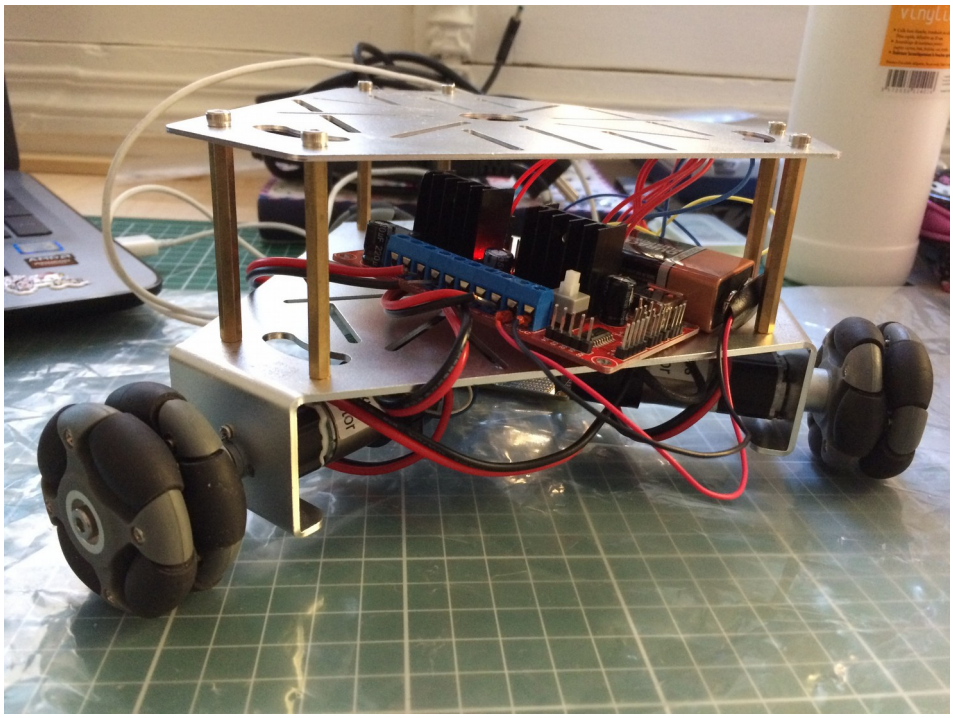
MUSICAL-E ROBOT

POLYTECH NICE SOPHIA

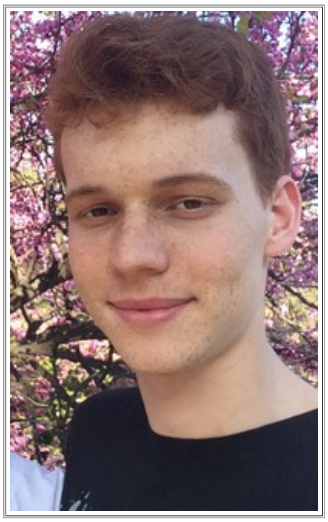
projet réalisé dans le cadre de la formation de prépa intégrée de Polytech'Nice Sophia



Sommaire

I. Présentation du binôme

a. Jehan Bouazzaoui



Élève en 2ème année de prépa intégrée à Polytech Nice-Sophia, Jehan est passionné par la musique et souhaite l'année prochaine s'orienter dans le domaine de l'électronique. Il était donc tout à fait naturel d'inclure du son au projet, mais sous quelle forme ? La première idée de projet fut celle d'un robot se déplaçant sur une musique heavy metal avec un visage énervé. Mais cela semblait trop facile et l'idée qu'il y ait différentes musiques diffusées était plus intéressante. Le projet a donc évolué en la conception d'un robot à plusieurs options diffusant différents morceaux de musique.

Ce projet a permis à Jehan de mettre en application les connaissances acquises depuis ses années de lycée en SI (science de l'ingénieur). Le robot n'est peut-être complètement achevé, mais la partie musique ayant été faite, on peut dire que le défi d'associer musique et électronique a été remporté.

b. Léna D'Antoni

Léna est également élève en 2ème année de prépa intégrée à Polytech Nice-Sophia. Elle aime la musique, la science-fiction et les jeux : un petit robot interactif pouvant diffuser de la musique était une idée parfaite. Après moult réflexions, et prenant en compte son niveau moyen, il a été décidé d'une liste d'options facilement réalisables qui seraient ajoutées au robot au fur et à mesure de l'avancement. Même si l'électronique n'est pas son domaine de prédilection, Léna a su prendre plaisir à confectionner ce robot. La satisfaction de mettre en application la théorie vue en cours, mais surtout de voir que ce que l'on fait fonctionne, a été une très bonne expérience.

