YoupiLab

DEMYSTIFYING ELECTRONICS

GUIDE D'UTILISATION DU KIT YOUPILAB

OCTOBRE 2024

V.1

Préface

Notre compagnie

Fondée en 2015, GENERAL INVASION SARL, également connue sous le nom de YoupiLab, est une entreprise technologique béninoise spécialisée dans l'électronique, l'informatique et la vente de composants électroniques et équipement de laboratoires.

Nos bureaux sont situés à Godomey Togoudo, au Bénin, un pays en plein développement technologique.

Nous proposons une vaste gamme de produits, incluant des kits électroniques, des imprimantes 3D, des machines CNC et divers capteurs, tous conçus pour répondre aux besoins d'apprentissage à tous les niveaux.

Pour en savoir plus, rendez-vous sur notre boutique en ligne : youpilab.com/components

Découvrez également nos ressources pédagogiques sur : education.youpilab.com

Et notre plateforme dédiée à l'internet des objets sur : https://iot.youpilab.com.

Tutoriel

Ce tutoriel est spécialement conçu pour les débutants et a pour objectif de guider tous les utilisateurs de notre kit. Il vise à faciliter la familiarisation avec les différents composants, quel que soit le niveau de compétence de chacun. Vous y trouverez des informations essentielles et pratiques pour utiliser une carte UNO, ainsi que des conseils sur l'utilisation de tous les autres éléments inclus dans le kit. Grâce à ce guide, nous espérons que chaque utilisateur pourra naviguer avec aisance dans le monde de l'électronique et de l'informatique, en tirant le meilleur parti des ressources à sa disposition.

Service consommateur

Nous faisons la mise en place de nos kits avec le plus grand soin, en veillant constamment à maintenir des standards de qualité élevés.
Nous restons à votre disposition pour toute question ou demande. N'hésitez pas à nous écrire à l'adresse suivante : sales@youpilab.com
Nous sommes impatients de recevoir vos retours et suggestions.

YOUPILAB STARTER KIT

YoupiLab
DEMYSTIFYING ELECTRONICS


Arduino Uno R3 x 1


Cable usb x 1


5V Relay x 1


10K Potentiometer x 1


16x2 LCD x 1


LM7805 x 1


IR Flame Sensor x 1

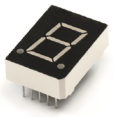

5VDC Motor x1


BreadBoard x 1


HC SR04 Ultrasonic Sensor x 1


Grenn Led x 5


Diode 1N4007 x 2


7 Segment Led x 1


Male header pinx 1


10k Résistor x 10

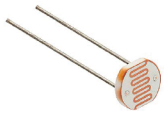


Photo Résistor x 4


Temperature humidity Sensor x 1


Piezo Buzzer x 1


IC-74hc595d x 1

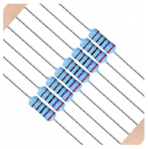

Tilt Sensor x 1


Push Button x 5


9V Battery Conector x 1


1K Résistor x 10


4x4 Matrix Keypad x 1


220V Ohm Résistor x 10


Red Led x 5


Red Led x 5


Bleue Led x 5


Yellow Led x 5


RC522-rfid kit x 1

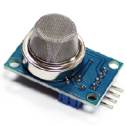

Servo Motor x 1


100K Thermistor x 1


Green Led x 5


Yellow Led x 5


Water Sensor x 1


Smoke Sensor x 1


Infrared Sensor x 1


Remote Control x 1


Pir Motion Sensor x 1


Jumper Wires x 40

Version : V6

SOMMAIRE

Leçon 0 Installation de l'environnement de programmation 6
Leçon 1 Ajouter une bibliothèque / utiliser le moniteur série 16
Leçon 2 Blink 26
Leçon 3 LED 36
Leçon 4 Digital Inputs 42
Leçon 5 Active buzzer 47
Leçon 6 Tilt Ball Switch 50
Leçon 7 Eight LED with 74HC595..... 54
Leçon 8 Photocell..... 59
Leçon 9 Servomoteur 62
Leçon 10 Capteur à ultrasons 66
Leçon 11 Capteur de température et d'humidité DHT11 69
Leçon 12 Clavier Matriciel 4x4 74
Leçon 13 Capteur de mouvement PIR 78
Leçon 14 Ecran LCD 16x2 80
Leçon 15 Capteur de niveau d'eau 84
Leçon 16 Module RFID 88
Leçon 17 Télecommande et Récepteur IR 91
Leçon 18 Module de Relais 1 canal 95
Leçon 19 Capteur de flamme 98
Leçon 20 Afficheur 7 segments 103
Leçon 21 Thermistance 107

Leçon 0

Installation de l'environnement de programmation

Introduction

L'IDE Arduino (Arduino Integrated Development Environment (ou IDE)) est un logiciel dédié de la plateforme Arduino.
Dans cette leçon, vous allez procéder à l'installation de celui-ci. Le logiciel est disponible pour Windows, MAC, LINUX.

Étape 1 : Rendez-vous sur https://www.arduino.cc/en/Main/Software

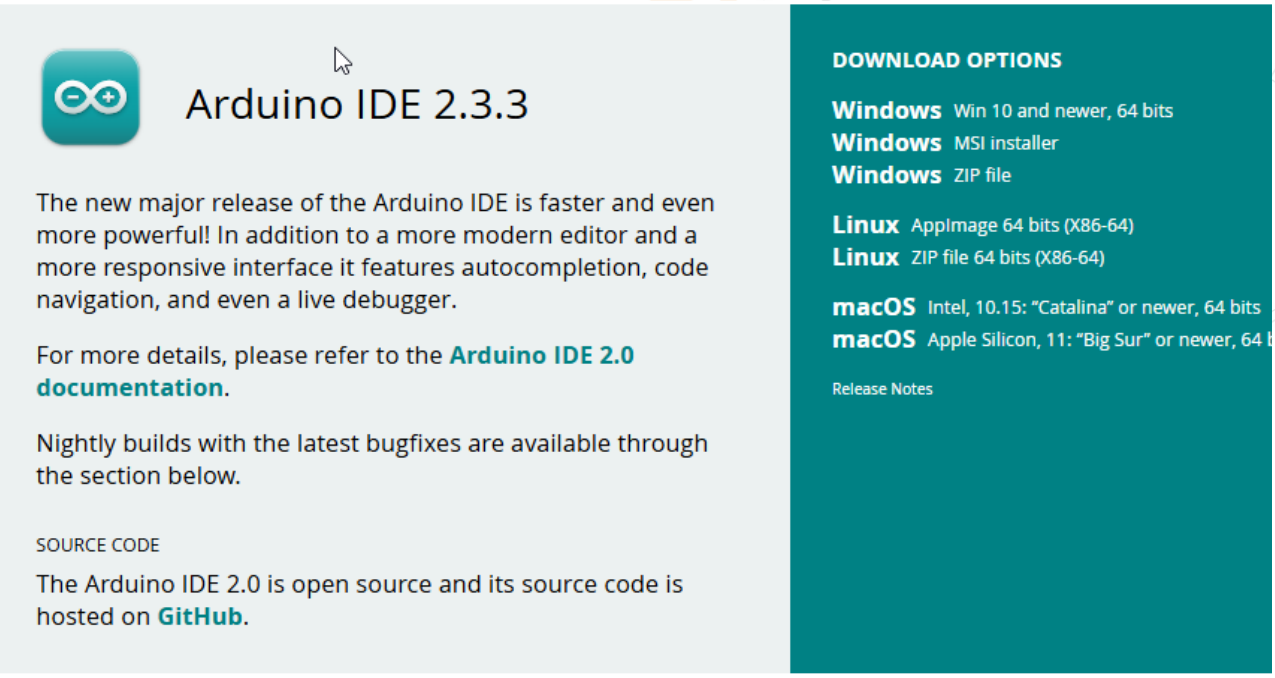


fig 0.1 :installation ide_1

Téléchargez la dernière version disponible (qui n'est plus forcément celle présente sur la capture d'écran).

Étape 2 : Téléchargez la version correspondant à votre ordinateur

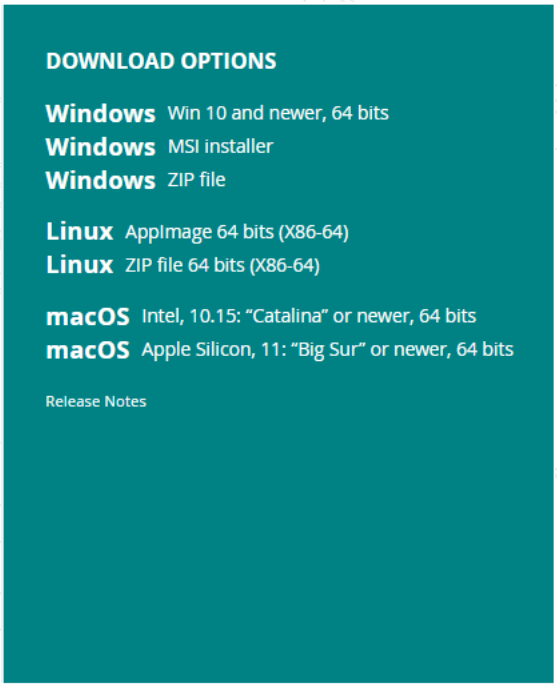


fig 0.2 :installation ide_2

fig 0.3 :installation ide_3

Si vous souhaitez faire un don pour soutenir la communauté, cliquez sur "contribute…" Si vous souhaitez téléchargez simplement, cliquez sur "just download".

