

PROGRAMMER ET DÉVELOPPER DES SYSTÈMES AVEC ARDUINO, INITIATION

Date : 21 juillet 2022

Auteur(s) : Sébastien Charles

Copyright : S. Charles, UVSQ - IUT de Mantes en Yvelines

Licence : CC 4.0 BY-NC-SA [<https://creativecommons.org/licenses/by-nc-sa/4.0/deed.fr>] + licence commerciale ET-LIOS [<https://et-lios.s-mart.fr/licencecommerciale/>]

Table des matières

Introduction	3
1. L'environnement de développement	4
2. La carte Arduino Uno	11
3. Réaliser son premier montage : contrôleur de LED	16
3.1. Introduction	16
3.2. Comprendre le principe de MLI	16
3.3. Déclaration d'une broche comme entrée	16
3.4. Délai entre deux exécutions de boucles	17
3.5. Réalisation du montage contrôleur de LED	18
4. Contrôleur de LED RGB	24
4.1. Introduction	24
4.2. Principe du RVB ou RGB	24
4.3. Montage et exécution du contrôleur de LED RGB	25
5. Pilotage de LED par des interrupteurs	28
5.1. Introduction	28
5.2. Montage et pilotage de LED par interrupteurs	28
6. Le moniteur série	30
6.1. Introduction	30
6.2. Afficher dans le moniteur série le caractère entré par l'utilisateur	30
6.3. Afficher dans le moniteur série des variables	32
7. Multimètre à 2 voies	34
7.1. Introduction	34
7.2. Montage, mesure de variations de tensions et traçage de courbes	34
8. Buzzer actif	39
8.1. Introduction	39
8.2. Montage et pilotage d'un buzzer actif	39
9. Buzzer passif	41
9.1. Introduction	41
9.2. Montage et pilotage d'un buzzer passif	41
10. Tilt	44
10.1. Introduction	44
10.2. Montage et allumage d'une LED par tilt	44
Glossaire	46

Introduction

Objectifs pédagogiques

L'objectif de cette ressource est de vous initier à l'utilisation des solutions Arduino pour développer des systèmes interactifs simples. Des notions d'électronique seront également abordées dans ce contenu.

Déroulement

Durée : 12 heures

Arduino propose un environnement de développement logiciel et une carte électronique multifonction permettant de réaliser en open source des prototypes électroniques interactifs rapidement et à moindre coût.

L'environnement de développement permet de programmer des fonctions électroniques par le biais d'un langage de haut niveau simplifié, proche du langage C. En complément de cet environnement, il existe de nombreuses bibliothèques facilitant l'usage de composants électroniques actifs et passifs.


La carte principale utilise un microcontrôleur pour traiter l'information et commander les composants et est équipée de connecteurs d'entrée / sortie numériques et analogiques pouvant être reliés à d'autres composants par des fils de type Dupont.

Site officiel Arduino ^[<https://www.arduino.cc/>]

1. L'environnement de développement

En premier lieu, il faut soit télécharger l'environnement de développement Arduino, soit utiliser la version web de l'IDE qui ne nécessite aucune installation.

Lien : Télécharger l'IDE [<https://www.arduino.cc/en/Main/Software>].

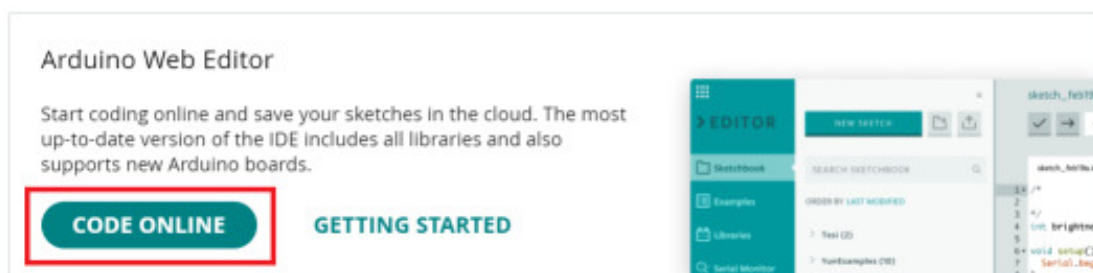
Si vous n'avez pas les droits d'administrateur sur vos sessions Windows, il vous faudra télécharger la version  Windows Zip file for non admin.



IDE pouvant être installé sans disposer des droits d'administrateur

Ceci est possible car l'IDE est codé en java, qui est un langage interprété par une machine virtuelle (non compilé) et donc compatible avec tous les systèmes d'exploitation et exécutable sans installation.



Pour utiliser la version web, cliquer ici [<https://www.arduino.cc/en/Main/Software>], puis choisir .



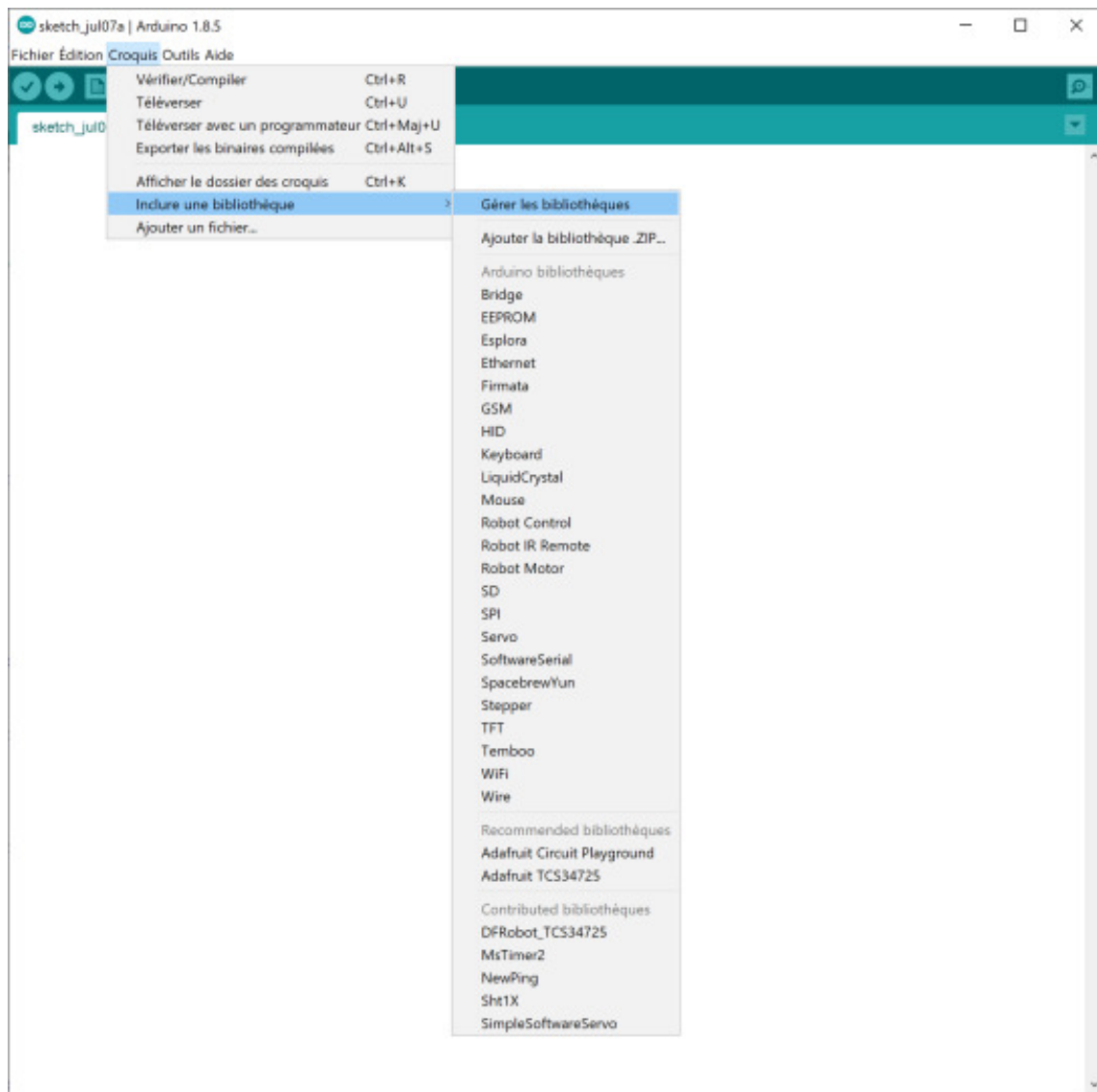
IDE version web

L'IDE est un environnement de développement qui permet d'implémenter un algorithme de contrôle / commande dans le micro-contrôleur de la carte Arduino (généralement un ATMEGA 328) par le biais d'un connecteur USB qui émule le port série traditionnel.

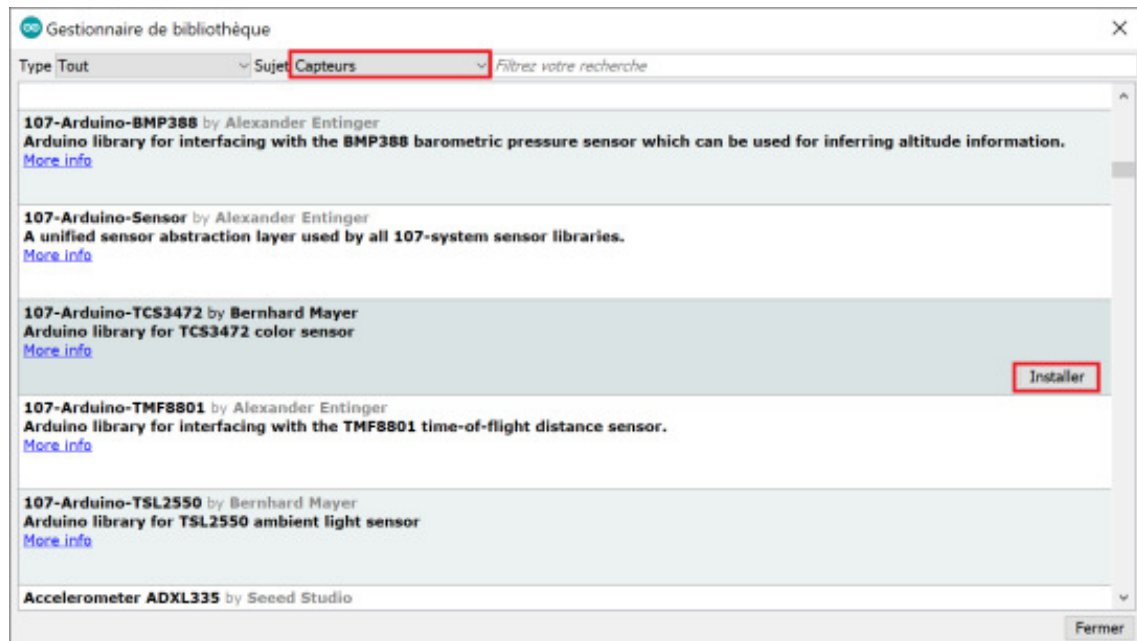
Certains programmes nécessitent des bibliothèques de fonctions qui simplifient la programmation.

Pour installer une nouvelle bibliothèque, le plus simple est d'utiliser le gestionnaire de bibliothèque en cliquant sur le menu  Croquis, puis  Inclure une bibliothèque/Gérer une

bibliothèque :



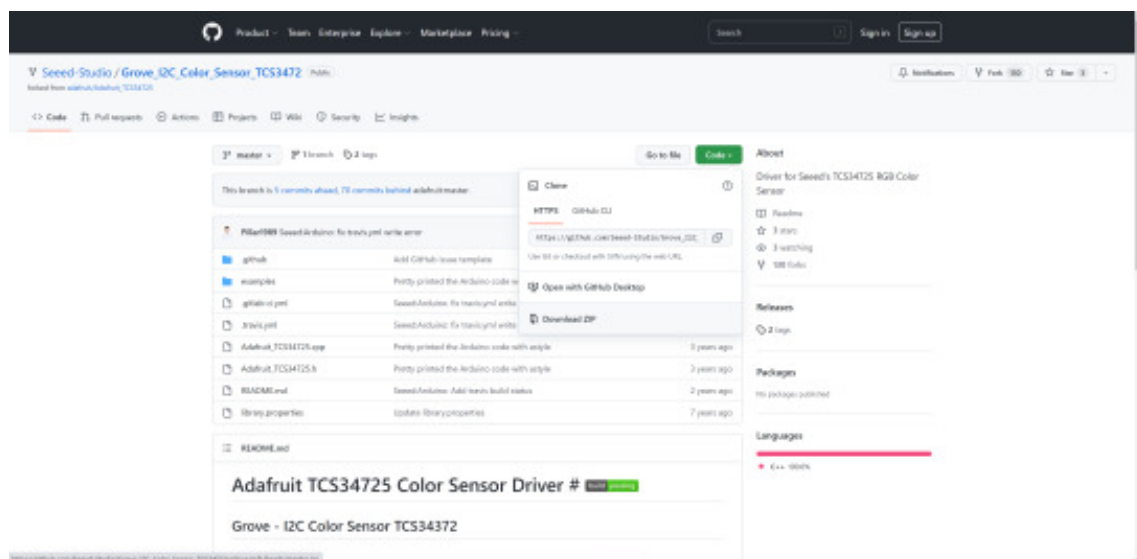
Ensuite, il vous suffit de rechercher la bibliothèque souhaitée, par mot clé ou par catégorie, par exemple dans la capture d'écran ci-dessous, nous ajoutons la bibliothèque d'un capteur de couleur TCS3472 en cliquant sur **Installer** :



Cette méthode nécessite toutefois l'utilisation de ports réseaux particuliers qui ne seront peut-être pas ouverts au niveau du proxy de votre établissement. Depuis votre connexion personnelle, il ne devrait pas y avoir de blocage. Si vous souhaitez installer des bibliothèques depuis votre établissement seule la méthode manuelle fonctionnera.

