphone, d'allumer ou d'éteindre le chauffage, de renseigner la température souhaitée, de choisir la couleur de la lumière du salon, ou encore de verrouiller ou déverrouiller la porte d'entrée.

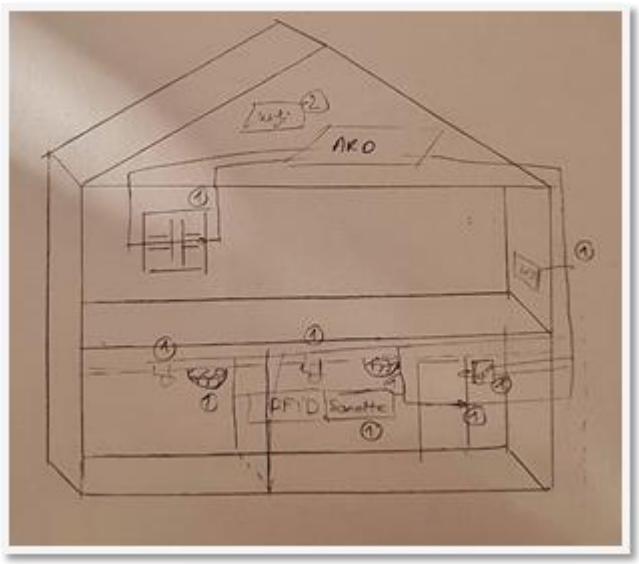
A toutes ces options de contrôle viennent s'ajouter les fonctions d'alerte. En effet, cette maison dispose d'une sonnette, d'un capteur de pluie, de capteurs d'effraction et de mouvement, afin d'avertir l'utilisateur, à l'aide d'une notification qu'il recevra directement sur son téléphone, de la détection d'évènements importants. Evidemment, toutes ces fonctions d'alerte sont activables et désactivables à souhait directement depuis l'application.

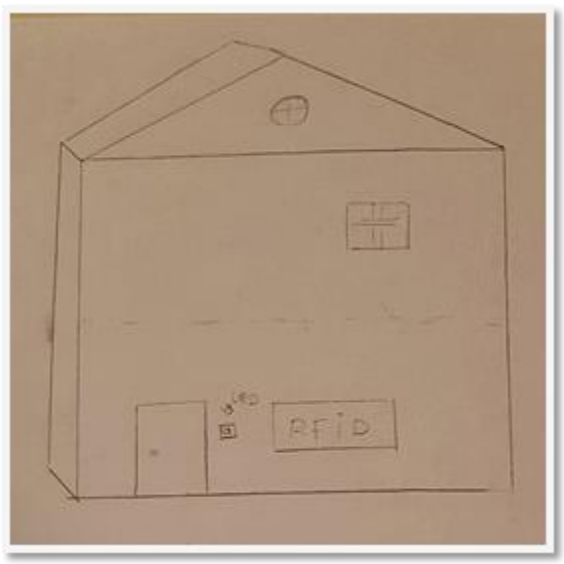
Enfin, nous devions rendre le tout le plus fluide possible et le plus simple d'utilisation, avec une application de contrôle la plus intuitive possible.

II. Réalisation

A. Développement de l'idée

Pour débuter, il a fallu définir ce que nous voulions vraiment faire comme maison et ce que nous allions utiliser pour la « connecter ». Vient donc le dessin de la maquette que nous pensions réaliser au départ :





