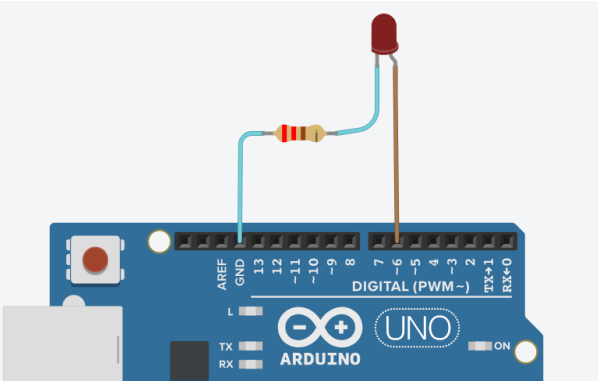
Arduino - Les composants

à venir une liste d'exemple d'utilisation des compo de base.

DEL

Une Diode Electro-Luminescente (ou LED an anglais) est un composant passif qui émet de la lumière quand il est alimenté.



exemple de clignotement d'une DEL:

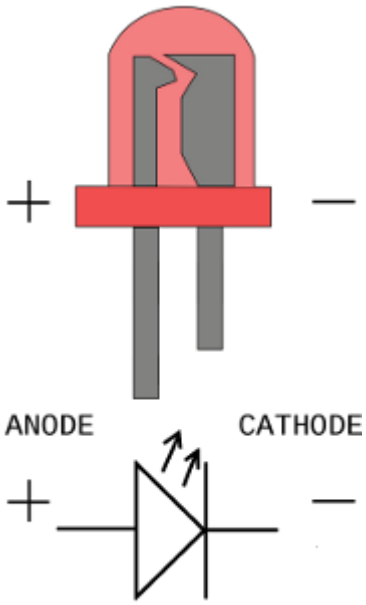
```c
#define pin_led 6
void setup()
{
  pinMode(led, OUTPUT);
}

void loop()
{
  digitalWrite(pin_led, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(pin_led, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

Polarité

Étant un dipôle polarisé la DEL ne fonctionne que si elle est alimenté dans le bon sens. Sont **Anode** doit être reliée au **+** et sa **Cathode** au **GND**.

Pour reconnaître le sens d'une DEL:

- le plus est la patte la plus longue
- si on regarde le dessus de la DEL , elle a une forme de "D", le coté plat est le moins.
- sur les schéma, la trait représente le moins



source: https://www.deco.fr/bricolage-travaux/electricite/qr/598408-a-quoi-correspondent-les-couleurs-des-fils-electriques.html

Je me sens dans l'obligation de vous partager un moyen mémo-technique trouvé sur ce site : https://jeretiens.net/polarite-de-lanode-et-de-la-cathode/

