Nathan
DJIAN-MARTIN
G2

Rapport de séance n°1 17/02/2020

Avant cette séance j'ai un peu avancé le programme afin de détecter d'éventuels problèmes avant la séance et de me familiariser avec la gestion de plusieurs fichiers dans un même projet Arduino.

1 - Programme

En me documentant sur le comportement d'Aruino quant à la gestion de plusieurs fichiers j'ai appris plusieurs choses qui m'ont permise d'avancer un peu le programme.

Compilation

La compilation est un moment crucial pour la portée des variables (quelles variables on va pouvoir utiliser dans quel fichier). Lorsque l'IDE compile les fichier il va les concaténer dans un certain ordre. Il compile d'abord le fichier .ino principal (celui qui porte le même nom que le dossier contenant le projet Arduino), puis il compile les autres fichiers .ino dans l'ordre alphabétique. Il concatène ensuite le tout dans un seul fichier qui sera transmit à la carte Arduino.

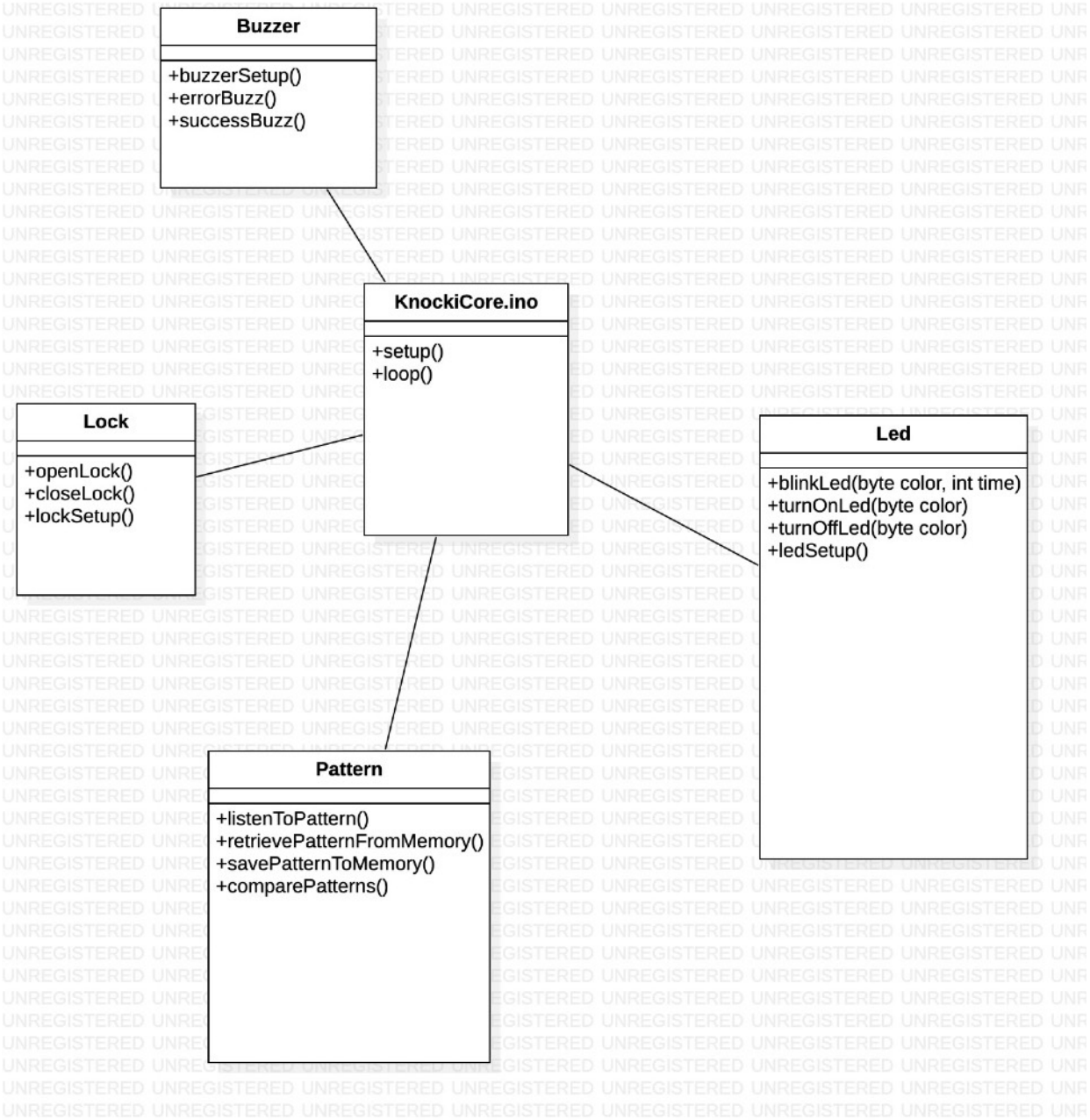
Portée des variables (scope)