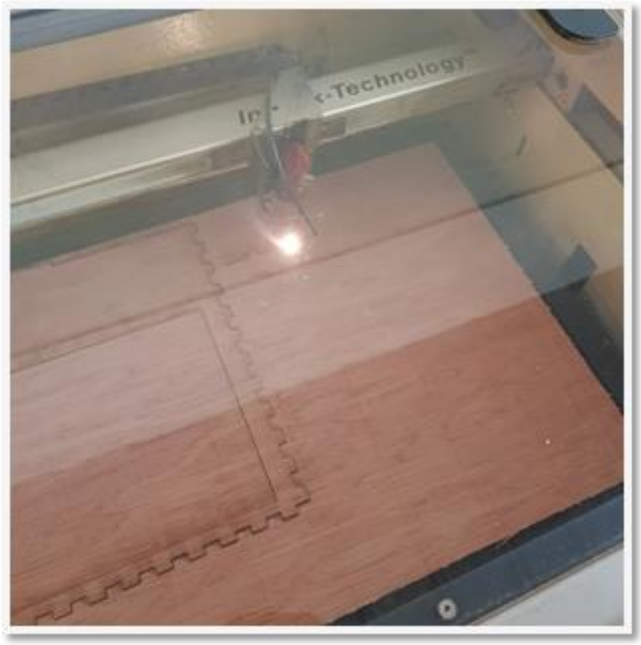
Nous avons donc défini les différents modules qui nous seraient utiles, ainsi que leurs différents emplacements. La carte Arduino se retrouvait cachée dans le toit, les modules répartis dans différentes pièces en fonction de leur utilité.

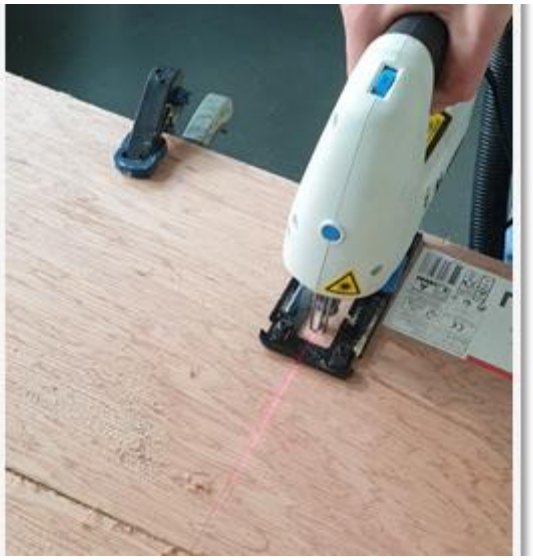
Ce schéma en main, nous nous lançons dans la réalisation d'une maquette ébauche en carton pour vérifier la taille de chaque pièce en fonction des composants. Cette maquette, ne représentant qu'une seule pièce, nous a apporté de nombreuses informations utiles pour la suite du projet. Premièrement, elle était trop petite pour faire rentrer tous les modules que nous avions, nous avons donc revu à la hausse les dimensions de la pièce. Deuxièmement, nous avons pu tester différentes positions pour chaque module, notamment le capteur de mouvement et le servo-moteur (faisant office de verrou) pour que ceux-ci soient des plus efficaces.

B. Confection de la maquette en bois

Une fois les dimensions correctement notées, nous sommes passés à la réalisation de la maquette finale. Pour cela, nous avons décidé d'utiliser la découpe laser du FabLab pour

confectionner la maquette avec des planches de bois. C'est ici que le projet a totalement changé de forme. En effet, nous n'avions pas connaissance d'une taille maximale pour les planches à découper dans la découpe laser, il aurait donc fallu séparer les façades de la maison en deux ce qui l'aurai grandement fragilisé. Nous avons donc décidé de changer l'agencement intérieur et extérieur et de diminuer la hauteur à un unique étage avec une seule pièce et en rendant le toit plat pour pouvoir rentrer la carte Arduino plus facilement tout en gardant une solidité de l'ensemble.





La réalisation du plan pour la découpe laser s'est faite avec le logiciel Inkscape et l'aide du site carrefour-numerique : Générateur de boites qui génère une boite aux dimensions demandées dans un fichier lisible par Inkscape. Il a fallu ensuite indiquer l'emplacement des trous pour les modules et les composants de la maison (portes, fenêtre, …). Puis nous sommes passés à la découpe des planches pour qu'elles rentrent dans la machine et avons laissé le laser travailler.

