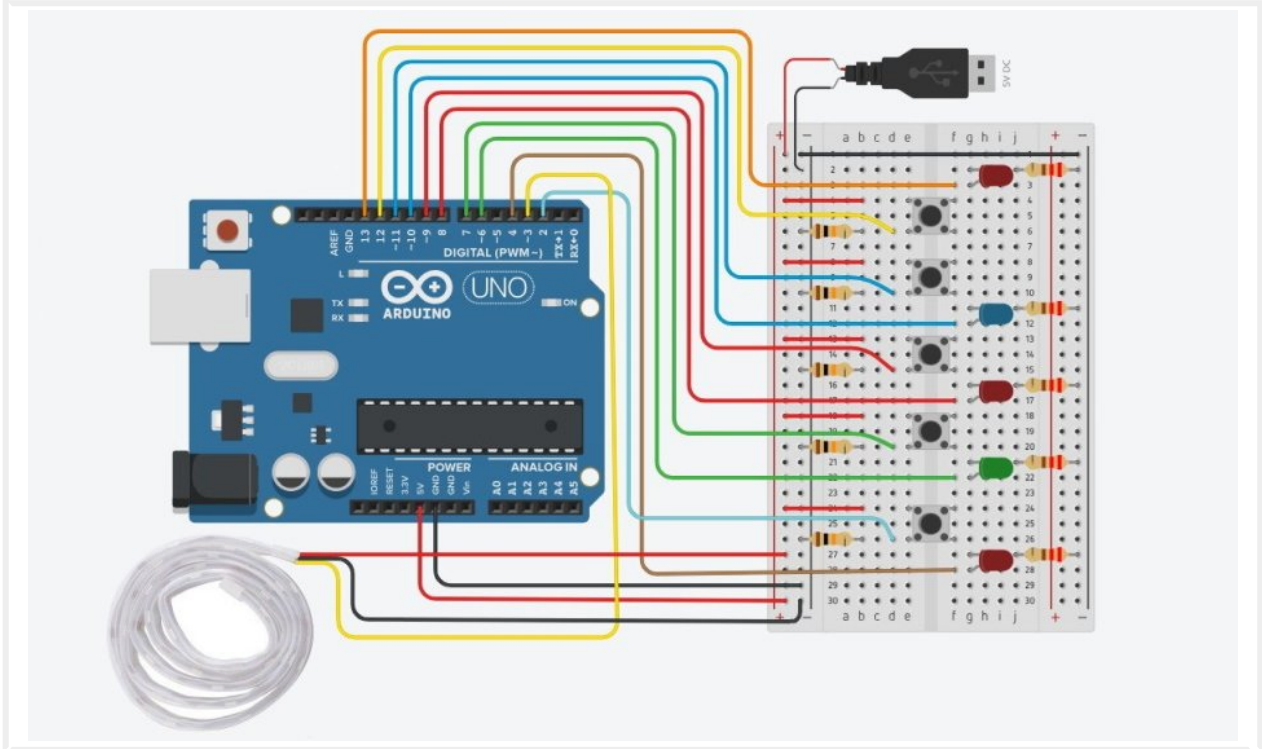


# *Cordon DEL RVB*

*(Gestion de l'éclairage d'un cordon de 30 DELs RVB)*



## Liste des composants

- . 1 Cordons 30 DELs RVB Grove
- . 3 DELs rouges
- . 1 DEL verte
- . 1 DEL bleue
- . 5 Résistances de 220  $\Omega$
- . 5 Boutons poussoir
- . 5 Résistances de 10 k $\Omega$
- . 1 plaque d'essais
- . Fils de connexion

## Objectif

L'objectif du montage est de contrôler un cordon lumineux de 30 DELs RVB Grove pour l'éclairage d'un aquarium :

- Par défaut, après un appui sur le premier bouton poussoir, les DELs RVB éclairent en blanc ou en rouge ou en vert ou en bleu (la première DEL rouge du montage est alors allumée),

- un maintien du premier bouton poussoir permet de régler l'intensité lumineuse des DELs (par défaut, celle-ci est au maximum),
- le deuxième bouton poussoir permet d'éclairer certaines DELs en bleu (la DEL bleue du montage est alors allumée),
- le troisième bouton poussoir permet d'éclairer certaines DELs en rouge (la deuxième DEL rouge du montage est alors allumée),
- le quatrième bouton poussoir permet d'éclairer certaines DELs en vert (la DEL verte du montage est alors allumée),
- le cinquième bouton poussoir lance un minuteur. Les DELs sont alors éteintes à la fin d'une durée défini par le programme, puis rallumées au bout d'une autre durée également définie,
- un second appui sur le premier bouton poussoir éteint le cordon lumineux.

