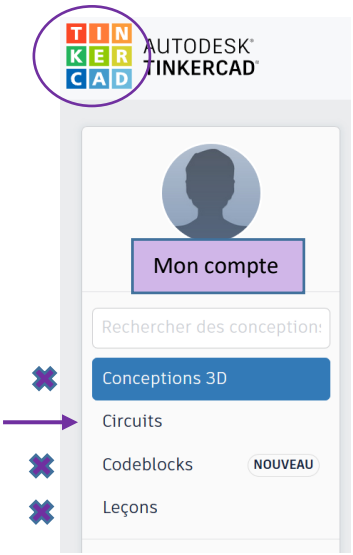
TP arduino : simulation avec tinkercad

Nous allons utiliser un site en ligne permettant de simuler des circuits électroniques comprenant des arduino ainsi qu'une interface permettant de modifier leur programme.

Rendez-vous donc sur https://www.tinkercad.com/ .

1. tinkercad



Le site permet également de faire de la conception de modèle 3D que vous pouvez explorer sur votre temps libre.

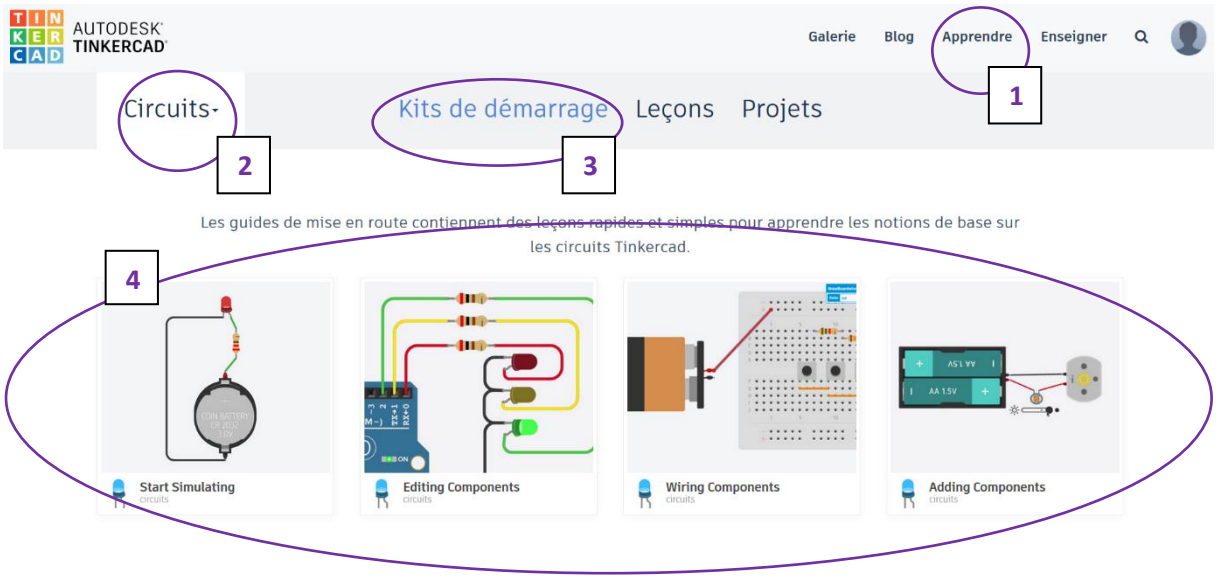
Faites un compte personnel (il n'est pas nécessaire de faire un compte étudiant).

Et connectez-vous, vous arrivez sur votre page personnelle de conceptions, à laquelle vous pouvez retourner en cliquant sur le logo tinkercad en haut à gauche.

Vous êtes par défaut dans la section conception 3D et vous devez passer à la section circuits.

Cliquez ensuite sur le bouton créer un circuit pour commencer votre simulation.