```
Anode => Anus => Plus
Cathode => Catin => Moins

Ou… MOKA => Moins / Cathode.
```

Résistance

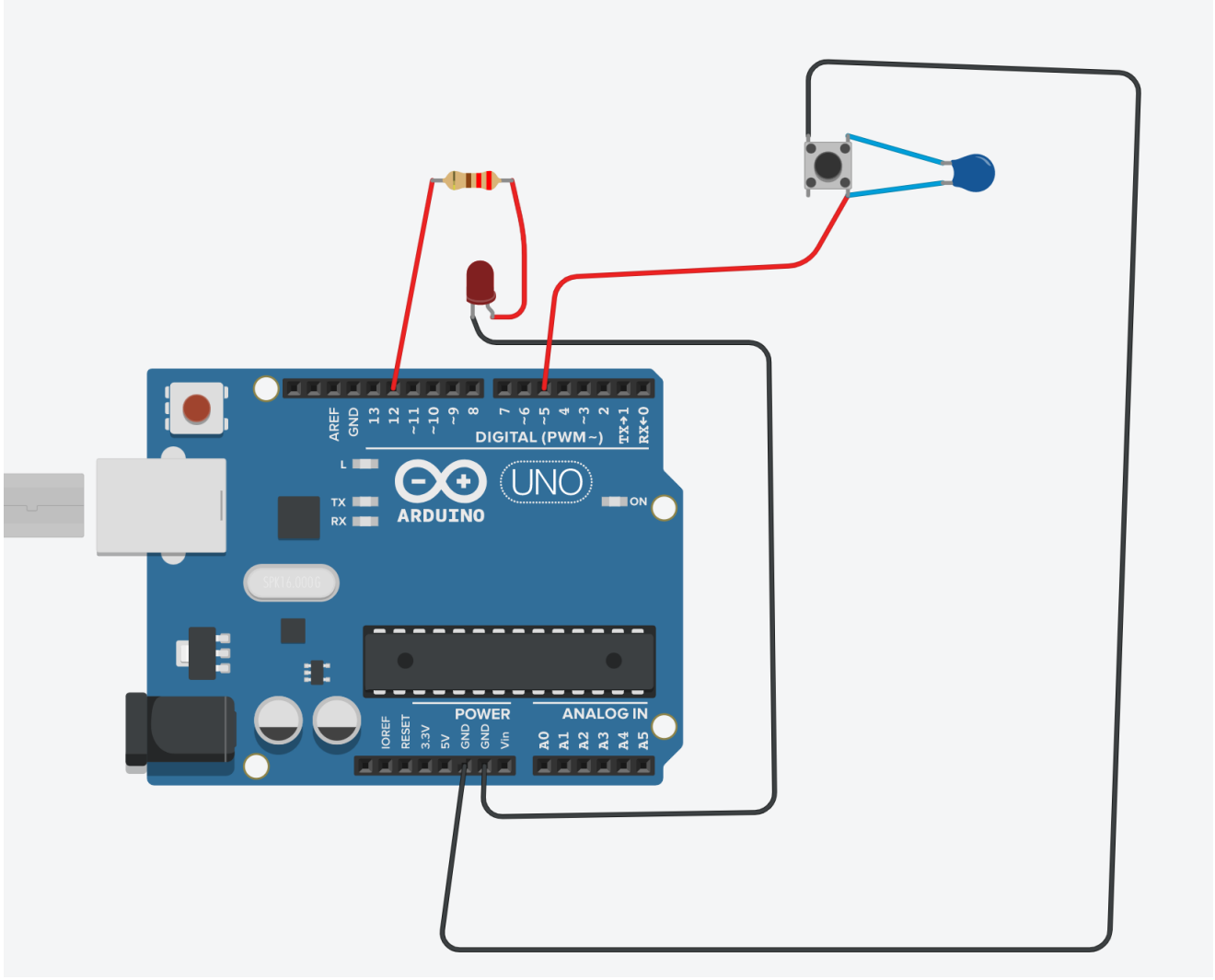
La DEL, si'il n'y a pas de résistance en série avec elle, va cramé.
Il faudra donc lui en mettre une, en calculant la valeur par rapport avec la fiche technique de celle-ci.
En utilisant la loi d'Ohm: **U = R x I**

Par exemple, nous aurons pour une DEL qui supporte 20mA:
R = 5 / 0,02 = 250 ohms

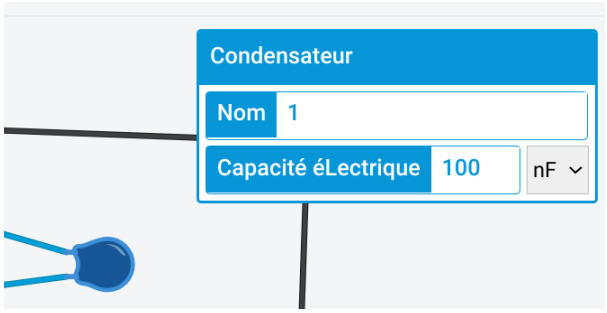
250 n'existe pas comme valeurs dans la série E12 donc nous prendrons 270 ou 220.

Boutons

En raison de contraintes mécanique lié à la conception même des boutons il peut être nécessaire d'effectuer quelque ajustement si vous utilisez un bouton tout simple.



Un condensateur permet de limité l'effet rebond du signal éléctrique.