Vu comment l'IDE compile le projet, il est préférable de déclarer toutes les variables en haut du fichier .ino principal pour y avoir accès dans les autres fichiers (lors de la compilation elles se retrouveront tout en haut du fichier compilé). Cependant pour des raisons de lisibilité j'ai stocké les variables contenant les pins des LEDs de différentes couleurs dans un enum que j'ai rangé dans un fichier .h (c'est un fichier header qu'il faut « #include » dans le fichier principal), je ne maitrîse pas encore les différents types de fichiers C/C++ (.cpp, .h etc…) et le rôle et l'utilité qu'ils ont dans l'environnement Arduino mais l'enum fonctionnait correctement. Finalement j'ai déclaré les variables des pins des LEDs en haut du fichier principal pour être consitant avec les autres déclarations de variables et parce que l'utilisation de l'enum n'était pas forcément justifiée, son utilité dans mon cas était de légèrement améliorer la lisibilité du code.

Avancement du programme

J'ai commencé à remplir les fichiers avec les noms des fonctions. Les plus simples (LEDs, buzzer) sont déjà remplies et nous implémenterons les autres prochainement. Il y a de plus une méthode de setup (ledSetup, buzzerSetup etc…) par fichier qui sont appelées

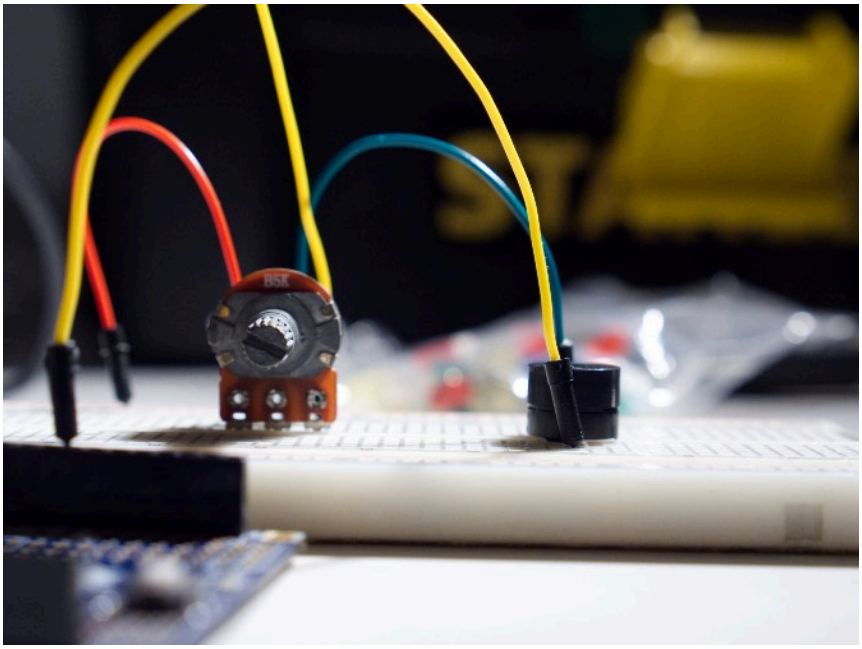
dans la fonction setup du fichier principal. J'ai aussi déclaré quelques variables et constantes dans le fichier principal. J'ai finalement modifié le diagramme UML car en programmant les fonctions je me suis rendu compte qu'il fallait légèrement modifier la structure de fichier que nous avions définie au préalable. Voici le nouveau diagramme (aussi disponible sur le GitHub).