## [. Le programme](#)

Voici le code de l'activité :

Cordon\_DEL\_RVB

```
#include <FastLED.h>
#define NUM_LEDS 30
#define LED_TYPE WS2812B
#define COLOR_ORDER BRG//RGB

CRGB leds[NUM_LEDS];

const int stripPin = 3;
unsigned long previousMillis = 0;
const long interval = 28800000;
const long interval2 = 57600000;
int ledState = LOW;
int BRIGHTNESS = 250;
unsigned long startTime = 0;

const int OnOffSwitchPin = 12;
const int OnOffSwitchTimePin = 2;
const int PinLEDOnOff = 13;
const int PinLEDTimeOnOff = 4;

const int PinLEDBlueOnOff = 10;
const int OnOffSwitchPinBlue = 11;

const int PinLEDRedOnOff = 8;
const int OnOffSwitchPinRed = 9;

const int PinLEDGreenOnOff = 6;
const int OnOffSwitchPinGreen = 7;
```

```

int OnOffSwitchState = 0;
int previousOnOffSwitchState = 0;
int OnOffSwitchTimeState = 0;
int previousOnOffSwitchTimeState = 0;
int state=0;
int previousstate=0;
int stateTime=0;
int previousstateTime=0;

int OnOffBlueState = 0;
int previousOnOffBlueState = 0;
int Bluestate=1;
int previousBluestate=1;

int OnOffRedState = 0;
int previousOnOffRedState = 0;
int Redstate=1;
int previousRedstate=1;

int OnOffGreenState = 0;
int previousOnOffGreenState = 0;
int Greenstate=1;
int previousGreenstate=1;

void AllumLeds() {
    int LedBlue = 3;
    int LedGreen = 5;
    int LedRed = 1;
    for (int i = 0; i < NUM_LEDS; i++) {
        if (i!=LedBlue and i!=LedGreen and i!=LedRed){
            leds[i].setRGB(255, 255, 255);
        }
        else{
            if (i==LedBlue){
                if (Bluestate==1){
                    leds[i].setRGB(0, 255, 0);
                }
                else{
                    leds[i].setRGB(255, 255, 255);
                }
            }
            LedBlue = LedBlue + 4;
        }
        else {
            if (i==LedGreen){
                if (Greenstate==1){
                    leds[i].setRGB(0, 0, 255);
                }
                else {
                    leds[i].setRGB(255, 255, 255);
                }
            }
            LedGreen = LedGreen + 8;
        }
        else {

```

```

        if (Redstate==1){
            leds[i].setRGB(255, 0, 0);
        }
        else {
            leds[i].setRGB(255, 255, 255);
        }
        LedRed = LedRed + 8;
    }
}
}
FastLED.show();
ledState = HIGH;
}

void EteintLeds() {
    ledState = LOW;
    for (int j = 0; j < NUM_LEDS; j++) {
        leds[j].setRGB(0,0,0);
    }
    FastLED.show();
}

void LedScenario() {
    unsigned long currentMillis = millis();
    if (ledState == LOW) {
        if (currentMillis - previousMillis >= interval2) {
            previousMillis = currentMillis;
            AllumLeds();
        }
    }
    else {
        if (currentMillis - previousMillis >= interval) {
            previousMillis = currentMillis;
            EteintLeds();
        }
    }
}

void setup() {
    FastLED.addLeds<LED_TYPE, stripPin, COLOR_ORDER>(leds, NUM_LEDS);
    FastLED.setBrightness(BRIGHTNESS);
    pinMode (PinLEDOnOff, OUTPUT);
    pinMode (PinLEDTimeOnOff, OUTPUT);
    pinMode (PinLEDBlueOnOff, OUTPUT);
    pinMode (PinLEDRedOnOff, OUTPUT);
    pinMode (PinLEDGreenOnOff, OUTPUT);
    pinMode (OnOffSwitchPin, INPUT);
    pinMode (OnOffSwitchTimePin, INPUT);
    pinMode (OnOffSwitchPinBlue, INPUT);
    pinMode (OnOffSwitchPinRed, INPUT);
    pinMode (OnOffSwitchPinGreen, INPUT);
}

