

PROJETS ARDUINO - PEIP2

Année scolaire 2018-2019

Robot-réveil



Etudiants : Maltese Salomé, Collie Kilian

Encadrant : Mr Masson Pascal

REMERCIEMENTS

Nous remercions notre professeur Mr Masson Pascal ainsi que Mr Abderrahmane Nassim qui nous ont aidés tout au long de notre projet.

Nous remercions également Mr Forner Marc, qui nous a aidé à imprimer nos pièces en 3D au FabLab.

LIENS PRATIQUES

Ici vous trouverez les liens pratiques comme notre chaîne YouTube, ou notre GitHub :

Github : <https://github.com/kilian-salome/Robot-reveil>

Youtube : https://www.youtube.com/channel/UCtZrPfM_xAn4shzOnV4YiYg

SOMMAIRE

| | |
|--|----|
| Introduction et inspirations..... | 4 |
| Chapitre I : Objectif et résultat | |
| I.1. Cahier des charges..... | 5 |
| I.2. Résultat final..... | 5 |
| I.3. Plannings initial et final : comparaison..... | 6 |
| Chapitre II : Fonctionnement | |
| II.1. Algorithme..... | 8 |
| II.2. Modules..... | 8 |
| Chapitre III : Perspectives | |
| III.1 Problèmes rencontrés..... | 12 |
| III.2. Points à améliorer..... | 12 |
| Conclusion..... | 14 |
| Bibliographie..... | 15 |

Introduction

Le projet Arduino permet d'avoir un avant-goût du travail en entreprise. En effet, nous avons conçu un projet avec un cahier des charges, un nombre de séances fixé donc un timing à respecter, avec également des libertés, puisque nous avons pu commander des pièces, mais aussi des restrictions avec un budget limité. Nous avons eu des imprévus, ainsi que des nouveautés à gérer plus ou moins seuls. En effet, nous avons utilisé du matériel déjà vu en cours, comme les moteurs ; mais aussi du matériel nouveau, comme les écrans OLED dont nous avons du apprendre le fonctionnement par nous-même. Cela nous a permis de nous faire une idée de ce qui nous attendra dans le futur.

Inspirations

Concernant l'idée de ce projet, disons que [cette vidéo](#) a été la principale source d'inspiration et que cette idée a éveillé en nous la détermination nécessaire à ce projet...

C'est un réveil, nommé Clocky, qui s'enfuit lorsqu'il sonne. Cette idée nous a tout de suite plu. Tout d'abord, elle joint l'utile à l'agréable, puisque l'idée de devoir se lever pour arrêter un réveil est assez amusante, et l'utilisation d'un réveil le matin est très répandue. Nous étions tous les deux d'accord sur le fait que se lever le matin, c'est souvent une étape compliquée, qui passe par le programmation de plusieurs réveils : le premier étant trop dur, on en met un deuxième pour se laisser un peu le temps de se réveiller, puis un troisième parce que le deuxième est encore trop difficile et enfin encore un au cas où...



Finalement, se forcer à se lever pour arrêter le réveil nous a semblé assez efficace pour avoir à programmer un unique réveil et se lever une bonne fois pour toutes.

Durant ce rapport, nous allons vous parler de ce projet en détail. Dans un premier temps, nous allons voir l'objectif que nous nous étions fixé en vous montrant notre cahier des charges ainsi que notre résultat, ce que nous avons finalement réalisé. Nous allons également comparer notre planning initial au final. Après cela, nous allons expliquer le fonctionnement de notre réveil, avec d'abord la partie programmation puis nous allons détailler l'agencement des différents modules. Pour finir, nous allons parler des perspectives du projet en expliquant les problèmes que nous avons rencontrés et ce que nous aurions pu améliorer.

Chapitre I : Objectif et résultat

I.1. Cahier des charges

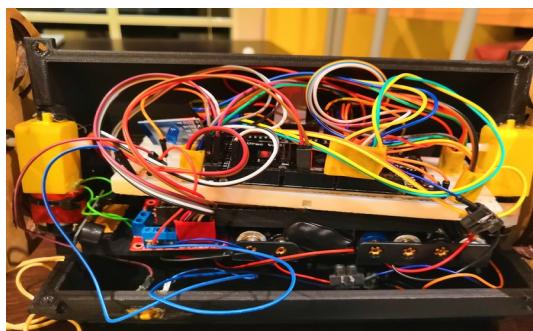
Robot-réveil : quésako ?