```c
#define pin_led 12
#define pin_bt 5


void setup()
{
  pinMode(pin_led, OUTPUT);
  pinMode(pin_bt,INPUT_PULLUP); //résistance de pull-up
interne
  Serial.begin(9600);
}

void loop()
{
        bool etat_bt = digitalRead(pin_inter);
        Serial.println(etat_bt);
        if (etat_bt == 1){
                digitalWrite(pin_led, HIGH);
        }
        else{
                digitalWrite(pin_led, LOW);
        }
}
```

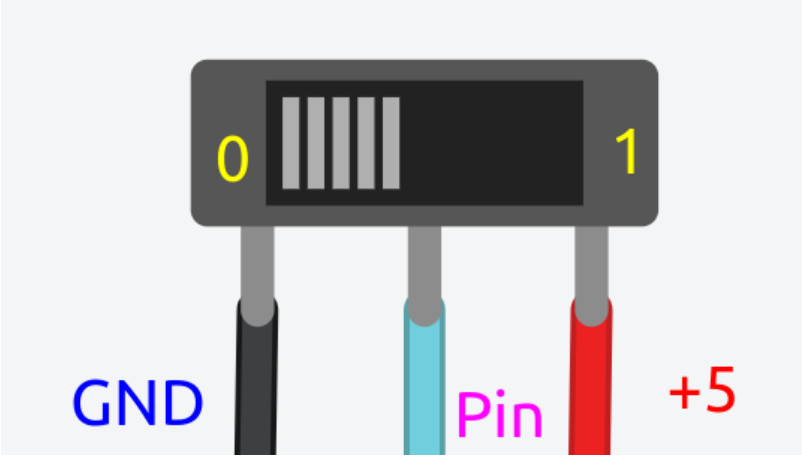
lien tinkercad: https://www.tinkercad.com/things/880U1MaYGWw?sharecode=SX7362kxqkKA3PTwOZM1hilpSDfv_Jsx2e4XWeeSC8Q

Une façon encore plus efficace de les gérer serai avec des *interruptions* : Arduino - Les interruptions

Interrupteurs

Interrupteurs 3 PINs

Composants bistable permettant de récupérer un booléen en fonction de la position de l'interrupteur.



exemple: voir Arduino - Les composants > Afficheur 7 segments

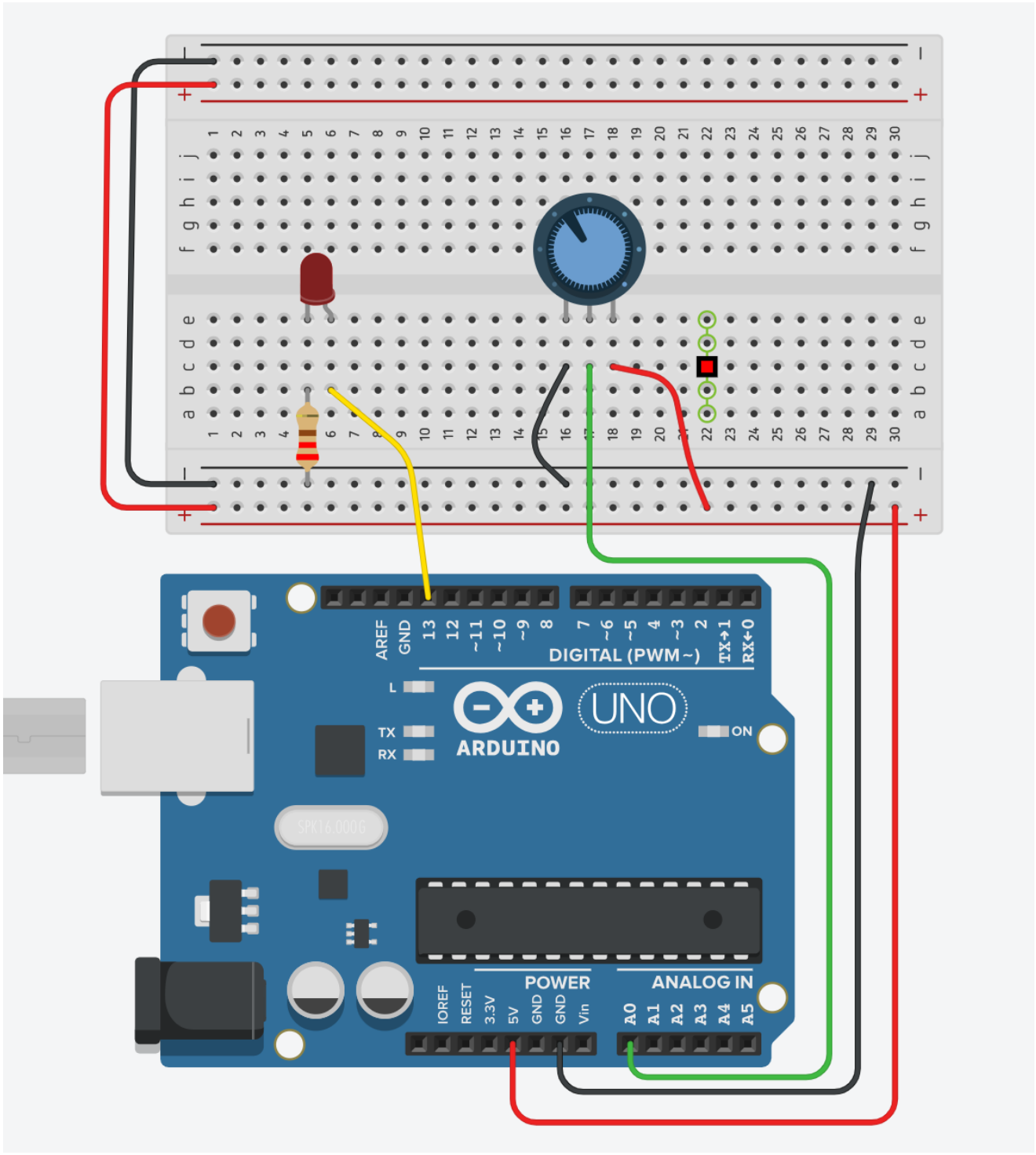
Interrupteur avec un bouton

Nous pouvons utiliser un bouton comme un interrupteur, dans ce cas il faut utilisé une mémoire pour permettre de sauvegarder son état précédent.
Une interruption serait plus adaptée pour gérer cela, car elle permet la gestion des fronts montant et descendant: Arduino - Les interruptions.