Buzzer

En parallèle je me suis interessé au fonctionnement des buzzers car j'en avais chez moi. En plus des LEDs, cela rajoute un moyen d'interagir avec l'utilisateur et de lui donner un feedback (sonore) sur ses actions. Sans passer trop de temps à me documenter j'ai choisi un buzzer actif car le tutoriel que j'ai regardé expliquait que les buzzers actifs

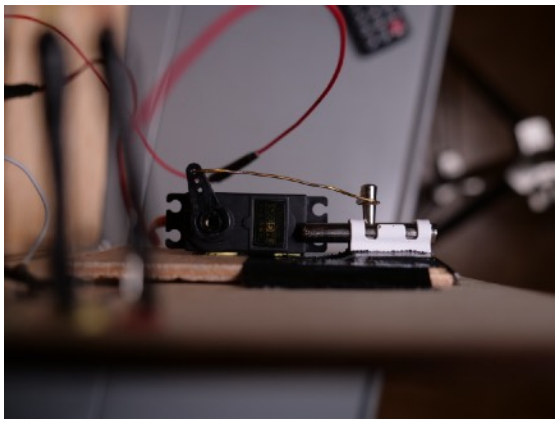
fonctionnaient avec du courant continu alors que les passifs avec de l'alternatif. J'ai donc rajouté le fichier Buzzer ainsi que quelques fonctions permettant d'emettre des séquences sonores pré-définies dans le code en fonction de ce que l'utilisateur fait. Sur l'image suivante je me suis amusé à câbler un potentiomètre avant le buzzer ce qui permet de gérer son volume.



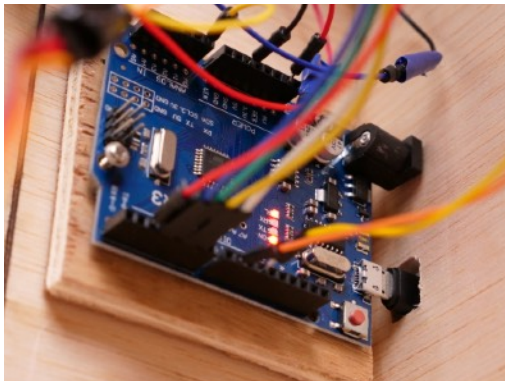
2 - Assemblage du projet (en cours)

Lors de la séance nous avons essayé d'avancer au maximum l'assemblage de notre maquette, chose que nous ne pouvons pas faire chez nous par manque d'outils. Nous avons percé un trou en haut de la boîte pour laisser passer les câbles du récepteur IR que nous avons décidé d'installer à l'extérieur de la porte.

Nous avons par la suite sur-élevé l'ensemble servomoteur + verrou car le verrou étant trop proche de l'extrémité de l'encastrement de la porte le trou allait être trop fragile. Le fait de sur-élever cet ensemble permet de l'éloigner de l'extrémité de l'encastrement et ainsi de solidifier le trou dans lequel vient se bloquer le verrou. Nous avons testé le servomoteur en l'actionnant avec la télécommande IR et nous avons ajusté le battement de l'angle dans lequel doit se positionner le servomoteur pour ouvrir le verrou. Pas de démarche particulière pour cet ajustement, nous l'avons fait à l'œil.