- Ce projet consiste à créer un robot réveil qui s'enfuit lorsqu'il sonne. Il faut ainsi se lever pour l'arrêter. Le robot peut tomber des meubles. Il peut aussi éviter des murs ou autres obstacles. Lorsqu'il en rencontre, il recule un peu puis tourne pour changer de direction.
- Le réveil sera programmable depuis un téléphone portable. Il aura donc plusieurs fonctionnalités : alarme et connexion Bluetooth. Il pourra aussi donner l'heure lorsqu'on appuie sur un bouton. Sinon, il se mettra en veille.
- La voiture pourra tourner, avancer, reculer. Elle sera constituée de deux roues. Il faudra donc mettre un contre-poids pour qu'elle ne tourne pas, que ce soit uniquement les roues qui tournent.

Étapes à suivre :

- Premièrement, il faut faire le réveil. D'abord, réussir à afficher l'heure avec Arduino, puis utiliser le module Bluetooth pour afficher l'heure depuis notre téléphone portable. Il faut aussi utiliser un buzzer pour pouvoir le faire sonner et s'arrêter lorsqu'on appuie sur un bouton.
- Ensuite, il faut faire la voiture. Concernant la structure, on peut soit la dessiner sur un logiciel puis l'imprimer en 3D, soit la construire nous-même. Pour la faire fonctionner, nous allons utiliser des petits moteurs (comme en TD). Pour relier les roues au moteur, il nous faut des suspensions.
- Finalement, il faut assembler le réveil avec la voiture. Nous allons programmer la voiture de façon à ce qu'elle se mette en marche lorsque le réveil sonne. Le trajet de la voiture est aléatoire. La voiture s'arrête en même tant que l'alarme.

I.2. Résultat final



Suite à des changements au cours des séances ainsi qu'à quelques dysfonctionnements, le résultat de notre projet diffère de ce que l'on avait prévu.

Concernant les fonctionnalités du réveil, il donne l'heure grâce au capteur de distance : lorsqu'on s'approche à une certaine distance, l'heure s'affiche. Sinon, les écrans affichent des yeux (c'est le mode « veille »). Le réveil est programmable depuis un téléphone portable grâce au Bluetooth. Nous avons créé une application qui permet de paramétriser l'alarme. Celle-ci sonne grâce au buzzer.

Concernant la voiture : elle possède deux moteurs, elle peut avancer, reculer, tourner et détecter les obstacles grâce au capteur de distance. Elle est constituée d'une coque et de deux roues.

Étapes par étapes, voilà comment le réveil fonctionne : il faut d'abord se connecter au Bluetooth via un téléphone portable sur lequel il faut préalablement installer l'application MIT AI2 Companion (dont nous verrons le fonctionnement un peu plus un détail).

Ensuite, il faut programmer l'heure du réveil sur l'application. Le réveil va alors recevoir l'information et donc sonner et rouler. Il va également afficher le message « c'est l'heure » sur les écrans. Tant qu'il ne rencontre pas d'obstacles, il avance. A la rencontre d'un



obstacle, il change de trajectoire. Il commence par ralentir, ensuite il recule, accélère à nouveau et tourne, puis il avance jusqu'au prochain obstacle où la procédure se répète. Pour l'arrêter, il suffit d'appuyer sur un bouton. Dès qu'on appuie, il s'arrête de sonner et de rouler. L'écran affiche des yeux à nouveau (ou l'heure si on se rapproche).

I.3. Plannings initial et final : comparaison

Planning initial :

| | Séance 1 | Séance 2 | Séance 3 | Séance 4 | Séance 5 | Séance 6 | Séance 7 | Séance 8 |
|---------------------------|----------|----------|----------|----------|----------|----------|----------|---------------|
| Affichage heure actuelle | Kilian | Kilian | | | | | | |
| Programmation buzzer | Salomé | Salomé | | | | | | |
| Relation bluetooth/réveil | | | Kilian | Kilian | | | | |
| Roues et suspensions | | | | Salomé | Salomé | | | |
| Contrepoids | | | | | | Salomé | | |
| Moteurs | | | | | Kilian | Kilian | | |
| Radar | | | | | | | Salomé | |
| Encapsulage | | | | | | | | Kilian/Salomé |
| Assemblage robot+réveil | | | | | | | | Kilian/Salomé |