Malgré des bugs qui ont largement retardés l'avancement du projet, la satisfaction fut tout même présente.

c. Planning prévisionnel

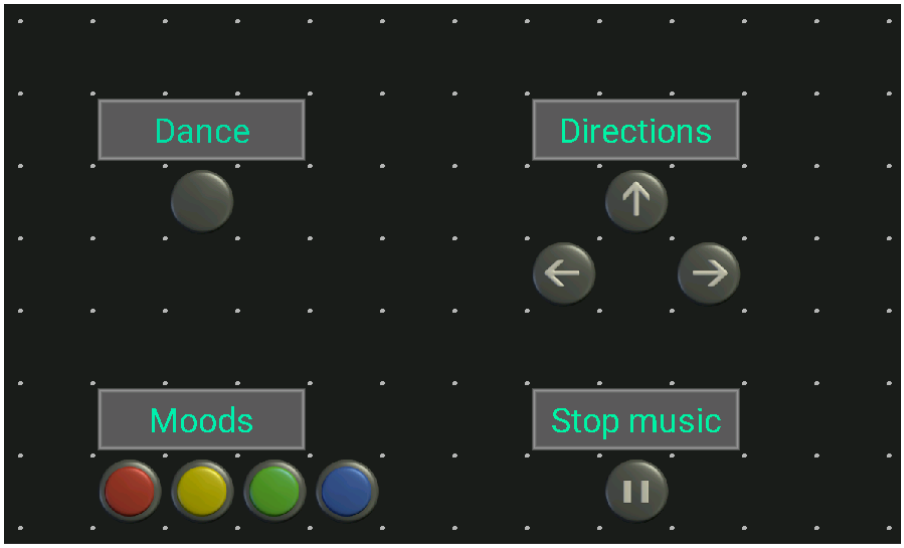
	JEHAN	LÉNA
Semaines 1 à 3	- recherche d'idées de projet - récupération du matériel nécessaire au démarrage du projet	
Semaines 4 à 7	- gestion des distances	- contrôle des moteurs
Semaines 8 à 11	- gestion de la musique - écran LCD	- fonction danse
Semaines 12 à 16	- assemblage	- bras motorisés
Semaines 17 à 24	- assemblage - correction des bugs - design	

d. Planning réel

	JEHAN	LÉNA
Semaines 1 à 3	- première idée de projet	
Semaine 4	- évolution du projet : Musical-e Robot - récupération du matériel nécessaire au démarrage du projet	
Semaines 5 à 10	- prise en main de l'écran - carte arduino cramée donc attente d'une nouvelle	- prise en main de L298N - création des fonctions : • forward() • left() • right()
Semaines 11 à 15	- code des visages - recherche de diffusion du son	- création de la liaison BT - amélioration des fonctions - création de dance()
Semaines 16 à 21	- code de la musique	- bugs récurrents des roues - code servo-moteurs
Semaines 22 à 23	- assemblage	
Semaine 24	- écriture du rapport	

II. Présentation du projet

Musical-e Robot est un robot possédant plusieurs fonctionnalités gérées grâce à un smartphone. Il a été pensé pour divertir et permettre de toucher à différents domaines : la mécanique, la connexion sans fil, le son, l'image. Les options du robot sont initialement : la commande des moteurs pour lui permettre de se déplacer dans l'espace (« directions ») ; la gestion de son humeur qui se traduit par différentes têtes affichées sur un écran (« moods ») ; la danse représentée par le robot effectuant une chorégraphie (« dance »). A chaque instant le robot diffuse une musique qui représente son humeur mais l'utilisateur peut décider de couper le son grâce à un bouton stop (« stop music »).



