Compte Rendu Arduino

Projet : Mad-Simon

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

Université Nice Sophia Antipolis

POLYTECH NICE SOPHIA

Sommaire

1) But du projet

2) Matériels utilisés
    a) La carte Arduino
    b) Les Leds
    c) Le module 16 pwm
    d) Le Piezzo Buzzer
    e) Les puces Bluetooth

3) L’emballage

4) Le code
    a) La structure générale
    b) Le jeu
    c) Les erreurs
    d) Le Bluetooth

5) Les problèmes rencontrés

6) Conclusion

1) But du projet

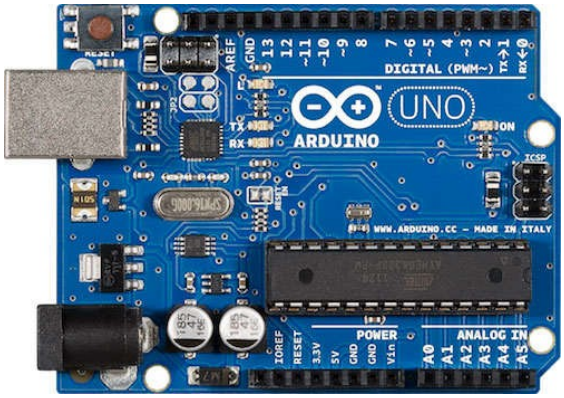
Notre but principal était de proposer un projet interactif et utile, en utilisant l'Arduino ainsi qu'une puce Bluetooth.
Après quelques semaines de recherche, nous décidons de proposer plusieurs amélioration a un jeu déjà existant, Simon.
Pour ceux qui ne connaissent pas ce jeu, je vais le décrire.
De base, le jeu est constitué d'un module qui, lorsqu'il est en action, émet une suite de son et une lumière de couleur (toutes deux corrélés) simultanément et le but du joueur est de reproduire cette série jusqu'à qu'il se trompe. Il existe 4 couleur/son différent.

Nous voulions rendre ce jeux plus attractif : tout d'abord, nous voulons pouvoir jouer à deux joueurs, ces deux joueurs devrons jouer avec leurs téléphone portable et nous voulions aussi augmenter le nombre de couleurs/sons.

Pour celà, nous allons avoir besoin d'une boite contenant tous le matériel nécessaire afin de pouvoir créer notre jeu, le Mad-Simon.

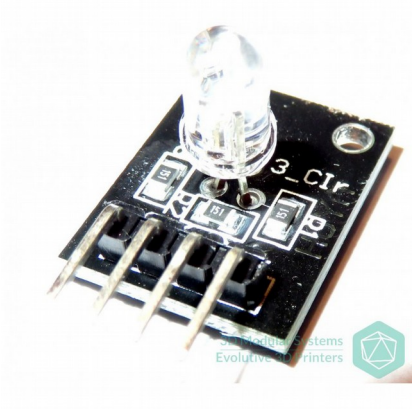
2) Matériel utilisé

a) La carte Arduino.



Tout d'abord, nous avions besoin d'une carte qui puissent enregistrer un code et l'exécuter par la suite. Nous avons utilisé pour ce projet la carte Arduino uno.
Cette carte possède un port micro-USB qui permet le transport de données (code) ainsi que l'alimentation de la carte en 5V, une dizaine d'entrée numérique ainsi que 6 entrée analogique.
Elle possède ensuite d'autre entrée qui sont présente pour pouvoir transmettre aux composant du 5V ou 3V.

b) Les LED

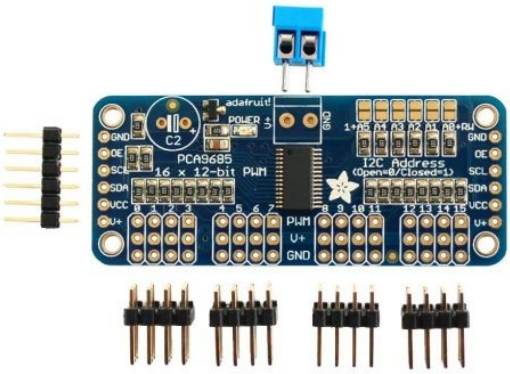


Nous avions besoin de 6 LED RGV car nous voulions pouvoir changer la couleurs des LED si nous le voulions.
Nous avons donc choisis ce module ci contre qui est conçu pour l'Arduino, nous permettant d'avoir les LED et les résistances adéquates (pour du 5V), nous évitant ainsi de faire 3 soudures supplémentaires par LED, soit 18 soudures. De plus, les soudures manuelles sont moins solides que les soudures industrielles, donc il y avais plus de risques de mauvais fonctionnement si nous avions choisis des LED et soudé sur ces LED les résistances adéquates que de choisir ces modules.