Planning final :

| | Séance 1 | Séance 2 | Séance 3 | Séance 4 | Séance 5 | Séance 6 | Séance 7 | Séance 8 |
|----------------------------|---------------|----------|----------|----------|----------|----------|----------|---------------|
| Cahier des charges | Kilian/Salomé | | | | | | | |
| Module RTC / écrans LCD | Kilian/Salomé | | | | | | | |
| Capteur de distance | | | Kilian | | | | | |
| Bluetooth + application | Kilian | Kilian | | | | | | |
| Buzzer + bouton poussoir | Salomé | Salomé | | | | | | |
| Optimisation code | | | | Kilian | | | | |
| Test des modules ensembles | | | | Kilian | | Kilian | | Kilian/Salomé |
| Ecrans OLED | | | | Salomé | Salomé | Salomé | Salomé | |
| Moteurs | | | | | Kilian | | | |
| Inventor / impression 3D | | | | | | | Kilian | |
| Assemblage | | | | | | Kilian | | Kilian/Salomé |

Comme on peut le voir ci-dessus, il y a une assez grande différence entre ce que l'on avait prévu et ce qu'on a finalement fait. Tout d'abord, nous avions prévu deux séances pour réussir à afficher l'heure et utiliser un buzzer. Ces deux tâches ont été finalement assez faciles à faire. Afficher l'heure nous a pris moins d'une séance (grâce au module RTC). Pour le buzzer, nous avons mis deux séances, mais c'est dû à l'utilisation du buzzer avec le bouton poussoir qui a généré quelques difficultés. Ensuite, nous avions prévu du temps pour les roues et suspensions, alors que nous nous sommes occupés de la carrosserie de notre voiture et donc des roues en fin de projet et n'avons finalement pas fait de suspensions. Nous avons également introduit des choses au fil des séances auxquelles nous n'avions pas pensé, comme la création d'une application. De plus, nous n'avions pas anticipé l'optimisation de notre code. Celle-ci s'est avérée nécessaire puisque notre code fait plus de 600 lignes maintenant. De plus, nous avons ajouté les écrans OLED qui ont pris beaucoup de temps alors que dans notre planning initial nous n'avions même pas prévu de séance pour ça. Nous n'avions également pas d'idée précise sur comment on allait faire l'esthétique de notre réveil. Nous avons introduit une séance dans le planning pour faire le modèle sur un logiciel afin de l'imprimer en 3D.

Chapitre II : Fonctionnement

I.1. L'algorithme

Pour trouver l'algorithme vous pouvez aller sur notre GitHub avec le lien donné précédemment.

On peut décomposer ce programme en 3 grandes parties. Chaque partie étant représentée par la valeur d'une variable nommée « cond » (pour condition). Tout d'abord, cond est initialisée à 1. Si cond =1, alors le robot est en stand-by c'est-à dire qu'il va attendre qu'une information soit envoyée pour changer de grande partie. De plus, lorsque cond=1, le réveil affiche des yeux sur les écrans et si la main de l'utilisateur est proche du robot (si le capteur de distance capte quelque chose en dessous de 20 cm), il affiche l'heure actuelle.

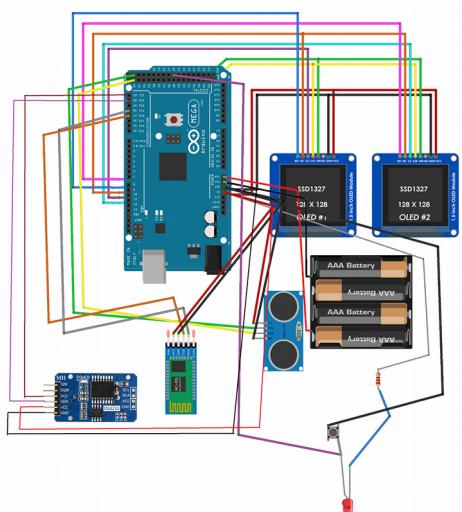
Lorsqu'on utilise l'application mobile et que l'on envoie l'heure et la minute programmées alors l'Arduino les reçoit avec le Bluetooth et cond est égale à 0. Si cond =0, on regarde si l'heure actuelle est la même que l'heure programmée et si la minute actuelle est la même que celle programmée. Dans ce cas on met cond à 2. Si l'heure n'est pas la même on attend et le réveil ne fait rien.