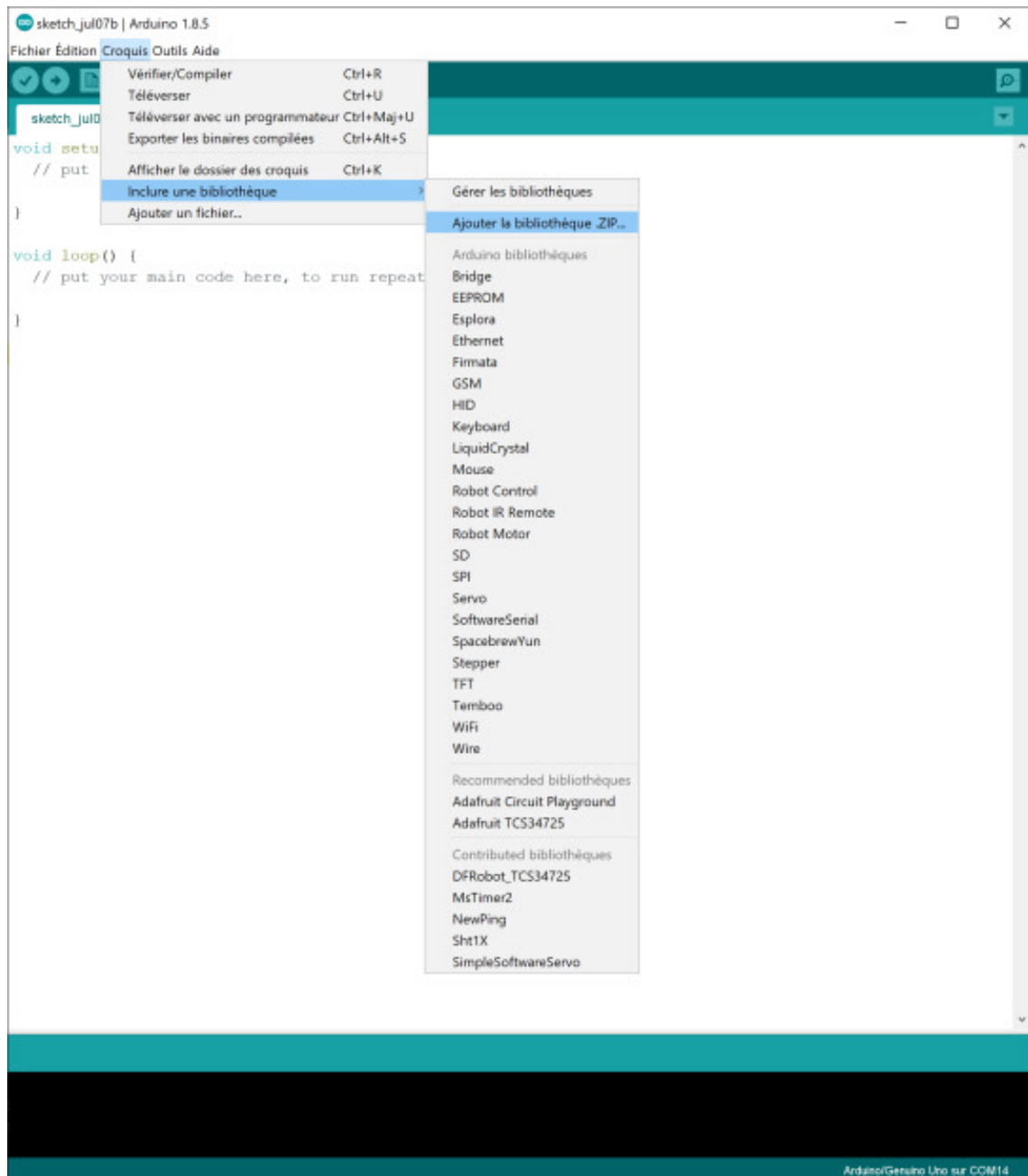
Pour ajouter manuellement des bibliothèques dans l'IDE Arduino, vous pouvez la télécharger sur un site, puis cliquer sur le menu **Croquis**, puis **Inclure une bibliothèque/Ajouter une bibliothèque .ZIP**.

Dans l'exemple ci-dessous, nous téléchargeons le zip de la bibliothèque du capteur en allant sur GitHub, puis en cliquant sur **Code**, puis sur **Download ZIP** :

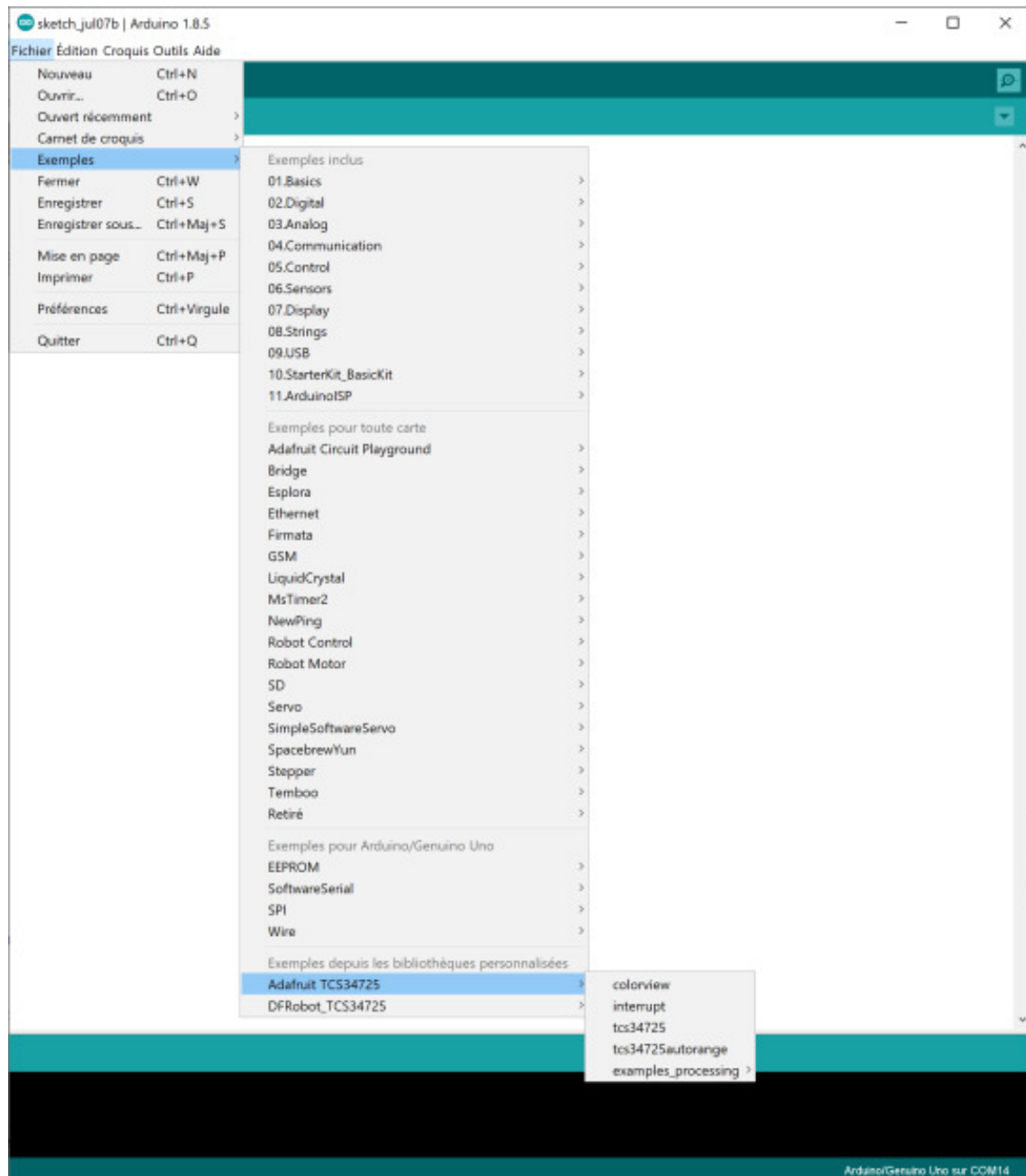


Le lien vers la page web ci-dessus : [GitHub \[https://github.com/Seeed-Studio/Grove_I2C_Color_Sensor_TCS3472\]](https://github.com/Seeed-Studio/Grove_I2C_Color_Sensor_TCS3472)

Ensuite, il suffit de cliquer sur le menu **Croquis**, puis **Inclure une bibliothèque /Ajouter une bibliothèque .ZIP** et de choisir le fichier téléchargé.



Une fois une bibliothèque téléchargée, vous obtenez de nouveaux exemples de code Arduino « Clés en main » en cliquant sur le menu **Fichiers/Exemples**, les bibliothèques téléchargées sont présentées en fin de liste.



⚠ Attention

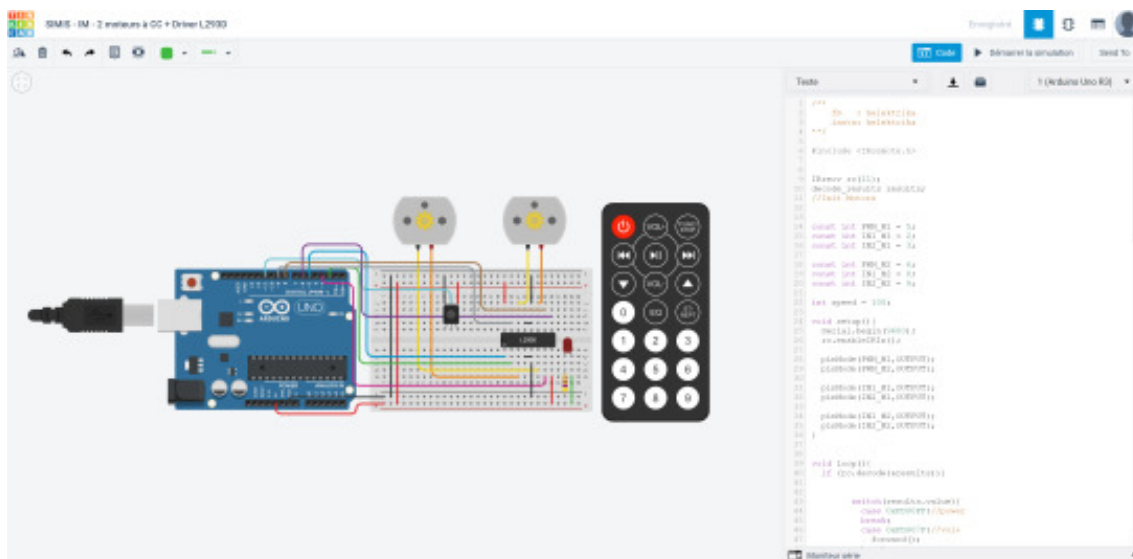
Les exemples de programmes présents dans l'IDE Arduino sont très utiles pour développer rapidement un code permettant de faire fonctionner un composant électronique.

Vous pouvez aussi installer une bibliothèque en l'incluant dans le dossier dédié à l'IDE. Pour ce faire, il suffit de fermer l'IDE, de coller les nouvelles bibliothèques dans le dossier intitulé `libraries` situé à la racine du dossier de votre IDE et de le relancer.

Cette ressource a pour objet de vous initier à l'Arduino et de vous apprendre à utiliser des capteurs et actionneurs courants pour prototyper un système mécatronique. Pour compléter vos compétences en Arduino, consultez le cours OpenClassrooms qui est bien plus complet : Programmez vos premiers montages avec Arduino [\[https://openclassrooms.com/fr/courses/2778161-programmez-vos-premiers-montages-avec-arduino\]](https://openclassrooms.com/fr/courses/2778161-programmez-vos-premiers-montages-avec-arduino)

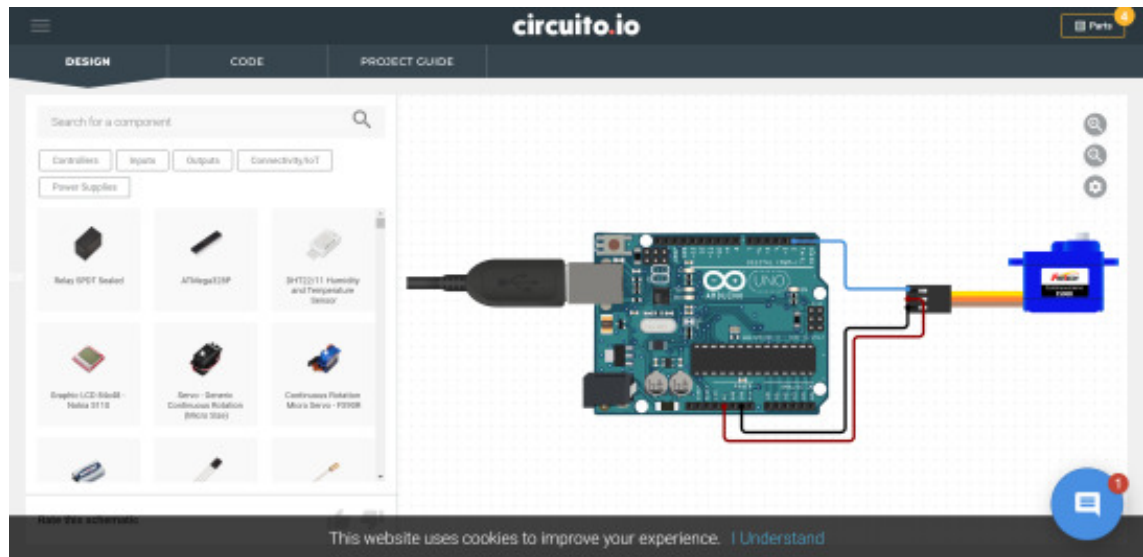
❖ Pour aller plus loin Si vous ne disposez pas de matériel électronique

Si vous ne disposez pas de carte Arduino, vous pouvez toujours réaliser le TP en allant sur le site Tinkercad [<https://www.tinkercad.com/>] qui, de manière collaborative, permet de simuler le fonctionnement d'une carte Arduino de manière réaliste (il s'agit d'une émulation). Cette solution entièrement web permet aussi de créer des modèles 3D et de les exporter au format STL pour les imprimer par exemple par fabrication additive. L'environnement web propose également de partager de manière communautaire les maquette 3D, les montages électroniques et les programmes réalisés.