Mais nous avons un petit problème : chaque LED utilisent 3 entrée PWM, donc on avait besoin de 18 PWM sur l'Arduino or il n'en avait pas autant.
Pour résoudre ce problème, nous avons utilisé le module ci-dessous.

Nous avons due ensuite faire un câble qui puisse brancher chaque base de la LED à la base de l'Arduino, penser aussi à la base du buzzer et aux deux bases des puces Bluetooth, ce qui fait un câble à 10 sorties.

c) Le module 16 PWM.



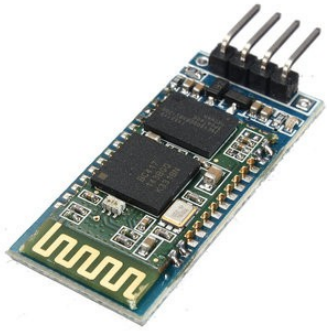
Ce module, en réalité, est fait pour des moteurs Arduino. Il permets d'augmenter le nombre de PWM de l'Arduino et aussi fournir au moteur le courant nécessaire grâce a l'entrée V+.
Nous avons utilisé juste le fait que ce module augmente le nombre de PWM. Pour cela nous avons branché 5 LED sur ce module et une sur l'Arduino de base.
Ainsi nous avons pu connecter toutes nos LED à l'Arduino.

c) Le Piezzo buzzer

Le Piezzo buzzer est un module dont le but est d'émettre un son (basique). Ce buzzer a besoin d'être alimenté en tension 5V et aussi d'être connecté à l'Arduino par les entrée PWM. L'utilisation du buzzer est assez simple grâce à la fonction tone de l'Arduino

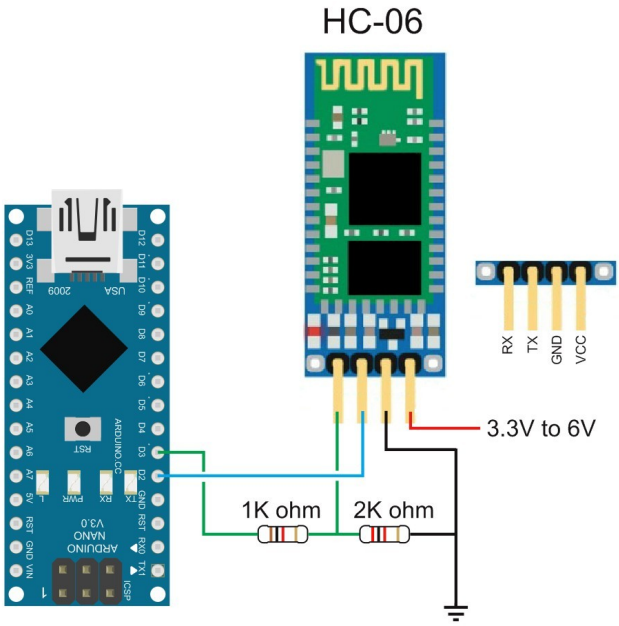
Pour plus d'informations :
https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/

d) Les puces Bluetooth



Nous avons utilisé deux puces Bluetooth HC06. Ces puces permettent d'envoyer et de recevoir de l'information entre elle et le portable auquel elles seront connecter .