Lorsque cond =2 on fait sonner le buzzer, on affiche le message « c'est l'heure ! » sur les écrans et on fait rouler le robot. Dans la fonction qui fait rouler le robot, on regarde la distance que capte le capteur de distance. Si cette distance est entre 0 et 20 centimètres, alors on rentre dans une boucle qui fait reculer le robot puis le fait tourner légèrement. Dans cette petite partie du code, on a rajouté des boucles. Elles servent à faire décélérer les moteurs, puis accélérer, afin de faire des départs progressifs. Cela permet d'aider les moteurs, puisque c'est un gros problème que nous avons rencontré. Ensuite, tant que le bouton poussoir n'est pas appuyé, on reste dans cette boucle. Lorsque le bouton est appuyé, on sort de cette grande boucle en mettant cond à 1. Ainsi on revient au début de l'algorithme.

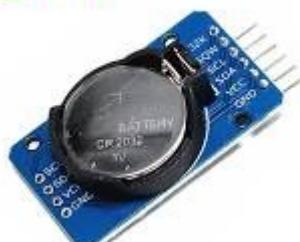
I.2. Les modules

Voici la présentation de tous les modules dont nous avons eu besoin pour notre projet.

D'abord, un schéma du montage final (sans les moteurs) :



[Module RTC \(Real Time Clock\) :](#)



La carte Arduino mesure le temps écoulé depuis sa mise sous tension en millisecondes. Elle peut donc calculer des intervalles de temps avec précision. C'est bien pratique pour réaliser des temporisations mais pas suffisant si on a besoin de l'heure . Une instruction comme millis() par exemple, retourne le temps écoulé depuis le dernier démarrage de l'Arduino mais sera réinitialisée dès le prochain démarrage. Afin que notre carte Arduino connaisse l'heure de façon permanente, nous avons utilisé le module RTC DS3231. Ce module comporte une puce qui gère l'heure de façon extrêmement précise grâce à un quartz et une pile qui permet de sauvegarder cette heure. Elle communique avec le module Arduino via le bus I2C (broches SDA & SCL).

Câblage du module DS3231 avec Arduino UNO :

VCC -> Arduino 5V

GND -> Arduino GND

SCL -> SCL (ou A5)

SDA -> SDA (ou A4)

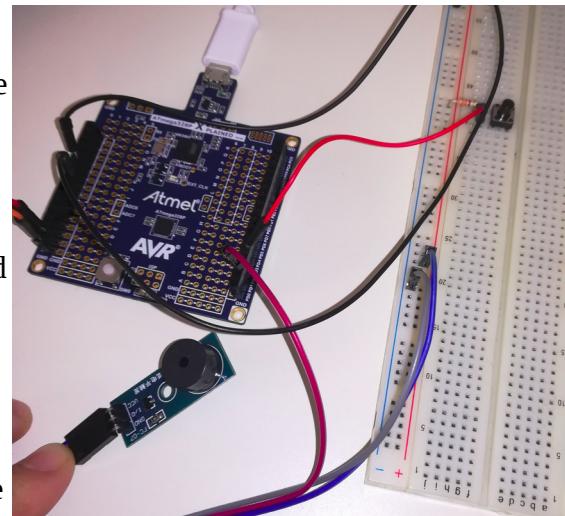
Nous avons importé la librairie ds3231, qui nous a permis de réaliser un programme simple. Elle contient une structure de temps dans laquelle les variables concernant la date et l'heure sont définies. On a pu ainsi récupérer les heures, minutes et secondes facilement.

Buzzer :

Afin de faire sonner le réveil, nous avons utilisé un buzzer. Il fonctionne en numérique. Il faut donc le relier à une sortie digital. Le montage est simple : il consiste à relier l'alimentation (5V et GND) et une sortie digital de l'Arduino au buzzer.

Nous avions au départ utilisé la fonction tone(), qui génère une onde carrée à la fréquence spécifiée (en Hertz) sur une broche. Sa syntaxe est la suivante : tone(broche, fréquence, durée) où broche correspond à la broche sur laquelle la note est générée (sortie digital sur laquelle le buzzer est branché). On introduit ensuite la fréquence de la note produite, en hertz (Hz), puis la durée de la note en millisecondes (optionnel). Cependant, plus tard dans notre projet lorsque nous avons voulu rajouter le code du buzzer dans notre code principal, nous avons eu des erreurs de compilation. La fonction tone utilise le timer n°2 et il y a donc des conflits de librairies... Nous avons donc trouvé une nouvelle librairie NewTone (que nous avons téléchargé sur internet) qui fonctionne de la même manière mais sans les conflits.

Nous avons également introduit un bouton poussoir permettant d'arrêter de faire sonner le buzzer manuellement.