Sélectionnez le fichier correspondant à votre environnement de travail.

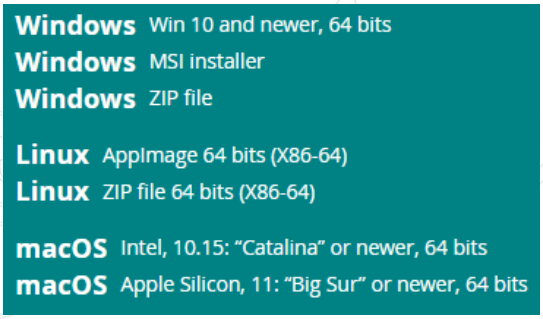


fig 0.4 :installation ide_4

Installation sous Windows

Exécutez le fichier.

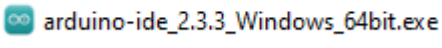


fig 0.5 :installation ide_5

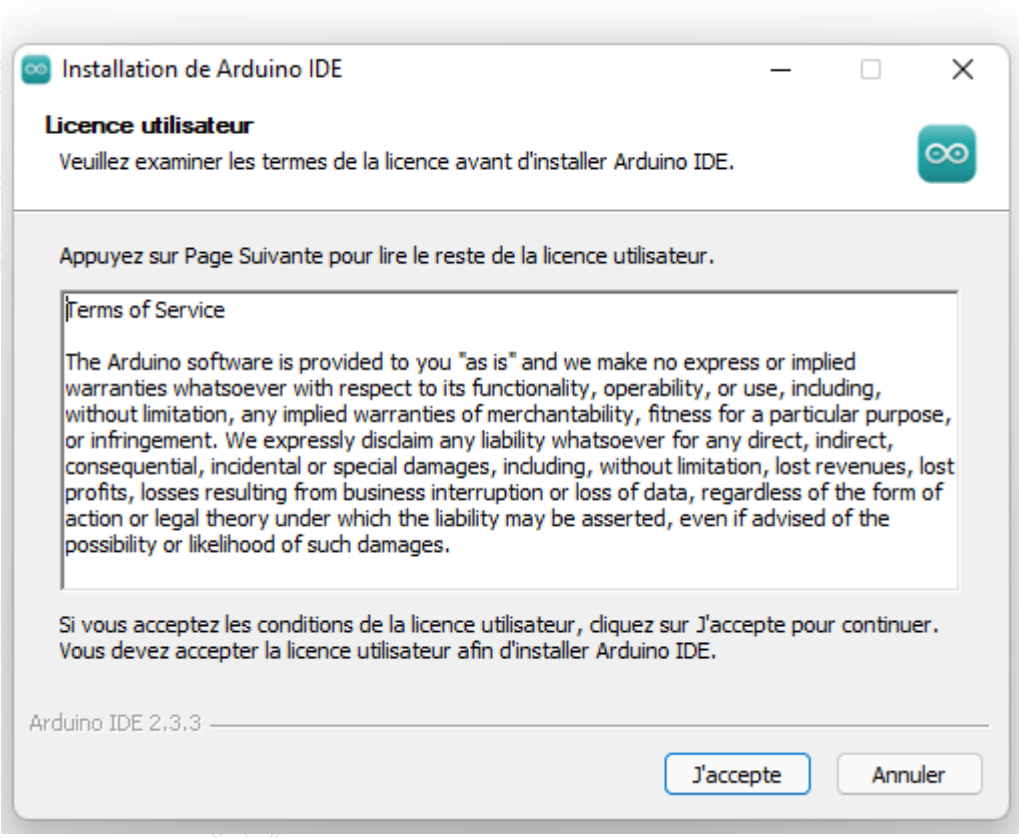


fig 0.15 :installation ide_15

Cliquez sur *J'accepte*

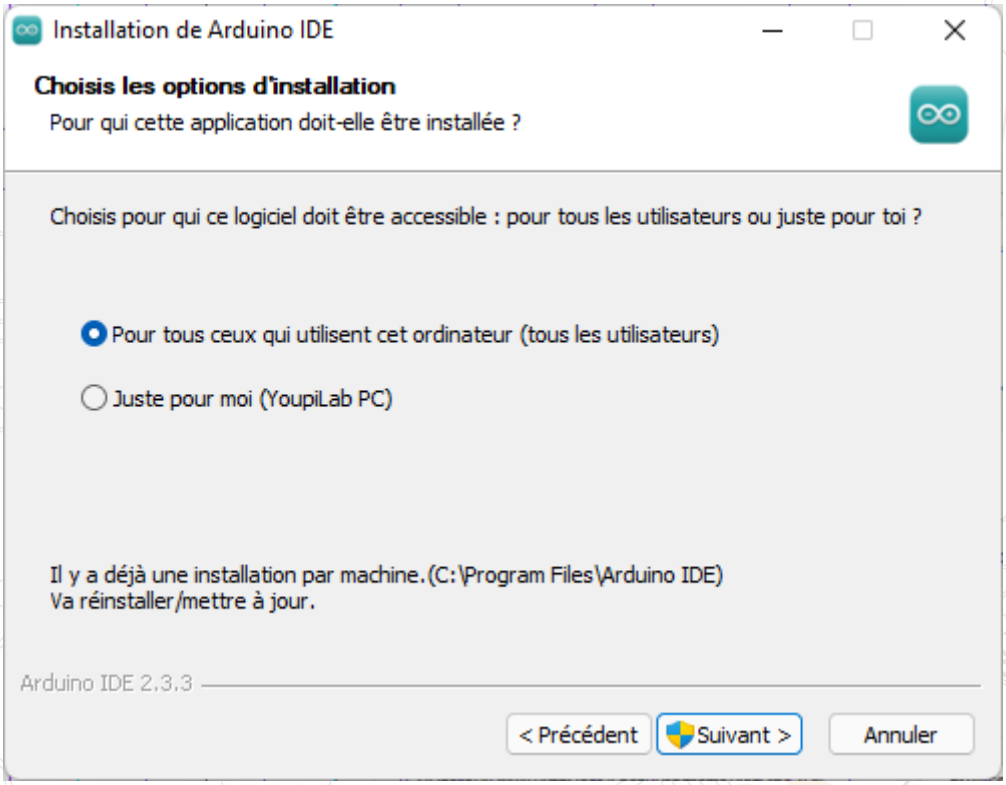


fig 0.7 :installation ide_7

Cliquez *Suivant*

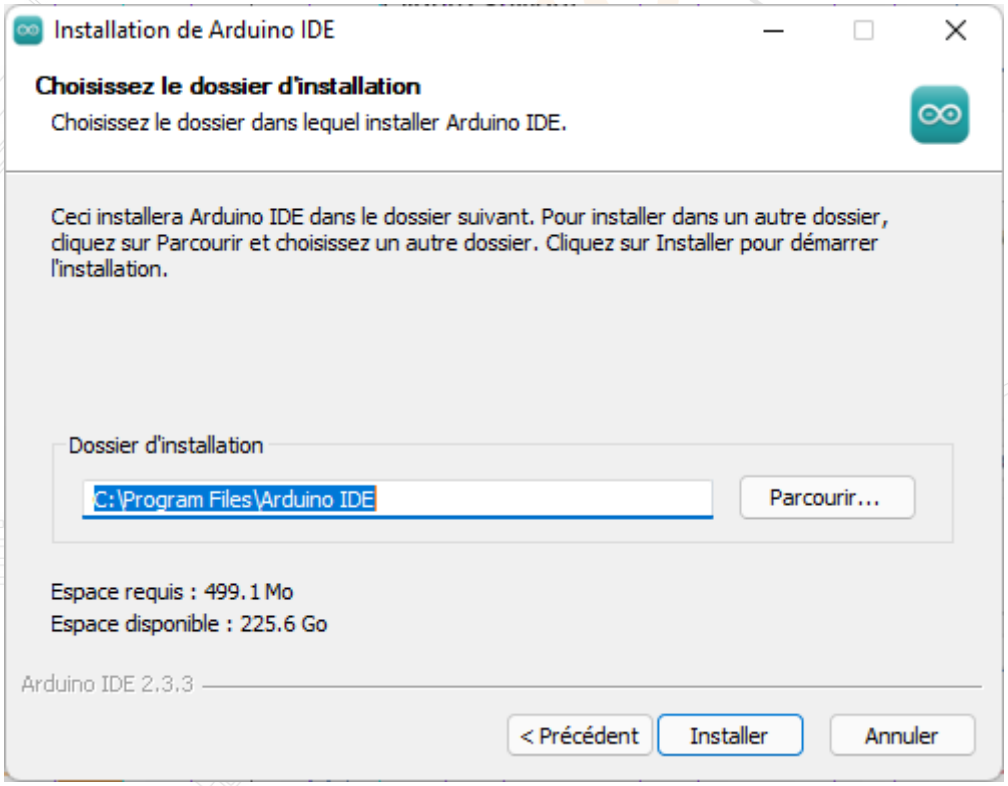


fig 0.8 :installation ide_8

Cliquez sur *Parcourir* si vous souhaitez définir un autre répertoire.