Pendant qu'un membre du binôme s'occupait de la maquette, l'autre s'occupait de la partie « connectée » de la maison qui sera expliquée juste après. Il fallait aussi faire fonctionner les modules un par un avant de tous les regrouper dans la maquette finale qui était en préparation, nous nous sommes donc réparti les tâches pour que chaque partie du projet avance au même rythme.

La maquette en bois finie, nous avons pu placer tous les composants et la carte à leur place. Nous avons choisi une carte Arduino MEGA dans un premier temps car les modules prenaient trop de pins pour une UNO, surtout le digicode. Nous sommes finalement repassés à une carte UNO car nous avons pu connecter une partie des composants sur l'ESP 32, qui permet la communication entre l'Arduino et l'utilisateur. Une fois tous les modules

fonctionnels séparément, nous les avons branchés ensemble à l'Arduino et l'ESP et avons eu pour défi de tous les faire marcher avec un même code.

III. Les 11 composants utilisés

Notre maquette est composée de **11** composants électroniques que l'on va classer en deux catégories :

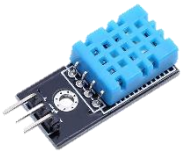
Les détecteurs

- HC-SR501 : Détecteur infrarouge de mouvement. Capteur tout ou rien qui renvoie **1** si un mouvement est détecté, **0** sinon.

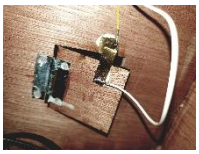
- Capteur de pluie : Module qui renvoie une valeur analogique ou digitale, au choix. Dans notre projet, nous avons configuré le capteur en mode digitale car seule la valeur binaire nous intéresse. L'utilisateur reçoit une notification si le capteur envoie **1**.

- Bouton de sonnette : Même montage qu'un bouton poussoir simple, permettant de détecter un appuie. Lorsque le bouton est enfoncé, on lit **0** et une notification est envoyée sur le téléphone, informant que quelqu'un sonne à la porte. Une LED témoin clignote pour confirmer que l'appuie a été pris en compte.

- DHT11 : Capteur de température connecté à l'Arduino. Il permet de contrôler le chauffage en mode automatique. Le chauffage s'active tant que la température souhaitée renseignée dans l'application n'est pas atteinte. La puissance du chauffage est proportionnelle à la différence entre la température renseignée et la température de la pièce.



- Contacteur : Permet d'être alerté lorsqu'une porte ou une fenêtre est ouverte. Fonctionne exactement sur le même principe qu'un bouton poussoir. Renvoie **1** si la porte (ou la fenêtre est ouverte).



- Clavier numérique : Permet d'ouvrir la porte à l'aide d'un code dans le cas ou l'utilisateur n'a pas son téléphone, qu'il n'a plus de batterie, etc.

Les « récepteurs »

- LED RGB : Caractérise la lumière du salon. On envoie grâce à l'application trois valeurs comprises entre 0 et 255 pour le rouge, le vert, et le bleu. La LED est connectée à trois pins PWM, qui permet de contrôler l'intensité de chaque couleur indépendamment.

- Servo moteur : Fait office de verrou pour la porte. Peut-être actionné soit par le digicode, soit via l'application Android. Le Servo se met à 90° pour déverrouiller la porte, et à 0° pour la verrouiller.

- Buzzer : Emet un son chaque fois que l'on appuie sur une touche du digicode. Lorsque le code est correct, émet un son adéquat, de même lorsque le code est inexact.

- Chauffage : Caractérisé par une LED rouge, on utilise un port PWM pour indiquer la puissance à laquelle on met le chauffage. Se met en route si la température souhaitée est supérieure à la température actuelle.