Capteur de distance (module HC-SR04) :

Câblage: 3 entrées = VCC, GND et trigger ; une sortie = echo

Comme en TD, nous avons utilisé la librairie NewPing avec la fonction NewPing sonar(trig, echo, distance), où la distance est la distance maximale à mesurer, en cm. Ce module nous a servi d'une part à afficher l'heure. En effet, comme vu dans le résultat final, nous avons sur notre réveil deux écrans OLED qui affichent soit des yeux, soit l'heure lorsqu'on se rapproche à une certaine distance, grâce au capteur de distance. D'autre part, il permet au robot de détecter les obstacles. Nous avons dans notre algorithme une fonction qui permet de faire rouler le robot et dans celle-ci, nous utilisons le capteur de distance. Cela nous permet de changer la trajectoire du réveil et ne pas se prendre de murs ou autres obstacles potentiels.

Bluetooth (module HC-06) :

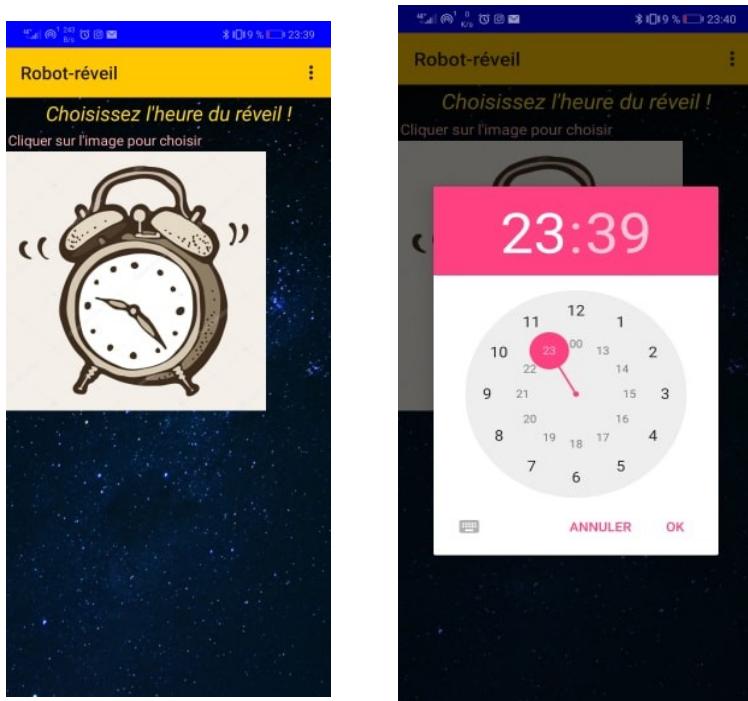


Câblage: 3 entrées = VCC, GND et RX(réception) ; une sortie = TX(transmission)

Ce module permet la communication en Arduino et un téléphone portable, ce dont nous avions besoin pour programmer le réveil. Il peut fonctionner en esclave ou en maître donc contrôler d'autres modules bluetooth. Ici, il est esclave. Nous avons utilisé la librairie SoftwareSerial. La communication Arduino/module est similaire à la communication PC/Arduino. Ce module peut fonctionner en esclave ou en maître donc contrôler d'autres modules bluetooth. Ici, il est esclave.

En parallèle, nous avons créé une application. le but étant de faire un réveil connecté, ce qui est plus pratique. Pour cela nous avons utilisé le site [App Inventor](#) développé par Google et le MIT.

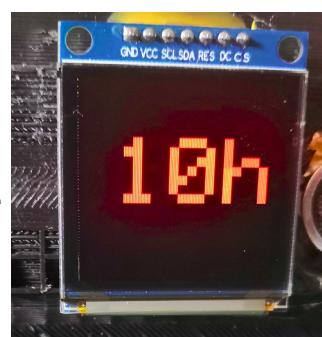
Il faut aller sur le site d'APP inventor 2. Après nous être inscrits, nous avons pu créer l'apparence de notre application. Ensuite, nous avons fait la partie programmation avec du code. Sur le site, la partie est facilement accessible à tous niveaux puisque c'est de la programmation par blocs. Finalement, il faut générer un QR CODE depuis le web puis scanner le QR CODE sur application compagnon de notre téléphone que nous avons téléchargé. Cette application installe directement l'application sur notre téléphone et donc exécute notre code. Nous pouvons finalement, sur l'application, nous connecter au Bluetooth et ensuite choisir l'heure à laquelle on va programmer l'alarme. Nous avons ensuite un peu améliorer l'esthétique. Voici le rendu final :



Ecrans OLED 128x128 :