Capture d'écran de l'interface de gestion des options

Liste du matériel nécessaire pour la réalisation du robot :

- Une carte Arduino Mega 2560
- Un châssis avec des roues omnidirectionnelles qui permettent le déplacement du robot
- un écran LCD qui représente la tête
- 2 servo-moteurs pour articuler les bras
- un capteur HC-SR04 ultra-sons pour éviter les obstacles
- une carte L298N (H-Bridge) qui contrôle les moteurs des roues
- un module bluetooth HC-06 qui assure la liaison entre le smartphone et le robot
- un module MicroSD Card Adapter avec une carte SD
- une enceinte qui diffuse la musique
- une batterie rechargeable Li-poly RC

La confection du robot semblait abordable pour nos niveaux respectifs, c'est ce qui nous a poussé à faire ce choix. Le but était de réaliser un maximum d'options tant qu'on en avait le temps pour obtenir un robot aussi complet que possible. Malheureusement tout ne s'est pas passé comme prévu à cause de bugs et de contre-temps qui nous ont retardés.

a. Commande des moteurs

La commande des moteurs se fait grâce à une connexion bluetooth permise par le module HC-06. Nous avions déjà appris à utiliser ce module lors du cours Électronique avec Arduino, il a donc suffi de réutiliser le code fait en TP et de l'adapter au projet. Pour créer l'interface de gestion des options nous avons utilisé une application Android utilisée en cours : Bluetooth Electronics.

Les moteurs sont gérés à l'aide de la carte L298N, une carte spécialisée dans le contrôle des moteurs qui utilise des ponts en H. On visse la paire de fils de chaque moteur aux PIN réservés sur la carte L298N : un fil gère le sens horaire, l'autre le sens anti-horaire. Puis on relie L298N à la carte Arduino Mega en branchant les fils sur les sorties PMW de la Mega. On alimente les moteurs grâce à une batterie rechargeable Li-poly RC car la carte seule n'est pas assez puissante pour les alimenter correctement.

Pour gérer la vitesse du moteur il faut utiliser analogWrite et associer une valeur comprise entre 0 (vitesse nulle) et 255 (vitesse très élevée) à chaque pin des moteurs.

Il existe plusieurs fonctions qui permettent au robot de se déplacer :

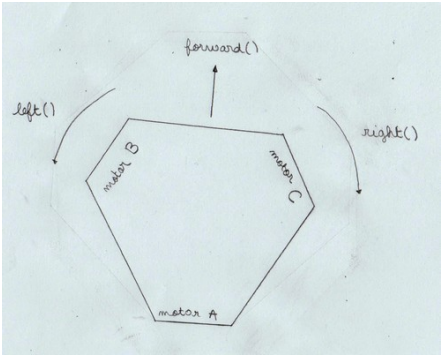
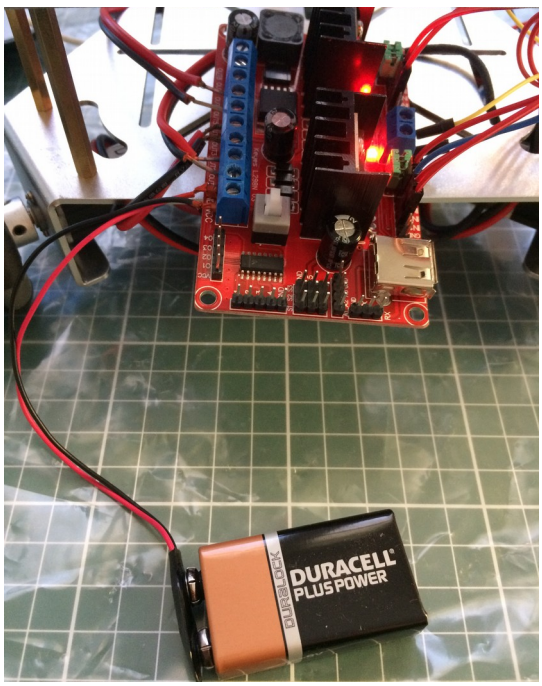


schéma en vue du dessus

forward() : permet au robot de se déplacer tout droit. Le moteur B dans le sens antihoraire et le moteur C dans le sens horaire.
left(): permet au robot de tourner à gauche. Tous les moteurs tournent dans le sens horaire.
right() : permet au robot de tourner à droite. Tous les moteurs tournent dans le sens antihoraire.
stopRobot() : aucune des roues ne tourne. Le robot est à l'arrêt.