❖ Pour aller plus loin Pour facilement composer vos propres montages

Vous pouvez utiliser le site Circuito.io [<https://www.circuito.io/>] qui propose un environnement web permettant de réaliser des montages électroniques avec Arduino et un large choix de composants. Cet environnement intelligent génère automatiquement les montages et programmes associés par simple glisser/déplacer des composants dans la fenêtre de création.

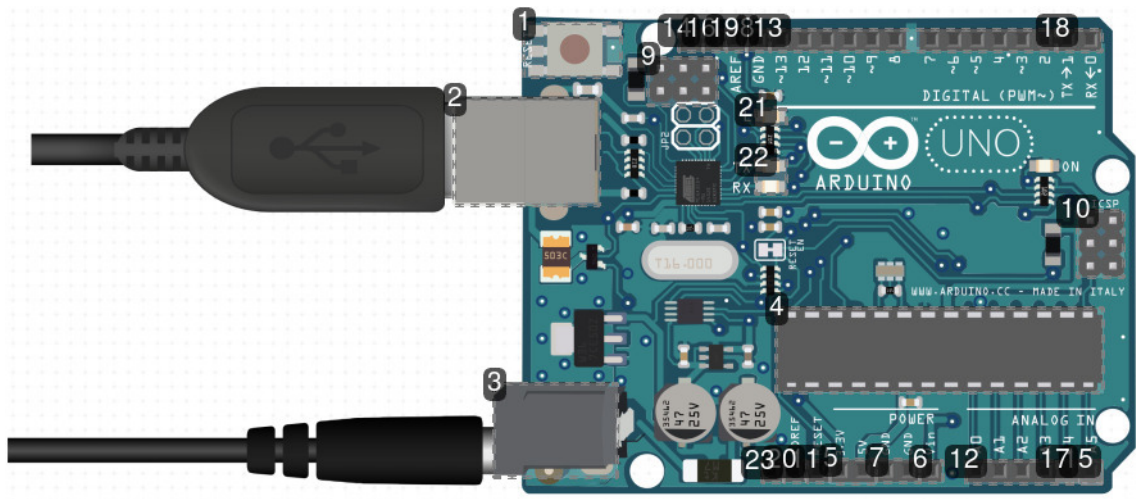


Pour concevoir des montages Arduino et plus généralement des circuits électroniques, la solution Fritzing [<https://fritzing.org/>] peut aussi vous être utile.

2. La carte Arduino Uno

La carte Arduino est une carte électronique permettant de programmer en langage C de manière simplifiée. Si nous utilisons une carte de programmation, c'est qu'elle nous permet de faire des branchements électroniques et donc de relier l'hardware au software.

Constitution d'une carte Arduino Uno

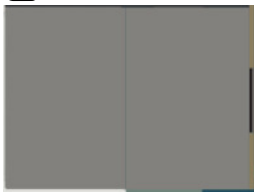


1 Bouton de remise à zéro du programme « RESET »



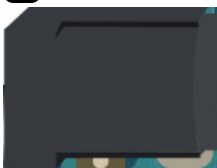
Ce bouton sert à réinitialiser le programme pour qu'il s'exécute depuis la première ligne de code.

2 Connecteur USB Type B (5V - 500mA max)



Ce connecteur permet d'alimenter la carte et de communiquer avec le microcontrôleur via un port COM virtuel (car la liaison physique USB fonctionne en série alors qu'un COM est utilisé pour des liaisons en parallèle).

3 Connecteur d'alimentation externe (7-12V)



Connecteur Rayon 2.1mm + au milieu

4 Microcontrôleur ATMEL de type ATMEGA 328



5 Broches d'alimentation de composants externes (3.3V et 5V - 500mA max)



6 Broche d'alimentation de la carte



7 Broche de masse (0V)



8 Broche de masse (0V)



9 Broches ICSP

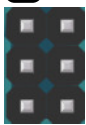


Permet de brancher une sonde de programmation ICSP pour programmer le microcontrôleur sans passer par le connecteur USB.

Cela permet de reprogrammer le microcontrôleur sans passer par l'USB et pour exécuter son programme en mode debug pour suivre en temps réel l'exécution des instructions et vérifier l'état de la mémoire.

Ces pins ICSP sont aussi mappés sur un bus SPI.

10 Broches ICSP



Permet de brancher une sonde de programmation ICSP pour programmer le microcontrôleur sans passer par le connecteur USB.

Cela permet de reprogrammer le microcontrôleur sans passer par l'USB et pour exécuter son programme en mode debug pour suivre en temps réel l'exécution des instructions et vérifier l'état de la mémoire.

Ces pins ICSP sont aussi mappés sur un bus SPI.

11 Broches pour piloter le reset de la carte



12 Broches Analogiques A0 à A5



13 Broches Numériques D2 à D13



Certaines de ces broches peuvent être pilotées de manière numérique ou analogique (à préciser lors de l'initialisation des entrées / sorties) : 3,5,6,9,10,11,13 pour piloter des sorties en PWM.

Les broches 2 et 3 peuvent être utilisées pour gérer les interruptions, respectivement 0 et 1.

14 Broche I2C (SCL)



15 Broche I2C (SCL)



16 Broche I2C (SDA)



17 Broche I2C (SDA)



18 Broches de communication en série asynchrone reliées au connecteur USB



D0 : Rx : entrée

D1 : Tx : sortie

19 Broche AREF



La broche AREF permet de fixer la valeur haute de la tension de référence pour les entrées analogiques. Si cette broche n'est pas reliée à une source de tension, la tension de référence est alors de 5V pour une UNO.

20 Broche IOREF



Cette broche fournit une tension similaire à celle utilisée par les entrée / sortie de la carte, par exemple 5V pour une UNO ou 3.3V pour une DUE.

21 LED intégrée reliée au connecteur numérique D13



Permet de tester le fonctionnement de la carte sans connecter de composants externes.

22 2 LED indiquant l'usage de la liaison série TX et de RX

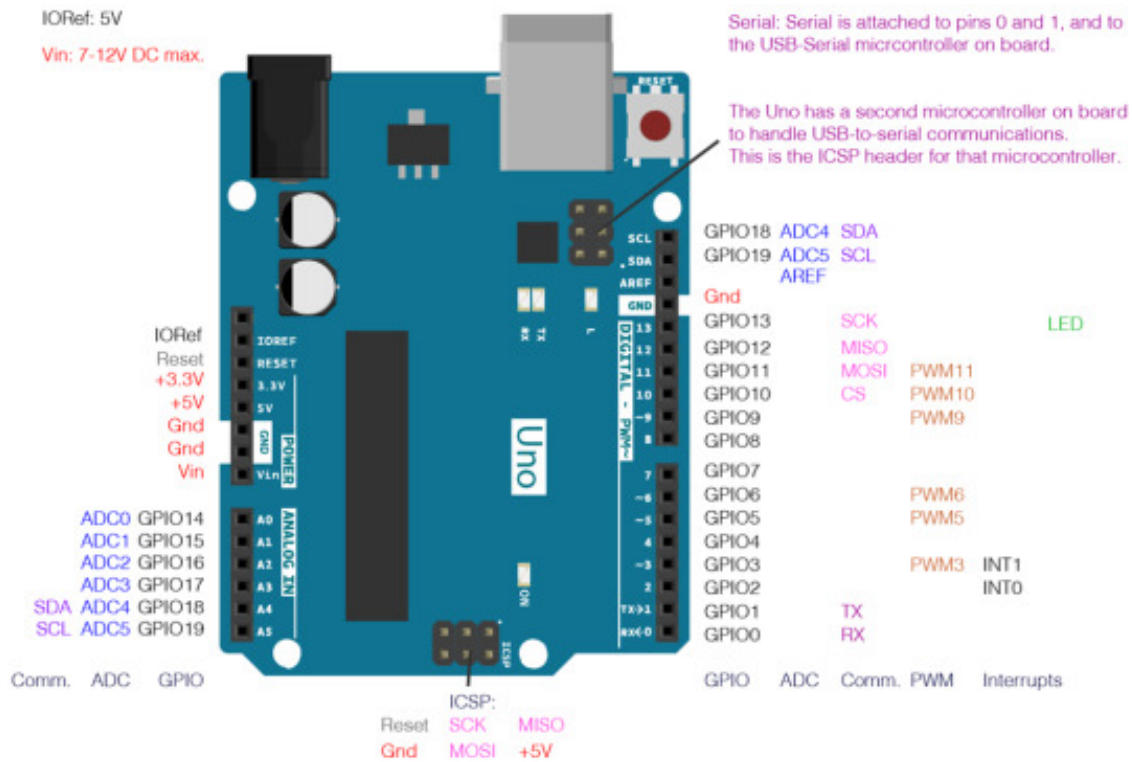


23 Broche inutile



Cette broche n'est pas reliée à la carte.

Sa présence s'explique sans doute pour éviter une découpe coûteuse du connecteur 8 broches lors de la fabrication en série



Détails des connecteurs de la carte

Une carte Arduino, bien que pratique, est limitée à l'usage de composants qui fonctionnent en basse tension.

Ses limites sont les suivantes :

- La carte peut supporter une tension d'entrée de 20 Volts au grand maximum.
- Intensité maximale disponible par connecteur d'entrée/sortie sous une tension de 5V est de 40 mA (avec un total maximal de 200 mA sur l'ensemble de ses connecteurs).
- Intensité maximale disponible pour la sortie 3,3V : 50 mA.
- Intensité maximale disponible pour la sortie 5V : 500 mA en cas d'alimentation par le port USB seul, sinon c'est en fonction de l'alimentation utilisée.

Si vous souhaitez contrôler des composants nécessitant des tensions ou des intensités plus importantes, il faut utiliser des cartes de puissance telles que des motor shield, driver ou relais. La carte

Arduino assure dans ce cas uniquement le rôle de contrôle/commande.

3. Réaliser son premier montage : contrôleur de LED

3.1. Introduction

Le premier exercice va consister à réaliser un contrôleur de LED par Modulation de Largeur d'Impulsions (MLI ; en anglais : Pulse Width Modulation, soit PWM) qui va faire accroître puis décroître son intensité lumineuse.

Pour réaliser ce montage, il faudra vous munir des composants suivants :

- une carte arduino
- un câble USB
- une LED
- un résistor de 220 Ohms
- des fils Dupont Mâle/Mâle
- une platine de prototypage rapide à trous de type Labdec

La plupart des montages et programmes présentés dans le module d'initiation sont basés sur le kit de développement de UCTRONICS, que vous pouvez vous procurer sur : UCTRONICS [<https://www.uctronics.com/maker-kits/arduino-kits/uctronics-advanced-starter-kit-for-arduino-with-instruction-booklet-uno-r3-uno-r3-protoshield-v3-relay-breadboard-power-supply-sg90-9g-servo-remote-controller-and-ir-receiver.html>]

3.2. Comprendre le principe de MLI

Le principe de MLI est très utilisé pour piloter des composants électroniques analogiques, c'est pourquoi il convient de le comprendre avant de l'utiliser dans le premier montage.

Consultez l'article MLI sur le site Wikipédia [https://fr.wikipedia.org/wiki/Modulation_de_largeur_d%27impulsion] et la vidéo Qu'est-ce que la PWM ? [<https://www.youtube.com/watch?v=CSReyYwbGRY>]

Question

Expliquez le principe de MLI en quelques lignes.

3.3. Déclaration d'une broche comme entrée

Le premier bloc d'un code Arduino contient les déclarations des variables du programme :

```
1 int ledpin=11; //précise que la broche numérique 11 sera utilisée par le programme
```

Le deuxième bloc contient une fonction qui est appelée une seule fois à chaque exécution du programme :

```
1 void setup () // ce bloc du code n'est exécutée qu'une seule fois à chaque
  initialisation
2 {
3   pinMode(ledpin,OUTPUT); // déclare que la broche 11 sera uniquement utilisée
  comme sortie
4 }
```

⚠ Attention

Pour comprendre et connaître la liste des instructions disponibles en Arduino, veuillez consulter le site de références Arduino [<https://www.arduino.cc/reference/en/#page-title>].

Question

Quelle est l'instruction pour déclarer qu'une broche est utilisée comme entrée ?