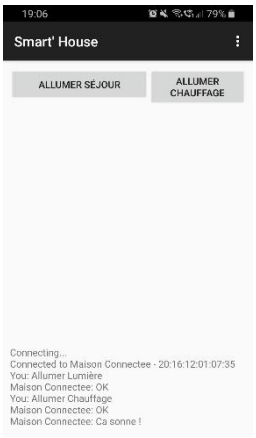
IV. Communication

A. Création de l'application

Nous avons décidé que la communication entre l'utilisateur et la maison se ferait à travers une application spécialement créée pour le projet. Cette application est écrite à l'aide de l'environnement de développement Android Studio, logiciel permettant d'écrire des applications spécialement pour Android. Une application est principalement composée d'une partie XML, qui permet de définir les boutons, zones de texte, interrupteurs, et les curseurs permettant de choisir la couleur de la LED RGB. Une fois que tous les éléments sont définis et mis à la bonne place sur l'écran, le code Java permet d'exécuter l'action associée à chaque bouton.

B. Les débuts en Bluetooth

Pour commencer, nous avons créé une première version d'application qui utilisait une communication Bluetooth avec l'Arduino. En effet, nous avions déjà largement utilisé ce type de communication en TD. Il était alors plus facile de passer par le Bluetooth au début du projet, pour avoir quelque chose qui commence à marcher assez rapidement. De ce fait, nous avons pu présenter un projet déjà fonctionnel (mais avec beaucoup moins d'options qu'actuellement) lors de la première présentation qui s'est déroulé au bout de 4 séances.



C. Le passage à une communication Wifi

Dès la 5ième séance, nous nous sommes concentrés sur une communication bien plus avantageuse pour notre projet : une communication Internet. Alors qu'une communication Bluetooth n'est possible que lorsque l'utilisateur se trouve à quelques mètres du dispositif, une communication par Internet enlève toutes contraintes de distance. Cette communication était indispensable surtout pour les fonctions d'alerte de notre maison connectée. En effet, si nous sommes notifiés d'une effraction alors que nous sommes déjà qu'à quelques mètres de l'Arduino, l'intérêt est moyen.

L'ESP32 est un Arduino en lui-même, doté d'une puce Wifi, qui lui permet de récupérer des données d'internet, d'en envoyer, etc. Deux méthodes s'offraient à nous pour commander notre maison à l'aide de l'ESP. La première consistait à faire de l'ESP en serveur Web. Il suffisait d'envoyer des requêtes HTTP à ce dernier pour lui dire quoi faire (ex. d'utilisation : allumer une lumière : ***IP_ESP?lum=true***).

L'autre solution, celle que l'on a choisi, consiste à connecter l'ESP et l'application Android à une même base de données, stockée sur un serveur externe. Google propose justement un outil qui permet cela, en plus de biens d'autres services : Firebase. Nous avons

créé une base de données accessible et modifiable par l'ESP et l'appareil Android qui stocke tous les états de la maison (cf. Capture d'écran base de données ci-dessous). Firebase permet aussi d'envoyer des notifications aux téléphones qui ont l'application Smart'House sur leur smartphone. Pour se faire, il suffit juste d'envoyer une requête HTTP aux serveurs de Firebase avec en paramètre une clé de sécurité, le destinataire, et le contenu de la notification.