Seulement à force d'ajouter des lignes de code au programme principal et de câbler de nouveaux modules sur la carte Arduino, les moteurs ont commencé à se dérégler. Au début on pensait à un manque de puissance, on a donc échanger le pile 7V contre une batterie lithium 12V mais sans résultats. Nous avons ensuite recâblé les fils des moteurs mais il n'y a pas eu de changements non plus. Pourtant, lorsqu'on teste chaque roue indépendamment les unes des autres elles tournent toutes les 3 dans les deux directions. Comme nous perdions beaucoup de temps à tenter de faire marcher les 3 roues correctement nous avons décidé d'orienter la face du robot du côté des 2 roues encore performantes.



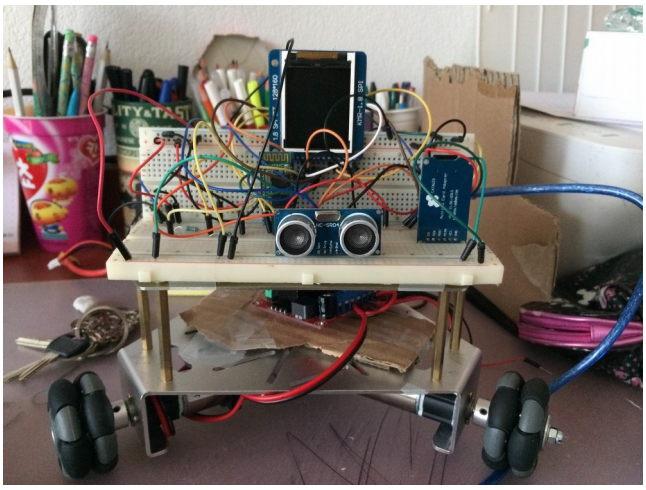
Moteurs alimentés par une pile 7V

b. Gestion de la distance

La gestion de la distance se fait grâce au module HC-SR04 déjà utilisé en cours. Nous l'avons donc adapté au projet afin qu'à partir d'une distance limite, ici 15cm, le robot s'arrête. Il peut ainsi éviter les obstacles et d'être abîmé. Théoriquement le code fonctionne, nous l'avons testé avec un Serial.println(). En effet à partir d'une certaine distance, il est affiché sur le moniteur série la distance. Mais une fois dans le code principal, et au lieu d'afficher la distance on demande l'arrêt du robot avec stopRobot(), cela ne fonctionne pas.

Dans un premier temps, nous avons créé une fonction distance() qui devait, lorsqu'elle était appelée, stopper le robot à l'approche d'un obstacle ou d'un mur. Seulement le robot continuait de rouler peu importe sa distance à l'obstacle. Nous avons donc ensuite directement inclus les lignes de code de la fonction distance() dans les fonctions forward(), left() et right(). Après cela le robot ne se stoppait toujours pas lorsqu'il roulait et qu'on lui mettait soudainement un obstacle. En revanche lorsque l'obstacle était présent avant de faire rouler le robot, celui-ci ne démarrait pas.

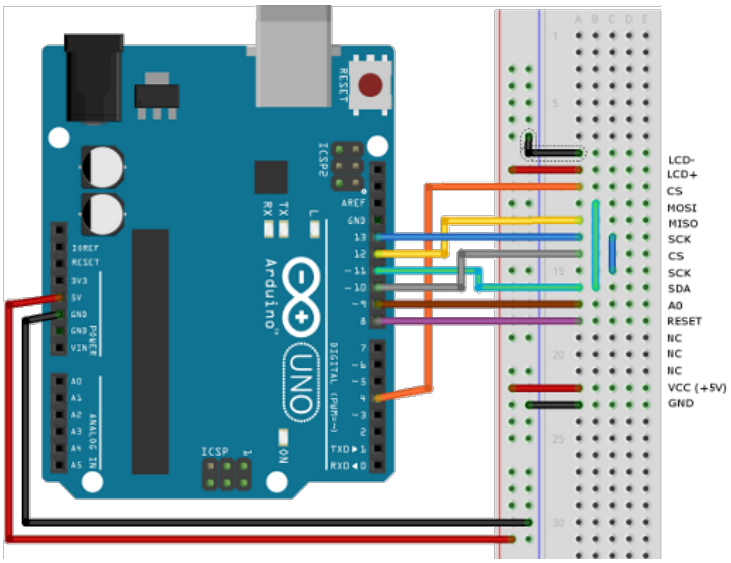
Puisqu'on ne trouvait pas de solution à ce problème, et que le module prenait de la place et de la puissance, nous avons décidé de retirer la partie gestion de la distance.