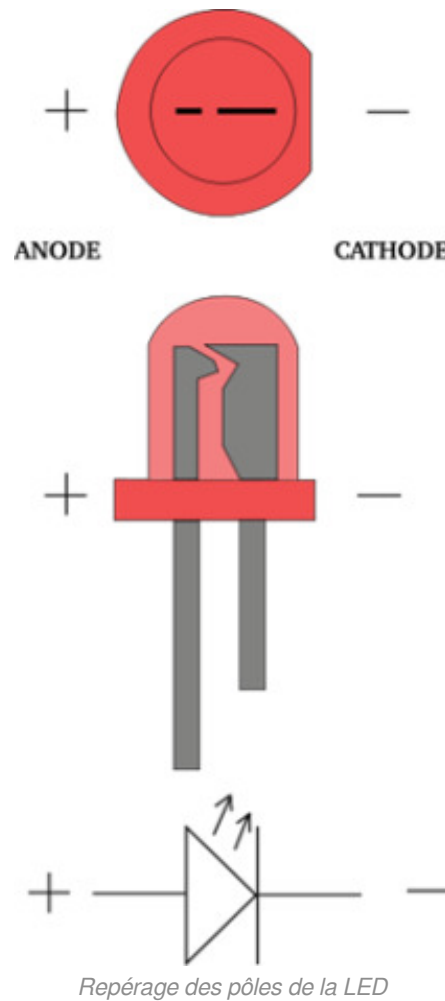
3.4. Délai entre deux exécutions de boucles

Le troisième bloc contient une fonction exécutée en boucle jusqu'à l'appui du bouton reset ou l'arrêt de l'alimentation de la carte. Le micro-contrôleur tourne naturellement à 16 MHz, et les boucles tournent généralement à environ 100 000 instructions par seconde (variable en fonction de la complexité des instructions) si aucun minuteur (delay) ne vient ralentir son exécution.

```
1 void loop() // ce bloc est exécuté en boucle jusqu'au débranchement de la carte ou
  appui sur le bouton reset
2 {
3   for (int a=0; a<=255;a++) // cette boucle permet d'incrémenter l'éclairage de
  la led par le biais d'un contrôle de type PWM (analogique), ceci est possible car le
  connecteur ou pin 11 a la particularité de pouvoir être utilisé en analogique ou
  numérique
4   {
5     analogWrite(ledpin,a); // cette boucle envoie la valeur de PWM à la sortie 11
  qui est déclarée comme analogique, 255 correspond à une valeur de 100 % en PWM
6     delay(15); // ajoute une temporisation de 15 ms entre deux incréments de
  niveau de luminosité afin que le changement reste perceptible
7   }
8   for (int a=255; a>=0;a--) // cette boucle décrémente la valeur de PWM jusqu'à l'
  extinction (valeur = 0)
9   {
10    analogWrite(ledpin,a);
11    delay(15);
12   }
13   delay(100); // temps de pause entre deux cycles d'augmentation et réduction de
  la luminosité
14 }
15
```

Une LED est un composant polarisé qui ne laisse passer le courant que dans un sens, comme le ferait une diode. Le courant circule, et donc la LED s'allume, quand sa patte la plus longue est reliée au 5V (ou

au 3.3V) et quand la plus courte est reliée au GND sur une carte Arduino :



Question n°1

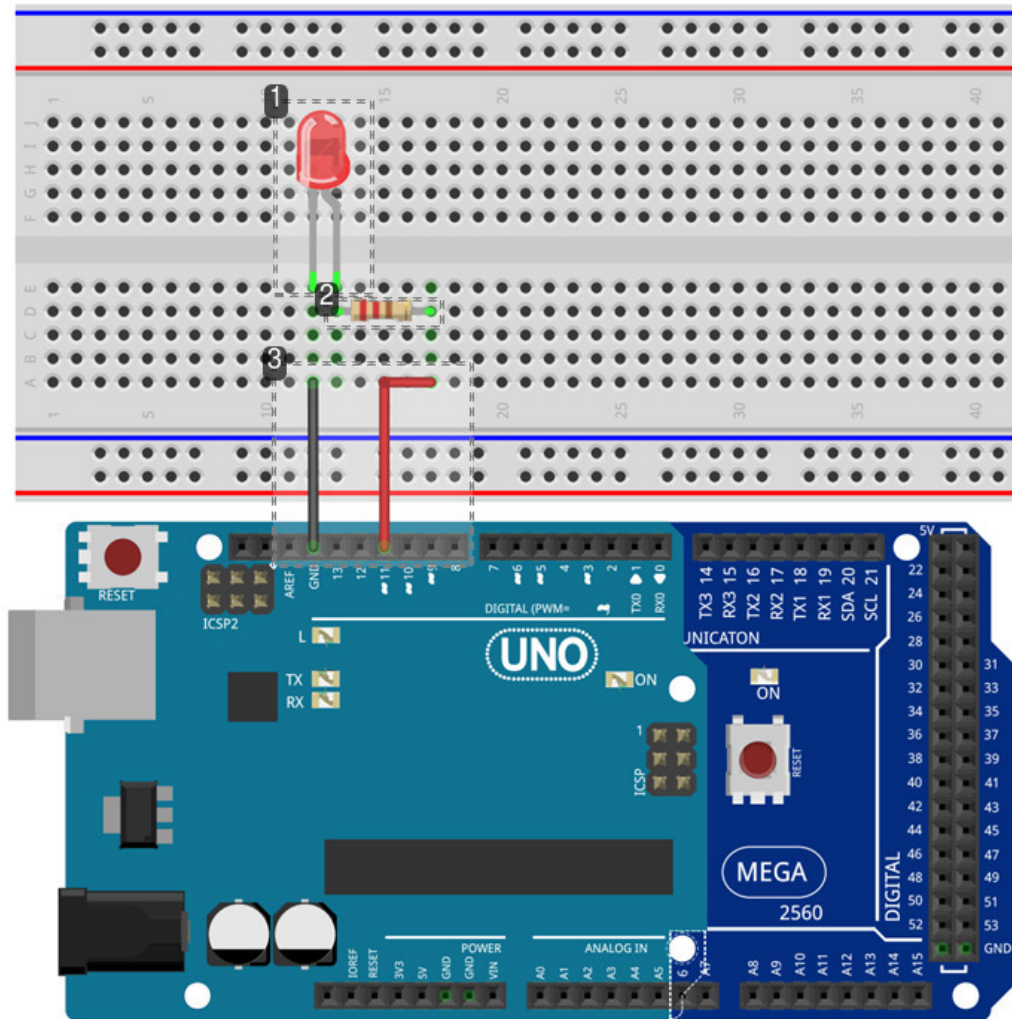
Comment fonctionne une boucle FOR en langage Arduino ?

Question n°2

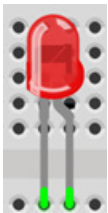
Pourquoi imposer un délai de 15 ms entre deux exécutions successives de boucles ? Il faut savoir que l'œil humain peut percevoir des changements allant jusqu'à 2 000 images par seconde.

3.5. Réalisation du montage contrôleur de LED

Réalisez le montage ci-dessous, puis copiez/collez les 3 blocs du code présentés auparavant dans l'IDE en respectant l'ordre indiqué et téléversez-le dans la carte Arduino.



1 LED



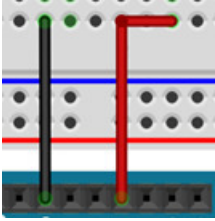
Diode électroluminescente

2 Résistor 220 ohm



Permet de baisser la tension dans un circuit.

3 Fils Dupont



Permet de relier les composants entre eux.

Platine Labdec

≈ Breadboard

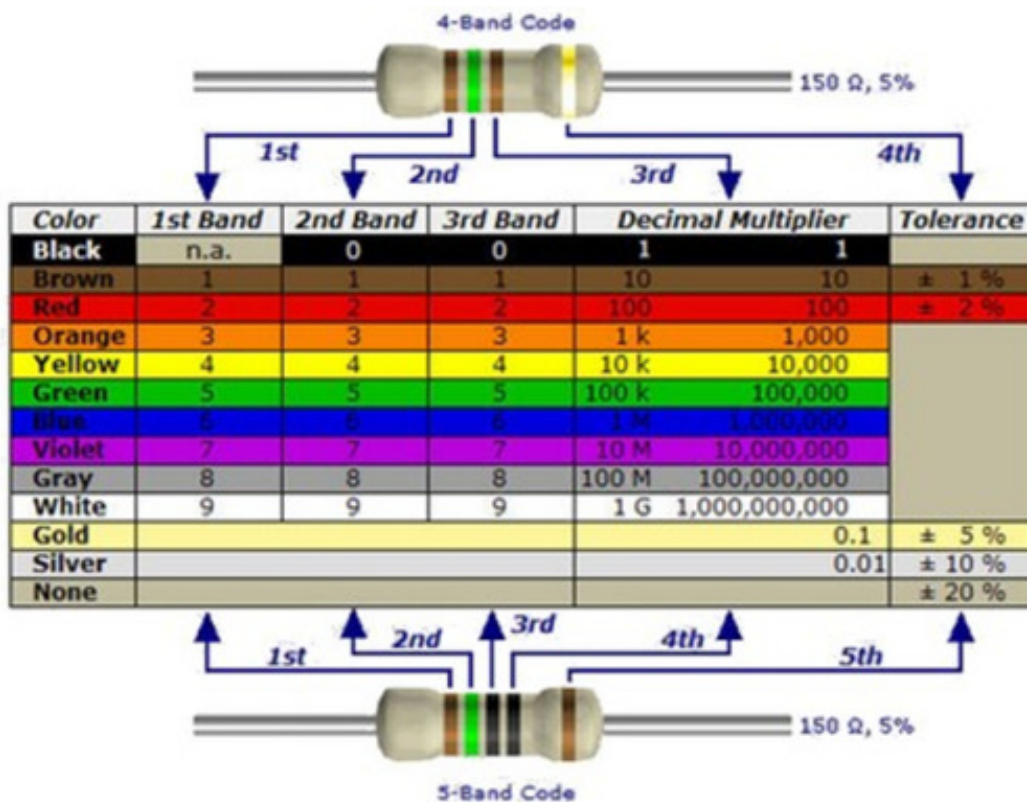
La « breadboard » ou « platine Labdec » simplifie le montage de composants. Cette platine est très utilisée pour le prototypage électronique afin d'éviter d'avoir recours à des cartes électroniques et des soudures.

Tous les connecteurs positifs et négatifs latéraux (entourés des lignes rouges et bleues) sont reliés entre eux. Tous les connecteurs des lignes A à E et des lignes F à J sont reliés entre eux (mais il n'y aucune liaison entre A-E et F-J).

Le schéma ci-dessous illustre les liaisons internes de la platine.

La résistance d'un résistor et la tolérance associée à cette valeur sont clairement identifiables à travers un code couleur normalisé.

Le code couleur des résistances est présenté ci-dessous.



Les lignes colorées précisent la valeur de la résistance et la tolérance associée

Résistance

Une résistance est un composant électronique ou électrique dont la principale caractéristique est d'opposer une plus ou moins grande résistance (*mesurée en ohms*) à la circulation du courant électrique.

La résistance électronique est l'un des composants primordiales dans le domaine de l'électricité.

Information technique



Lorsqu'une résistance est placée dans un circuit électrique on obtient deux types de phénomènes :

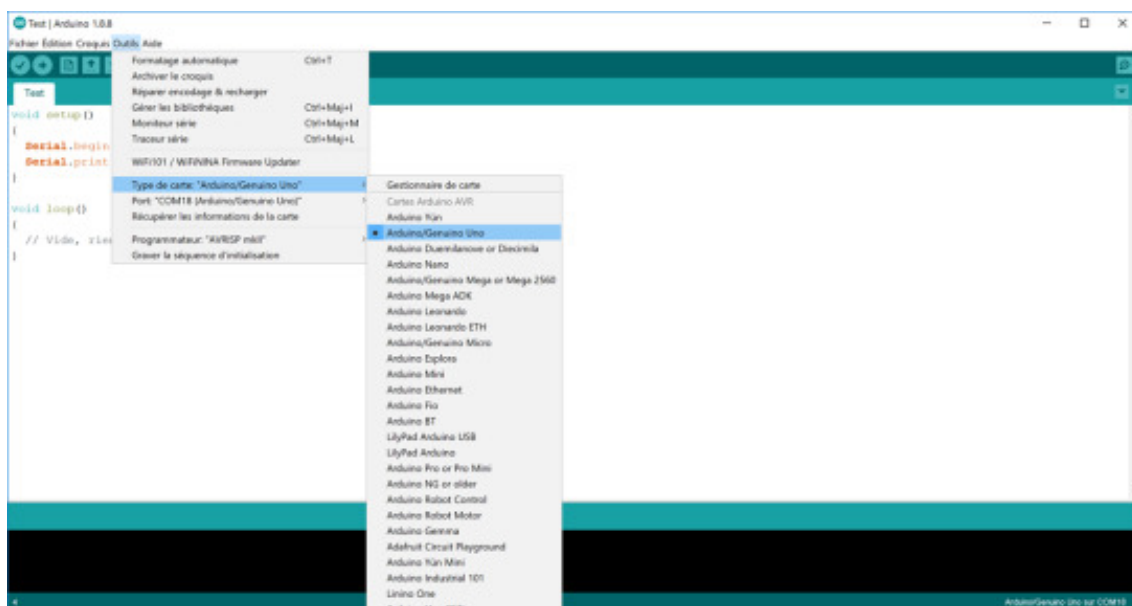
- La résistance a une influence sur l'intensité du courant continu, plus la résistance est élevée et plus l'intensité est faible.

Cette influence sur le courant est mise à profit dans de nombreux appareils électriques et électronique pour modifier l'intensité du courant. Elles permettent de protéger les dipôles qui ne supportent pas des intensités trop élevées. Dans ce cas les résistances servent à réguler l'intensité du courant continu.

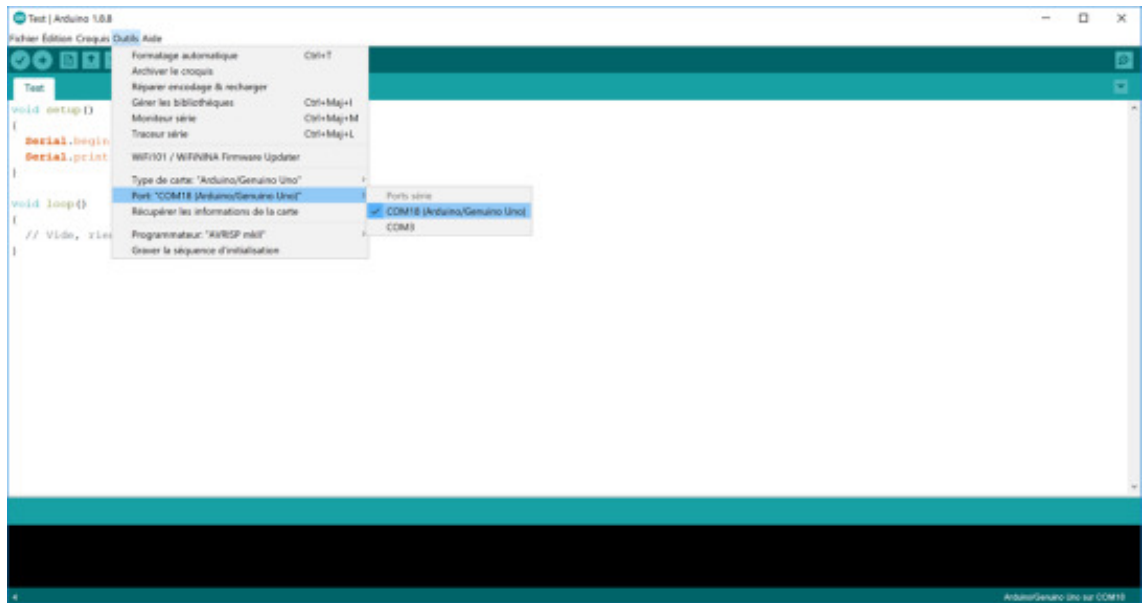
- La résistance parcourue par un courant électrique donne naissance à l'effet joule^[p.46].