Je vous recommande de commencer par les tutoriels de base du site qui vous permettront de prendre en main l'ajout et l'édition de composants, le câblage ainsi que le démarrage et l'arrêt d'une simulation. Pour les trouver suivez les étapes 1 à 4 ci-dessous :



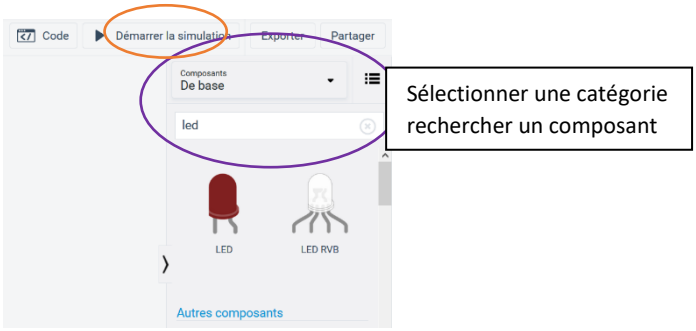
Remarques diverses:

- Vous pouvez faire pivoter votre composant avec la touche r.
- Vous pouvez placer des points intermédiaires sur vos fils pour mieux contrôler leur position.

- certaines parties des tutoriels sont en anglais mais vous pouvez chercher les composants par leur nom français (ex : résistance plutôt que resistor, platine plutôt que breadboard). Voici la liste des composants susceptibles d'être utilisés pendant ce TP :
    - Résistance
    - Photorésistance
    - Potentiomètre
    - Bouton poussoir
    - Condensateur
    - Led
    - Led rvb
    - Arduino uno r3

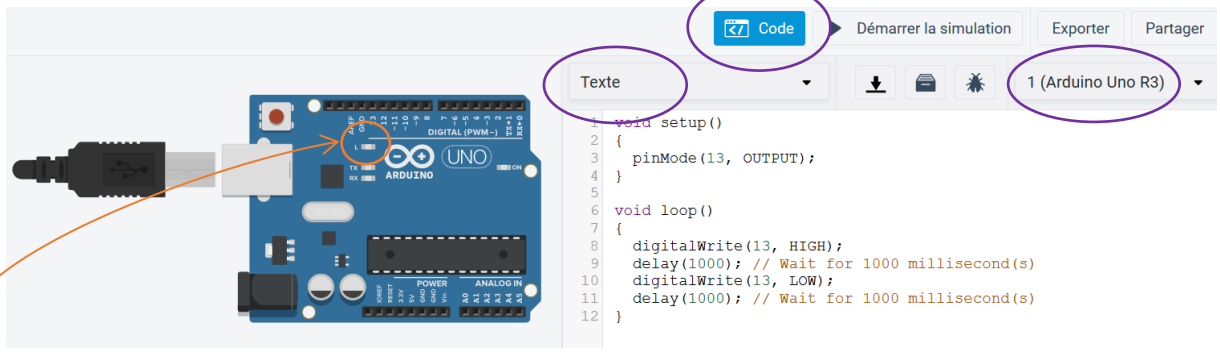
Votre interface se compose d'un grand espace vide dans lequel vous pouvez ajouter des composants pour créer votre circuit et de plusieurs boutons d'option :

En particulier à droite vous pouvez trouver des composants. Il suffit de cliquer-déplacer sur l'un d'entre eux pour l'ajouter à votre projet.



Quand un composant est sélectionné vous pouvez modifier ses propriétés (ex : couleur d'une led, valeur d'une résistance, couleur d'un fil…)

Vous pouvez démarrer la simulation, ce qui correspond à démarrer l'alimentation. A partir de ce moment vous ne devez plus toucher au circuit (dans tinkercad vous ne pouvez rien modifier) au risque de détériorer un composant. Pour effectuer une modification vous devez arrêter la simulation.



Après avoir placé au moins un arduino sur votre plan de travail vous pouvez éditer son code. Le code est par défaut en mode block. Passez en mode texte et vérifiez que vous éditez le bon arduino (même nom que le composant) dans le cas ou vous en avez mis plusieurs.