```c
const int buttonPin = 2;     // Broche du bouton
const int ledPin = 13;       // Broche de la LED

bool buttonState = false;    // État actuel du bouton
bool previousState = false; // État précédent du bouton

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);  // Configuration de la broche du bouton en entrée avec résistance de pull-up interne
  pinMode(ledPin, OUTPUT);             // Configuration de la broche de la LED en sortie
}

void loop() {
  buttonState = digitalRead(buttonPin);  // Lecture de l'état actuel du bouton

  if (buttonState != previousState) {
    if (buttonState == LOW) {
      // Si le bouton est pressé (état bas), inverse l'état de la LED
      //La valeur du bouton est inversé à cause de la pullup
      digitalWrite(ledPin, !digitalRead(ledPin));
    }
    delay(50);  // Délai pour éviter les rebonds du bouton
  }

  previousState = buttonState;  // Mise à jour de l'état précédent du bouton
}
```

Potentiomètre

Un potentiomètre est une résistance qui varie en fonction de la position de la partie rotative, ils sont souvent utilisés pour permettre de régler des valeurs rapidement. Pour lire cette valeur il faut utiliser une **entrée analogique**.



Exemple de réglage de la luminosité d'une DEl en fonction de la valeur:

C

```c
const int potPin = A0;    // Broche analogique utilisée pour lire la valeur du potentiomètre
const int ledPin = 9;     // Broche de la LED

void setup() {
  pinMode(ledPin, OUTPUT);   // Configuration de la broche de la LED en sortie
  Serial.begin(9600);        // Initialisation de la communication série
}

void loop() {
  int potValue = analogRead(potPin);   // Lecture de la valeur analogique du potentiomètre
  int ledBrightness = map(potValue, 0, 1023, 0, 255);   // Mappage de la valeur lue sur une plage de luminosité de la LED

  analogWrite(ledPin, ledBrightness);   // Réglage de la luminosité de la LED en utilisant la valeur mappée

