PROJET ARDUINO PEIP2

Année 2018-2019

KUBO

Cube de LED

Ombéline Carcouet et Jeanne Léauté

Université Nice Sophia Antipolis

POLYTECH NICE-SOPHIA


Université
Nice
Sophia Antipolis


POLYTECH
NICE-SOPHIA

Sommaire


Université
Nice
Sophia Antipolis


POLYTECH
NICE-SOPHIA

Introduction

Dans la cadre de l'enseignement électronique de notre parcours PeiP, nous avions un projet à réaliser avec une carte Arduino. Notre choix s'est donc porté sur un cube de 8x8x8 LEDs, l'objectif étant de créer un objet décoratif dans une maison ou appartement. L'idée était donc de proposer 3 modes différents contrôlables sur le téléphone par Bluetooth. Premièrement, le mode "veille" permet d'obtenir une lumière d'ambiance avec différents motifs et couleurs ; le mode "horloge" permet d'afficher l'heure sur le cube à fréquence souhaitée ; enfin, le mode "fête" permet d'afficher des motifs plus dynamiques afin d'animer une fête. De plus, ce dernier mode pourrait interagir avec la musique écoutée et allumer les LED en fonctions de la fréquence des sons obtenus.

Dans un premier temps, nous exposerons une vision globale du projet avec les différents modules, puis nous présenterons les difficultés rencontrées et enfin nous conclurons sur ce qui a été réalisé et nous exposerons nos perspectives si nous avions des séances supplémentaires.

Description du projet

Le projet final est un cube de 8x8x5 LED avec deux modes contrôlables par Bluetooth avec l'application Bluetooth Electronics sur téléphone Android. Le "mode veille" permet d'afficher sur le cube des animations lentes et diffuses comme un arc en ciel. Le "mode fête" permet d'afficher des animations dynamiques comme des clignotements rapides mais n'interagit pas avec la musique.
Enfin, on a programmé l'affichage de l'heure en faisant défiler du fond vers l'avant le nombre d'heure puis le nombre de minutes de l'heure courante. D'une autre part, nous avons configuré le module RTC pour qu'il affiche l'heure courante dans le moniteur série. Cependant, on rencontre des difficultés lorsque l'on demande à l'Arduino d'afficher deux nombres quelconques et en parallèle de faire fonctionner le module RTC pour récupérer l'heure courante et l'afficher sur le moniteur série. En effet, l'heure défile bien sur le moniteur série mais les nombres ne s'affichent pas sur le cube. Nous n'avons donc pas réalisé le mode "horloge".

Les modules utilisés

Module Bluetooth

Dans le cadre du projet Kubo, nous avons dû utiliser un module Bluetooth afin de connecter un téléphone à notre cube. Ce module branché à la carte Arduino permet une connexion Bluetooth entre le téléphone et l'ordinateur. Nous avons utilisé l'application Bluetooth Electronics pour créer une interface permettant à l'utilisateur de choisir le mode dans lequel il désir mètre le cube en appuyant sur les différents boutons.

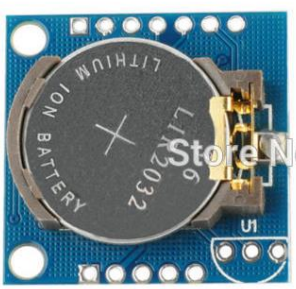
Lorsque l'utilisateur appuie sur l'un des boutons ou le relâche, le module Bluetooth permet une transmission d'information, l'ordinateur reçoit une lettre, ce qui lui permet d'entrer dans l'une ou l'autre des boucles if() du programme et ainsi d'exécuter le mode demander.