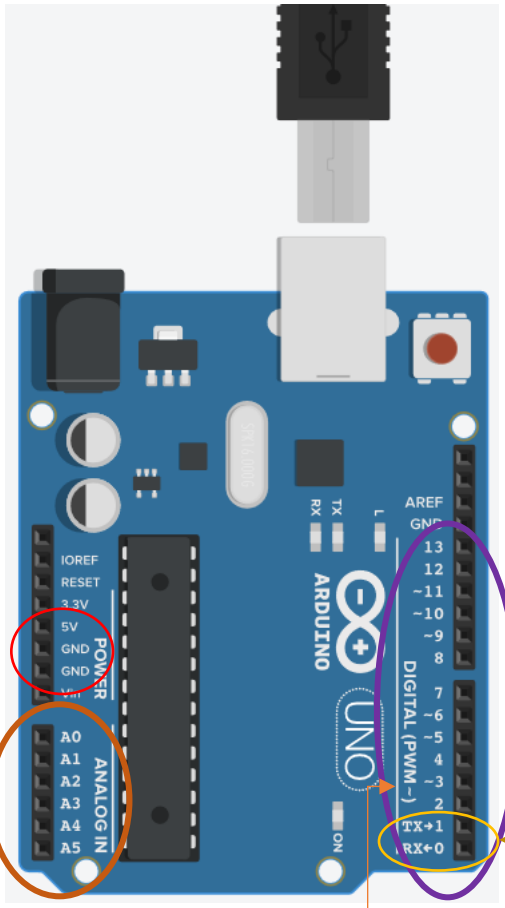
Le code par défaut fait clignoter la led interne de l'arduino (également reliée à la pin 13), vous pouvez le tester en démarrant la simulation.

2. Arduino le composant

La carte seule

C'est un composant semblable à mini-ordinateur programmable avec de nombreuses broches d'entrée-sorties (pin) numérotées sur les bords. Il fonctionne sur la base de signaux électriques de tension entre 0 et 5V qui peuvent être logique ou analogiques.

Les signaux logiques n'ont que deux valeurs (signal binaire) High/Low, 5V/0V tandis que les signaux analogiques peuvent prendre n'importe quelle valeur entre 0 et 5V.



Sortie **5V** et sortie **0V** (la masse / ground = gnd)

Entrées analogiques : peuvent convertir une valeur de tension réelle en signal binaire sur 10 bits donc entre 0 et 1023 à rapporter à l'intervalle de référence [0 – 5]V

Peuvent toutes être utilisées soit en entrée soit en sortie pour des signaux logiques 0/5 V

Les numéros précédés d'un ~ peuvent également être utilisés comme sortie pour un signal analogique (disposent de PWM)

Valeur entière (un octet) entre 0 et 255 -> convertie entre 0 et 5V

Interruptions :
Int 0 : pin2
Int 1 : pin3

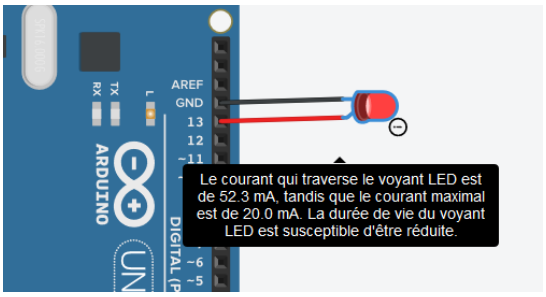
Communication série (avec ordinateur par exemple)

Au sein d'un circuit

La loi d'ohm doit être gardée à l'esprit pour concevoir un circuit même basique :

U = RI (en Volts, Ohms et Ampères)

Par exemple avec le code initial faisant clignoter la led interne sur la broche 13, si l'on essaie de reproduire le branchement de cette led avec un composant externe on peut obtenir l'exemple ci-joint.



Remarque 1) En étant branché à la masse (0V), lorsque que le signal logique délivré par la broche 13 est HIGH (5V), il y a une tension u = 5V – 0V appliquée à la led et celle-ci s'allume. Lorsque que le signal est LOW par contre la tension est nulle (0V-0V), il n'y a donc pas de courant et la led est éteinte.