Nathan
DJIAN-MARTIN
G2

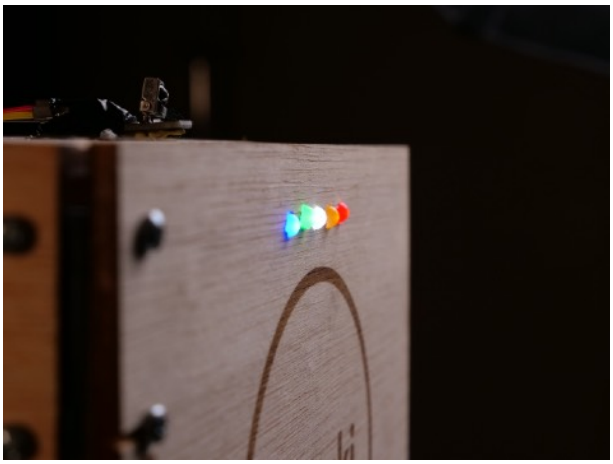


Le mécanisme fonctionnant très bien nous avons attaqué la fixation de la carte. Pour percer la boîte et ainsi laisser passer le câble USB nous avons aussi du sur-élever la carte avec une plaque en bois car le trou était trop proche de l'angle de la boîte. Pour fixer la carte de manière non définitive j'ai pré-percé la plaque de bois puis j'ai utilisé des visses à bois pour sécuriser la carte Arduino.

Comme nous avons besoin de LEDs pour communiquer avec l'utilisateur nous avons percé des trous sur le haut de la porte pour laisser passer les LEDs. Pour que la taille du trou soit parfait nous avons d'abord percé une autre planche de bois avec des forets de différents diamètres pour trouver le bon. Une fois le bon foret déniché nous avons percé la porte et glissé les LEDs dans les trous (il y a 5 LEDs au total). Nous aurions pu utiliser une unique LEDs RGB mais nous trouvions les 5 LEDs mises côte à côte plus esthétiques et nous utilisons peu de pins I/O de notre carte donc aucun souci de ce côté là.

3 - Finalisation de l'assemblage (l'après-midi)

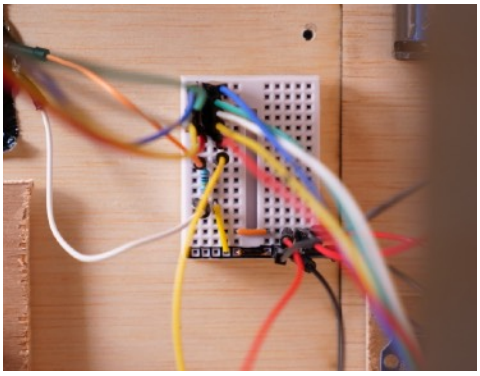
L'après-midi nous tenions à terminer l'assemblage, le cablage et les soudures. Nous avons soudé toutes les LEDs une premère fois et nous les avons sécurisées dans les trous avec du gaffer que nous avons collé par dessus. Cependant ceci empêchait l'accès aux LEDs facilement en cas de remplacement donc nous avons tout ressoudé de manière plus propre. Dans la foulée nous avons testé les LEDs et elles fonctionnaient parfaitement.



Nous avons ensuite fixé le récepteur IR sur le haut de la porte. Nous voulions le fixer avec un pistolet à colle mais la colle n'accroche pas très bien le bois, nous avons donc utilisé du gaffer et de la pâte adhésive. Nous avons scotché les cables que nous avons branché sur le récepteur car les broches étaient très proches et nous trouvions le risque de court-circuit trop important en utilisant des soudures. Nous l'avons aussi testé lors du test avec le servomoteur.

Nathan
DJIAN-MARTIN
G2

Nous avons finalement câblé le capteur piezo électrique que nous avons testé juste après afin d'être certain qu'il détecte bien les coup une fois la porte dans sa configuration finale. Nous l'avons câblé sur une mini table de prototypage que nous avons aussi utilisée afin de dupliquer nos pins I/O de la masse et du +5V



Nous avons décidé de supprimer le buzzer pour le moment car le code utilisé pour la fonction « tone() » cause, d'après ce que l'on a compris, des conflits avec la library du récepteur infrarouge. Nous n'avons pas plus creusé cette erreur pour le moment car nous voulions une maquette fonctionnelle à la fin de l'après-midi. De plus le buzzer était un ajout plus amusant que réellement utile. Nous nous pencherons sur l'erreur si nous en avons le temps.

Finalement, à la fin de cette après-midi la maquette est finalisée, correctement cablée et chaque composant fonctionne parfaitement, le piezo détecte les coups, les LEDs s'allument, la récepteur IR permet d'actionner le servomoteur avec la télécommande et le mécanisme du verrou fonctionne très bien.