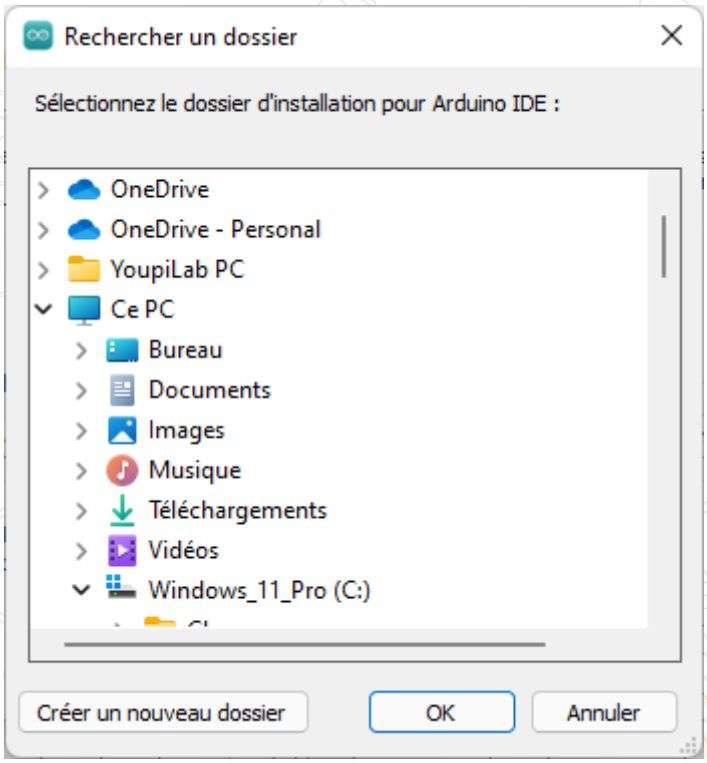


fig 0.9 : installation ide_9

Cliquez sur *Installer*

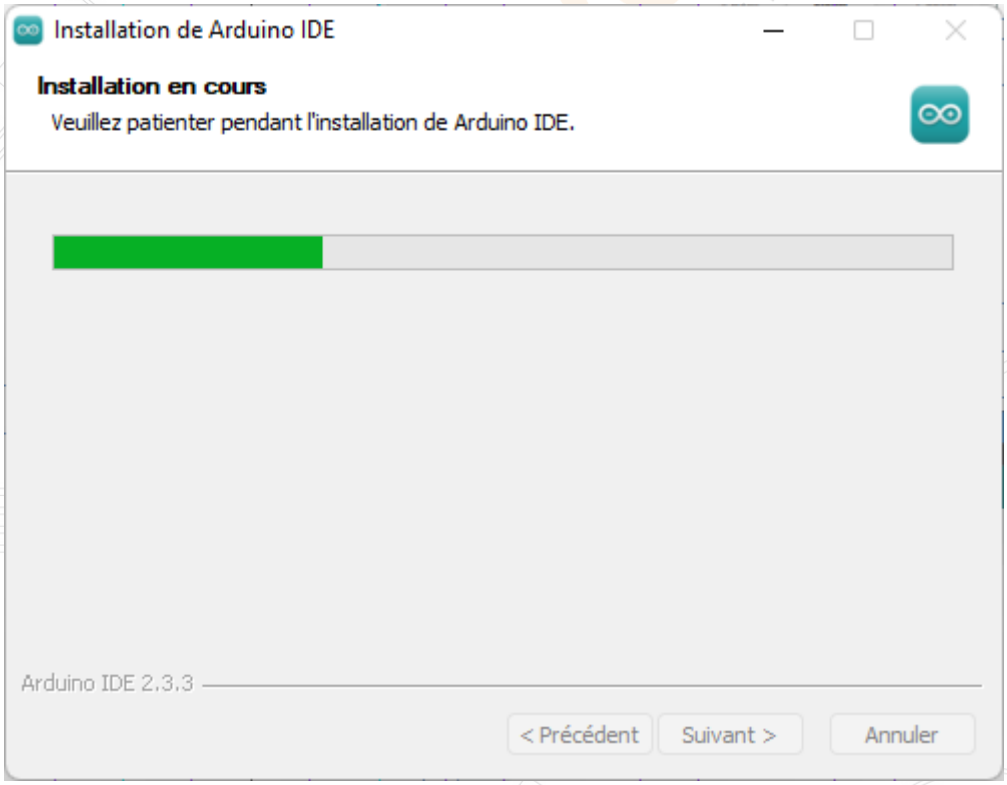


fig 0.10 : installation ide_10

Enfin, cliquez de nouveau sur Install.

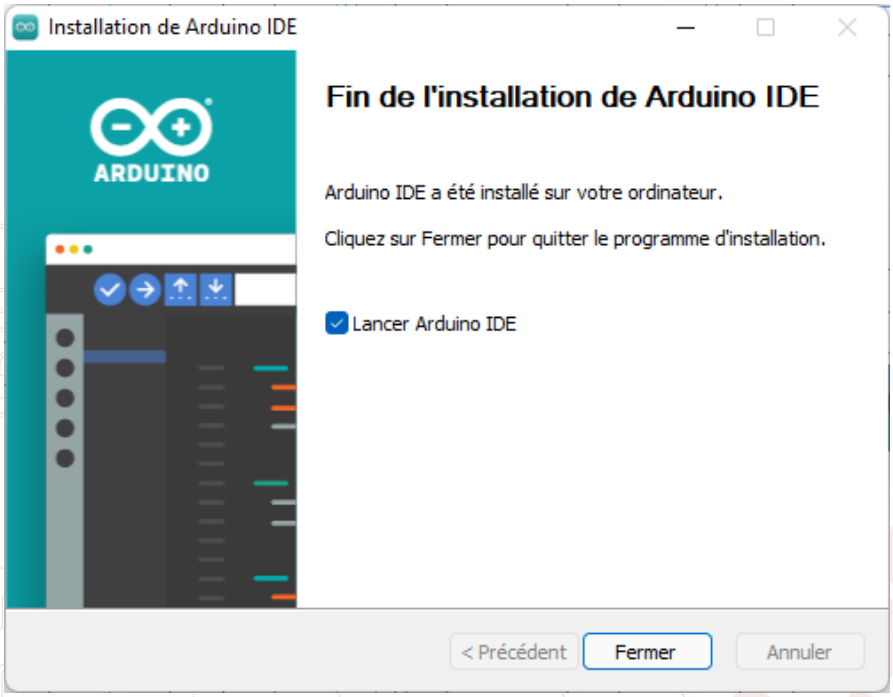


fig 0.11 : installation ide_11

L'icône suivant apparait sur votre bureau

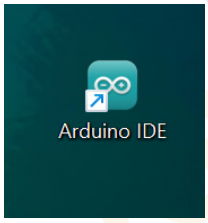


fig 0.12 : installation ide_12

Double cliquez pour lancer l'IDE

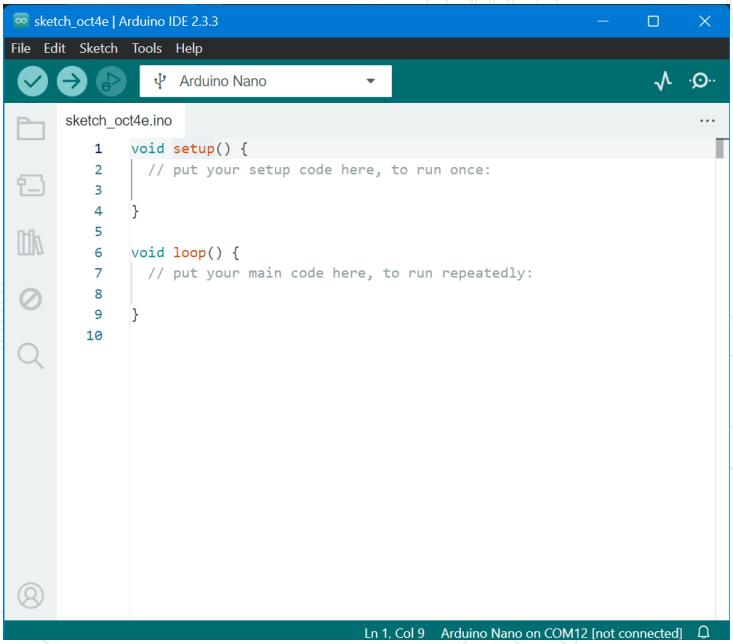


fig 0.13 : installation ide_13

Ceci est la fenêtre de départ. Elle contient l'ossature de code commune et obligatoire.