Remarque 2) La résistance d'une led n'est pas très élevée et elle ne supporte pas un courant de trop grande intensité, ce circuit endommage donc le composant. Pour éviter cela il faut diminuer le courant donc augmenter la résistance.

Question 1 : Faites le branchement précédent et insérez une résistance en série avec la led.

Modifiez les valeurs de la résistance et dites dans chacun des cas suivant si la led s'allume faiblement/fortement/pas du tout et si elle est en condition de s'endommager.

Valeur	20 pΩ	200 mΩ	20 Ω	200 Ω	20 KΩ	2 MΩ	2GΩ
Allumée ?	Fortement	fortement	fortement	fortement	faiblement	faiblement	très faiblement voire éteint
Danger ?	oui	oui	oui	non	non	non	non

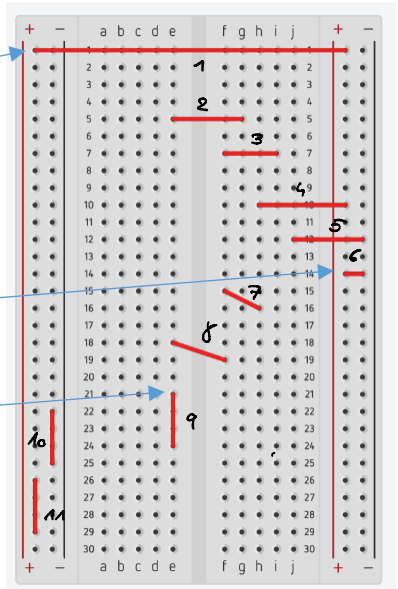
Avec une breadboard

En présence de nombreux composants, il est plus simple d'utiliser une breadboard (platine).
Ce composant est une plaque pleine de trous sur lesquels accrocher nos composants et certains de ces trous sont déjà reliés entre eux. En plaçant le curseur sur un trou le logiciel indique tous ceux à qui il est relié.

Question 2 :

Indiquez pour chacun des câbles suivants, numérotés de haut en bas, s'ils ont un effet ou non (extrémités déjà connectées)

numéro	utile	pas d'effet
1		x
2	X	
3		x
4	x	
5	x	
6		x
7	x	
8	x	
9	x	
10		x
11		x



3. Arduino : le code

Il s'agit essentiellement de révisions sous forme de synthèse.