```

```

void loop() {
  OnOffSwitchState = digitalRead(OnOffSwitchPin);
  delay(10);
  if ((OnOffSwitchState == HIGH) && (previousOnOffSwitchState == LOW))
  {
    state=1-state;
    startTime= millis();
    delay(10);
  }
  else
  {
    if ((OnOffSwitchState == LOW) && (previousOnOffSwitchState == HIGH))
    {
      delay(10);
    }
    else {
      if ((OnOffSwitchState == HIGH) && (previousOnOffSwitchState == HIGH)) {
        if (state == 1 && (millis() - startTime) > 500) {
          BRIGHTNESS = BRIGHTNESS + 25;
          if (BRIGHTNESS > 250) {
            BRIGHTNESS = 25;
          }
          FastLED.setBrightness(BRIGHTNESS);
          FastLED.show();
          delay(500);
        }
      }
    }
  }
  previousOnOffSwitchState = OnOffSwitchState;

  OnOffSwitchTimeState = digitalRead(OnOffSwitchTimePin);
  delay(10);

  if ((OnOffSwitchTimeState == HIGH) && (previousOnOffSwitchTimeState == LOW))
  {
    stateTime=1-stateTime;
    delay(10);
  }
  else
  {
    if ((OnOffSwitchTimeState == LOW) && (previousOnOffSwitchTimeState == HIGH))
    {
      delay(10);
    }
  }

  previousOnOffSwitchTimeState = OnOffSwitchTimeState;

  OnOffBlueState = digitalRead(OnOffSwitchPinBlue);
  delay(10);
  if ((OnOffBlueState == HIGH) && (previousOnOffBlueState == LOW))
  {
    Bluestate=1-Bluestate;
    delay(10);
  }
  else

```

```

{
  if ((OnOffBlueState == LOW)&&(previousOnOffBlueState == HIGH))
  {
    delay(10);
  }
}
previousOnOffBlueState = OnOffBlueState;

OnOffRedState = digitalRead(OnOffSwitchPinRed);
delay(10);
if ((OnOffRedState == HIGH)&&(previousOnOffRedState == LOW))
{
  Redstate=1-Redstate;
  delay(10);
}
else
{
  if ((OnOffRedState == LOW)&&(previousOnOffRedState == HIGH))
  {
    delay(10);
  }
}
previousOnOffRedState = OnOffRedState;

OnOffGreenState = digitalRead(OnOffSwitchPinGreen);
delay(10);
if ((OnOffGreenState == HIGH)&&(previousOnOffGreenState == LOW))
{
  Greenstate=1-Greenstate;
  delay(10);
}
else
{
  if ((OnOffGreenState == LOW)&&(previousOnOffGreenState == HIGH))
  {
    delay(10);
  }
}
previousOnOffGreenState = OnOffGreenState;

if (state==1) {
  if (previousstate==0) {
    AllumLeds();
    digitalWrite(PinLEDOff, HIGH);
    digitalWrite(PinLEDBlueOnOff, Bluestate);
    digitalWrite(PinLEDRedOnOff, Redstate);
    digitalWrite(PinLEDGreenOnOff, Greenstate);
    previousstate=1;
  }
  else{
    if (stateTime==1) {
      if (previousstateTime==0) {
        previousMillis = millis();
        previousstateTime=1;
        digitalWrite(PinLEDTimeOnOff, HIGH);
      }
      else {
        LedScenario();
      }
    }
  }
}