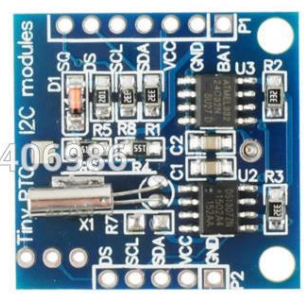
Les LED

Les led qu'on utilise possèdent 4 pattes, la plus longue est la cathode, et les trois autres sont, la masse, l'entrée qui reçoit l'information de la carte Arduino et la sortie qui passe l'information à la led suivante. A l'intérieur du boitier, il y a en fait 3 led de couleurs différentes : une rouge, une bleue et une verte. La librairie NeoPixel d'Adafruit permet de donner une valeur entre 0 et 255 à chaque pixel avec la fonction setPixelColor(i,r,g,b) où i la position du pixel dans la face. L'utilisation de ces led particulières permet donc d'obtenir une infinité de couleurs.

Module RTC

Pour réaliser le mode "horloge", nous avons dû utiliser un module RTC ds1307. Celui-ci permet de récupérer l'heure courante de l'ordinateur et de la stocker. En effet, la carte
Arduino n'a pas la fonctionnalité de garder l'heure en mémoire. On a donc programmé le module RTC une fois puis l'heure courante est sauvegardée dans le module tant qu'on ne retire pas la pile. On peut ensuite récupérer différentes informations comme l'heure, les minutes, ou même l'année avec des fonctions de la librairie RTClib telles que la fonction now().



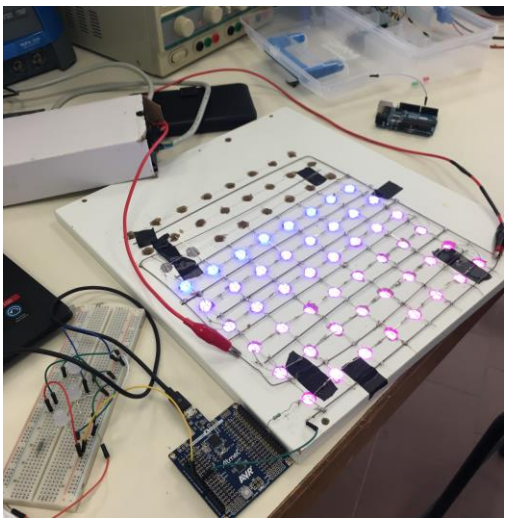


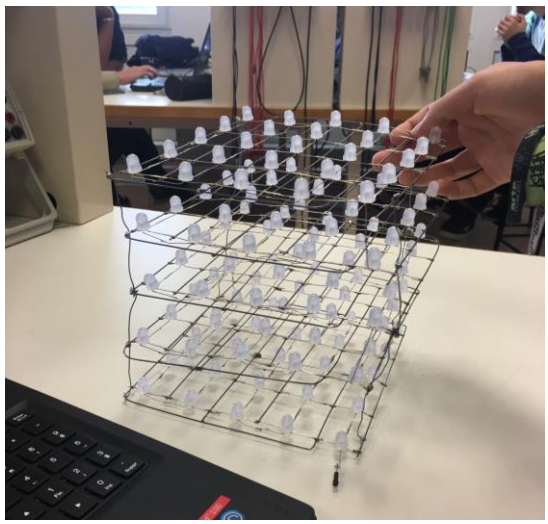
Structure du cube

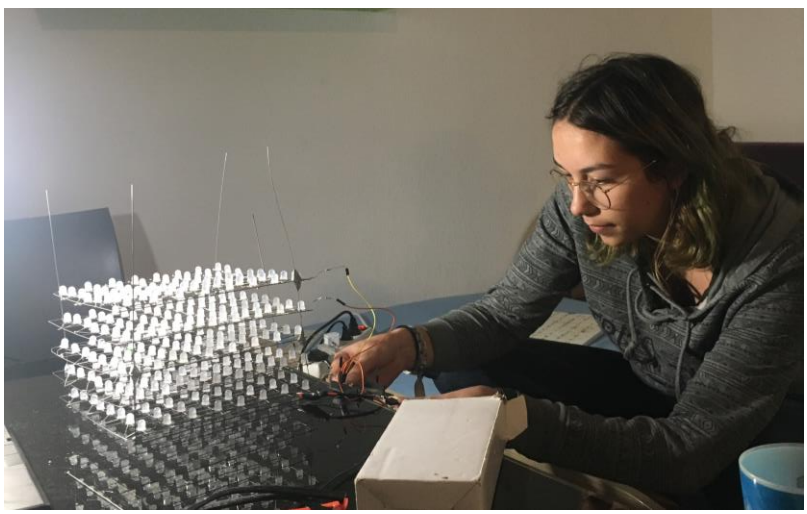
Le cube de led a été construit à l'aide d'un support en bois que nous avons percé de 64 trous aux dimensions correspondant aux led. Nous avions préalablement tracé un quadrillage 8x8 sur cette planche afin d'espacer toutes les led de 3 cm.