1. Le code de l'arduino peut se découper en trois parties : La fonction **setup ()** exécutée en premier et une seule fois, la fonction **loop()** exécutée en boucle à l'infini et **le reste** à l'extérieur des deux premières fonctions, qui peut contenir les ajouts de librairie, des déclarations de variables globales et des fonctions supplémentaires crées à votre convenance.

```
int BuiltInLED = 13;

void setup()
{
  pinMode(BuiltInLED, OUTPUT);
}

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

2. Les entrées/sortie :
    a. On déclare dans le setup si l'on compte utiliser une **broche (pin) comme entrée (INPUT) ou sortie (OUTPUT)** avec la commande pinMode(int numero, commande). Par souci de lisibilité, on renomme généralement les numéros de pin utilisées dans le code.
    b. On peut envoyer un signal en sortie logique avec la commande digitalWrite(int numPin, commande) commande pouvant prendre uniquement les valeur HIGH et LOW.
    Pour un signal de sortie analogique la commande est analogWrite(int numPin, int value) avec value entier entre 0 et 255.
    c. Pour lire en entrée les fonctions sont respectivement int digitalRead(int numPin) et int analogRead(int numPin).
    d. Il existe des entrées avec un fonctionnement différent et plus immédiat, les interruptions qu'il faut déclarer également dans le setup avec attachInterrupt(int numeroInterruption, nomfonction, mode), attention le nom de fonction ne prend pas de parenthèse dans cette instruction, le numéro d'interruption est différent du numéro de la pin
3. Faire attendre le programme :
    a. Void delay (int): reste sur cette ligne de code tant que le délai (en ms) passé en paramètre n'est pas écoulé.
    b. Int milis () : retourne le nombre de millisecondes écoulées depuis le démarrage de l'arduino. Ce n'est pas une instruction bloquante mais elle peut être utilisée dans un test pour déclencher un effet après un certain temps.

Ces deux fonctions sont perturbées par les interruptions qui mettent en pause l'horloge interne de l'arduino.

Exercice 3 : allumer une led avec sortie logique et analogique

Q3.1) Parmi les valeurs de résistance étudiées question 1, laquelle permet de faire briller la led le plus vivement sans l'endommager ? 200 Ω

Utilisez-la dans votre montage et vérifiez que le code initial permettant de faire clignoter une led fonctionne bien.
Modifiez alors la fonction digitalWrite en analogWrite sans modifier ses arguments.

Q3.2) Parmi les 13 pin, donnez toutes celles qui permettent à la led de s'allumer ? 3, 5, 6, 9, 10, 11

Q3.3) Modifiez la valeur HIGH en différentes valeurs entières, commentez ce qu'il se passe pour les valeurs 1, 100, 255, 256, 257, 511 et 512.

512 —> éteint
511 —> allumé fort
257 —> allumé faible
256 —> éteint
100 —> allumé fort
1 —> allumé faible

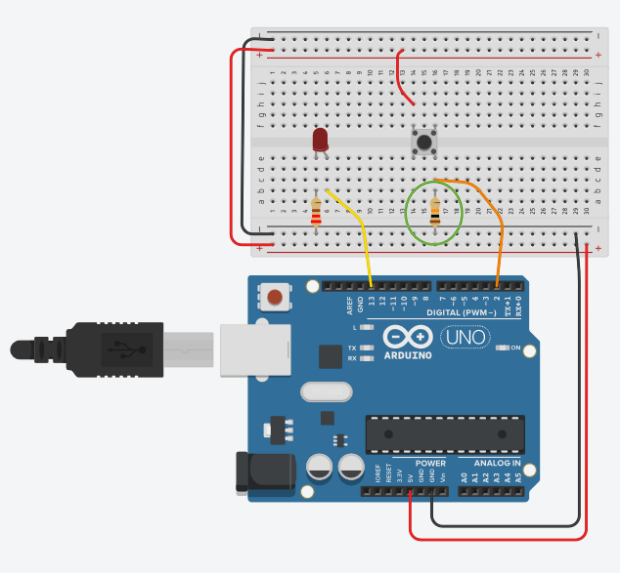
Q3.4) Ecrivez une fonction void decrease (int pin, int timems, int stepNumber) qui prend en paramètre une pin, et diminue sa valeur de sortie depuis la valeur maximale jusqu'à 0 en stepNumber étapes de timems millisecondes.
Utilisez cette fonction pour diminuer progressivement la luminosité de votre led.

Code :

```
1  void decrease(int pin, int timems, int stepNumber) {
2    int volt_step = 512 / stepNumber;
3    int ETA = 511;
4    for (int i = 0; i <= stepNumber; i++) {
5      analogWrite(pin, ETA);
6      ETA = ETA - volt_step;
7      delay(timems);
8    }
9  }
```

Exercice 4 : ajouter un bouton en entrée

Pour détecter si un bouton est appuyé ou non on le place de la manière suivante :



L'entrée de l'arduino est reliée à un coté du bouton ainsi qu'à une borne 0 ou 5v par l'intermédiaire d'une résistance de valeur élevée (10 K Ω) appelée **résistance de pull-up ou de pull-down** selon la borne qu'elle relie. L'autre borne est connectée à l'autre côté du bouton. Dans l'exemple, lorsque le bouton est appuyé le courant passant par la résistance est négligeable et l'entrée de l'arduino lit bien le signal de 5V. Lorsque le bouton est ouvert, l'entrée de l'arduino est reliée à la masse par la résistance et peut lire 0V au lieu de rester dans l'état indéterminé qu'elle peut avoir en l'absence de tout branchement.

Q 4.1) Ecrivez le code permettant de déclarer la pin associée au bouton comme étant une entrée ainsi qu'une fonction loop qui ne déclenche votre fonction decrease que si le bouton est appuyé et ne fait rien sinon.

Code :