```

```

    }
    else {
        if (previousstateTime==1) {
            AllumLeds();
            previousstateTime=0;
            digitalWrite(PinLEDTurnOnOff, LOW);
        }
    }
}

if ((Bluestate==1 and previousBluestate==0) or (Bluestate==0 and previousBluestate==1)) {
    digitalWrite(PinLEDBlueOnOff, Bluestate);
    previousBluestate = Bluestate;
    AllumLeds();
}

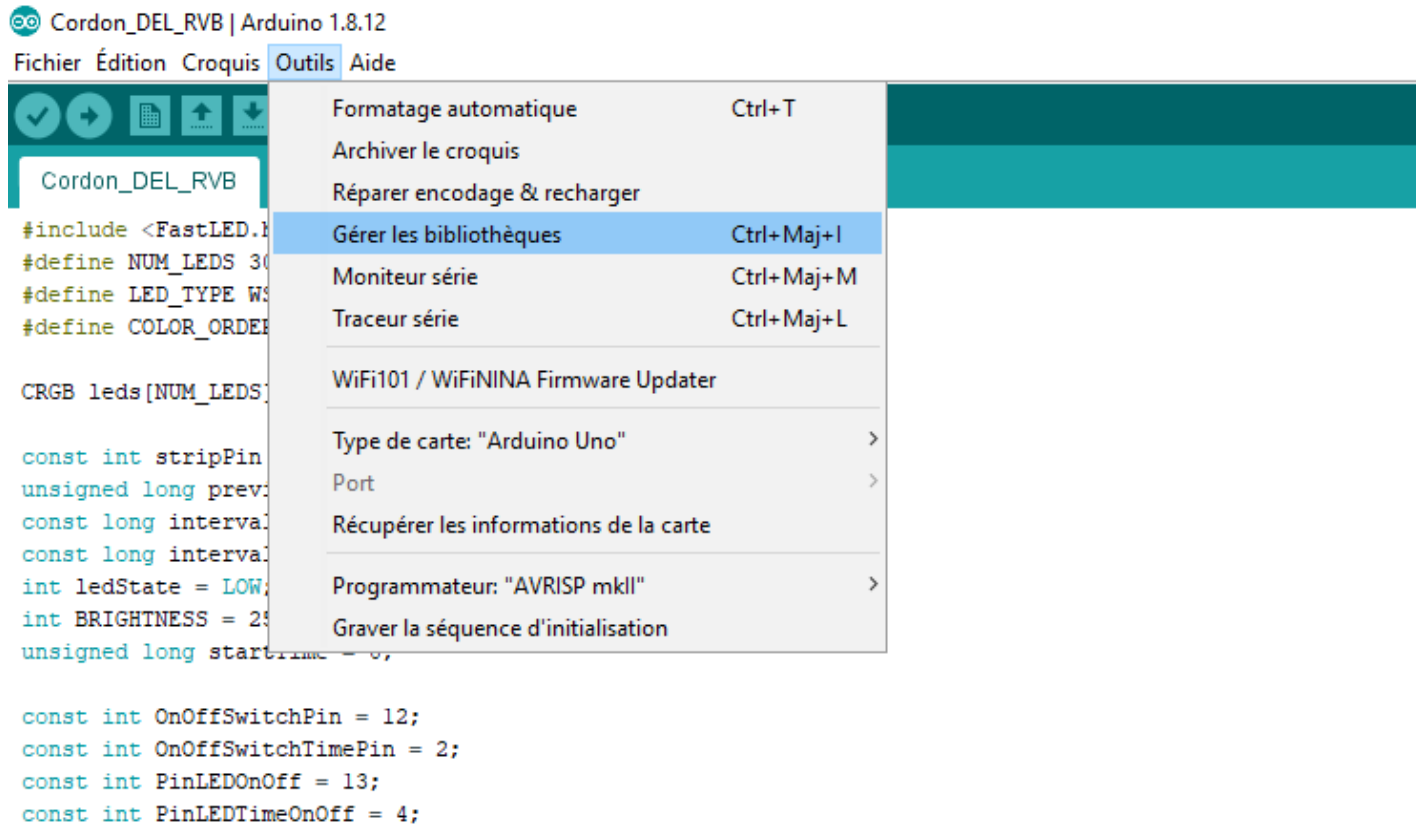
if ((Redstate==1 and previousRedstate==0) or (Redstate==0 and previousRedstate==1)) {
    digitalWrite(PinLEDRedOnOff, Redstate);
    previousRedstate = Redstate;
    AllumLeds();
}