Schéma de communication entre l'application et la maison connectée

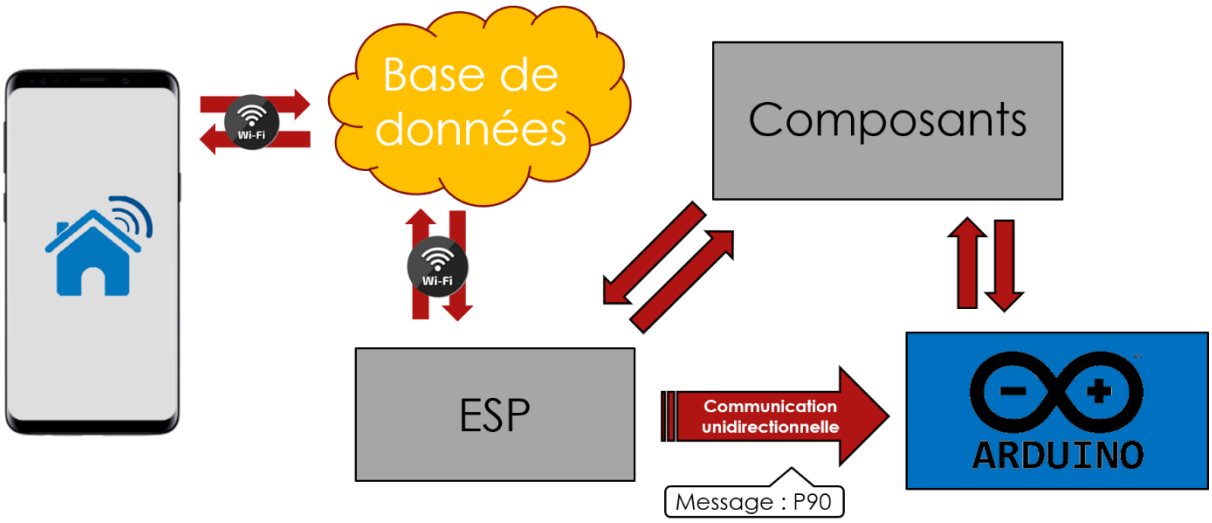
D. La communication ESP – Arduino

Une fois la base de données configurée et fonctionnelle avec l'ESP, il a fallu envoyer ses données à l'Arduino, et inversement récupérer l'état des modules branchés sur l'Arduino. L'ESP fonctionnant sur le même principe que l'Arduino, on a utilisé un « Serial Communication » qui est la communication série entre deux carte Arduino dont la bibliothèque « *SoftwareSerial* » est déjà intégrée. Il faut donc relier le respectivement le 'RX' et 'TX' de l'ESP sur le 'TX' et le 'RX' de l'Arduino. Ces ports sont définis directement dans le programme sur des Pins supportant le PWM.

Au niveau de l'électronique, c'est là qu'est apparu le gros problème de cette communication qui nous a bloqué un certain temps. Dans un premier temps, nous voulions mettre l'ensemble des modules sur l'Arduino et ne se servir de l'ESP que pour recevoir et faire transiter les informations de la base de données à l'Arduino. Cependant, il aurait fallu une

communication bidirectionnelle entre les deux cartes. Mais, contrairement à l'Arduino qui utilise du 5V, l'ESP utilise du 3.3V, on ne peut donc pas envoyer du 5V directement. Des solutions existent, comme un pont diviseur de tension que nous n'avons pas réussi à faire marcher, ou alors une autre solution : un régulateur 3.3V. Nous en avons commandé un mais il est malheureusement arrivé trop tard pour la fin du projet, c'est pourquoi nous avons imaginé un autre système qui est celui que nous avons finalement gardé car fonctionnel.

Fonctionnant dans le sens ESP vers Arduino, nous avons gardé cette communication pour les modules ne nécessitant qu'un seul message sans attendre de retour de la part de l'Arduino, c'est le cas du servomoteur par exemple. Pour tous les autres modules, nous les avons donc déplacés sur l'ESP ce qui permet d'avoir une information sur leur état et ainsi l'envoyer dans la base de données. D'autres modules, dit « non-connectés » sont restés sur l'Arduino, comme le digicode ou le buzzer. Ce système impose cependant des limitations dans notre idée de départ. Tout changement ordonné par l'Arduino n'informe pas l'utilisateur sur l'état de sa maison, par exemple si la porte est ouverte avec le digicode, l'utilisateur de l'application continuera de voir la porte comme « verrouillée » alors qu'elle est ouverte. Nous obtenons alors le schéma de fonctionnement suivant :



Au niveau du programme, une simple commande, après initialisation de la communication, permet de lire les caractères envoyés par l'ESP :