```
void setup()
{
  pinMode(2, INPUT);
  pinMode(11, OUTPUT);
}

void loop()
{
  if (digitalRead(2) == HIGH) {
    decrease(11, 1000, 10);
  }
}
```

Q 4.2) Parmi les pin 0 à 13 (digital) lesquelles permettent de lier un signal à une interruption ?

0 et 1

Q 4.3) Reprenez votre fonction decrease mais retirez l'argument timems pour en faire une variable globale. Utilisez cette fonction afin de rallumer puis éteindre progressivement la led de manière répétée.

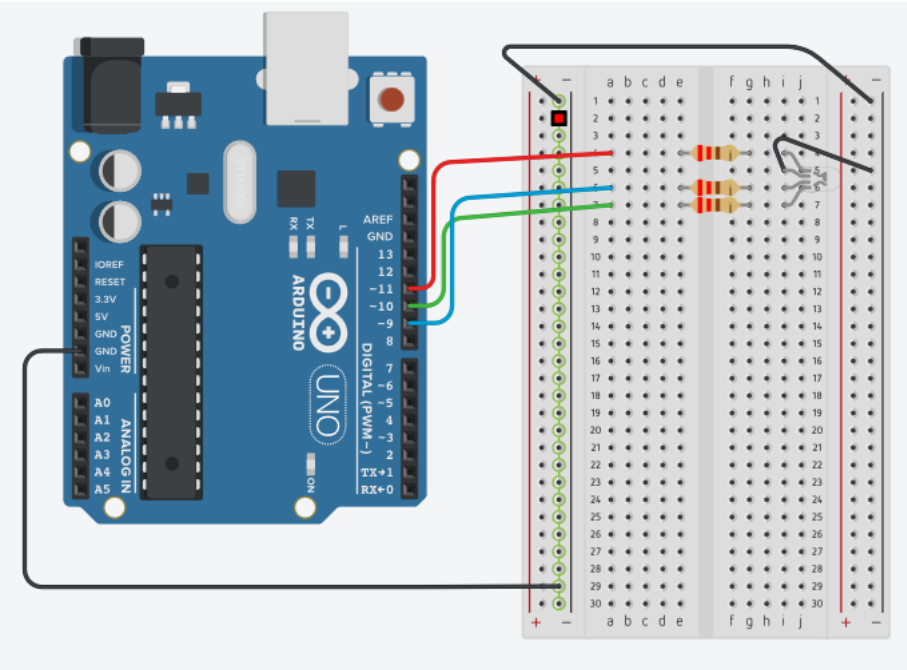
Rattachez le bouton à une interruption qui permet de changer cette variable timems pour passer d'une valeur à une autre en cycle entre trois modes lent(1sec) modéré(500ms) et rapide(100ms).

Code :

Snippet sur Teams

Exercice 5 : La led RVB

La led rvb est contrôlée par trois entrées, une pour chacune des valeurs primaires. Sur l'exemple ci-dessous les fils ont été choisis pour mettre en évidence la couleur commandée.



Q 5.1) On souhaite vérifier par un signal de sortie logique (digitalWrite) simple que le fil rouge est bien connecté à l'entrée régulant la couleur rouge de la led. Comment procédez-vous ?

On désactive les entrées bleues et vertes et on met la sortie rouge à HIGH.

Q 5.2) Toujours avec des signaux logiques testez ce que donne la combinaison de plusieurs entrées. Complétez avec les couleurs cyan magenta jaune et blanc :

- Rouge + bleu -> magenta
- Rouge + vert -> jaune
- Vert + bleu -> Cyan
- Rouge + vert + bleu -> blanc

Peut-on brancher les fils sur les entrées 7,6 et 5 pour ce test ? justifiez.

Aucun soucis. Les pins analogiques font également numérique.

Q 5.3) Utilisez maintenant des signaux analogiques pour allumer ou éteindre progressivement une couleur de la led. Ecrivez une fonction increase et une fonction decrease qui prennent un numéro de pin en paramètre et font varier la sortie entre sa valeur minimale et maximale. Le nombre d'étapes et la durée d'une étape seront définies en variables globales.

En supposant que l'entrée 9 ne fonctionne plus, quelle autre entrée proposez-vous pour le fil bleu et pourquoi ?

6, 5 et 3 car ce sont des pins analogiques.

Utilisez ces fonctions pour allumer la led selon la séquence suivante en utilisant des transitions douces : éteinte -> rouge -> jaune -> verte -> cyan -> blanc -> magenta -> bleue -> éteinte

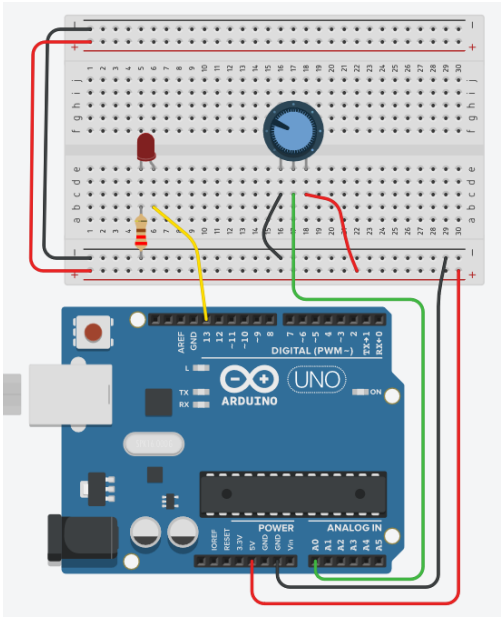
Code :

Q 5.4) Rajoutez votre bouton d'interruption pour modifier la vitesse de transition d'une couleur à l'autre.

Copie d'écran du circuit (windows+shift+s par exemple) et Code source à joindre.

Exercice 6 : entrée analogique : un potentiomètre

Voici un exemple d'utilisation du potentiomètre, il permet ici de réguler la fréquence à laquelle la led clignote. Vous pouvez tester le composant sur ce circuit.