Au départ, lorsque nous avons utilisé le module RTC, nous avions simplement affiché l'heure sur le moniteur, puis sur un écran LCD (dont le fonctionnement avait été vu en TD). Pour avoir plus de ports disponibles et pour des raisons esthétiques, nous avons décidé d'utiliser un écran OLED. Celui fourni par le professeur au départ avait besoin de seulement quatre ports. Cependant, nous avons finalement eu

l'idée d'en utiliser deux afin de faire des yeux à notre réveil, lorsque celui-ci est en mode "veille", pour le rendre plus convivial. De plus, nous en avons finalement commandé des nouveaux écrans car ceux fournis n'étaient pas en couleur RGB (ils étaient en orange et bleu). Les nouveaux que nous avons reçus nécessitent sept ports au lieu de quatre. Ceci est dû au fait qu'ils n'ont pas le même



mode de communication avec l'Arduino. Ceux que nous avons commandés ont une communication SPI, où il y a toujours un composant maître (ici l'Arduino) qui commande le ou les composants périphériques. Le câblage est le suivant :

GND, VDD

SCK (Serial Clock) : cette ligne transmet les impulsions de signal d'horloge générées par le composant maître SDA, RST,

MOSI(Master Out Slave In) : c'est la ligne utilisée pour envoyer des données du maître vers le(s) périphérique(s).

RES : réinitialise le module

DC : broche de commande de données

CS(Chip Select) : la ligne de sélection de l'esclave

Sur l'arduino mega, il faut utiliser ces broches : 51 : MOSI, 52 : SCLK, 53 : CS

Nous avons installé deux bibliothèques: Adafruit_SSD1351, qui gère la communication, et Adafruit_GFX, qui permet d'ajouter des fonctions graphiques telles que des lignes, des cercles et du texte. Ceci nous a permis d'afficher l'heure ou un message lorsque le réveil sonne assez facilement. Pour afficher un œil, nous avons d'abord pensé à le dessiner nous-mêmes... Nous avons finalement trouvé une méthode plus simple grâce à ce site :

<http://javl.github.io/image2cpp/>. Il convertit une image en liste de pixels. Ensuite on a importé cette liste de pixels. Dans la librairie adafruit, il existe une fonction "display.drawBitmap" qui permet de dessiner un pixel particulier. Nous avons donc choisi une image, que l'on a convertie en une liste de pixels via le site, puis nous avons utilisé la fonction drawBitmap. Pour que les deux écrans fonctionnent correctement, il a fallu faire des branchements spécifiques. En effet, pour l'affichage de l'heure nous voulions qu'un écran affiche les heures et l'autre les minutes. Pour cela, il faut que l'arduino communique avec les deux écrans séparément. Nous sorties, exceptées les CS et RES que nous avons bran de communiquer avec ses deux esclaves non simultan de chacun (RES).

Moteurs :

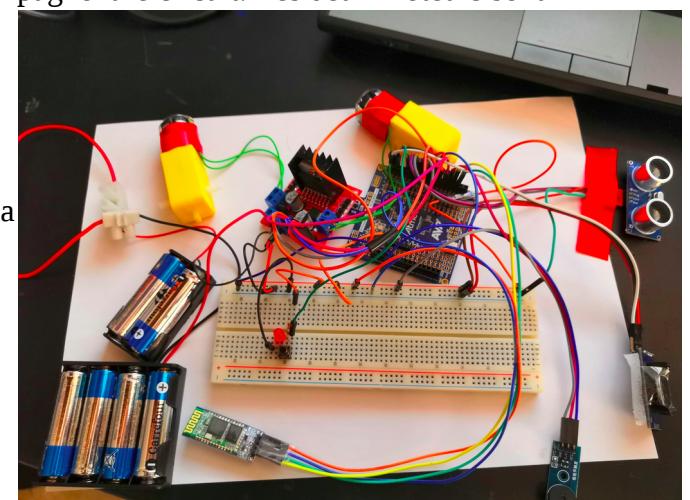
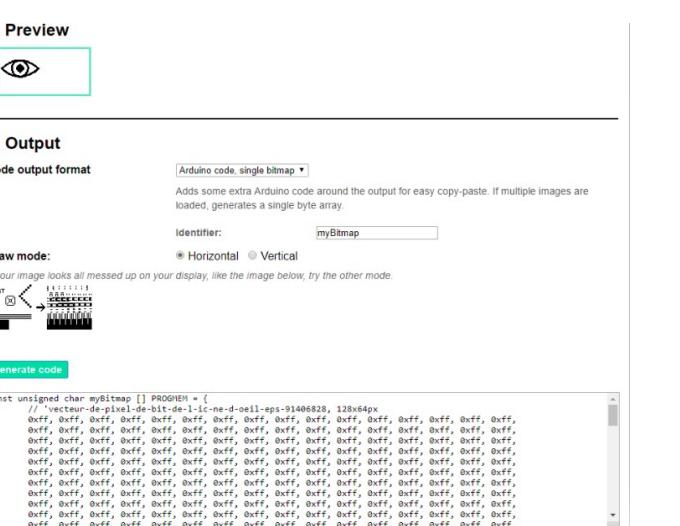
Nous pouvons voir ci-dessous les deux moteurs connectés au driver. On voit également le capteur de distance sur la droite ainsi que les autres modules.