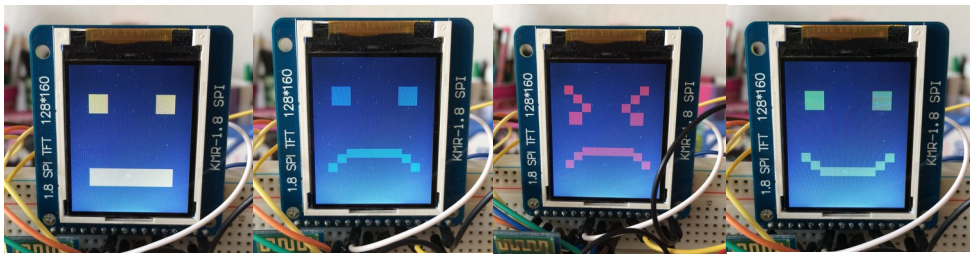
Musical-e Robot muni du module HC-SR04

c. Visage du robot

Pour représenter le visage du robot nous utilisons un écran LCD KMR-1.8 SPI. Nous avons trouvé sur internet le schéma de câblage de cet écran puis nous l'avons reproduit sur notre carte Arduino. La prise en main de l'écran a été assez rapide et les 4 visages représentants les 4 humeurs du robot ont été rapidement codés. Il a suffi de télécharger les bibliothèques TFT.h et SPI.h et d'utiliser les fonctions qui permettent de créer des rectangles (myScreen.rect) et de remplir (myScreen.fill) afin de créer des visages colorés.



Câblage de l'écran LCD (http://www.benbarbour.com/arduinolcd/)



Il existe 4 fonctions qui permettent de dessiner les 4 visages :

neutre() : dessine une visage neutre jaune. Il faut appuyer sur le bouton jaune sur l'application pour que ce visage s'affiche. Une musique en relation avec cette émotion se déclenche également.

triste() : dessine un visage bleu triste. Il faut appuyer sur le bouton bleu sur l'application pour que ce visage s'affiche. Une musique en relation avec cette émotion se déclenche également.

colere() : dessine un visage rouge en colère. Il faut appuyer sur le bouton rouge sur l'application pour que ce visage s'affiche. Une musique en relation avec cette émotion se déclenche également.

joie() : dessine un visage vert content. Il faut appuyer sur le bouton vert sur l'application pour que ce visage s'affiche. Une musique en relation avec cette émotion se déclenche également.

d. Bras motorisés

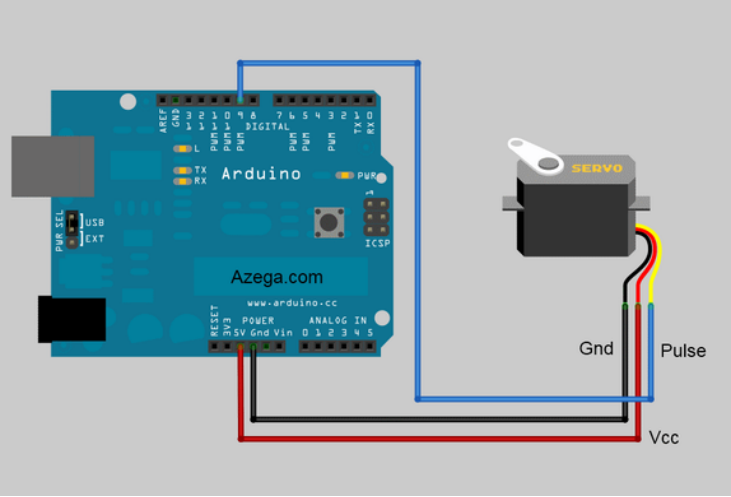
Lorsqu'on appuie sur le bouton « dance » le robot devait tourner en rond, activer deux servo-moteurs représentant ses bras, et diffuser de la musique. Pour cela plusieurs fonctions devaient se déclencher :

servo() : les deux servo-moteurs sont programmés pour monter puis descendre grâce à deux boucles : l'une faisant monter le servo d'un angle de 35° à 100° avec un pas de 1, l'autre faisant le mouvement inverse. Pour cela on utilise myservo.write(pos) qui dit au servo d'aller à la position pos (pos allant de 100° à 35° ou inversement).

stopServo() : qui se déclenche à la fin de la boucle de servo() et qui permet de remettre le servo à son angle initial.

dance() : qui fait tourner le robot en rond.