Procédure : Téléverser un programme Arduino par le port USB

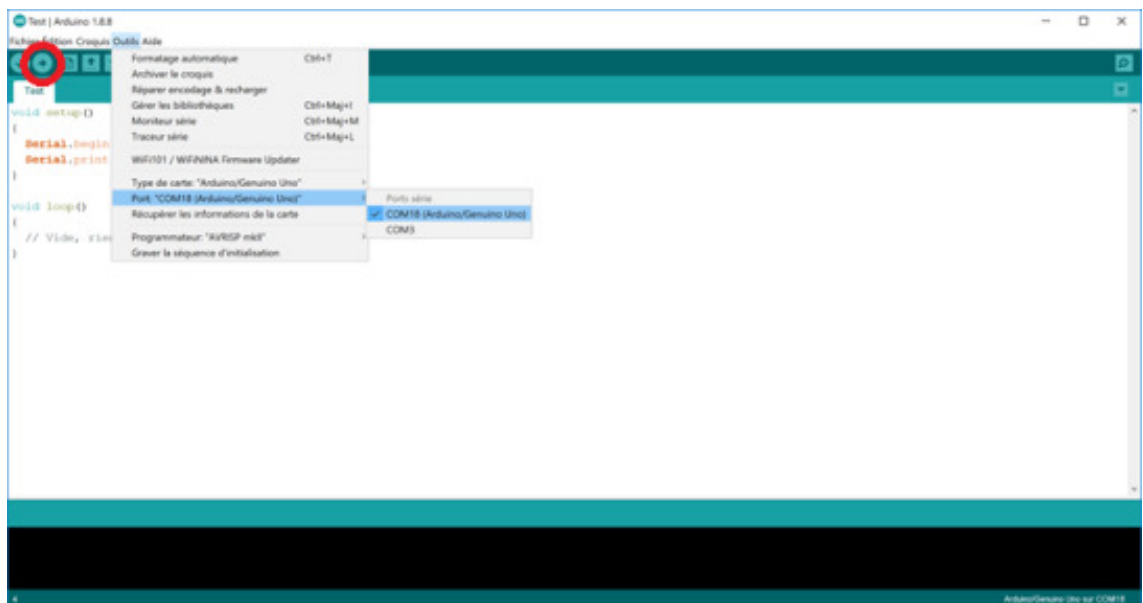
- 1** S'assurer que  Outils / Type de carte est bien configuré sur  Arduino /Genuino Uno.



2 S'assurer que le port COM est bien celui qui est connecté à votre carte Arduino. Si la carte est bien détectée, son nom est généralement écrit à côté du port COM associé.



3 Cliquer sur la flèche dirigée vers la droite.



Si le texte **Téléversement terminé** n'apparaît pas dans le bandeau du bas, c'est qu'il y a eu un problème de communication.

Question n°1

Réalisez le montage, téléversez le programme puis constatez son fonctionnement.

Question n°2

A quoi sert la résistance de 220 Ohm branché sur l'anode (+) de la LED (pate la plus longue) ?

Question n°3

Pourquoi appelle-t-on anode la parte reliée au 5V alors que sur une batterie, la borne + s'appelle une cathode ?

Question n°4

Modifiez le programme afin que la LED s'allume 1s, s'éteigne 1s, s'allume 3s et s'éteigne 1s et constatez le fonctionnement.

4. Contrôleur de LED RGB

4.1. Introduction

Objectifs pédagogiques

L'objectif de cet exercice est de piloter une LED RGB de façon à faire varier sa couleur.

Pour réaliser ce montage, il faudra vous munir des composants suivants :

- une carte arduino
- un câble USB
- une LED RGB
- 3 résistors de 220 Ohms
- des fils Dupont Mâle/Mâle
- une platine de prototypage rapide à trou de type Labdec

4.2. Principe du RVB ou RGB

RVB

≈ RGB

Rouge, vert, bleu, abrégé en RVB ou en RGB (de l'anglais « red, green, blue ») est un système de codage informatique des couleurs, le plus proche du matériel.

Les écrans d'ordinateurs reconstituent une couleur par synthèse additive à partir de trois couleurs primaires : rouge, vert et bleu, formant sur l'écran une mosaïque trop petite pour être aperçue.

Le codage RVB indique une valeur pour chacune de ces couleurs primaires.

Des paramètres plus intuitifs tels que la teinte, la saturation et la luminosité exigent du système informatique qu'il calcule ces valeurs.

Pour chacune des couleurs primaires, la valeur s'exprime dans un intervalle compris entre 0 et le maximum, qui est soit 1 ou 100 %, exprimé en codage informatique par 255 ou 0xFF (hexadécimal).

Les trois primaires en quantité égale codent du gris, et au maximum donnent du blanc.

Exemple Codage d'un ton saumon

Le code RVB indique rouge = 100 %, vert = 80 %, bleu = 60 %.

4.3. Montage et exécution du contrôleur de LED RGB

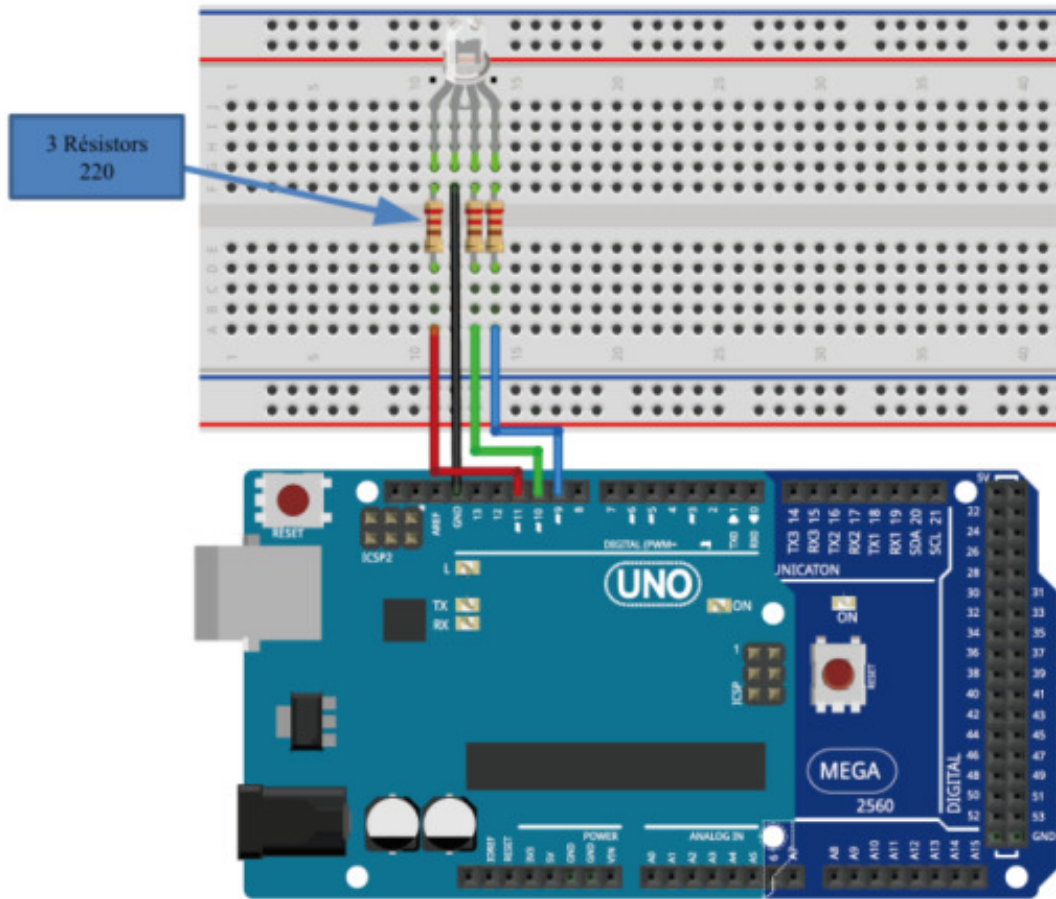
Le programme ci-dessous permet de contrôler une LED RGB :

```

1 int redPin = 11; // Rouge
2 int greenPin = 10; // Vert
3 int bluePin = 9; // Bleu
4 void setup()
5 {
6   pinMode(redPin, OUTPUT);
7   pinMode(greenPin, OUTPUT);
8   pinMode(bluePin, OUTPUT);
9 }
10 void loop()
11 {
12   // Couleurs brutes
13   color(255, 0, 0); // Active la couleur rouge de la LED RGB
14   delay(1000);
15   color(0,255, 0); // Active la couleur verte de la LED RGB
16   delay(1000);
17   color(0, 0, 255); // Active la couleur bleue de la LED RGB
18   delay(1000);
19
20   // Couleurs variées
21   color(255,255,0); // Active la couleur jaune de la LED RGB
22   delay(1000);
23   color(255,255,255); // Active la couleur blanche de la LED RGB
24   delay(1000);
25   color(128,0,255); // Active la couleur mauve de la LED RGB
26   delay(1000);
27   color(0,0,0); // Eteint la LED RGB
28   delay(1000);
29 }
30
31 void color (unsigned char red, unsigned char green, unsigned char blue)
32 {
33   analogWrite(redPin, red); // sortie MLI
34   analogWrite(greenPin, green); // sortie MLI
35   analogWrite(bluePin, blue); // sortie MLI
36 }
37

```

Le montage à réaliser est le suivant :



Correspondance des différentes pattes d'une LED RGB

Lumen

En physique, un lumen est la quantité de lumière interceptée par 1 m² de surface interne d'une sphère creuse de 1 m de rayon, au centre de laquelle on a placé une bougie. Pour définir l'éclairement d'un objet, on utilise le lux qui correspond à un flux d'un lumen tombant sur une surface de 1 m² de l'objet.

Question n°1

Réalisez le montage, téléversez le code et vérifiez son fonctionnement.

Question n°2

L'intensité lumineuse de la couleur rouge est plus faible que les autres, pour quelle raison ? Que faudrait-il changer pour augmenter sa production de lumens ?

Question n°3

Modifiez le programme de façon à produire d'autres couleurs.

5. Pilotage de LED par des interrupteurs

5.1. Introduction

Objectifs pédagogiques

L'objectif est d'apprendre à utiliser des interrupteurs pour piloter le programme, dans le cas présent pour piloter l'allumage d'une LED.

Pour réaliser ce montage, il faudra vous munir des composants suivants :

- une carte arduino
- un câble USB
- une LED RGB
- 1 résistor de 220 Ohms
- 2 interrupteurs
- des fils Dupont Mâle/Mâle
- une platine de prototypage rapide à trou de type Labdec

5.2. Montage et pilotage de LED par interrupteurs

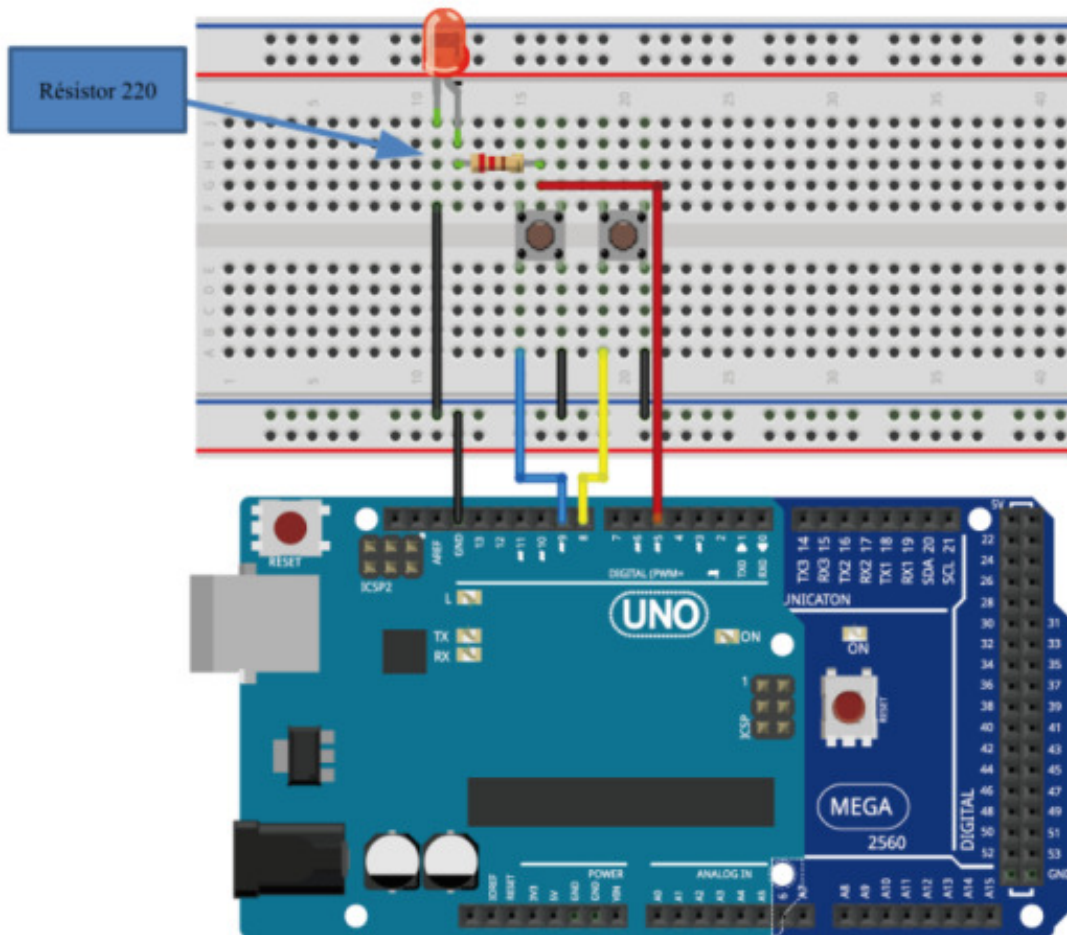
Le programme ci-dessous permet de contrôler une LED en utilisant deux interrupteurs.

```
1 int ledPin = 5;
2 int buttonApin = 9;
3 int buttonBpin = 8;
4
5 byte leds = 0;
6
7 void setup()
8 {
9   pinMode(ledPin, OUTPUT);
10  pinMode(buttonApin, INPUT_PULLUP);
11  pinMode(buttonBpin, INPUT_PULLUP);
12 }
13
14 void loop()
15 {
16   if (digitalRead(buttonApin) == LOW)
17   {
18     digitalWrite(ledPin, HIGH);
19   }
20   if (digitalRead(buttonBpin) == LOW)
```

```

21 {
22   digitalWrite(ledPin, LOW);
23 }
24 }
    
```

Le montage à réaliser est le suivant :



Question n°1

Pourquoi faut-il préciser que les entrées 8 et 9 sont en INPUT_PULLUP ?