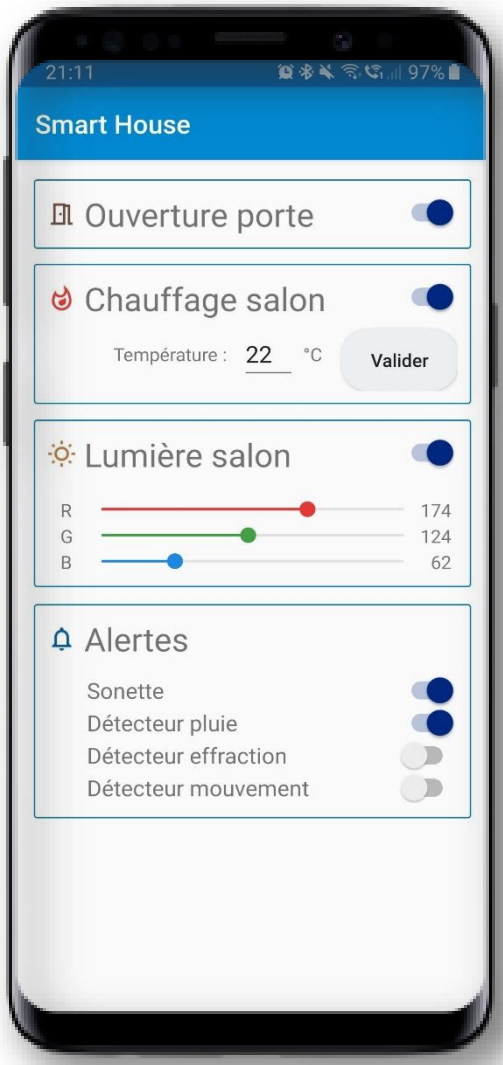
```
10  void setup() {
11   Serial.begin(9600);
12   mySerial.begin(9600);
13
14  }
```

```
21  while (mySerial.available()){
22    recu += char(mySerial.read());
23  }
24  Serial.println(recu);
```

Viens ensuite un code qui vient déchiffrer les caractères reçus. Nous avons décidé d'un système simple pour comprendre, dans un premier temps le module auquel s'adresse le message, ensuite le contenu du message lui-même. L'ESP envoie toujours 3 caractères, pas plus de 3 car si l'ESP envoie trop de messages d'affilés l'Arduino ne les lit pas assez rapidement, il fallait donc réduire au maximum le nombre de caractères. Dans ces 3 caractères, le premier est une lettre correspondant à l'initial du module, et les deux suivant une valeur numérique correspondant au message en lui-même. Par exemple, pour dire au verrou de la porte, nous envoyons 'P90' : 'P' pour Porte et '90' pour le degré que l'on indique au servomoteur.

E. Résultat : Application Android *Smart'House*

Ci-joint l'aperçu final de l'application Android ***Smart'House***. Cette application nous a demandé un grand travail, pour obtenir une interface claire, avec chaque élément bien aligné, et une certaine intuitivité. Voici la liste exhaustive des fonctions qu'intègre notre application :

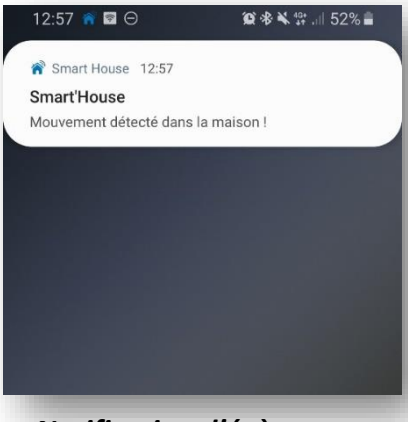


Visuel final de l'application

- ✓ *A l'ouverture de l'application* : Communication avec la base de données et synchronisation de **tous** les éléments avec celle-ci (ex. *colorR* = 174 dans la base de données ⇒ le curseur rouge mis à la bonne valeur)

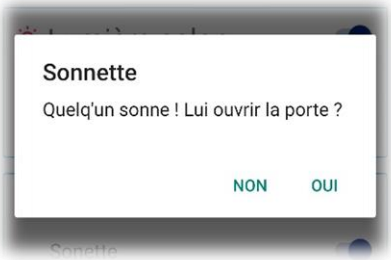
- ✓ *Fonctions de contrôle :*
  - Ouverture de la porte : Interrupteur permettant de changer l'état d'ouverture de la porte.
  - Chauffage de salon : Permet de contrôler le chauffage. On entre la température que l'on souhaite et le chauffage va démarrer en conséquence.
  - Lumière salon : Permet de contrôler l'état et la couleur de la lumière du salon. L'icone change de couleur et permet de prévisualiser la couleur transmise à la LED.