Le problème est qu'un seul servo-moteur prenait énormément d'énergie à la carte Arduino. En effet, lorsqu'on le branchait, l'écran se brouillait puis devenait blanc. Il aurait fallu le brancher sur une batterie, comme pour les moteurs, sauf comme dit précédemment, nous avons entrepris cette partie trop tard et quand nous nous en sommes rendus compte nous ne pouvions plus récupérer de matériel. Nous avons donc décidé de retirer cette partie du projet pour éviter que cela ne se répercute sur les autres options.

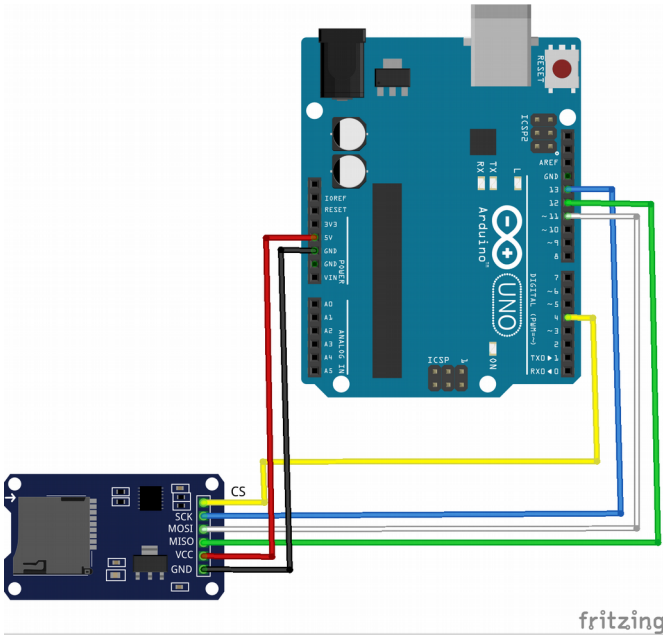


Câblage d'un servo-moteur

e. Diffusion de la musique

Pendant plusieurs semaines nous nous sommes demandés de quelle manières nous allions diffuser la musique. D'abord nous avions pensé inclure un buzzer qui reproduirait des mélodies dans un style 8-bit. Puis nous avons trouvé une vidéo d'une personne diffusant une musique sur une enceinte reliée à la carte Arduino. Cette idée nous a plu et dès lors nous avons fait de nombreuses recherches.

Pour diffuser de la musique à partir d'une enceinte directement reliée à Arduino il faut une carte micro SD contenant des fichiers musiques .wav, un module adaptateur de carte micro SD, une enceinte et deux bibliothèques Arduino : TMRpcm.h et SD.h. Il a aussi fallu souder deux fils à la sortie de l'enceinte, un fil pour la masse et l'autre pour le pin de sortie de l'Arduino.



Câblage du MicroSD Adapter

La prise en main de la bibliothèque a été intuitive et des sites listaient les fonctions les plus utilisées. Les deux fonctions que nous utilisons et qui sont issues de la bibliothèque sont :

audio.setVolume(n) avec n compris entre 0 et 7 qui gère le volume du morceau.

audio.play(fichier.wav) qui joue le fichier.wav.

audio.stopPlayback() qui arrête la musique.