if ((Greenstate==1 and previousGreenstate==0) or (Greenstate==0 and previousGreenstate==1)) {
    digitalWrite(PinLEDGreenOnOff, Greenstate);
    previousGreenstate = Greenstate;
    AllumLeds();
}
}
else
{
    EteintLeds();
    digitalWrite(PinLEDTurnOnOff, LOW);
    digitalWrite(PinLEDTurnOnOff, LOW);
    digitalWrite(PinLEDBlueOnOff, LOW);
    digitalWrite(PinLEDRedOnOff, LOW);
    digitalWrite(PinLEDGreenOnOff, LOW);
    state=0;
    previousstate=0;
    stateTime=0;
    previousstateTime=0;
    Redstate=1;
    previousRedstate=1;
    Bluestate=1;
    previousBluestate=1;
    Greenstate=1;
    previousGreenstate=1;
}
}

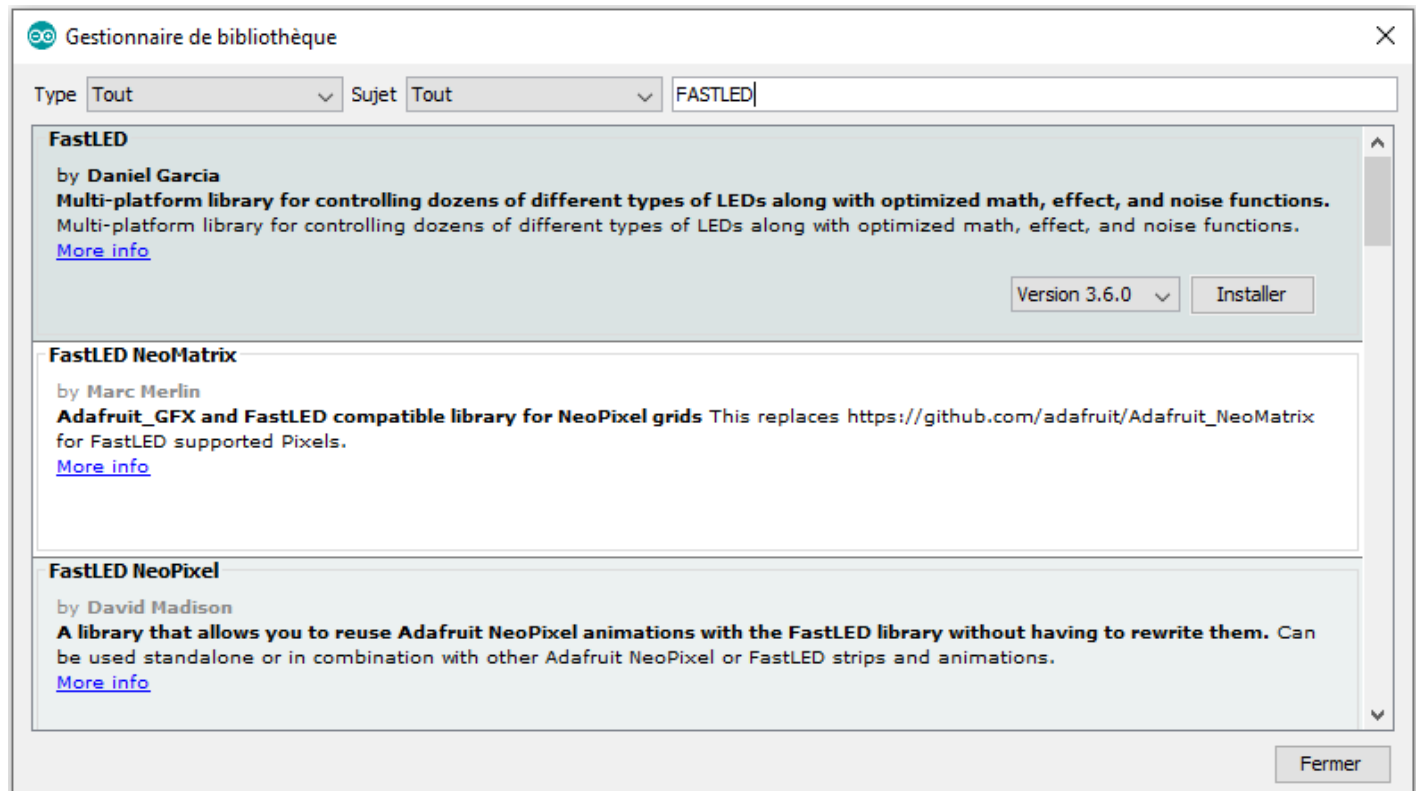
```

