TP Bandeau de LED

Bruno Caillier - Nicolas Garric

10 mars 2017

1 Compétences visées

Allumer éteindre des LED WS2812 Neopixel.
Faire varier les couleurs.
Gérer les interruptions.
Réaliser des boucles et des fonctions.

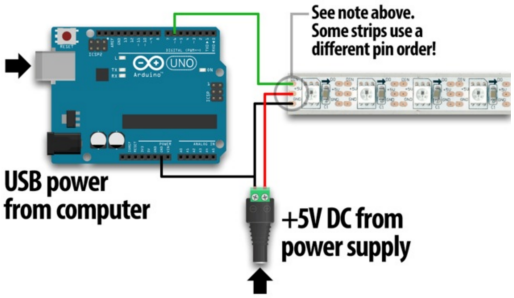
2 Prérequis

Avoir vu l'environnent arduino et sa syntaxe : boucle, fonction, définition de variables, etc...
Savoir uploader son code dans le microcontroleur.
Lecture de l'état d'une broche on/off.
Lecture de l'état analogique d'une entrée.

3 Montage - Matériel

Un support avec :

- Bandeau de LED NeoPixels WS2812 1m RGB adressable - 36 à 144 led/m `https://hackspark.fr/fr/1m-30leds-ws2812-led-strip-neopixel-compatible.htmlhttps://www.adafruit.com/products/1138`
- Alimentation pour le bandeau leds (36 à 120 * 60mA) 5V, à allumer AVANT le microcontrôleur de type au moins `https://www.adafruit.com/products/276` suivant le bandeau acheté.
- Résistance de 470Ω entre le premier pixel et la carte Arduino
- Capacité de lissage 1000μF
- 4 boutons poussoirs
- un potentiomètre
- Carte Arduino
- Connectique



4 Expérimentation

La programmation depuis « zéro » de bandeau de LED est un challenge si l'on part de rien. Nous utiliserons la bibliothèque développée par Adafruit pour nous consacrer à plus de « fun ». Chaque NeoPixel demande environ 3 bytes de RAM.

Il faut dans un premier temps importer la librairie (à récupérer sur *moodle*) puis croquis –> importer une librairie.

Il faut ensuite dans le sketch définir la borne de connexion et le nombre de pixel par mètre.

```
#include <Adafruit_NeoPixel.h>
#define PIN 6 // Parameter 1 = number of pixels in strip
// Parameter 2 = pin number (most are valid)
// Parameter 3 = pixel type flags, add together as needed:
// NEO_KHZ800 800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
// NEO_KHZ400 400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers) // NEO_GRB Pixels are wired for GRB bitstream (most NeoPixel products)
// NEO_RGB Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)
Adafruit_NeoPixel strip = Adafruit_NeoPixel(60, PIN, NEO_GRB + NEO_KHZ800);
```

Le codage peut démarrer. On va initialiser le bandeau et mettre tous les pixels off :

```
void setup() {
strip.begin();
strip.show(); // Initialize all pixels to 'off'
}
```

Pour configurer l'état d'un pixel, on peut utiliser ces deux syntaxes :

```
strip.setPixelColor(n, red, green, blue);
```

ou

```
strip.setPixelColor(n, color);
```

avec :

- n le numéro du pixel (le premier est le zéro)
- red, green, blue un nombre entre 0 et 255 qui donne la luminosité dans la couleur
- color est une variable de type 32-bit regroupant les informations de couleurs.

```
uint32_t magenta = strip.Color(255, 0, 255);
```

Il faut ensuite appliquer cette couleur, ici à tout le bandeau :

```
strip.show();
```

5 Méthodes classiques

1. Coder l'allumage et l'extinction de la 11éme LED en Magenta.
2. Réaliser une boucle permettant de faire un chenillard monochrome aller et retour avec delay fixe.
3. Réaliser une méthode prenant en paramètre d'entrée (début,fin,) et faisant le chenillard entre début et fin.
4. Le chenillard doit être maintenant de longueur paramétrable l=5.

6 Lecture Potentiomètre

1. Réaliser la lecture du potentiomètre en utilisant une entrée Analog, (ne pas oublier les déclarations) :

```
poten = analogRead(PinNb);
```

poten prendra une valeur entre 0 et 1023.

2. À partir de la lecture de poten, réaliser un curseur de niveau : pour poten= lu alors 0 pixel affiché pour poten=1023 alors tout le bandeau affiché.

7 Jeu de couleurs

Nous allons maintenant mettre en œuvre différents effets de couleurs :

1. Réaliser un dégradé linéaire pour une même couleur sur TOUT le bandeau, 0 sur le pixel 0 et le maximum sur le dernier pixel
2. Le maximum va maintenant pouvoir se déplacer à l'aide du potentiomètre, il faut conserver le dégradé.
3. Le pixel de référence (le max) va maintenant passer par toutes les couleurs possibles (), le dégradé doit suivre.

8 Gestion des interruptions

Nous allons maintenant rajouter des interruptions avec des boutons branchés. Ne pas oublier d'activer le button :

```
pinMode(BPlus,INPUT_PULLUP);
```

Il faut aussi déclarer la vitesse en volatile :

```
volatile int vitesse = 1000;
```

1. Un appui sur le bouton 1 permet l'arrêt du chenillard,
2. Un nouvel appui sur le bouton 1 permet le redémarrage.
3. Un appui sur le bouton 3 augmente la vitesse du chenillard.
4. Un appui sur le bouton 4 diminue la vitesse du chenillard.
5. Lecture d'entrée analogique : le potentiomètre doit permettre le réglage de la luminosité du chenillard.

9 Aller plus loin : réaliser un jeu type ping-pong

1. Mettre le pixel 8 et 42 en blanc, tout en conservant le chenillard actif.
2. L'appui sur le bouton 1 et 2 permet maintenant de changer la direction du chenillard, uniquement si le chenillard se trouve entre la bordure et le pixel blanc
3. Finaliser un jeu sur cette base (conditions de victoire, affichage des points, coups spéciaux, combo, etc...).