Les led sont toutes branchées en série, ce qui correspond à leur fonctionnement puisque lorsque leur mémoire est pleine, les led transmettent les informations à la suivante sans les prendre en compte. Elles sont également toutes reliées à l'alimentation et à la masse du générateur grâce à des tiges métalliques qui permettent de maintenir la structure en place. Toute la structure est fixée à l'aide de soudure.

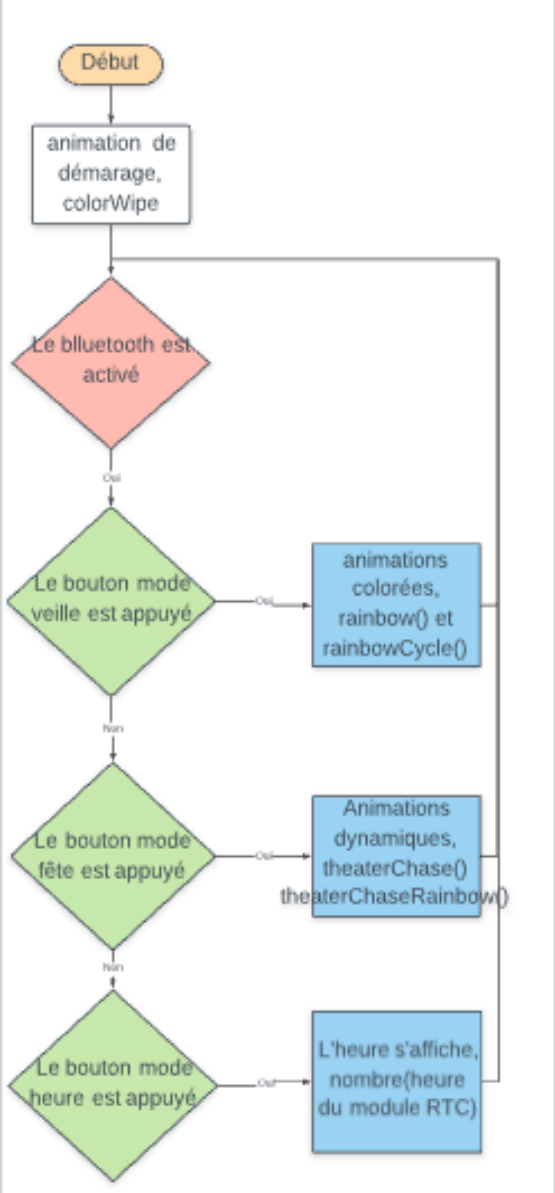
Avec de nouvelles tiges, nous avons superposé les faces pour construire le cube. Toutes les faces sont indépendantes les unes des autres afin de faciliter la programmation. Ainsi toutes les faces sont liées à la carte Arduino via une résistance.







Algorithme du programme



Fonctions du programme

Mode veille :

- La fonction Rainbow parcourt d'une part toutes les valeurs possibles des pixels pour chaque LED de chaque face. Ainsi, elle colore chaque LED du cube d'une couleur différente avec un léger décalage ce qui produit un arc-en-ciel, celui-ci se déplaçant sur le cube.

- La fonction RainbowCycle effectue la même chose que la fonction précédente mais avec un décalage plus grand de telle manière que l'on voie sur le cube une seule couleur de l'arc-en-ciel à la fois avec ses nuances. Les couleurs défilent de sorte que cela parcourt toutes les couleurs de l'arc-en-ciel.