Pour installer le logiciel depuis un fichier zip, veuillez suivre les instructions suivantes.

Windows ZIP file

fig 0.14 : installation ide_14

Téléchargez le fichier zip. Faites l'extraction des fichiers dans un dossier. Lancer le fichier *arduino.exe*

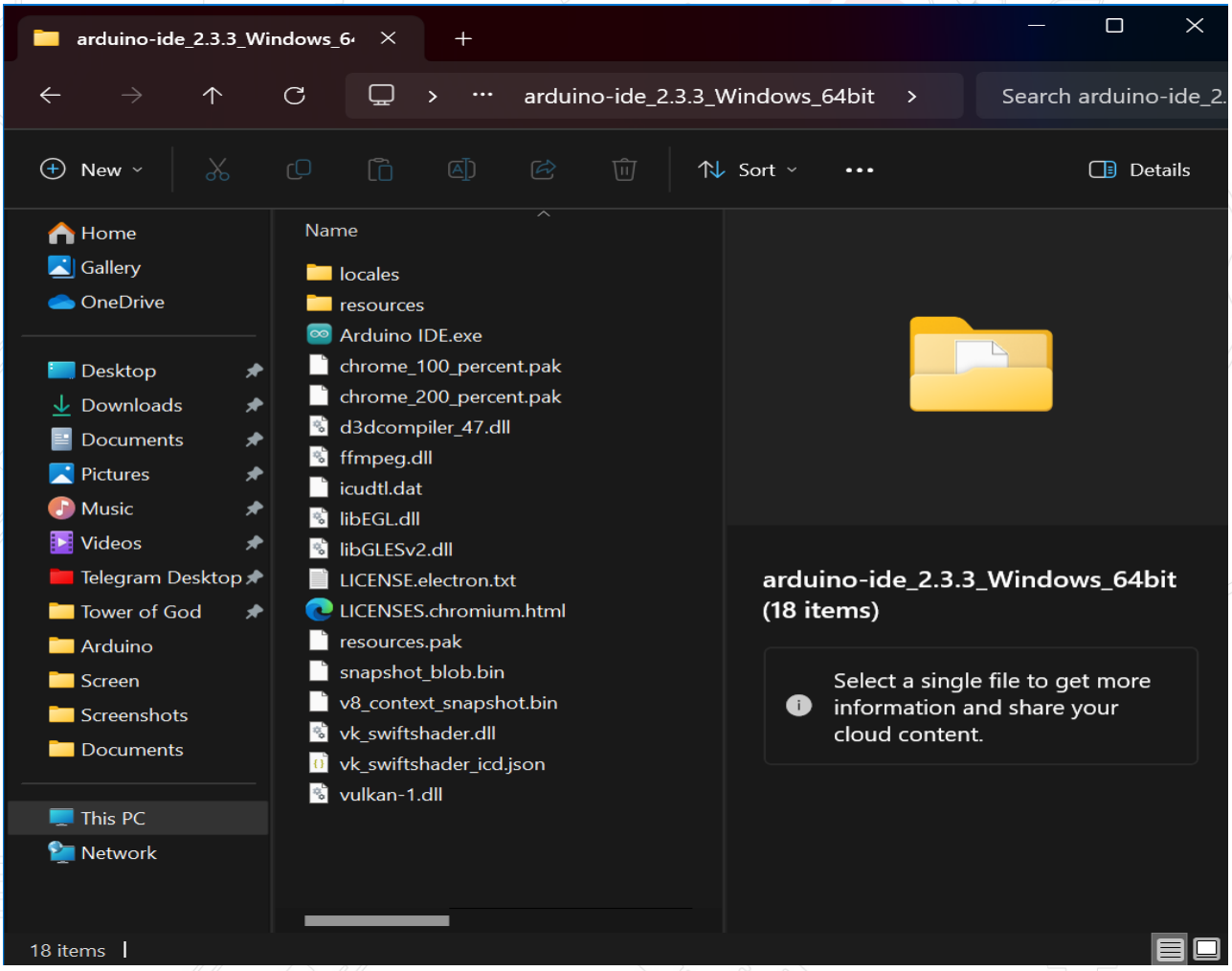


fig 0.15 : installation ide_15

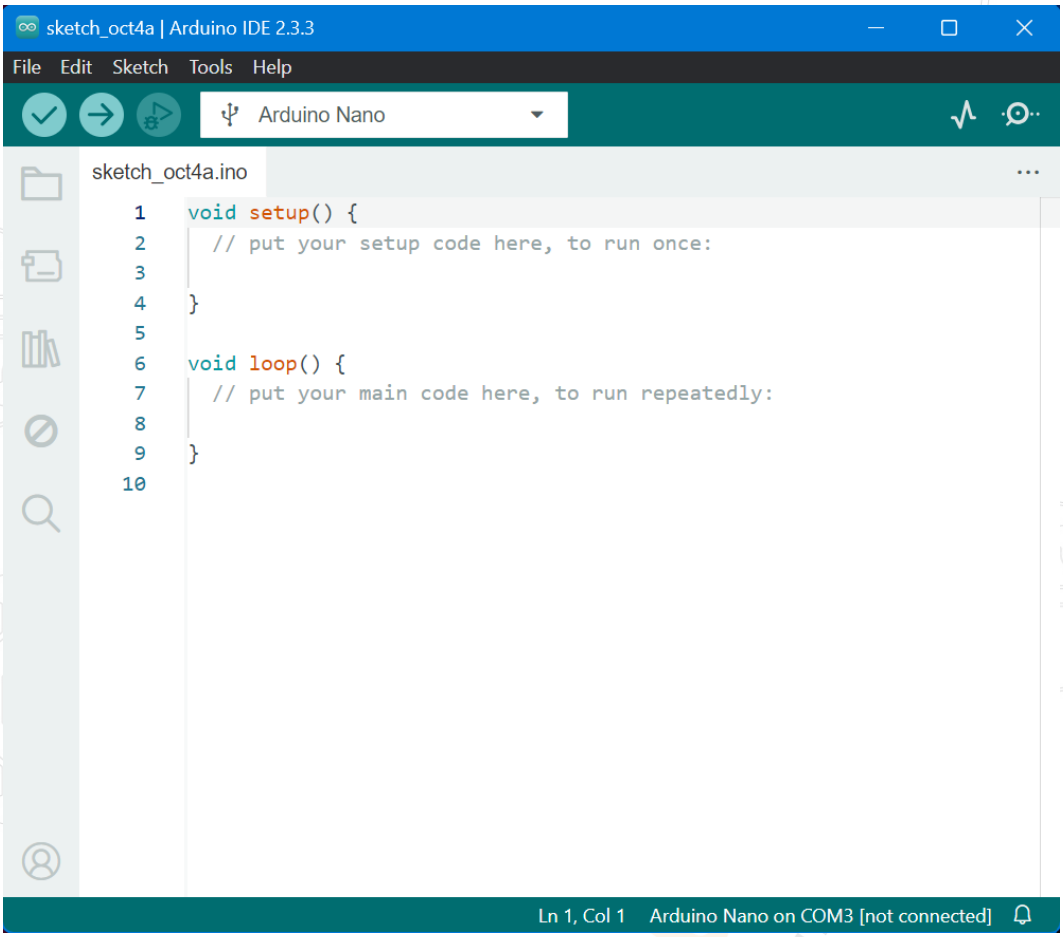


fig 0.16 : installation ide_16

Si vous procédez de cette dernière manière, il faut installer les drivers de la carte. Le répertoire dé zippé contient à la fois les fichiers nécessaires au bon fonctionnement de l'IDE, mais aussi ceux nécessaires à l'installation des drivers USB de la carte UNO.

Branchez une carte UNO à un port USB de votre ordinateur. Vous allez certainement voir un message apparaître mentionnant que Windows a découvert un nouveau matériel. Ignorez ce message et fermez les tentatives de Windows de faire l'installation.

La méthode la plus fiable est d'aller dans le gestionnaire de périphérique et de faire l'installation manuellement.