  Serial.print("Potentiomètre : ");
  Serial.print(potValue);
  Serial.print(" | Luminosité LED : ");
  Serial.println(ledBrightness);

  delay(100);   // Délai pour une lecture plus lente
}```

Exemple de réglage de la fréquence de clignotement d'une DEL en fonction du potentiomètre:
```C
// C++ code
//
int sensorValue = 0;

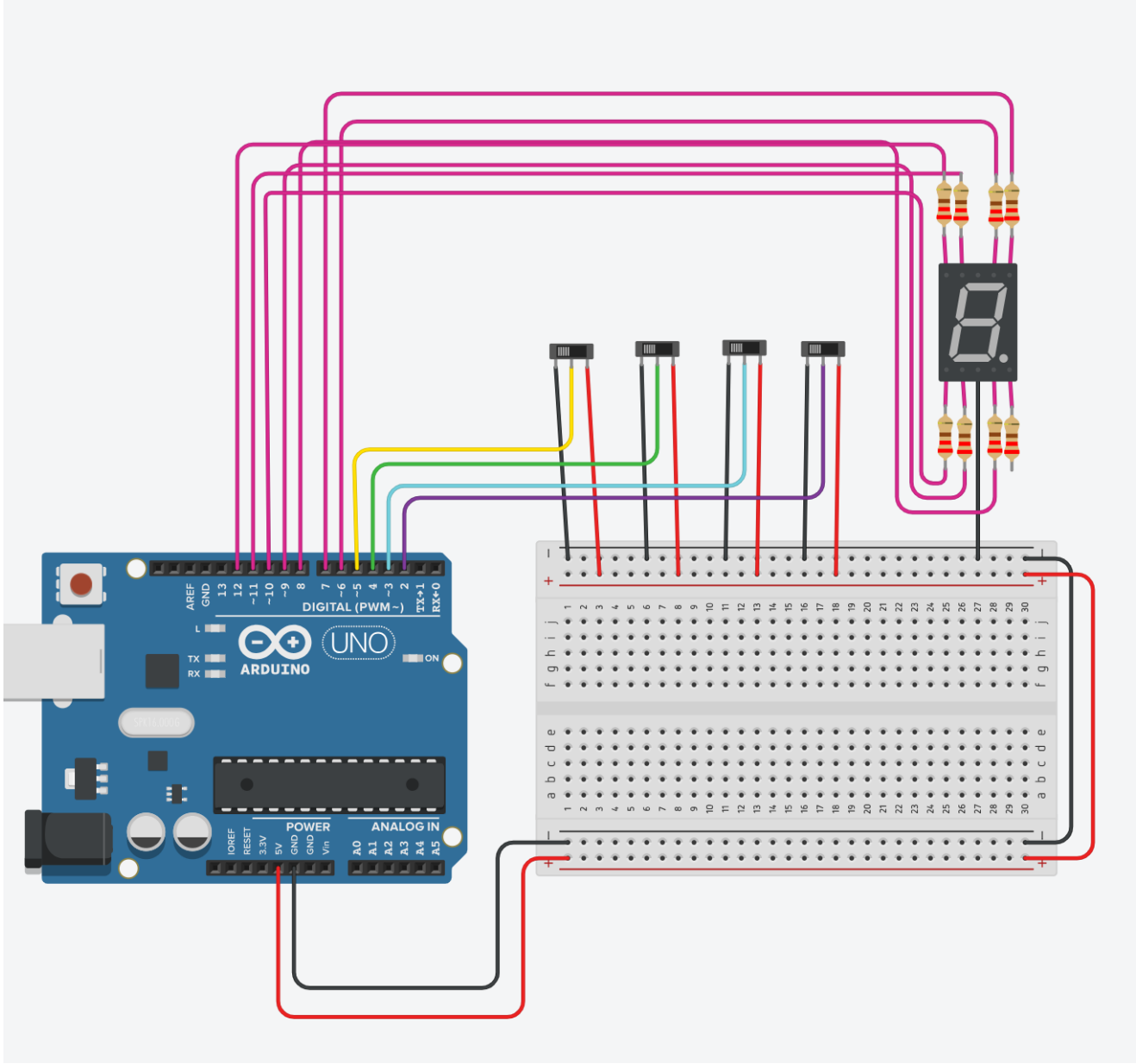
void setup()
{
  pinMode(A0, INPUT);
  pinMode(LED_BUILTIN, OUTPUT);
}
```
```