Question n°2

Expliquez la différence entre un montage PULL-UP et un PULL-DOWN.

Question n°3

Réalisez le montage, téléversez le code puis constatez le fonctionnement.

Question n°4

Modifier le programme pour que la LED s'allume uniquement quand les deux interrupteurs sont activés puis testez.

6. Le moniteur série

6.1. Introduction

La liaison série de type RS232 peut être utilisée pour transmettre des informations depuis la carte vers l'ordinateur ou inversement. Ces informations peuvent être visualisées via le moniteur série.

Dans le cas de la carte Arduino Uno, elle est "simulée" et passe par la liaison USB.

Elle permet de transmettre une information sur 8 bits de manière asynchrone.




Sur les systèmes d'exploitation MS-DOS et Windows, les ports RS-232 sont désignés par les noms COM1, COM2... qui sont utilisés pour communiquer avec les cartes Arduino.

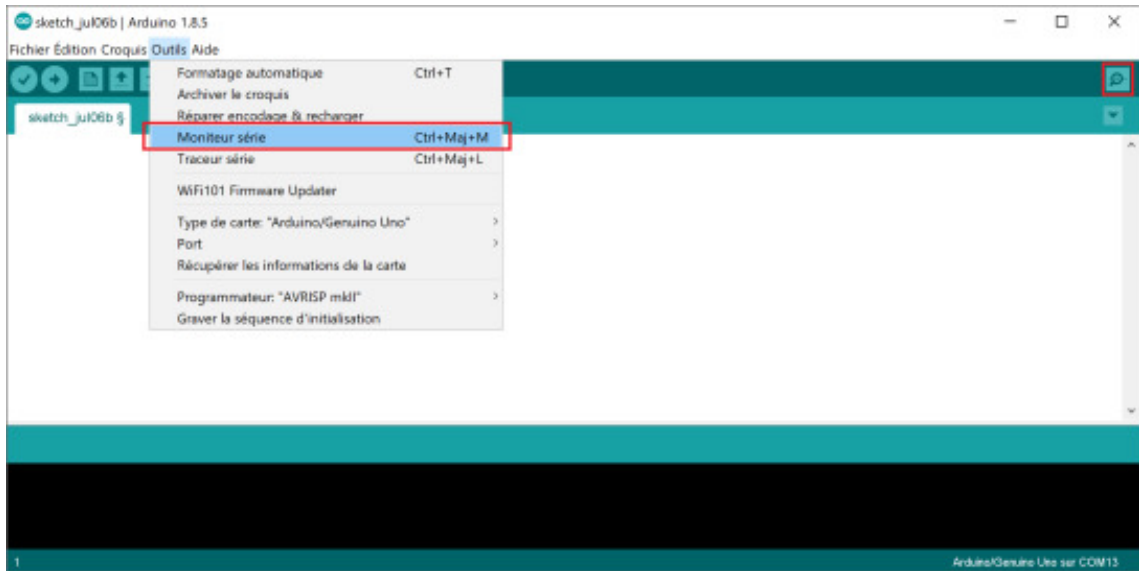
Le moniteur série a plusieurs usages intéressants, il permet par exemple d'afficher des messages de débogage depuis la carte vers l'ordinateur ou de créer une Interface Homme Machine entre l'utilisateur, le programme et la carte Arduino pour piloter son fonctionnement au moyen de variables entrées au clavier.

6.2. Afficher dans le moniteur série le caractère entré par l'utilisateur

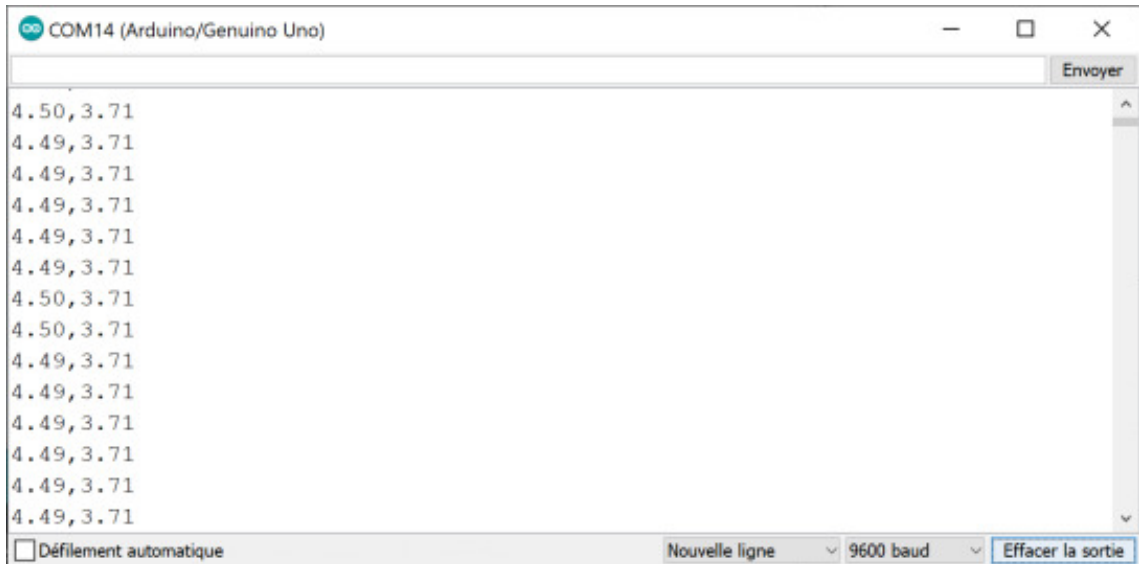
Le programme ci-dessous permet de faire communiquer le programme et l'utilisateur :

```
1 void setup() {
2   Serial.begin(9600); // Ouvre le port série et fixe le débit de données à 9600
   bauds
3 }
4
5 void loop() {
6   if (Serial.available() > 0) { // Vérifie le fonctionnement de la liaison série
7     char c = Serial.read(); // Attribue à la variable c le caractère entré au
   clavier par l'utilisateur du moniteur
8     Serial.print("Caractere recu : ");
9     Serial.println(c); // Affiche dans le moniteur le caractère entré par
   l'utilisateur
10  }
11 }
```

Le moniteur série peut être affiché dans Windows depuis le menu  Outils/Moniteur série ou en appuyant sur  ou en cliquant sur l'icone  en haut à droite.



La fenêtre du moniteur série est la suivante :



⚠ Attention

Veillez à bien régler le nombre de baud du moniteur série à celui que vous avez fixé dans le setup du programme, dans le cas contraire les informatiques affichées seront illisibles. Ce réglage est disponible en bas à droite de la fenêtre. Dans le cas présent, c'est 9600 bauds.

Dans cette fenêtre il est possible de cocher le défilement automatique pour que les valeurs affichées s'actualisent au cours du temps

Question n°1

Qu'est-ce-que le nombre de bauds ?

Question n°2

Dans le programme ci-dessus, combien de temps s'écoule entre deux changements d'état ?

Question n°3

Copiez le programme dans l'IDE Arduino et testez-le.

6.3. Afficher dans le moniteur série des variables

Le programme ci-dessous permet de créer un menu à choix multiple :

```

1 int variable=0;
2
3 void setup() {
4   Serial.begin(9600); // Ouvre le port série et fixe le débit de données à 9600
   bauds
5   menu();
6 }
7
8 void loop() {
9   if (Serial.available() > 0) { // Vérifie le fonctionnement de la liaison série
10    char c = Serial.read();
11    if (c == '1') {
12      variable++;
13      menu();
14    }
15    else if (c == '2') {
16      variable--;
17      menu();
18    }
19    else if ((c!='1') && (c!='2') && (c!='\n')){
20      Serial.println("Pas la bonne touche !"); Serial.println();
21    }
22  }
23 }
24
25 void menu(){
26   Serial.print("La variable vaut : ");
27   Serial.println(variable); Serial.println();
28   Serial.println("Menu à choix multiple");
29   Serial.println("1: +");
30   Serial.println("2: -");
31 }

```

Dans ce programme, le menu est appelé sous la forme d'une fonction.

Une fonction s'écrit s'ainsi : `variablesRenvoyées NomDeLaFonction(variablesReçues)`.

Dans le cas présent :

- `void` signifie que la fonction ne renvoie pas de valeur de variable
- `menu` est le nom de la fonction
- `()` que la fonction ne reçoit aucune valeur

Question n°1

Copiez le programme dans l'IDE Arduino et testez-le.

Question n°2

Dans le code ci-dessus, le programme vérifie que la réponse de l'utilisateur n'est pas '\n'. A quoi sert cette vérification ?

Question n°3

Rajoutez une troisième option dans le menu pour remettre à 0 la variable.

7. Multimètre à 2 voies

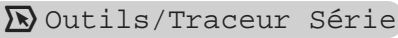

7.1. Introduction

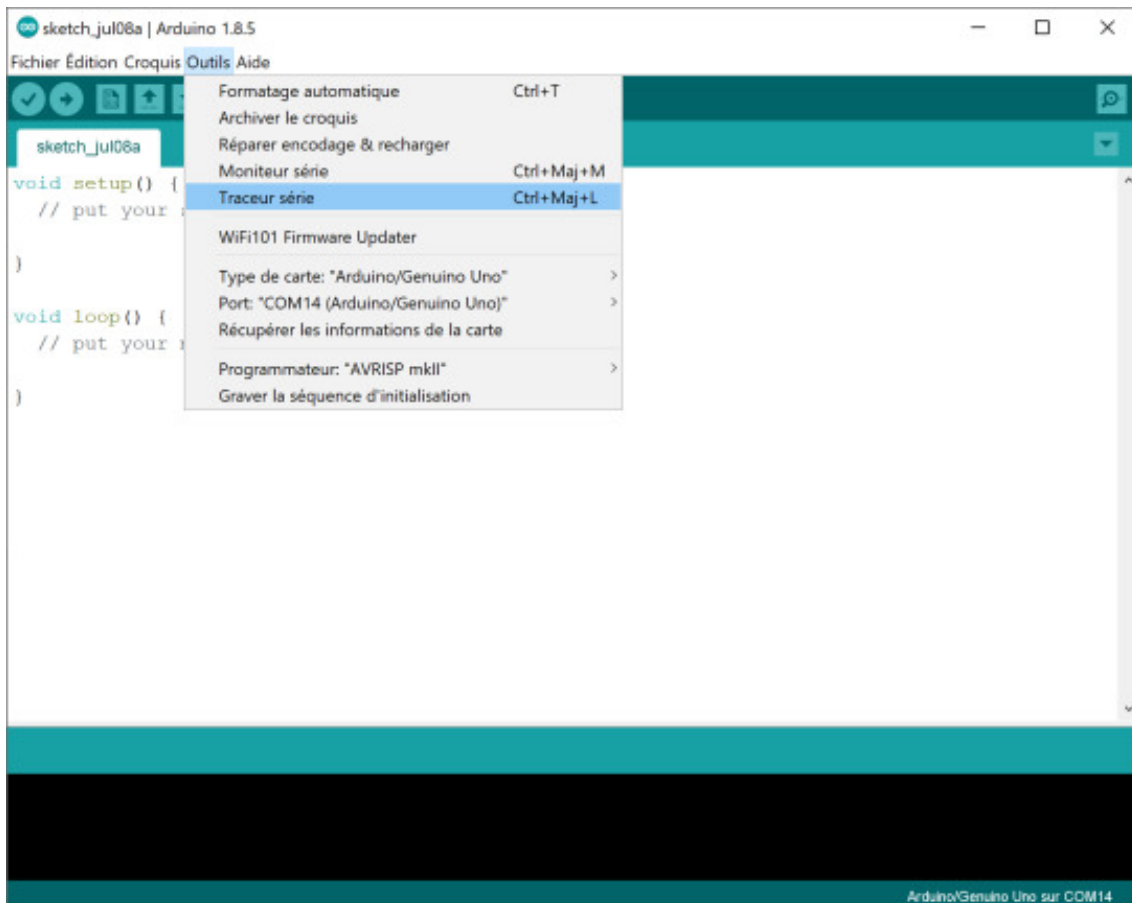
L'objectif de cette ressource est de mesurer deux variations de tension simultanées et de tracer leurs courbes en temps réel dans l'IDE.