Nous avons utilisé deux moteurs CC (moteur à courant continu) avec le driver L298, qui possède des modules qui intègrent déjà tous les composants qui accompagnent le circuit. Les deux moteurs sont alimentés par des piles. Nous avons ajouté un condensateur afin de stabiliser l'alimentation lorsqu'on alterne entre marche avant et marche arrière.

Chaque moteur possède trois ports :

Enable, qui permet d'activer la commande des moteurs, on la met sur une sortie PWM afin de pouvoir contrôler la variation de la rotation (valeur entre 0 et 255).

Les deux autres sont les broches de commande digitales (1 et 2) qui sont utilisées pour commander le sens de rotation du moteur.



Chapitre III : Perspectives

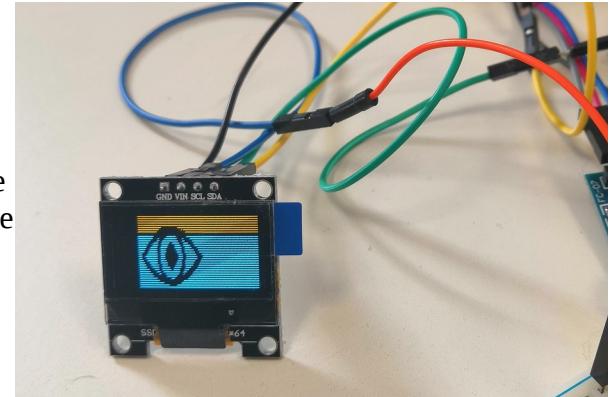
III.1. Problèmes rencontrés

Comme souvent dans un projet, nous avons eu quelques imprévus.

Notre premier souci a été avec le buzzer. Nous voulions qu'il sonne lorsqu'il reçoit l'information envoyée par le Bluetooth puis qu'il s'arrête une fois qu'on a appuyé sur le bouton poussoir.

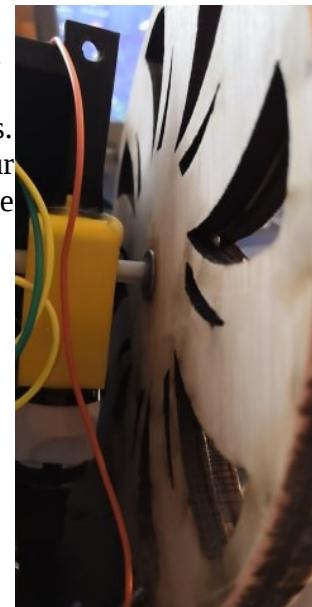
Ensuite, nous avons eu des complications avec les écrans. Au départ, on avait pas d'écrans en couleur (orange et bleu) mais nous n'étions pas sûr d'en recevoir en couleur, donc nous avons essayé de faire fonctionner ceux là comme on le voulait, ce qui nous a pris du temps. En effet, nous n'arrivions pas à afficher deux choses différentes sur chaque écran simultanément. Le problème venait de l'adresse, qui était la même pour chaque écran. Leur adresse était définie selon le positionnement de la résistance. Pour changer l'adresse d'un des deux écrans, il a fallu déssouder sa résistance pour la ressoudre à un autre endroit.

Nous avons finalement reçu les écrans en couleur, mais nous avons également quelques soucis. Ils n'avaient pas le même mode de communication (communication SPI alors que les anciens avaient une communication i2c) donc on ne savait pas comment utiliser les 2 séparément. Il suffisait finalement d'utiliser deux sorties différentes pour les broches CS et reset.