Mode fête :

- La fonction TheaterChase parcourt chaque LED de chaque face et sur 4 LEDs elle fait s'allumer puis s'éteindre les LED chacune leur tour. On voit donc tout le cube "clignoter" d'une couleur.

- La fonction TheaterChaseRainbow fait la même chose que la fonction précédente mais en parcourant toutes les couleurs de l'arc-en-ciel.

Mode heure :

- La fonction nombre permet, grâce à l'utilisation d'un tableau à trois dimensions composées de 0 et de 1, d'afficher deux chiffres sur une face latéral du cube. Les chiffres sont codés de manière assez explicite dans le tableau et la fonction chiffre allume les led du cube correspondant aux positions ayant un 1 dans le tableau.

- On récupère l'heure et la minute courante avec la fonction "now" de la librairie RTClib et on l'affiche sur le moniteur série.

Difficultés

Tout au long des séances, nous avons rencontré quelques difficultés qui ont compromis le bon déroulement du projet.
Tout d'abord, au début de la construction nous avons très vite remarqué qu'à mesure que l'on fixe les led, certaines soudures se brisent, il y avait quatre ou cinq endroits déconnectés à chaque séance qu'il fallait trouver puis ressouder, on a donc perdu environ trente minutes par séance pour tester les faces. Le professeur nous a finalement proposé de mettre du vernis pour fixer les soudures malheureusement on n'a pas pris le temps d'en trouver.
De plus, il y a eu quelques fois des led qui ont explosé probablement à cause de courts-circuits ; parfois nous avons trouvé la cause du court-circuit mais pour certaines fois la raison reste un mystère et nous n'avons pas pu apprendre de nos erreurs.
Par ailleurs, nous avons mal géré notre stock de led puisqu'à la fin, lorsqu'on en a manqué, il n'y en avait plus aucune disponible sur le marché. Pour pallier ce problème, nous aurions dû nous préparer à cette éventualité en commandant plus de led plus tôt.

Nous avons aussi rencontré quelques difficultés ponctuelles, par exemple lors du dernier week-end avant les soutenances, le générateur de tension qui nous permettait d'alimenter toutes les led nous a lâché, puisque le câble de la borne + s'est détaché, nous avons alors dû dévisser le pied de vis et dénuder le câble afin de le rebrancher. Pour cela, nous avons demandé de l'aide à un étudiant de 4ème année en département électronique afin d'être sûr de bien effectuer les manipulations sur le générateur.

Enfin, dans l'essence même du projet, la plus grosse difficulté rencontrée est dans la programmation. En effet, après avoir envisagé toutes les possibilités, nous n'avons pas réussi à comprendre pourquoi la fonction nombre() interfère avec le reste du programme, que ce soit avec le module Bluetooth ou le module RTC, alors qu'elle fonctionne parfaitement toute seule. On aurait donc dû essayer de coder les nombres d'une autre manière mais malheureusement nous n'avons pas eu assez de temps.

Plannings

Comme les plannings ci-contre le montrent, nous avons eu des difficultés à prévoir le temps que nous prendrait la construction du cube. Effet, elle ne s'est achevée que le week-end précédent la présentation. Cela a forcément influencé les autres étapes du projet. La programmation notamment puisque nous avons dû coder sur une seule face sans pouvoir visualiser le rendu final. Nous avons dû abandonner une partie du projet par manque de temps et ne construire que cinq faces au lieu de huit.