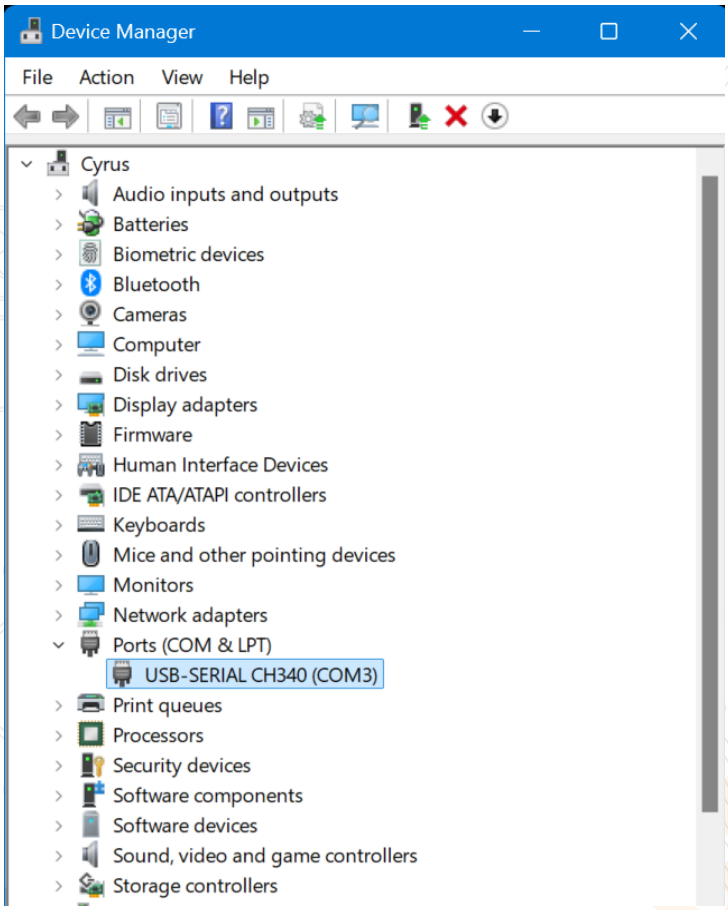


fig 0.17 : installation ide_17

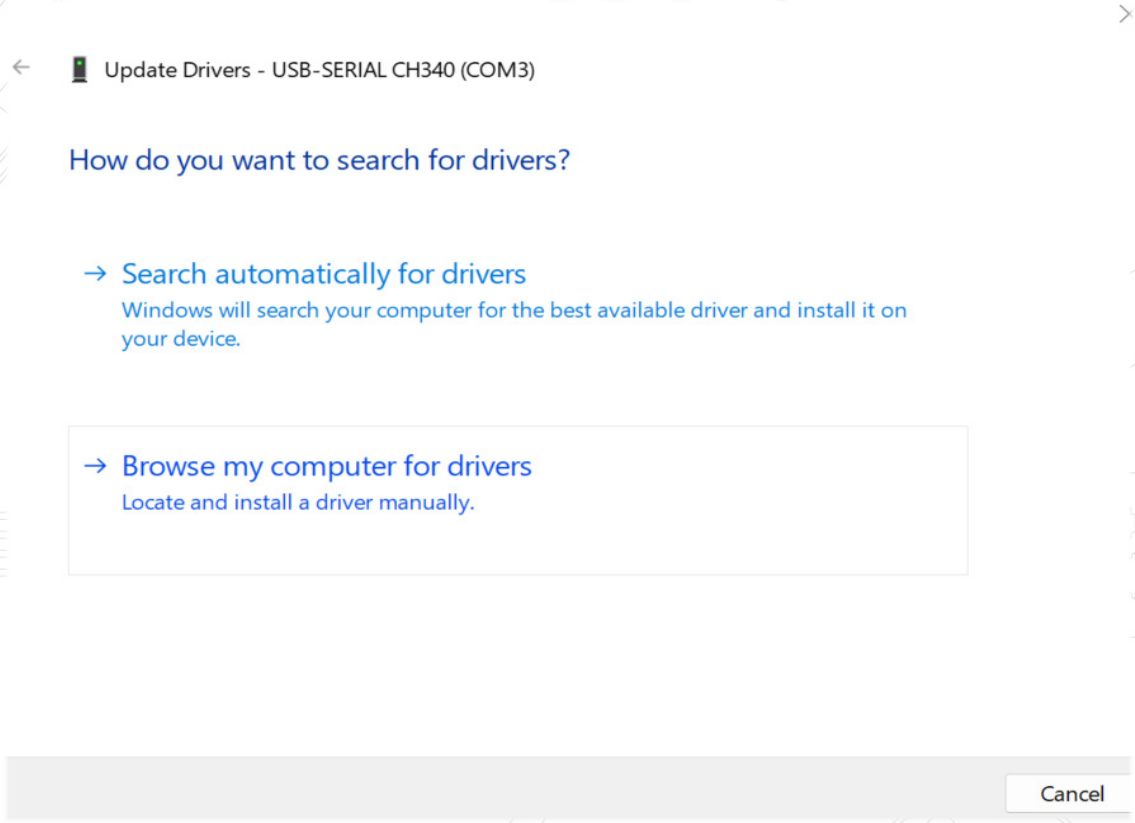


fig 0.18 : installation ide_18

Faites un clic droit et sélectionnez le menu *"Mettre à jour le driver"*. Sélectionnez ensuite *parcourir* et allez chercher le répertoire dans lequel vous avez dé zippé le fichier Arduino.

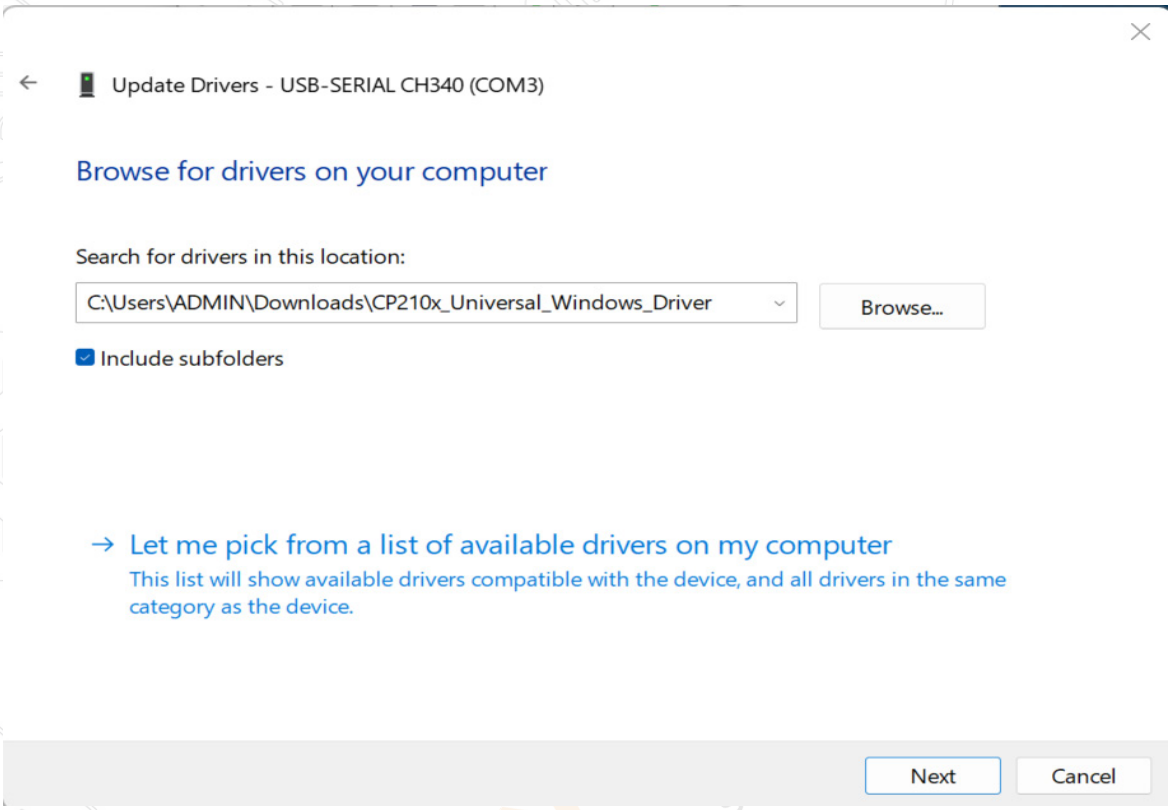


fig 0.19 : installation ide_19

Cliquez *'Suivant'*. Après un message de sécurité, vous obtenez l'écran de confirmation de la bonne installation.

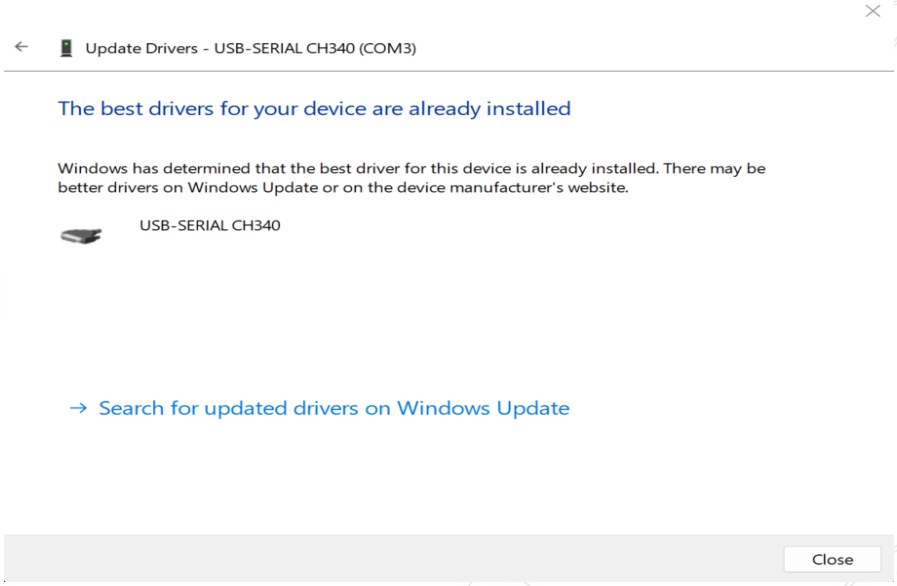


fig 0.20 : installation ide_20

Leçon 1

Ajouter une bibliothèque / utiliser le moniteur série

Installer des bibliothèques complémentaires

Lorsque vous aurez bien saisi les fonctions intégrées de l'environnement Arduino, vous aurez certainement le besoin ou l'envie d'aller encore plus loin, avec, pourquoi pas des bibliothèques complémentaires.