De plus, nous nous sommes aperçus qu'il nous manquait une sortie pour pouvoir tout brancher sur notre carte Arduino (Uno). Il nous manquait aussi des sorties PWM (dont on a besoin pour les moteurs et les écrans). Il a donc fallu changer de carte et prendre une Mega.

Enfin, nous avons eu des problèmes avec les moteurs. Nous avons eu tout d'abord du retard sur l'impression de notre modèle en 3D. Nous avions besoin d'une pièce assez grande, que nous avons imprimée en deux fois, avec 9h d'impression pour chaque moitié. Cependant, suite à quelques soucis avec l'imprimante 3D, lorsque nous avons récupéré notre deuxième pièce, il ne restait plus de séances. Nous avons du faire l'assemblage chez nous. Cependant, nous nous sommes aperçus que les moteurs n'étaient pas assez puissants : leur couple était trop faible. Nous avons rajouté des piles. D'abord, une pile de 5 volt, puis une de 9 volt, puis deux de 9 volt, et à la fin 2 piles de 9 volt et enfin une de 5 volt. Après ça, nous avons réussi à faire avancer le réveil au sol, mais pas de manière optimale. De plus, lors de l'assemblage, nous avons directement mis nos roues sur les axes des moteurs. Nous avons ensuite rajouté une rondelle au bout des axes pour diminuer les frottements. Nous avions aussi rajouté un contour en caoutchouc sur nos roues que nous avons finalement d'enlever pour essayé d'alléger les moteurs en réduisant le diamètre.



III.1. Points à améliorer

Il y a quelques points sur notre projet que nous aurions pu améliorer avec quelques séances supplémentaires.

Tout d'abord, comme nous l'avons vu dans le paragraphe précédent, le réveil ne roule pas correctement. Nous aurions donc pu utiliser des moteurs plus puissants, ainsi qu'alléger notre matériel. En effet, la coque de notre réveil est quand même assez lourde, et tout le matériel dont nous avons besoin pour faire fonctionner notre réveil aussi, notamment les piles qui sont nombreuses.

De plus, nous voulions au départ que notre réveil puisse tomber meubles. Or, par manque de temps et surtout car le réveil n'est pas encore capable de rouler correctement sans les chutes, nous n'avons pas essayé. Mais, avec un réveil qui fonctionne correctement, nous aurions pu d'abord essayé de le faire tomber d'une hauteur minime, puis petit à petit d'un peu plus haut en remettant de la mousse sur les roues comme on l'avais essayé.



Nous aurions pu également améliorer l'esthétique de notre réveil. En effet, si notre projet était industriel, nous n'aurions pas pu le mettre en vente avec un design comme celui-ci. Nous avons conçu la coque de notre réveil de façon à avoir la place de mettre tout le matériel à l'intérieur et que les moteurs soient fixes. Cependant elle n'est pas forcément « jolie ». On aurait peut-être pu ajouter une bouche à notre réveil puisqu'il a déjà des yeux.

Enfin, nous aurions pu essayer de mettre autre chose que le buzzer en guise d'alarme, comme des musiques par exemple, que le client puisse configurer lui-même .

Conclusion

Durant ce projet, nous avons plus ou moins tenu notre cahier des charges. Hormis que notre réveil ne tombe pas des meubles, nous avons exploité toutes les fonctionnalités que nous voulions. Nous n'avons pas tenu notre planning, que nous avons fait sûrement un peu trop rapidement. La partie réveil sans la voiture fonctionne : le réveil donne l'heure, il est programmable sur un téléphone grâce au Bluetooth, il sonne à l'heure choisie, et s'arrête après l'appui sur le bouton. La voiture fonctionne partiellement pour des raisons citées plus haut. Nous aurions peut-être pu la faire fonctionner correctement en anticipant certaines choses, comme le matériel trop lourd pour nos moteurs. Nous aurions pu aussi imprimer nos pièces plus tôt, nous nous sommes par rendu compte qu'imprimer en 3D pouvait prendre beaucoup de temps. C'est des choses qui sont bonnes à savoir pour nos projets futurs.

Bibliographie

Les différents sites qui ont contribué au développement de notre projet :

- **Pour le module RTC :**

<https://www.arduinolibraries.info/libraries/ds3231>

<https://www.robot-maker.com/forum/tutorials/article/120-utilisation-du-module-rtc-ds1307/>

- **Pour les écrans :**

<https://www.instructables.com/id/Utiliser-Un-%C3%89cran-OLED-124x68-Sur-Arduino/>

- **Pour tous les autres modules :**

<http://users.polytech.unice.fr/~pmasson/Enseignement-arduino.htm>