Planning initial

Etapes du projet	Séance 1	Séance 2	Séance 3	Séance 4	Oral de janvier	Séance 5	Séance 6	Séance 7	Séance 8
Réalisation du planning	J								
Réalisation du cahier des charges	O								
Tests de branchements et débuts de programmations sur quelques led		O J							
Montage du cube de led			J	O					
Programmation d'images sur le cube (mode veille)			O	J					
Application à la musique :interpretation du signal audio en signal de commande des LEDs									
Application à la musique : lecture du signal pour réaliser les animations des LEDs									
Option supplémentaire : afficher l'heure									
Bluetooth avec téléphone pour changer de mode									
Légende :									
O : Etapes réalisée par Ombéline Carcouet									
J : Etapes réalisée par Jeanne Léauté									
: Etapes non attribuée									

Planning final

étapes du projet	Séance 1	Séance 2	Séance 3	Séance 4	Oral de janv	Séance 5	Séance 6	Séance 7	Séance 8
Réalisation du planning	J								
Réalisation du cahier des charges	O								
Tests de branchements et débuts de programmations sur quelques led		JO							
Montage du cube de led		J	J	J	J	J	J	J	J
Programmation d'images sur le cube (mode veille)		O	O	O	O	O	O	O	O
Application à la musique :interpretation du signal audio en signal de commande des LEDs									
Application à la musique : lecture du signal pour réaliser les animations des LEDs									
Option supplémentaire : afficher l'heure							O	O	JO
Bluetooth avec téléphone pour changer de mode									J
Légende :									
O : Etapes réalisée par Ombéline Carcouet									
J : Etapes réalisée par Jeanne Léauté									
: Etapes non attribuée									

Conclusion

Notre projet avait un but principalement décoratif. Nous n'avons malheureusement pas eu le temps d'accomplir toutes les étapes que nous avions prévues mais une grande partie du projet fonctionne. En effet, le mode veille offre un rendu coloré et agréable lorsqu'on le déclenche via l'application du smartphone. Le mode fête n'est pas celui qui était prévu, c'est à dire des lumières changeant avec les sons, mais ils affichent un rendue dynamique lorsqu'il est activé. La plus grande déception de ce projet est le mode heure qui ne fonctionne pas. Le problème est que nous n'avons pas réussi à comprendre d'où venait le dysfonctionnement et ainsi à le corriger et apprendre de nos erreurs. Les différentes fonctions s'effectuent correctement toutes indépendamment les unes des autres mais pas lorsque la fonction nombre est appelée avec une autre. Ce problème pourrait probablement être résolu si nous programmions la fonction nombre d'une autre manière.

Avec l'expérience que nous avons acquise nous ferions probablement différemment certaines étapes du projet. Premièrement, nous aurions pu mieux estimer le temps à accorder aux différents modules, notamment au niveau de la construction du cube, ses dimensions étaient trop ambitieuses pour le temps imparti. Ensuite, nous partirions directement sur le codage d'un tableau à trois dimensions et des fonctions setPixels qui permettent de cibler les led à allumer plutôt que de perdre du temps sur des fonctions trop spécifiques et compliquées que nous n'avons finalement jamais utilisées.

Si nous avions la possibilité de continuer ce projet sur de nouvelles séances, nous pourrions nous concentrer sur le problème de la fonction nombre() et le résoudre et ajouter le code qui afficherait l'heure à fréquence régulière sur le cube, entrecoupé d'animations neutres. Nous prendrions également le temps de vernir toutes les soudures pour éviter qu'elles ne se détachent de nouveau. Nous pourrions dans un second temps réaliser le mode fête comme il était prévu à l'origine et écrire de nouvelles fonctions qui permettraient de varier davantage les animations du cube sur tous les modes.

Sitographie

https://www.instructables.com/id/Getting-Started-With-NeoPixle-WS2812-RGB-LED/

https://www.carnetdumaker.net/articles/utiliser-un-module-horloge-temps-reel-ds1307-avec-une-carte-arduino-genuino/

http://users.polytech.unice.fr/~pmasson/Enseignement/Elements%20de%20robotique%20avec%20arduino%20-%20Communications%20RF%20-%20Projection%20-%20MASSON.pdf