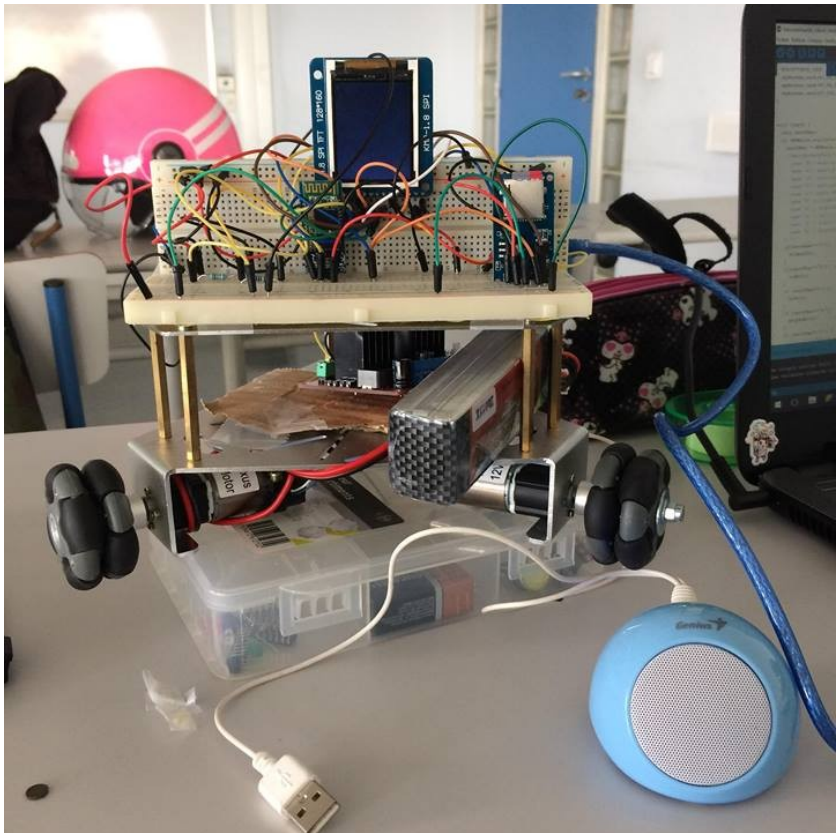
En dehors du code principal, l'ensemble marchait très bien Il fallait utiliser un switch sur BTSerie.read() et selon le caractère qui était envoyé il enclenchait telle ou telle musique, ou bien l'arrêtait complètement. Lorsqu'on inclut le switch dans le loop cela ne marche pas. En effet, certains caractère sont appelés deux fois, une fois pour activer le son dans le switch, une autre fois pour activer l'écran dans les if. Cela créait des contre-indications. Ensuite nous avons directement placé les audio.play() dans les if mais ça ne marche pas correctement : l'écran devient blanc, une roue tourne sans qu'on sache pourquoi. Nous avons essayé de corriger ce problème en utilisant la librairie SPI car, l'écran et la carte SD ayant tous les deux besoin des pins MISO , MOSI, et SCK, nous pensions que le problème pouvait être une erreur de

communication entre la carte et les deux modules. Nous avons affecté un CS à l'écran et un à la carte SD afin de pouvoir sélectionner sur quel module la carte devait envoyer les informations mais sans succès.

III. Conclusion

Ce projet a été une expérience enrichissante et nous a permis de mettre en application nos connaissances ainsi que d'être confrontés à la création d'un projet de A à Z, s'entend ici de l'idée jusqu'à un certain aboutissement. Il nous a aussi permis de comprendre de nouvelles utilisations et fonctions de la carte Arduino à travers les différentes librairies que l'on a utilisé. Toutes les spécificités que nous voulions inclure au robot n'ont pas été réussies à cause de problèmes rencontrés tout au long du projet. Nous sommes venus à la conclusion que notre choix de carte n'a pas été judicieux et que nous aurions du opter pour quelque chose de plus puissant qui aurait pu supporter l'ajout de plusieurs modules (ici le module bluetooth, l'écran LCD, 2 servo-moteurs, 3 moteurs continus, l'adapatateur microSD, le module à ultrasons). Si nous avions pu aller au bout de nos idées, nous aurions aimer créer un design afin de cacher les fils et les modules.

En outre nous avons également appris à travailler ensemble, à se répartir équitablement les tâches et à chercher nos propres réponses sans toujours demander à un professeur. Cette première confrontation à l'élaboration de projets et au travail en équipe fut enrichissante dans bien des domaines.



ANNEXE

Les PIN utiles de la Arduino Mega :

- PWM : 2 - 13 et 44 - 46.
- MISO (Master In Slave Out) : 50
- MOSI (Master Out Slave In) : 51
- SCK (Serial Clock) : 52

Librairies utilisées :

- SoftwareSerial : utilisée pour le BT
- Servo : utilisée pour les servo-moteurs
- TFT : utilisée pour les écrans
- SPI (Serial Peripheral Interface) : utilisée pour les modules périphériques
- SD : utilisée pour les cartes SD
- TMRpcm : utilisée pour les fichiers audio