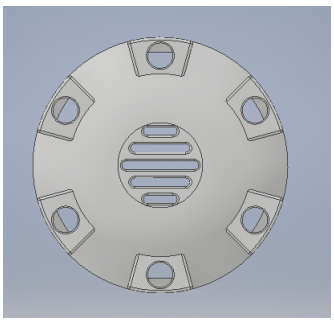
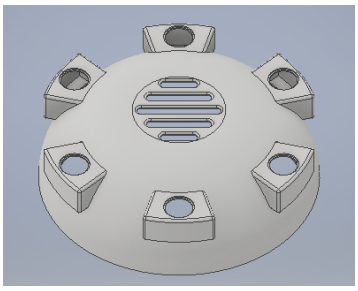
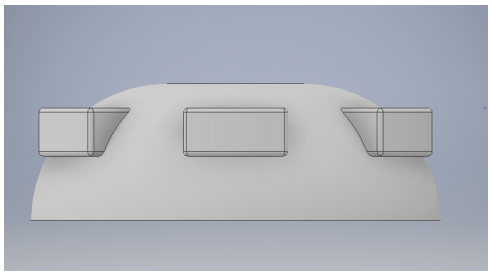
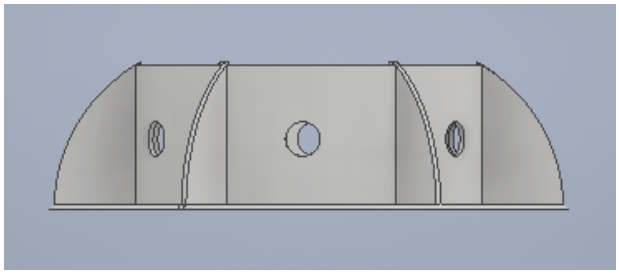
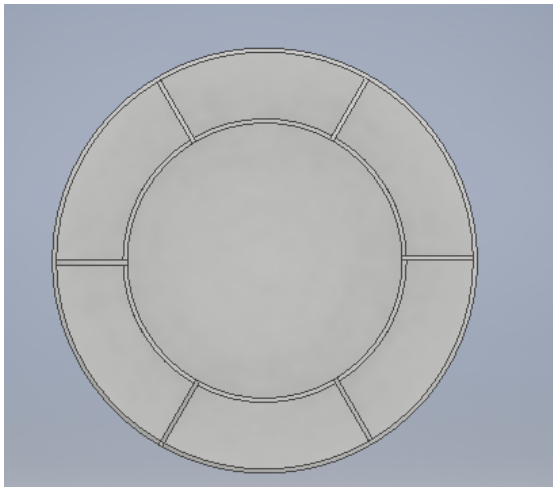
Cette puce fonctionne en 3.3V, nous avons donc fait un pont diviseur de tension avec des résistances comme montré ci-dessous :



3) L’emballage

Nous voulions avoir comme boite pour contenir tout le circuit électronique quelque chose qui sorte du commun. Au départ, nous voulions faire une structure avec des plaques de bois, mais nous nous sommes vite rendu compte que faire l’emballage en bois demandait beaucoup trop de temps pour un résultat pas terrible. C’est pourquoi nous avons choisis d’utiliser une imprimante 3D ainsi que le logiciel Autodesk Inventor pour modéliser notre boite sur l’ordinateur.

Voici a quoi ressemblais le croquis 3D sur Inventor :

Nous avons ensuite fait imprimer notre objet, cette impression a pris plus de 18 heures pour faire nos deux pièces. Le rendu est plutôt satisfaisant à part la partie supérieur du cache où l’on voit un peu les marques de fabrication. Nous avons voulu poncer cette partie mais le résultat était pire qu’au départ c’est pourquoi nous avons laisser la boite en état.

Il y a un seul petit problème, la boite étant noir, nous ne voyons pas trop les LED qui s’allument.

4) Le code

a) La structure générale

Pour la structure du code nous avons beaucoup réfléchi à un squelette propre qui nous permette de pouvoir coder plus facilement, nous avons pensé à plusieurs astuces qui nous on facilité le travail ensuite. Donc nous avons opté pour le choix d'associer chaque couleur à un nombre de 0 à 5 compris ce qui permet de ranger la suite de couleur dans une liste d'entier.

```
led 0 = VERT
led 1 = YELLOW
led 2 = BLEU
led 3 = MAGENTA
led 4 = CYAN
led 5 = ROUGE
```

La structure du code du Mad-Simon commence tout simplement avec les librairies que l'on importe. Il fallait donc que l'on importe le PWM Servo Driver pour gérer toutes les LED et Software Serial pour le Bluetooth.

```
#include<Adafruit_PWMServoDriver.h>
#include <SoftwareSerial.h>  //Software Serial Port
```

Ensuite nous avons créé le PWM Servo Driver et les deux Software Serial qui correspondent au module PWM et aux deux puces Bluetooth. Pour les données de chaque LED nous avons pensé à les stocker dans des listes.

```
int liste[100];
const int led[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,15,15};
const int coul[]= {0,1,0,1,1,0,0,0,1,1,0,1,0,1,1,1,1,1};
const int freq[]= {200,800,400,1000,600,1200};
```

En effet, les ports de chaque LED est stocké dans la variable led, leur couleur unique est stocké dans la variable coul et la fréquence du son associé a cette même LED est stocké dans la variable freq.