Le code de l'activité nécessite l'installation au préalable de la librairie " **FastLED** ".

Afin d'ajouter une librairie à l'IDE Arduino, il faut aller dans le menu « **Outils ->Gérer les bibliothèques** » :



Il suffit ensuite de rechercher et d'ajouter la librairie « **FastLED**» :



## Déroulement du programme :

### – 1. Insertion des bibliothèques :

```
#include <FastLED.h> (insertion de la bibliothèque FastLED)

#define NUM_LEDS 30 (initialisation du nombre de DEL)

#define LED_TYPE WS2812B (initialisation du type de DEL)

#define COLOR_ORDER BRG//RGB (initialisation de l'ordre des couleurs)

CRGB leds[NUM_LEDS] (Initialisation du nombre de DELs du cordon lumineux)
```

### – 2. Déclaration des constantes et variables :

```
. const int stripPin = 3 (constante nombre entier correspondant à la broche de contrôle du cordon lumineux)

. unsigned long previousMillis = 0 (variable nombre entier long pour le calcul de la durée d'appui sur le premier bouton poussoir)

. const long interval = 28800000 (constante nombre entier long correspondant à la durée d'éclairement en ms du cordon lumineux)

. const long interval2 = 57600000 (constante nombre entier long correspondant à la durée d'extinction en ms du cordon lumineux)

. int ledState = LOW (constante nombre entier correspondant à l'état d'éclairement des DELs)

. int BRIGHTNESS = 250 (constante nombre entier correspondant à l'intensité lumineuse des DELs, max = 250, min = 25)

. unsigned long startTime = 0 (variable nombre entier long pour stocker l'heure à laquelle le premier bouton poussoir a été appuyé).

. const int OnOffSwitchPin = 12 (constante nombre entier correspondant à la broche du premier bouton poussoir permettant d'allumer les DELs)

. const int OnOffSwitchTimePin = 2 (constante nombre entier correspondant à la broche du dernier bouton poussoir permettant de lancer la minuterie)

. const int PinLEDOnOff = 13 (constante nombre entier correspondant à la broche de la première DEL rouge indiquant que le cordon lumineux est allumé)

. const int PinLEDTimeOnOff = 4 (constante nombre entier correspondant à la broche de la dernière DEL rouge indiquant que la minuterie est en fonction)

. const int PinLEDBlueOnOff = 10 (constante nombre entier correspondant à la broche de la DEL bleue indiquant si certaines DELs du cordon sont allumées en bleu)
```

. **const int OnOffSwitchPinBlue = 11** (constante nombre entier correspondant à la broche du bouton poussoir permettant d'allumer certaines DELs en bleu)

. **const int PinLEDRedOnOff = 8** (constante nombre entier correspondant à la broche de la DEL rouge indiquant si les certaines DELs du cordon sont allumées en rouge)

. **const int OnOffSwitchPinRed = 9** (constante nombre entier correspondant à la broche du bouton poussoir permettant d'allumer certaines DELs en rouge)

. **const int PinLEDGreenOnOff = 6** (constante nombre entier correspondant à la broche de la DEL verte indiquant si les certaines DELs du cordon sont allumées en vert)

. **const int OnOffSwitchPinGreen = 7** (constante nombre entier correspondant à la broche du bouton poussoir permettant d'allumer certaines DELs en vert)

. **int OnOffSwitchState = 0** (variable nombre entier pour stocker la valeur du potentiel de la broche du premier bouton poussoir permettant d'allumer les DELs)

. **int previousOnOffSwitchState = 0** (variable nombre entier pour stocker la précédente valeur du potentiel de la broche du premier bouton poussoir permettant d'allumer les DELs)

. **int OnOffSwitchTimeState = 0** (variable nombre entier pour stocker la valeur du potentiel de la broche du dernier bouton poussoir permettant de lancer la minuterie)

. **int previousOnOffSwitchTimeState = 0** (variable nombre entier pour stocker la précédente valeur du potentiel de la broche du dernier bouton poussoir permettant de lancer la minuterie)

. **int state=0** (variable nombre entier correspondant à l'action à effectuer en relation avec le premier bouton poussoir)

. **int previousstate=0** (variable nombre entier correspondant à l'action effectuée précédemment en relation avec le premier bouton poussoir)

. **int stateTime=0** (variable nombre entier correspondant à l'action à effectuer en relation avec le dernier bouton poussoir)