Pour réaliser ce montage, il faudra vous munir des composants suivants :

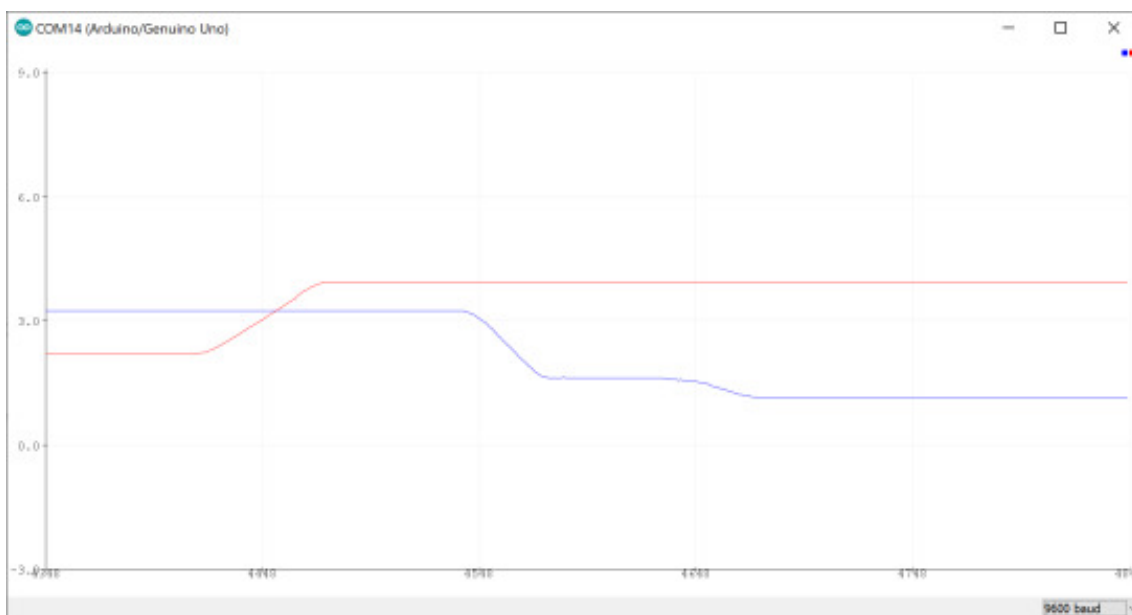
- une carte Arduino
- un câble USB
- deux potentiomètres
- des fils Dupont Mâle/Mâle
- une platine de prototypage rapide à trou de type Labdec

7.2. Montage, mesure de variations de tensions et traçage de courbes

L'IDE Arduino dispose d'une fonction pour tracer une ou plusieurs courbes simultanément, au moyen du traceur série disponible dans le menu  ou en appuyant simultanément sur .



Le traceur série se présente ainsi :



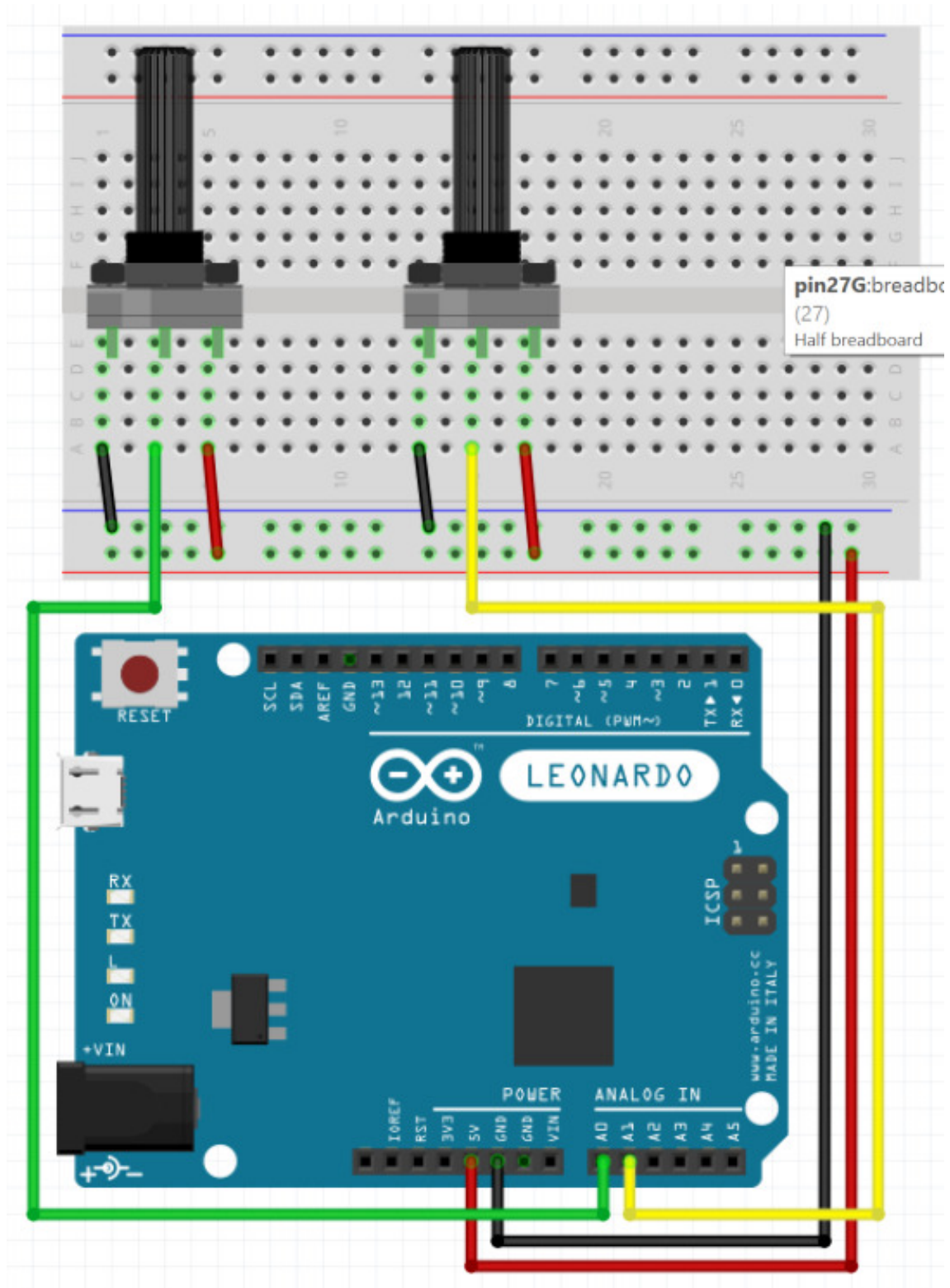
Le code ci-dessous permet de créer un multimètre à 2 voies :

```

1 // préciser les connecteurs reliés aux potentiomètres, le premier est branché sur
  le connecteur analogique 0 et le deuxième sur le 1
2 int potentiometre1 = 0;
3 int potentiometre2 = 1;
4
5 // déclaration des variables utilisées pour stocker la valeur des tensions mesurées
  analogiquement sur les entrées de la carte Arduino
6 int valeur1;
    
```

```
7 int valeur2;
8
9 // déclaration des variables qui vont accueillir les valeurs de tension sous forme
  de nombres réels
10 float tension1;
11 float tension2;
12
13 void setup() {
14   Serial.begin(9600);
15   while(!Serial);
16 }
17
18 void loop() {
19   valeur1=analogRead(potentiometre1);
20   valeur2=analogRead(potentiometre2);
21
22   // transformation des valeurs entières (5v=1023) en valeurs réelles en volts
23   tension1 = valeur1 * 5.0 / 1023;
24   tension2 = valeur2 * 5.0 / 1023;
25
26   // Affichage de la valeur mesure sur le traceur série
27   Serial.print(tension1);
28   Serial.print(",");
29   Serial.print(tension2);
30   Serial.println();
31 }
```

Le montage associé est le suivant :



Question n°1

Quelle est l'instruction pour déclarer qu'un connecteur est utilisé comme entrée ?

Question n°2

Réalisez le montage puis téléversez le code et observez dans le traceur série la variation de tension en tournant les potentiomètres.

Question n°3

Comment fonctionne un potentiomètre ?

8. Buzzer actif

8.1. Introduction

Objectifs pédagogiques

L'objectif de cet exercice est de piloter un buzzer en utilisant un transistor.

Pour réaliser ce montage, il faudra vous munir des composants suivants :

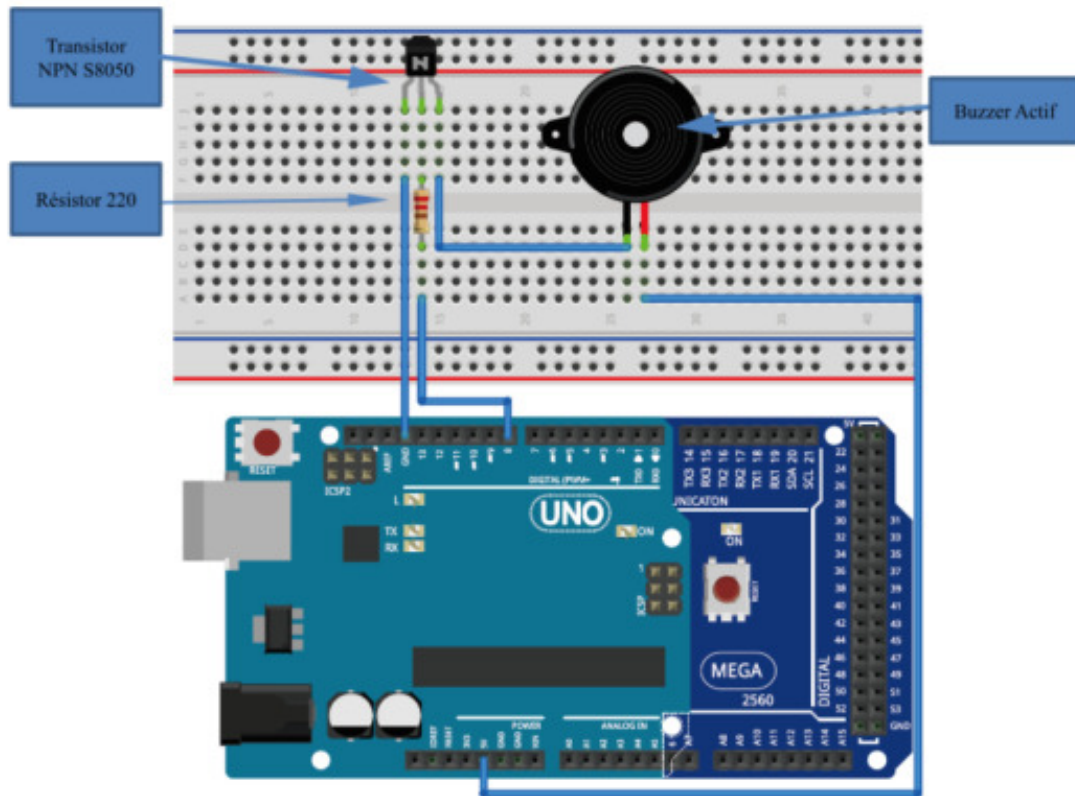
- une carte Arduino
- un câble USB
- un transistor NPN S8050
- un buzzer actif HYDZ
- un résistor 220 Ohms
- des fils Dupont Mâle/Mâle
- une platine de prototypage rapide à trou de type Labdec

8.2. Montage et pilotage d'un buzzer actif

Le programme ci-dessous permet de contrôler un buzzer actif :

```
1 int buzzerPin=8;
2 void setup()
3 {
4   pinMode(buzzerPin,OUTPUT);
5 }
6 void loop()
7 {
8   digitalWrite(buzzerPin,HIGH); //Force le connecteur à la valeur HIGH = 5 v
9   delay(2000);
10  digitalWrite(buzzerPin,LOW); //Force le connecteur à la valeur LOW = 0 v
11  delay(2000);
12 }
```

Le montage à réaliser est le suivant :



Question n°1

Quelle est la différence entre un buzzer ACTIF et un buzzer PASSIF ?

Question n°2

A quoi sert un transistor et comment fonctionne-t-il ?

Question n°3

Quelles sont les différences entre les transistors NPN et PNP ?

Question n°4

Réalisez le montage, téléversez le programme puis constatez son fonctionnement.

Question n°5

Réalisez un montage et un programme permettant d'allumer le buzzer en appuyant sur un bouton, puis vérifiez son fonctionnement

9. Buzzer passif

9.1. Introduction

Objectifs pédagogiques

L'objectif de cet exercice est d'apprendre à utiliser un buzzer passif pour réaliser une mélodie.