Ensuite il y a notre void setup, où l'on Set touts les ports de l'Arduino que l'on utilise et le Baud de chaque Software série.

```
void setup(){
  pwm.begin();
  pwm.setPWMFreq(50);
  pinMode(2, OUTPUT);
  pinMode(Tx1, OUTPUT);
  pinMode(Tx2, OUTPUT);
  pinMode(Rx1, INPUT);
  pinMode(Rx2, INPUT);
  BTSerie1.begin(9600);
  BTSerie2.begin(9600);
  Serial.begin(9600);
  Serial.println("debut");
  BTSerie1.write("choisissez votre type de partie !");
  BTSerie2.write("choisissez votre type de partie !");
}
```

Il y a quelques fonctions qui sont très utile comme la fonction random. Il faut savoir que la fonction random existe déjà en Arduino, mais elle n'est pas vraiment aléatoire, en effet c'est toujours la même liste de nombre qui passe même après avoir redémarré l'Arduino. C'est pourquoi notre fonction random se base sur l'horloge de l'Arduino pour être vraiment aléatoire.

```
int Random(int j){
  return millis()%j;
}
```

De plus il nous fallait un fonction qui allume une LED et une qui l'éteint avec son numéro de LED mis en paramètre.

```
void displayColor(int n){
  for(int j=0; j<=2; j++){
    pwm.setPWM(led[(n*3)+j],(coul[(n*3)+j])*4096,-4096+(coul[(n*3)+j])*4096);
  }
}

void undisplayColor(int n){
  for(int j=0; j<=2; j++){
    pwm.setPWM(led[(n*3)+j],0,4096);
  }
}
```

Grâce à ces deux fonctions ci-dessus on a pu faire d'autres fonctions essentielle au bon déroulement du programme. Comme par exemple la fonction error qui n'utilise que la fonction undisplayColor pour éteindre les LED après les avoirs allumés en rouge pour signaler une erreur. Il y a aussi la fonction SonEtLum qui utilise displayColor et UndisplayColor ainsi que tone et qui permet d'allumer une LED pendant un certain temps et de produire le son associé à cette LED pendant ce même temps.

```
void SonEtLum(int n){
  displayColor(n);
  tone(2, freq[n], 1000);
  delay(1000);
  undisplayColor(n);
}
```

On a aussi fait une fonction qui permet de vérifier si la tentative de jeu de l'utilisateur est correcte où si elle ne correspond pas aux couleurs de la suite courante. On la présentera plus bas.

```
void error(){
  int i = 0;
  ResetListe();
  tone(2, 150, 1500);
  while(i<8){
    for(int j=0; j<6; j++){
      pwm.setPWM(led[j*3+2],0,4096);
      pwm.setPWM(led[j*3+1],0,4096);
      pwm.setPWM(led[j*3],4096,0);
    }
    delay(100);
    for(int k=0; k<6; k++){
      undisplayColor(k);
    }
    delay(100);
    i++;
  }
}
```

b) Le jeu

La fonction Jeu est appelé pour débuter le type de jeu annoncé au début de la partie. Elle utilise toute les fonctions présenté précédemment. Elle créé la liste de nombres aléatoire au fur et à mesure de la progression de la partie.

```
void Jeu(int typePartie){
  for(int j: led){
    pwm.setPWM(j,0,4096);
  }
  ResetListe();
  liste[0]= Random(6);
  int verif = typePartie;
  while(verif!=0){
    int i =0;
    while(liste[i]!=12345){
      SonEtLum(liste[i]);
      delay(500);
      i++;
    }
    verif = Reponse(verif,i);
    delay(500);
    liste[i]= Random(6);
  }
}
```

c) Les erreurs

Durant tout le projet nous avons rencontré différentes erreurs qui vont de l'oublie du point virgule à l'erreur impossible à déceler.
Comme première erreur nous avons eu une NullPointer, en résumé notre liste qui contenait la suite de couleur était remplie de null ce qui posait problème quand on l'étudiait, c'est pourquoi nous avons pensé à faire une fonction qui reset la liste. Cette fonction remplie la liste de 12345 ce qui permet de faire des opérations sur celle-ci.

```
void ResetListe(){
  for(int k=0;k<100;k++){
    liste[k]=12345;
    }
}
```

L'autre erreur qui nous a demandé de la recherche, c'est cette fonction random dasn arduino, en effet la fonction random qui existait déjà dans Arduino a un problème majeur, c'est toujours la même. En effet la suite de nombre aléatoire ne changeait jamais (par exemple c'était toujours {1,1,2,5,1,3,5, … ,1})