Annexe :

```
//module
blutooth
#include<SoftwareSerial.h>
#define RX 10
#define TX 11
SoftwareSerial BlueT(RX,TX);
char Data;
//Leds
#include <Adafruit_NeoPixel.h>
#define intensite 150
#define PIN 6
#define NUMPIXELS 64
Adafruit_NeoPixel pixels0 = Adafruit_NeoPixel(NUMPIXELS, 3, NEO_GRB +
NEO_KHZ800);
Adafruit_NeoPixel pixels1 = Adafruit_NeoPixel(NUMPIXELS, 4, NEO_GRB +
NEO_KHZ800);
Adafruit_NeoPixel pixels2 = Adafruit_NeoPixel(NUMPIXELS, 5, NEO_GRB +
NEO_KHZ800);
Adafruit_NeoPixel pixels3 = Adafruit_NeoPixel(NUMPIXELS, 6, NEO_GRB +
NEO_KHZ800);
Adafruit_NeoPixel pixels4 = Adafruit_NeoPixel(NUMPIXELS, 7, NEO_GRB +
NEO_KHZ800);
Adafruit_NeoPixel pixels[6]={pixels0, pixels1 ,pixels2 ,pixels3 ,pixels4};
int cardef[10][6][4]
{
{ // 0
{0,1,1,1},
{0,1,0,1},
{0,1,0,1},
{0,1,0,1},
{0,1,1,1}
},
{ // 1
{0,0,0,1},
{0,0,0,1},
{0,0,0,1},
{0,0,0,1},
{0,0,0,1},
},
{ // 2
{0,1,1,1},
{0,0,0,1},
```

```
{0,1,1,1},
{0,1,0,0},
{0,1,1,1}
},
{ // 3
{0,1,1,1},
{0,0,0,1},
{0,0,1,1},
{0,0,0,1},
{0,1,1,1}
},
{ // 4
{0,1,0,1},
{0,1,0,1},
{0,1,1,1},
{0,0,0,1},
{0,0,0,1}
},
{ // 5
{0,1,1,1},
{0,1,0,0},
{0,1,1,1},
{0,0,0,1},
{0,1,1,1}
},
{ // 6
{0,1,1,1},
{0,1,0,0},
{0,1,1,1},
{0,1,0,1},
{0,1,1,1}
},
{ // 7
{0,1,1,1},
{0,0,0,1},
{0,0,0,1},
{0,0,0,1},
{0,0,0,1}
},
{ // 8
{0,1,1,1},
{0,1,0,1},
{0,1,1,1},
{0,1,0,1},
{0,1,1,1}
},
```

```
{ // 9
{0,1,1,1},
{0,1,0,1},
{0,1,1,1},
{0,0,0,1},
{0,0,0,1}
}
};
bool a = true;
void setup() {
//module blutooth
Serial.begin(9600);
delay(500);
Serial.println("Bonjour -Pret pour les commandes AT");
BlueT.begin(9600);
delay(500);
for(int i=0;i<8;i++){
pixels[i].begin();
pixels[i].show();
}
}
void loop() {
//module blutooth
while (BlueT.available()) {
Serial.print(char(BlueT.read())); }
while (Serial.available()) {
BlueT.write(char(Serial.read())); }
//animation allumage
if (a){
colorWipe(20,0,0,intensite);}
a = false;
if (BlueT.available()){
Serial.print("ok");
Data=BlueT.read();
Serial.print(Data);
//mode veille
if (Data=='Y') {
rainbow(20);
rainbowCycle(20);
}
//mode fête
if (Data=='B') {
theaterChase(20,127,127,127);
theaterChaseRainbow(20);
}
if (Data=='G') {
```

```
nombre(12,intensite,0,intensite);
nombre(54,0,intensite,intensite);
}
}
}
void colorWipe(int wait,int r, int g, int b) {
for (int i=0; i<NUMPIXELS; i++) {
for(int k=0;k<5;k++){
pixels[k].setPixelColor(i,pixels[k].Color(r,g,b));
pixels[k].show();
delay(wait);
}
}
}
void rainbow(int wait) {
for(int j=0; j<256; j++) {
for(int k=0;k<5;k++){
for(int i=0; i<NUMPIXELS; i++){
pixels[k].setPixelColor(i,Wheel((i+j)&255,k));
}
pixels[k].show();
}
//delay(wait);
}
}
void rainbowCycle(int wait) {
for(int j=0; j<256*5; j++) { // 5 cycles of all colors on wheel
for(int k=0;k<5;k++){
for(int i=0; i<NUMPIXELS; i++) {