Qu'est-ce qu'une bibliothèque ?

Une bibliothèque est une suite d'instructions qui rendent beaucoup plus facile l'utilisation de composants complexes. Cela peut être pour l'utilisation d'un écran à cristaux liquide, pour l'utilisation d'un servomoteur etc...

Comment installer ?

L'IDE contient un centre de management des bibliothèques qui permet de vérifier la bonne installation de telle ou telle bibliothèque ou d'en ajouter de nouvelles. Pour installer une nouvelle bibliothèque, cliquez sur le menu suivant :

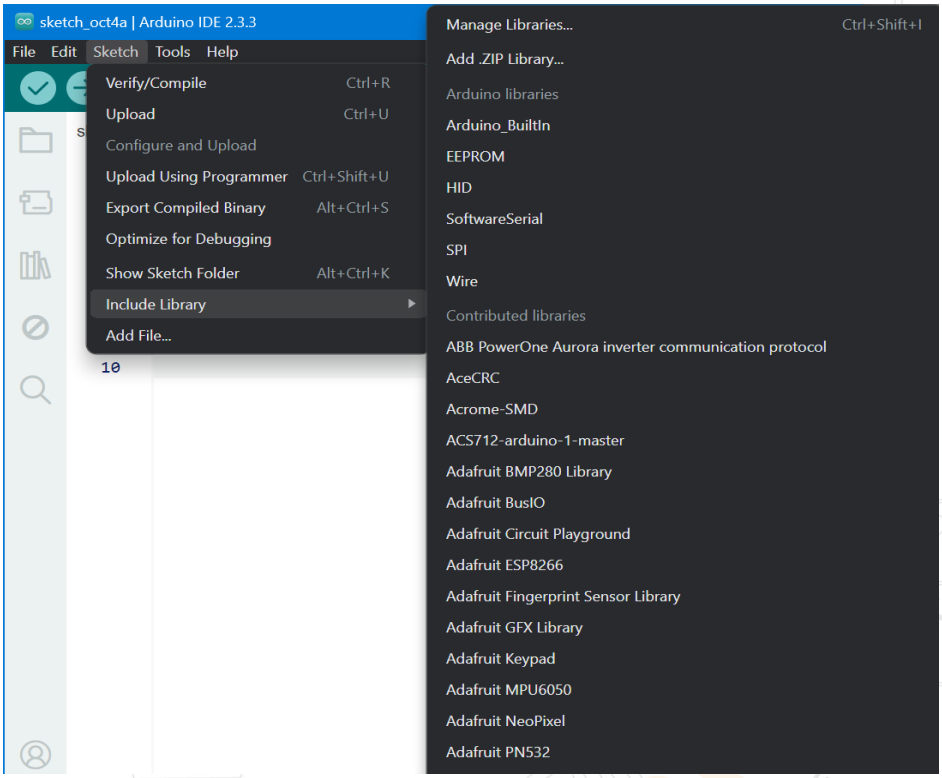


fig 1.1 : ajout d'une bibliothèque_1

Vous allez pouvoir voir l'ensemble des bibliothèques déjà présentes ainsi que la version utilisée :

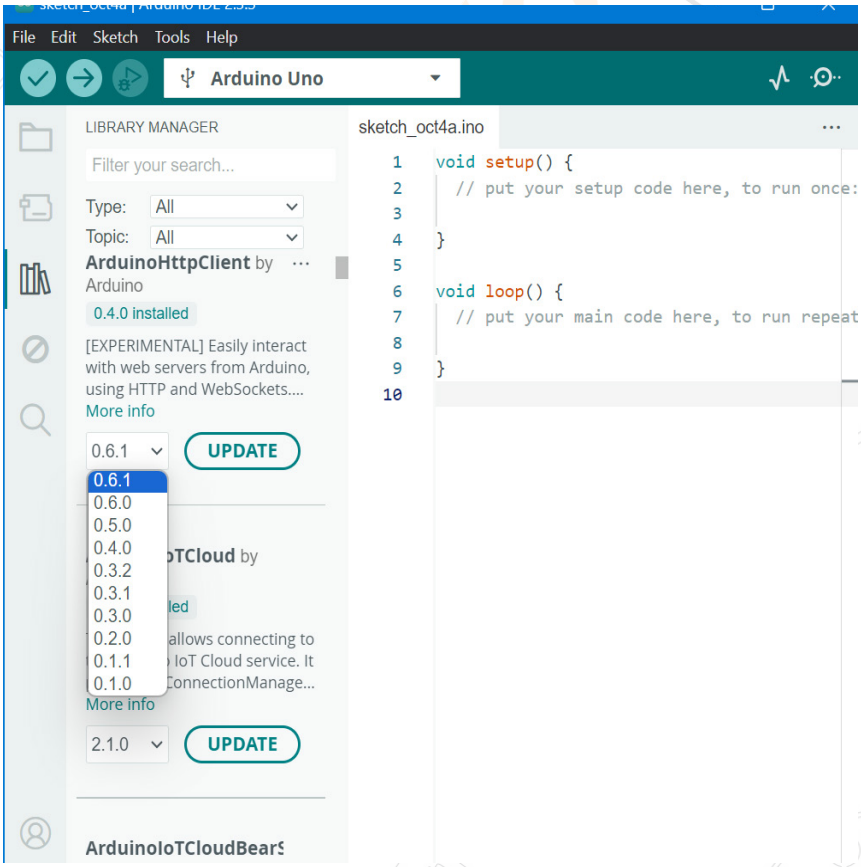


fig 1.2 : ajout d'une bibliothèque_2

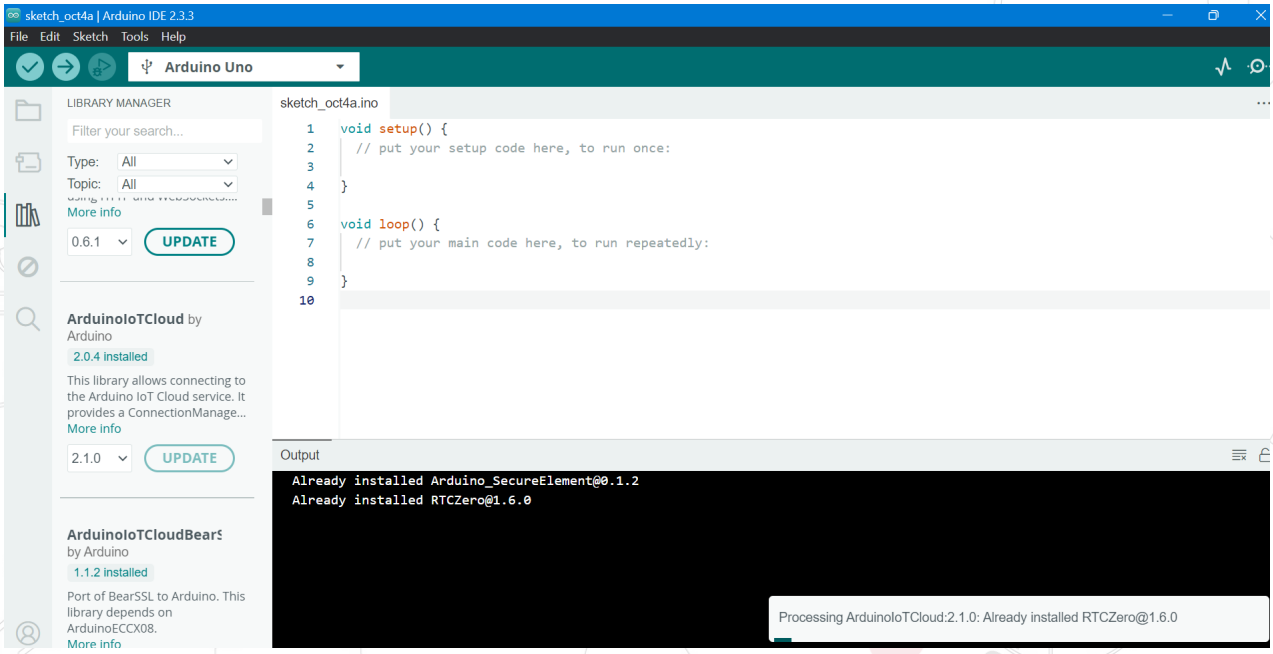


fig 1.3 : ajout d'une bibliothèque_3

Si la dernière version d'une bibliothèques n'est pas installée, le bouton "Installer" se dégrise, vous pouvez procéder à son installation.