. **int previousstateTime=0** (variable nombre entier correspondant à l'action effectuée précédemment en relation avec le dernier bouton poussoir)

. **int OnOffBlueState = 0** (variable nombre entier pour stocker la valeur du potentiel de la broche du deuxième bouton poussoir permettant d'allumer certaines DELs en bleu)

. **int previousOnOffBlueState = 0** (variable nombre entier pour stocker la précédente valeur du potentiel de la broche du deuxième bouton poussoir permettant d'allumer certaines DELs en bleu)

. **int Bluestate=1** (variable nombre entier indiquant si certaines DELs sont allumées en bleu)

. **int previousBluestate=1** (variable nombre entier indiquant si précédemment certaines DELs étaient allumées en bleu)

. **int OnOffRedState = 0** (variable nombre entier pour stocker la valeur du potentiel de la broche du troisième bouton poussoir permettant d'allumer certaines DELs en rouge)

. **int previousOnOffRedState = 0** (variable nombre entier pour stocker la précédente valeur du potentiel de la broche du troisième bouton poussoir permettant d'allumer certaines DELs en rouge)

. **int Redstate=1** (variable nombre entier indiquant si certaines DELs sont allumées en rouge)

. **int previousRedstate=1** (variable nombre entier indiquant si précédemment certaines DELs étaient allumées en rouge)

. **int OnOffGreenState = 0** (variable nombre entier pour stocker la valeur du potentiel de la broche du quatrième bouton poussoir permettant d'allumer certaines DELs en vert)

. **int previousOnOffGreenState = 0** (variable nombre entier pour stocker la précédente valeur du potentiel de la broche du quatrième bouton poussoir permettant d'allumer certaines DELs en vert)

. **int Greenstate=1** (variable nombre entier indiquant si certaines DELs sont allumées en vert)

. **int previousGreenstate=1** (variable nombre entier indiquant si précédemment certaines DELs étaient allumées en vert)

– 3. Déclaration des fonctions :

. **AllumLeds()** (fonction pour allumer une par une les DELS en blanc ou en rouge ou en bleu ou en vert en fonction de la valeur des variables d'éclairage)

. **EteintLeds()** (fonction pour éteindre une par une toutes les DELS)

. **LedScenario()** (fonction gérant l'éclairage ou l'extinction des DELs en fonction de la minuterie)

– 4. Initialisation des entrées et sorties :

. **Initialisation du cordon lumineux,**

. **Initialisation de l'intensité lumineuse des DELs,**

. **Initialisation des entrées et des sorties (DELs, bouton poussoirs).**

– 5. Fonction principale en boucle :

–> **Lecture de la valeur de la broche du premier bouton poussoir,**

–> **Mise à jour de la variable « state » si changement de la valeur du bouton poussoir,**

–> **Eclairage ou extinction du cordon en fonction de la variable « state »,**

--> **Réglage de la luminosité si le premier bouton poussoir est maintenu appuyé,**

- Lecture de la valeur de la broche du dernier bouton poussoir,
- Mise à jour de la variable « stateTime » si changement de la valeur du bouton poussoir,
- Mise en route ou arrêt de la minuterie en fonction de la variable « stateTime »,
- Lecture de la valeur de la broche du deuxième bouton poussoir,
- Mise à jour de la variable « Bluestate » si changement de la valeur du bouton poussoir,
- Eclairer en bleu de certaines DELs en fonction de la variable « Bluestate »,
- Lecture de la valeur de la broche du troisième bouton poussoir,
- Mise à jour de la variable « Redstate » si changement de la valeur du bouton poussoir,
- Eclairer en rouge de certaines DELs en fonction de la variable « Redstate »,
- Lecture de la valeur de la broche du quatrième bouton poussoir,
- Mise à jour de la variable « Greenstate » si changement de la valeur du bouton poussoir,
- Eclairer en vert de certaines DELs en fonction de la variable « Greenstate »,

#### **Caractéristiques du cordon 30 DELs RVB Grove :**

- Alimentation: 5 Vcc via carte Arduino + alimentation externe 5 Vcc recommandée
- Consommation: 1,8 A maxi en fonction des couleurs (0,06 A par led)
- Interface: digitale compatible Grove
- Type de Leds: WS2813
- Indice de protection: IP65