- ✓ *Fonctions de préférence :* Dans la catégorie *Alertes*, l'utilisateur choisi d'activer ou pas chaque alerte que propose notre maison connectée. Dès lors qu'une option est activée, l'utilisateur recevra une notification sur son téléphone lorsque l'évènement activé se produit.



Notification d'évènement

- ✓ Pop-Up lorsque quelqu'un sonne à la porte. Si l'alerte de sonnette est activée, l'utilisateur reçoit une notification (comme pour les autres alertes). Si l'application est ouverte au moment où quelqu'un sonne, un pop-up s'affiche à l'écran pour informer de l'évènement et permet à l'utilisateur d'ouvrir la porte directement, juste en cliquant sur ***Oui*** dans le pop-up.



V. Conclusion et perspectives

Finalement, ce projet nous a apporté sur différents plans :

Réalisation de projets :

Premièrement, la conception de cette maquette nous a appris le fonctionnement d'un projet en groupe, notamment le partage des tâches, le respect du cahier des charges mais aussi le respect, ou non, du planning préalablement défini. En effet, on prend considération des différentes difficultés que l'on peut rencontrer qu'elles soient techniques, temporelles ou de facteur humain. Cette expérience est donc enrichissante pour la réalisation future de projets en groupe, que ce soit pendant nos études ou en entreprise. Elle nous apporte une approche différente et plus maitrisée des problèmes et défis qui se présenteront à nous.

Adaptation à taille réelle :

Par rapport au projet, l'idée était de comprendre les défis que relèvent l'incorporation de la domotique dans une maison et surtout l'interconnexion des différents composants, ce qu'on appelle couramment l'IOT (Internet des Objet). Cependant, cette approche n'est pas totalement complète puisque d'autres problèmes viennent en addition lorsque l'on travaille à plus grande échelle. Par exemple, la moindre différence de tension dans notre projet nous a posée problème, mais à l'échelle d'une maison cette différence se fait sur chaque composant, rajoutant de nombreux défis. Le projet était donc une bonne approche pour ce problème qui semble donc réalisable à taille réelle une fois les difficultés passées.

Améliorations possibles :

Durant le projet nous avons donc eu des blocages que le temps nous a empêché de résoudre. Ces problèmes, nous les connaissons, et c'est par eux que nous commencerions à traiter si de nouvelles séances venaient à nous le permettre. En parallèle de ceci, la réalisation du projet nous a amené de nombreuses idées afin de mener notre projet à un stade plus complet tel que nous l'imaginons dans le meilleur des cas. L'ajout de modules ou encore la réadaptation de certains modules déjà présents dans notre maquette pourrait aboutir à une maison entièrement connectée ou l'utilisateur pourrait absolument tout contrôler depuis son téléphone ou tout autre interface qui pourrait lui plaire.

Ce dernier point amène bien sur le défi le plus important dans l'hypothèse de l'adaptation à la réalité qui est la sécurité. Pour l'instant seules les personnes disposants de l'« APK » (fichier contenant l'application) peuvent contrôler la maison mais cette solution n'est pas entièrement sécurisée et il faudrait créer une architecture réseaux totalement sécurisée pour déployer ce projet.

Annexe

Envoyer des notifications à l'application

- https://developers.google.com/cloud-messaging/http-server-ref
- https://www.codementor.io/flame3/send-push-notifications-to-android-with-firebase-du10860kb

Librairie utilisée sur l'ESP pour récupérer les informations de la base de données

- https://github.com/ioxhop/IOXhop_FirebaseESP32