```
int sensorValue = 0;

void setup()
{
  pinMode(A0, INPUT);
  pinMode(13, OUTPUT);
}

void loop()
{
  // read the value from the sensor
  sensorValue = analogRead(A0);
  // turn the LED on
  digitalWrite(13, HIGH);
  // pause the program for <sensorValue> millseconds
  delay(sensorValue); // Wait for sensorValue millisecond(s)
  // turn the LED off
  digitalWrite(13, LOW);
  // pause the program for <sensorValue> millseconds
  delay(sensorValue); // Wait for sensorValue millisecond(s)
}
```

Q 6.1) Faites un affichage avec des leds de la valeur d'une entrée commandée par un potentiomètre.

L'entrée analogique commandée par le potentiomètre permet de lire une valeur entre 0 et 1023.

Faites un circuit avec 7 leds côte à côte contrôlées indépendamment avec des sorties logiques.

Les leds s'allument dans l'ordre, une led ne doit pas être allumée si l'une des led à sa gauche est éteinte. Le nombre de led allumées doit être proportionnel à la valeur lue sur l'entrée liée au potentiomètre par rapport à sa valeur maximale.

Pour quelles valeurs lues doit-on changer d'état ?

0<->1	
1<->2	
2<->3	
3<->4	
4<->5	
5<->6	
6<->7	

Copie d'écran du circuit et Code source à joindre.

Q 6.2) Reprenez le circuit de l'exercice 5 mais commandez cette fois ci la durée d'un pas de temps avec un potentiomètre. La durée doit varier entre 100ms et 2sec.

Copie d'écran du circuit et Code source à joindre.

Justifiez l'endroit où vous placez l'instruction permettant de lire la valeur sur l'entrée analogique.

4. Références :

Pour plus de cours, dont des explications électroniques plus complètes, et d'autres exercices vous pouvez consulter https://eskimon.fr/extra/ebooks/arduino-premiers-pas-en-informatique-embarquee.pdf

Vous pouvez également consulter les différentes leçons et projets disponibles sur tinkercad.