Pour réaliser ce montage, il faudra vous munir des composants suivants :

- une carte Arduino
- un câble USB
- un transistor NPN S8050
- un buzzer passif HX
- une LED
- deux résistors 220 Ohms
- des fils Dupont Mâle/Mâle
- une platine de prototypage rapide à trou de type Labdec

9.2. Montage et pilotage d'un buzzer passif

Le programme ci-dessous permet de contrôler un buzzer passif pour lui faire jouer une mélodie :

```

1 #define NTD0 -1 // Basse 1# 2# 3# 4# 5# 6# 7#
2 #define NTD1 294 // A 221 248 278 294 330 371 416
3 #define NTD2 330 // B 248 278 294 330 371 416 467
4 #define NTD3 350 // C 131 147 165 175 196 221 248
5 #define NTD4 393 // D 147 165 175 196 221 248 278
6 #define NTD5 441 // E 165 175 196 221 248 278 312
7 #define NTD6 495 // F 175 196 221 234 262 294 330
8 #define NTD7 556 // G 196 221 234 262 294 330 371
9
10 #define NTDL1 147 // Alto 1 2 3 4 5 6 7
11 #define NTDL2 165 // A 441 495 556 589 661 742 833
12 #define NTDL3 175 // B 495 556 624 661 742 833 935
13 #define NTDL4 196 // C 262 294 330 350 393 441 495
14 #define NTDL5 221 // D 294 330 350 393 441 495 556
15 #define NTDL6 248 // E 330 350 393 441 495 556 624
16 #define NTDL7 278 // F 350 393 441 495 556 624 661
17 // G 393 441 495 556 624 661 742
18 #define NTDH1 589
19 #define NTDH2 661 // Aigu 1# 2# 3# 4# 5# 6# 7#

```

```

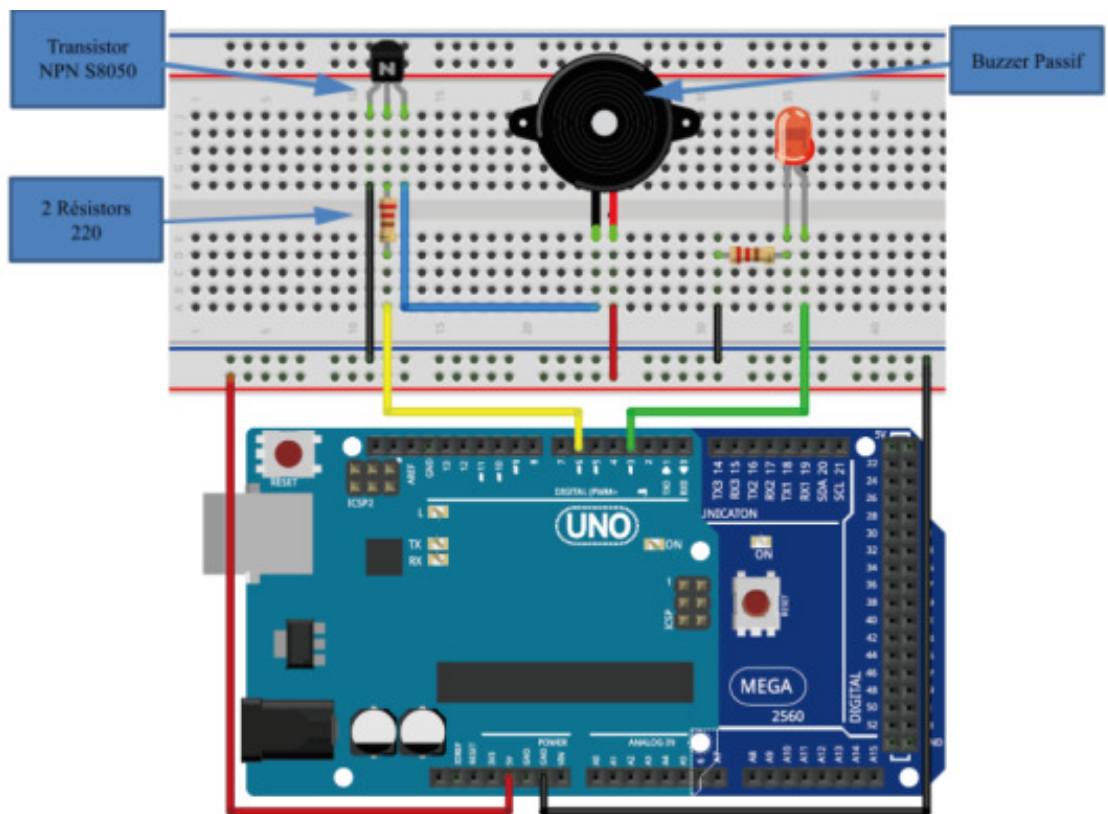
20 #define NTDH3 700 // A 882 990 1112 1178 1322 1484
    1665
21 #define NTDH4 786 // B 990 1112 1178 1322 1484 1665 1869
22 #define NTDH5 882 // C 525 589 661 700 786 882 990
23 #define NTDH6 990 // D 589 661 700 786 882 990 1112
24 #define NTDH7 112 // E 661 700 786 882 990 1112 1248
25 // F 700 786 882 935 1049 1178 1322
26 #define WHOLE 1 // G 786 882 990 1049 1178 1322 1484
27 #define HALF 0.5
28 #define QUARTER 0.25
29 #define EIGHTH 0.25
30 #define SIXTEENTH 0.625
31 int tune[]=
32 {
33   NTD3,NTD3,NTD4,NTD5,
34   NTD5,NTD4,NTD3,NTD2,
35   NTD1,NTD1,NTD2,NTD3,
36   NTD3,NTD2,NTD2,
37   NTD3,NTD3,NTD4,NTD5,
38   NTD5,NTD4,NTD3,NTD2,
39   NTD1,NTD1,NTD2,NTD3,
40   NTD2,NTD1,NTD1,
41   NTD2,NTD2,NTD3,NTD1,
42   NTD2,NTD3,NTD4,NTD3,NTD1,
43   NTD2,NTD3,NTD4,NTD3,NTD2,
44   NTD1,NTD2,NTDL5,NTD0,
45   NTD3,NTD3,NTD4,NTD5,
46   NTD5,NTD4,NTD3,NTD4,NTD2,
47   NTD1,NTD1,NTD2,NTD3,
48   NTD2,NTD1,NTD1
49 };
50 float durt[]= // Durée de chaque note
51 {
52   1,1,1,1,
53   1,1,1,1,
54   1,1,1,1,
55   1+0.5,0.5,1+1,
56   1,1,1,1,
57   1,1,1,1,
58   1,1,1,1,
59   1+0.5,0.5,1+1,
60   1,1,1,1,
61   1,0.5,0.5,1,1,
62   1,0.5,0.5,1,1,
63   1,1,1,1,
64   1,1,1,1,
65   1,1,1,0.5,0.5,
66   1,1,1,1,
67   1+0.5,0.5,1+1,
68 };
69 int length;
70 int tonepin=6; // Connecteur du buzzer
71 int ledp=3;
72 void setup()
73 {
74   pinMode(tonepin,OUTPUT);
75   pinMode(ledp,OUTPUT);
76   length=sizeof(tune)/sizeof(tune[0]); // Calcule le nombre total de notes
77 }
78 void loop()

```

```

79 {
80   for(int x=0;x<length;x++)
81   {
82     tone(tonopin,tune[x]); // Active le buzzer
83     digitalWrite(ledp, HIGH);
84     delay(400*durt[x]);
85     digitalWrite(ledp, LOW);
86     delay(100*durt[x]);
87     noTone(tonopin); // Eteint le buzzer
88   }
89   delay(2000);
90 }
    
```

Le montage à réaliser est le suivant :



Question n°1

Réalisez le montage, téléversez le code et constatez le résultat.

Question n°2

Accélérez la vitesse à laquelle est jouée la mélodie et modifiez quelques notes et constatez le fonctionnement.

10. Tilt

10.1. Introduction

Objectifs pédagogiques

Réaliser un montage utilisant un capteur de type Tilt pour allumer une LED si une inclinaison donnée est atteinte.

Pour réaliser ce montage, il faudra vous munir des composants suivants :

- une carte Arduino
- un câble USB
- un Tilt
- une LED
- des fils Dupont Mâle/Mâle
- une platine de prototypage rapide à trou de type Labdec

10.2. Montage et allumage d'une LED par tilt

Tilt

Un interrupteur Tilt permet de détecter l'orientation ou l'inclinaison d'un système. Il est souvent utilisé pour indiquer si un système (comme un véhicule agricole) dépasse sa plage d'inclinaison de fonctionnement ou pour détecter l'orientation d'un écran et, ainsi, modifier sa mise en page. Il ne donne pas autant d'information qu'un accéléromètre mais est plus robuste et ne nécessite pas de programme particulier pour être traité.

Le programme ci-dessous permet d'activer une LED par le biais d'un capteur d'accélération et/ou d'inclinaison appelé tilt :

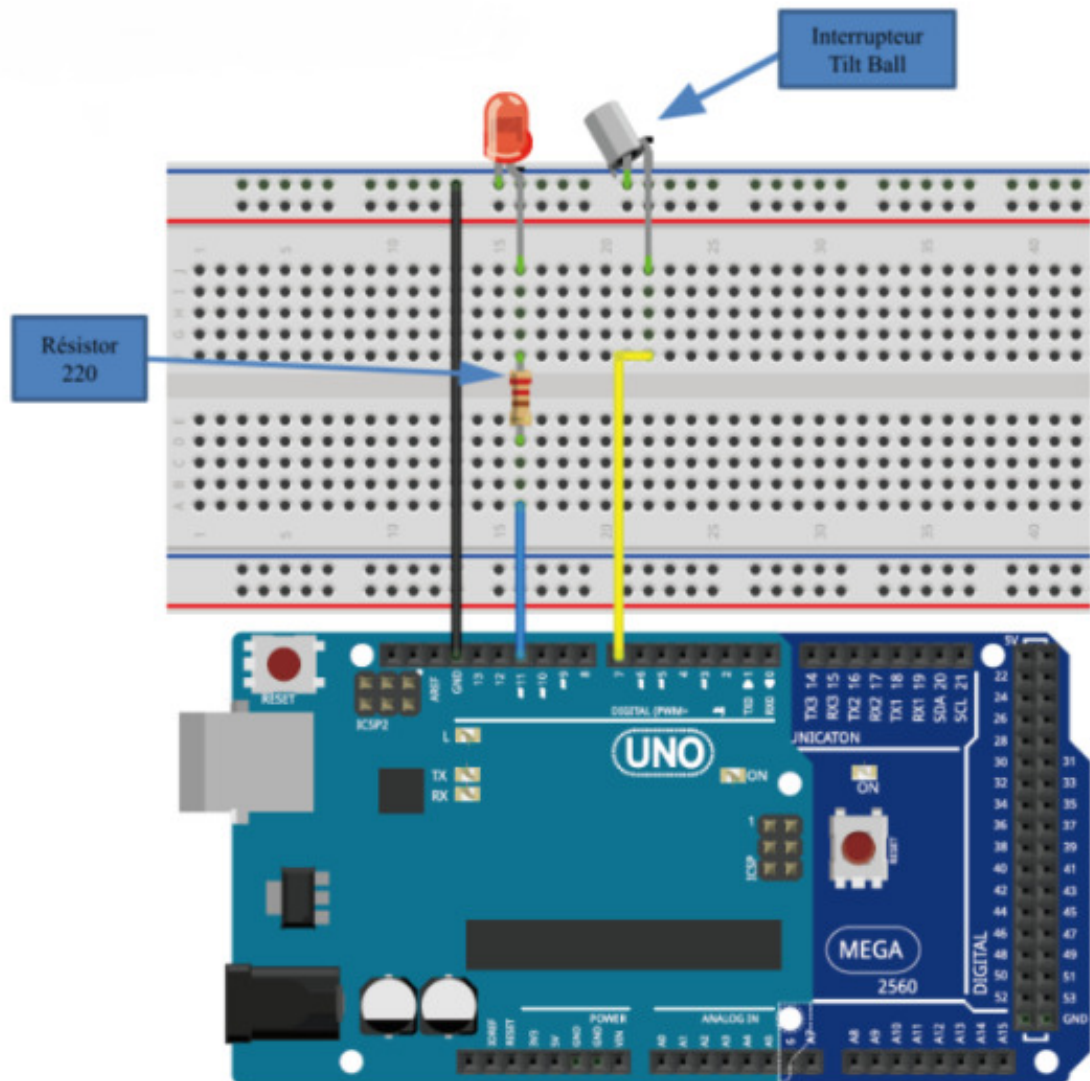
```
1 int ledpin=11;
2 int tiltSwitchpin=7;
3 int val; // Déclaration d'une variable de type entière
4
5 void setup()
6 {
7   pinMode(ledpin, OUTPUT);
8   pinMode(tiltSwitchpin, INPUT_PULLUP);
9 }
10
11 void loop()
```

```

12 {
13   val=digitalRead(tiltSwitchpin);
14   if(val==LOW) // Détecte si le tilt est inactif
15   { digitalWrite(ledpin,LOW); }
16   else // Si le tilt est déclenché, la LED est activée
17   { digitalWrite(ledpin,HIGH); }
18 }

```

Le montage à réaliser est le suivant :



Question n°1

Comment est constitué l'intérieur du Tilt utilisé ?

Question n°2

Réalisez le montage, téléversez le code et vérifiez son fonctionnement.

Glossaire

Effet joule	<p>L'effet joule est la transformation de l'énergie électrique reçue par la résistance en énergie thermique. La résistance parcourue par un courant se met alors à chauffer. Ce phénomène peut être gênant et non souhaité dans certains cas mais il peut être mis à profit pour produire de la chaleur. On trouve ainsi des résistances dans les fours électriques, les radiateurs électriques, les sèche cheveux, les fers à repasser, etc.</p>
Lumen	<p>En physique, un lumen est la quantité de lumière interceptée par 1 m² de surface interne d'une sphère creuse de 1 m de rayon, au centre de laquelle on a placé une bougie. Pour définir l'éclairement d'un objet, on utilise le lux qui correspond à un flux d'un lumen tombant sur une surface de 1 m² de l'objet.</p>
Platine Labdec <i>≈ Breadboard</i>	<p>La « breadboard » ou « platine Labdec » simplifie le montage de composants. Cette platine est très utilisée pour le prototypage électronique afin d'éviter d'avoir recours à des cartes électroniques et des soudures.</p> <p>Tous les connecteurs positifs et négatifs latéraux (entourés des lignes rouges et bleues) sont reliés entre eux. Tous les connecteurs des lignes A à E et des lignes F à J sont reliés entre eux (mais il n'y aucune liaison entre A-E et F-J).</p> <p>Le schéma ci-dessous illustre les liaisons internes de la platine.</p>
Résistance	<p>Une résistance est un composant électronique ou électrique dont la principale caractéristique est d'opposer une plus ou moins grande résistance (<i>mesurée en ohms</i>) à la circulation du courant électrique. La résistance électronique est l'un des composants primordiales dans le domaine de l'électricité.</p>
RVB <i>≈ RGB</i>	<p>Rouge, vert, bleu, abrégé en RVB ou en RGB (de l'anglais « red, green, blue ») est un système de codage informatique des couleurs, le plus proche du matériel.</p> <p>Les écrans d'ordinateurs reconstituent une couleur par synthèse additive à partir de trois couleurs primaires : rouge, vert et bleu, formant sur l'écran une mosaïque trop petite pour être aperçue.</p> <p>Le codage RVB indique une valeur pour chacune de ces couleurs primaires.</p>
Tilt	<p>Un interrupteur Tilt permet de détecter l'orientation ou l'inclinaison d'un système. Il est souvent utilisé pour indiquer si un système (comme un véhicule agricole) dépasse sa plage d'inclinaison de fonctionnement ou pour détecter l'orientation d'un écran et, ainsi,</p>

modifier sa mise en page. Il ne donne pas autant d'information qu'un accéléromètre mais est plus robuste et ne nécessite pas de programme particulier pour être traité.