```
void loop()
{
  // read the value from the sensor
  sensorValue = analogRead(A0);
  // turn the LED on
  digitalWrite(LED_BUILTIN, HIGH);
  // pause the program for <sensorValue> millseconds
  delay(sensorValue); // Wait for sensorValue millisecond(s)
  // turn the LED off
  digitalWrite(LED_BUILTIN, LOW);
  // pause the program for <sensorValue> millseconds
  delay(sensorValue); // Wait for sensorValue millisecond(s)
}
```

Afficheur 7 segments

Un afficheur sept segments est un ensemble de DEL (chacune représente un segment) qui permet d'écrire les chiffre de 0 à 9, certains ont un point en plus.

L'exemple suivant permet de convertir 4 bits en binaire représentés par des interrupteurs en un chiffre décimal sur l'afficheur
https://www.tinkercad.com/things/h056BxJQSHK


DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN

```c
// Définition des broches des interrupteurs
const int switchPin1 = 2;
const int switchPin2 = 3;
const int switchPin3 = 4;
const int switchPin4 = 5;

// Définition des broches pour le branchement de l'afficheur 7 segments (exemple)
//A > G
const int segmentPins[] = {6, 7, 8, 9, 10, 11, 12};
//const int segmentPins[] = {12, 11, 10, 9, 8, 7, 6};


void setup() {
  Serial.begin(9600);
  // Configuration des broches des interrupteurs en entrée avec résistance de pull-up interne activée
  pinMode(switchPin1, INPUT_PULLUP);
  pinMode(switchPin2, INPUT_PULLUP);
  pinMode(switchPin3, INPUT_PULLUP);
  pinMode(switchPin4, INPUT_PULLUP);

  // Configuration des broches de l'afficheur 7 segments en sortie
  for (int i = 0; i < 7; i++) {
    pinMode(segmentPins[i], OUTPUT);
  }
}

void loop() {
  // Lecture des états des interrupteurs
  int switchState1 = digitalRead(switchPin1);
  int switchState2 = digitalRead(switchPin2);
  int switchState3 = digitalRead(switchPin3);
  int switchState4 = digitalRead(switchPin4);

  // Calcul de la valeur décimale en fonction des états des interrupteurs
  int decimalValue = switchState1 + (switchState2 * 2) + (switchState3 * 4) + (switchState4 * 8);

  Serial.println(decimalValue);
```

```
  // Affichage de la valeur décimale sur l'afficheur 7
segments
  displayDecimal(decimalValue);

  delay(100);  // Délai pour éviter une mise à jour trop
rapide de l'affichage
}

// Fonction pour afficher la valeur décimale sur l'afficheur
7 segments
void displayDecimal(int value) {
  // Tableau des correspondances pour les segments
  //les valeurs des entiers suivants sont rentrées en binaire
  const int digitValues[] = {
    B11111100,  // 0
    B01100000,  // 1
    B11011010,  // 2
    B11110010,  // 3
    B01100110,  // 4
    B10110110,  // 5
    B00111110,  // 6
    B11100000,  // 7
    B11111110,  // 8
    B11110110   // 9
    //chiffres hexadecimaux à rajouter si vous le souhaitez
  };

  // Affichage du chiffre décimal sur l'afficheur 7 segments
  for (int i = 7; i >= 0 ; i--) {
    int segmentState = bitRead(digitValues[value], 7-i);   //
Lecture de l'état du segment correspondant

    digitalWrite(segmentPins[i], segmentState);   //
Activation ou désactivation du segment (niveau logique
inverse)
  }
}
```

DEL RVB

à vous de jouer !