Si la bibliothèque recherchée n'apparaît pas

Procurez-vous sur internet (notamment sur GITHUB) le zip de la bibliothèque que vous souhaitez ajouter. Il n'est pas nécessaire de dé zipper les fichiers, mais notez bien l'emplacement du fichier.

Dans le menu « Sketch », sélectionner « Inclure une bibliothèque ». Sélectionnez ensuite « Ajouter .ZIP Library ».

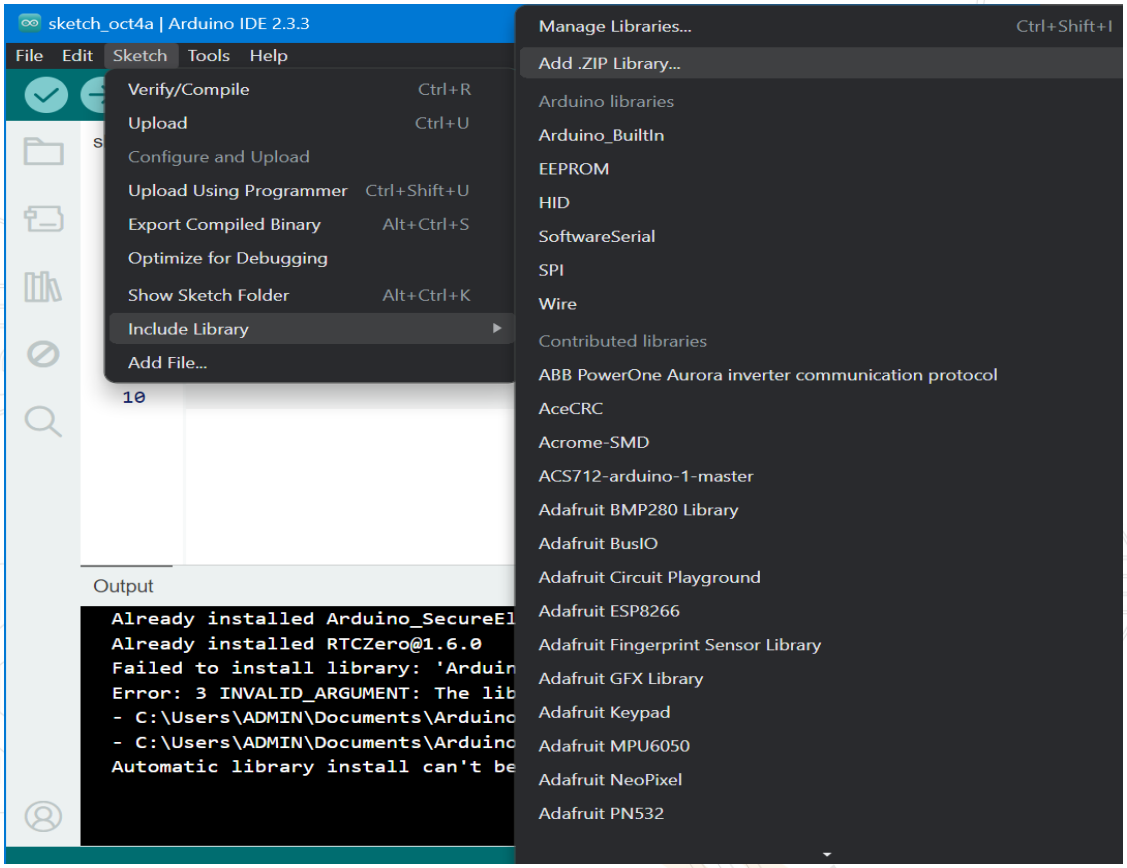


fig 1.4 : ajout d'une bibliothèque_4

Il vous suffit ensuite de sélectionner le zip contenant le fichier

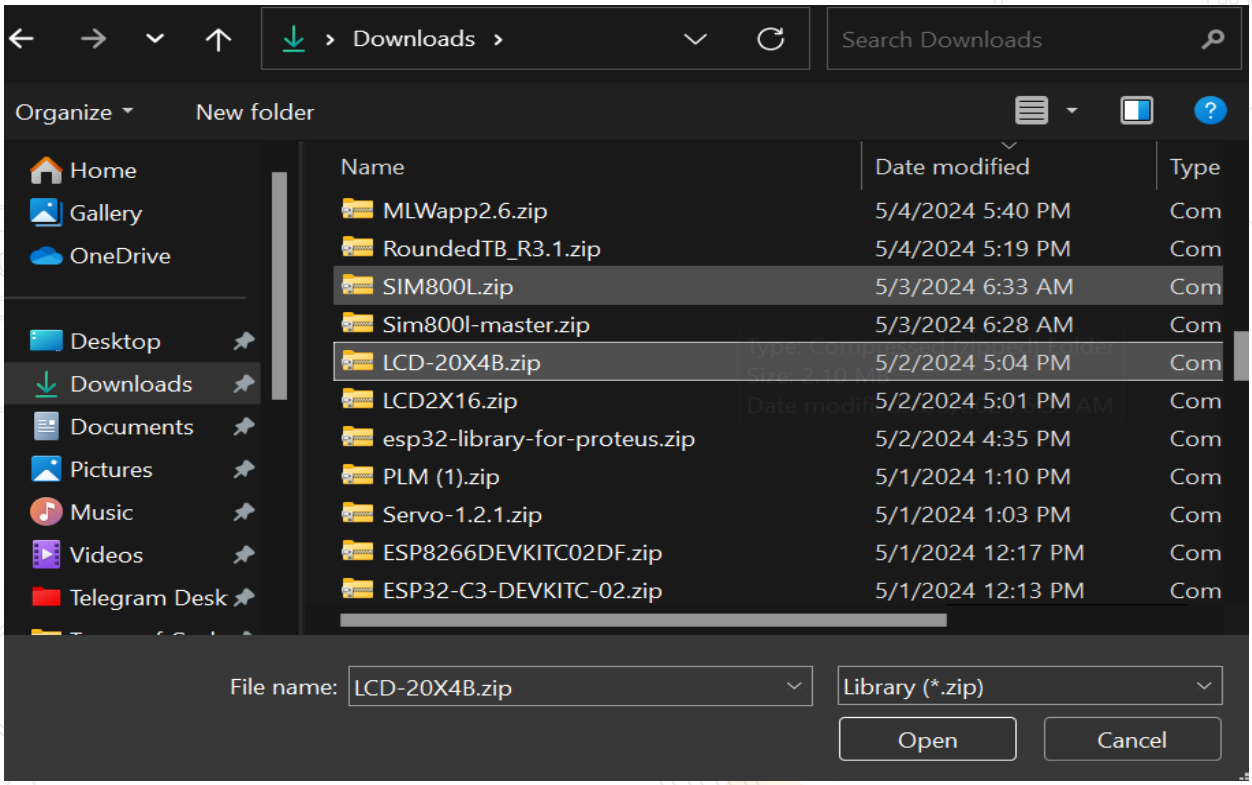


fig 1.5 : ajout d'une bibliothèque_5

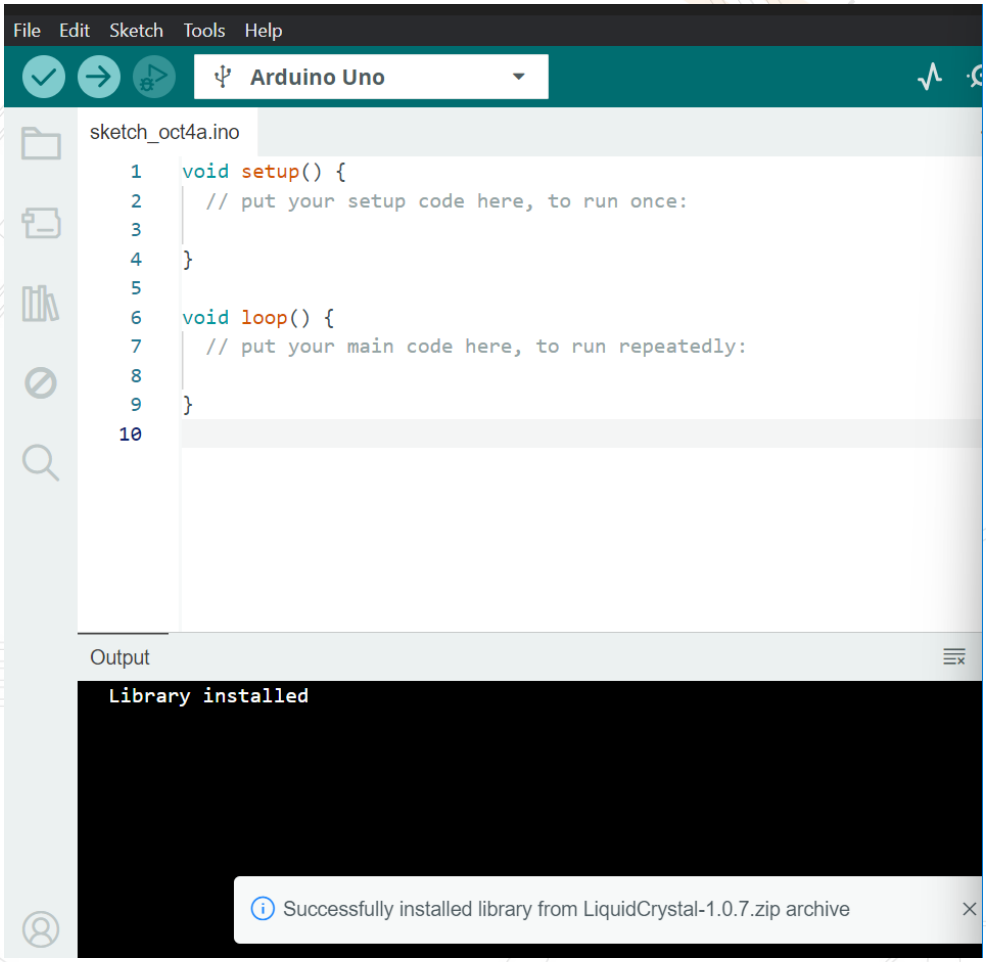


fig 1.6 : ajout d'une bibliothèque_6

Revenez au menu Sketch> Import Library. Vous devriez maintenant voir la bibliothèque en bas du menu déroulant. Il est prêt à être utilisé dans votre croquis. Le fichier zip sera extrait dans le dossier des bibliothèques d'Arduino.

NB: La bibliothèque sera disponible à utiliser dans des croquis, mais les exemples pour la bibliothèque ne seront pas exposés dans le fichier> Exemples jusqu'au redémarrage de l'IDE.

Ces deux sont les approches les plus courantes. Les systèmes MAC et Linux peuvent être gérés de la même manière. L'installation manuelle à introduire ci-dessous comme alternative peut être rarement utilisée et les utilisateurs sans besoins peuvent l'ignorer.

Installation manuelle

Pour installer la bibliothèque, quittez d'abord l'application Arduino. Ensuite, décompressez le fichier ZIP contenant la bibliothèque. Par exemple, si vous installez une bibliothèque appelée "ArduinoParty", décompressez ArduinoParty. zip. Il devrait contenir un dossier appelé ArduinoParty, avec des fichiers comme ArduinoParty.cpp et ArduinoParty.h à l'intérieur. (Si les fichiers .cpp et .h ne sont pas dans un dossier, vous devrez en créer un. Dans ce cas, vous devez créer un dossier appelé "ArduinoParty" et y déposer tous les fichiers qui se trouvaient dans le ZIP Fichier, comme ArduinoParty.cpp et ArduinoParty.h.)