pixels[k].setPixelColor(i,Wheel(((i*256/NUMPIXELS)+j)&255,k));
}
pixels[k].show();
}
//delay(wait);
}
}
void theaterChase(int wait,int r,int g,int b) {
for (int j=0; j<10; j++) { //do 10 cycles of chasing
for (int q=0; q<3; q++) {
for(int k=0;k<5;k++){
for (int i=0; i<NUMPIXELS; i=i+3) {
pixels[k].setPixelColor(i+q,pixels[k].Color(r,g,b)); //turn every third pixel on
}
pixels[k].show();
delay(wait);
for (int i=0; i<NUMPIXELS; i=i+3) {
```

```
pixels[k].setPixelColor(i+q,0); //turn every third pixel off
}
delay(wait);
}
}
}
}
void theaterChaseRainbow(int wait){
for(int j=0; j<256; j++){ // cycle all 256 colors in the wheel
for(int q=0; q<3; q++){
for(int k=0;k<5;k++){
for(int i=0; i < pixels[k].numPixels(); i=i+3){
pixels[k].setPixelColor(i+q, Wheel((i+j)%255,k)); //turn every third pixel on
}
pixels[k].show();
delay(wait);
for (int i=0; i<pixels[k].numPixels(); i=i+3){
pixels[k].setPixelColor(i+q,0); //turn every third pixel off
}
}
}
}
}
uint32_t Wheel(byte WheelPos, int face) {
WheelPos=255-WheelPos;
if(WheelPos<85) {
return pixels[face].Color(255-WheelPos*3,0,WheelPos*3);
}
if(WheelPos<170) {
WheelPos-=85;
return pixels[face].Color(0,WheelPos*3,255-WheelPos*3);
}
WheelPos-=170;
return pixels[face].Color(WheelPos*3,255-WheelPos*3,0);
}
// affiche dans l'axe de visualisation, 0 en bas à droite, x horizontal, y vertical, z
profondeur
void afficheVisu(int z, int x,int y,int v, int r, int b){
setpixel(y, x, z, v, r, b); // converti coordonnées visu en coordonnées plaque, avec
y no de plaque
}
void nombre(int a, int v, int r, int b){
int chiffre2=a%10;
int chiffre1=a/10;
for (int z=7; z>=0; z--) // affiche le nombre du fond vers l'avant
{
```

```
//clearZ(0); // efface le nombre precedent au depart. moins joli
chiffre(chiffre1,true, z, v, r, b);
chiffre(chiffre2,false, z, v, r, b);
if (z < 7) clearZ(z+1); // efface plan z précédent
}
}
// efface le plan de profondeur z
void clearZ(int z)
{
for (int x=0; x<8; x++)
{
for (int y=0; y<5; y++)
{
afficheVisu(z, x, y, 0,0,0);
}
}
}
void chiffre(int a, bool zone1, int z, int v, int r, int b){
int incrementX;
if (zone1) incrementX=4;
else incrementX=0;
for (int y=4;y>=0;y--){
for (int x=3;x>=0;x--){
if (cardef[a][y][x]==1){
//setpixel(3, 3-x+incrementX, 4-y, v,r,b);
//else setpixel(3, 3-x+incrementX, 4-y, 0,0,0);
afficheVisu(z,3-x+incrementX,4-y,v,r,b);
}
else afficheVisu(z,3-x+incrementX,4-y,0,0,0);
}
}
}
void setpixel(int face, int x, int y, int v, int r, int b)
{
//Adafruit_NeoPixel p = pixels[face];
bool pair = y%2==0;
int num;
if (pair) num = 8*y+x;
else num = 8*y+7-x;
pixels[face].setPixelColor(num,pixels[face].Color(v,r,b));
pixels[face].show();
return;
}
```


Université
Nice
Sophia Antipolis


POLYTECH
NICE-SOPHIA