On a eu un problème quand l'utilisateur ne répondait pas, donc il fallait lui mettre une limite de temps pour éviter que le programme ne fasse plu rien. Donc nous avons fait ces ligne de code pour que le programme considère que la réponse est fausse si il prend trop de temps à répondre.

```
unsigned long t = millis();
while(true){
  if(millis()-t>10000){
    error();
```

Nous avons aussi rencontré des erreurs et des problème avec les fonctions ainsi que les puces bluetooth.

d) Le Bluetooth

Pour le Bluetooth, nous avons dus utiliser les fonctions associé au package SoftwareSerial.h. Une des fonctions permet de regarder si la puce Bluetooth reçois des informations (BTSerie1.available( )), si oui on prend la valeur des données reçu

avec la fonction read( ), pour ensuite renvoyer des informations à l'appareil connecté à cette puce on utilise la fonction write( ).

```
if(BTSerie1.available()) {
  word rep = BTSerie1.read();
  if(rep == 48+6){
    Jeu(1);
    BTSerie1.write("choisissez votre type de partie !");
  }
```

Ici on voit que l'on vérifie si la puce Bluetooth 1 a reçue des donnée, si en effet elle en a reçue, on récupère ses données avec read( ), on les vérifies et ensuite si elles sont égales à 48+6 on exécute ce qu'il y a dans la boucle if.
Nous avons essayer de faire un mode de jeu à deux appareils, mais l'Arduino n'écoute que le port du dernier software qui a été begin avec la commande BTSerie2.begin(9600), en effet l'Arduino ne peut pas écouter deux ports en même temps.

5) Les problèmes rencontrés

Nous avons rencontré beaucoup de problèmes qui nous ont fait perdre énormément de temps dans notre projet :

- Nous n'avions pas assez réfléchis de base à comment assembler tous nos éléments et donc nous avons perdu beaucoup de temps sur des problèmes qui aurai pu être résolu bien avant (exemple : Pour le câble de la base qui comporte 10 sorties, nous avons du nous y reprendre a plusieurs fois car nous avons oublié au départ la base du buzzer et ensuite celles des deux modules HC06.

- Au niveau de l'assemblage, nous voulions que tout les composants, ainsi que les câbles qui les relient, puissent entrer dans la boite que nous avions fabriqué. C'est pourquoi nous avons choisis la soudure pour connecter les composants entre eux.
Mais nos soudures ne sont pas si solides, nous avons eu plein de problèmes de câbles qui se dessoudait et de faux-contact (nous en avons toujours).

- Au niveau de l'interface graphique du portable, nous utilisons le programme Bluetooth Electronics. Cette application nous permet le bon fonctionnement du jeu mais les boutons présents ne possèdent que 4 couleurs, or nous en avons besoin de 6. Nous avons cherché d'autres applications qui pourraient nous satisfaire mieux que celle-ci mais sans succès.

Pour l'instant nous utilisons donc cette applications avec deux autres boutons qui n'ont pas de couleurs.
Pour améliorer notre projet, nous pourrons développer notre propre application qui permettrais d'avoir notre interface.

- Au niveau des puces Bluetooth, Nous avons ressui a faire fonctionner notre jeu sur une seul des deux puces Bluetooth. Le jeu ne fonctionne pas sur l'autre puce Bluetooth alors que celle ci fonctionne bien.
Nous pensons que le problème viens du fait que nous ne pouvons pas déclarer deux software Serial sur la même Arduino.

- Au niveau de la boite, celle ci est noir et donc absorbe la lumière. Nos LED sont donc moins performantes et on ne vois pas trop la lumière si on met le couvercle. De plus, pour mettre la boite, il nous aurais fallu avoir une batterie externe de 5V pour la rentrer dans notre boite et éviter qu'il y ai un câble qui sorte.

- Au niveau de la connexion entre les puces et le portable, les chiffres qu'envoie le portable ne sont pas les même que ceux que la puce doit recevoir, la puce reçoit 48 lorsque le portable envoie 0. Nous avons vu que 0 est codé 48 en ASCII. Nous avons donc translaté tous nos nombre de 48.

6) Conclusion

Pour conclure, on peux dire que notre projet est relativement une réussite, Nous avons réussi a faire fonctionner notre projet. De plus, ce projet a été une vrai source d'enrichissement personnel pour nous, il nous a permis de savoir ce qu'était un projet sur une année.

Il nous a permis aussi de réfléchir sur la façon dont nous devons résoudre les problèmes que nous avons rencontré.
Malgré tout, il est encore possible d'améliorer notre projet, par exemple nous pourrions ordonner nos câbles, en remplacer quelques uns.