Faites glisser le dossier ArduinoParty dans ce dossier (votre dossier de bibliothèques). Sous Windows, il sera probablement appelé "Mes documents \ Arduino \ bibliothèques". Pour les utilisateurs de Mac, il sera probablement appelé «Documents/ Arduino / bibliothèques». Sur Linux, ce sera le dossier "bibliothèques" dans votre carnet de croquis.

Votre dossier de bibliothèque Arduino devrait maintenant ressembler à ceci (sous Windows):

Mes documents \ Arduino \ bibliothèques \ ArduinoParty \ ArduinoParty.cpp Mes documents \ Arduino \ bibliotecas \ ArduinoParty \ ArduinoParty.h Mes documents \ Arduino \ bibliotecas \ ArduinoParty \ examples

Ou comme ceci (sur Mac et Linux):

Documents / Arduino / bibliothèques / ArduinoParty / ArduinoParty.cpp Documents / Arduino / bibliothèques / ArduinoParty / ArduinoParty.h Documents / Arduino / bibliothèques / ArduinoParty / examples.

...

Il peut y avoir plus de fichiers que les fichiers .cpp et .h, assurez-vous qu'ils sont tous là. (La bibliothèque ne fonctionnera pas si vous mettez les fichiers .cpp et .h directement dans le dossier des bibliothèques ou si elles sont imbriquées dans un dossier supplémentaire. Par exemple: Documents \ Arduino \ bibliotecas \ ArduinoParty.cpp et Documents \ Arduino \ Les bibliothèques \ ArduinoParty \ ArduinoParty \ ArduinoParty.cpp ne fonctionneront pas.)

Redémarrez l'application Arduino. Assurez-vous que la nouvelle bibliothèque apparaît dans l'élément de menu Sketch-> Import Library du logiciel. C'est tout! Vous avez installé une bibliothèque!

Utilisation du moniteur série (Windows, Mac, Linux)

Le moniteur série est une petite fenêtre très utile qui va vous permettre d'interagir en temps réel avec la carte UNO. Vous pouvez lui envoyer des informations et elle pourra à son tour faire de même. Vous verrez dans les différentes leçons que le recours au moniteur série est très fréquent.

Se connecter

Pour l'ouvrir, c'est très simple, il suffit de cliquer sur le petit icône entouré en rouge.

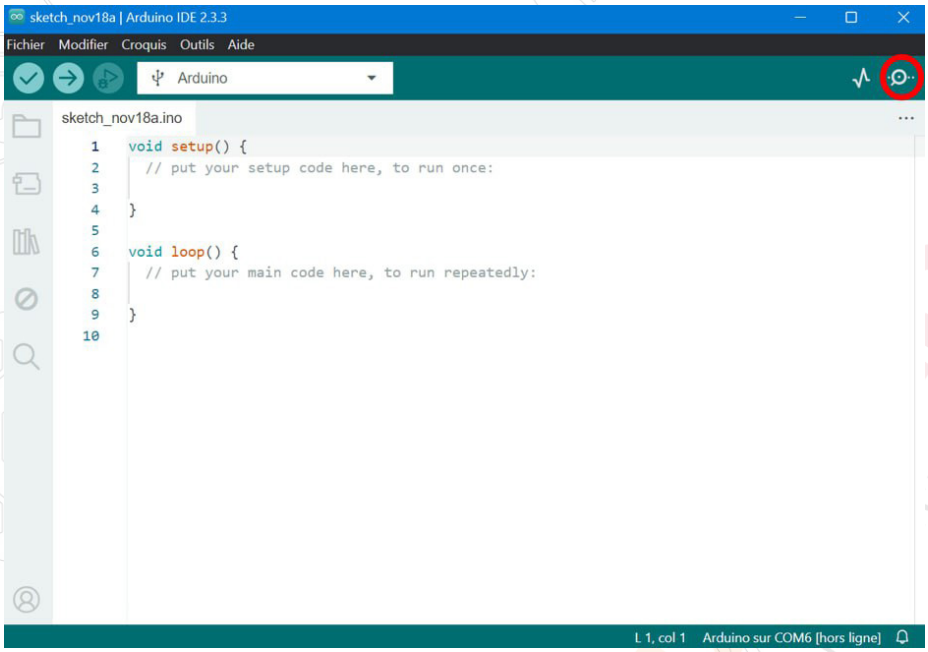


fig 1.7 : ajout d'une bibliothèque_7

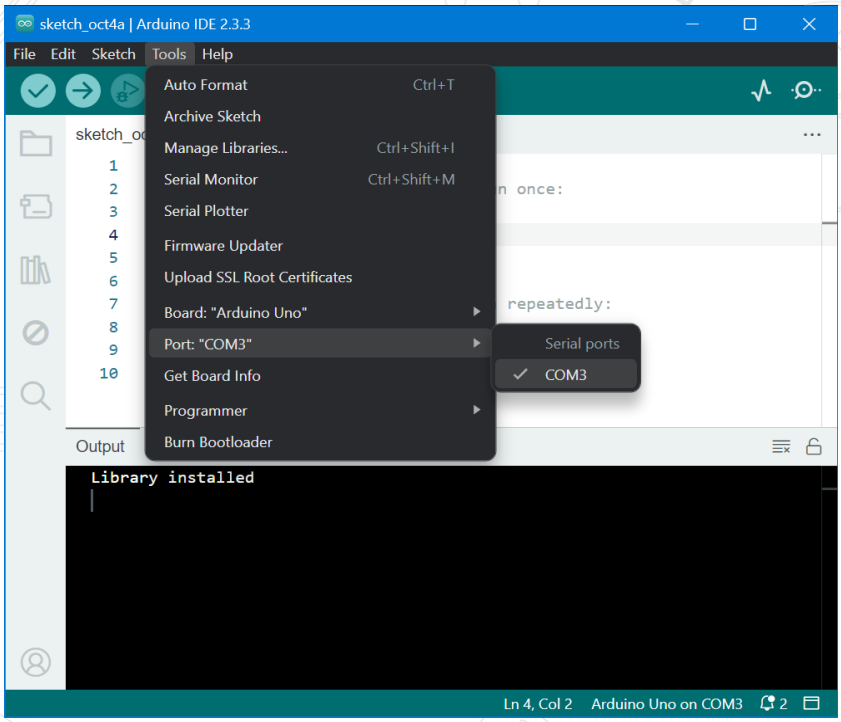


fig 1.8 : ajout d'une bibliothèque_8

A l'ouverture, vous devez voir cette fenêtre:

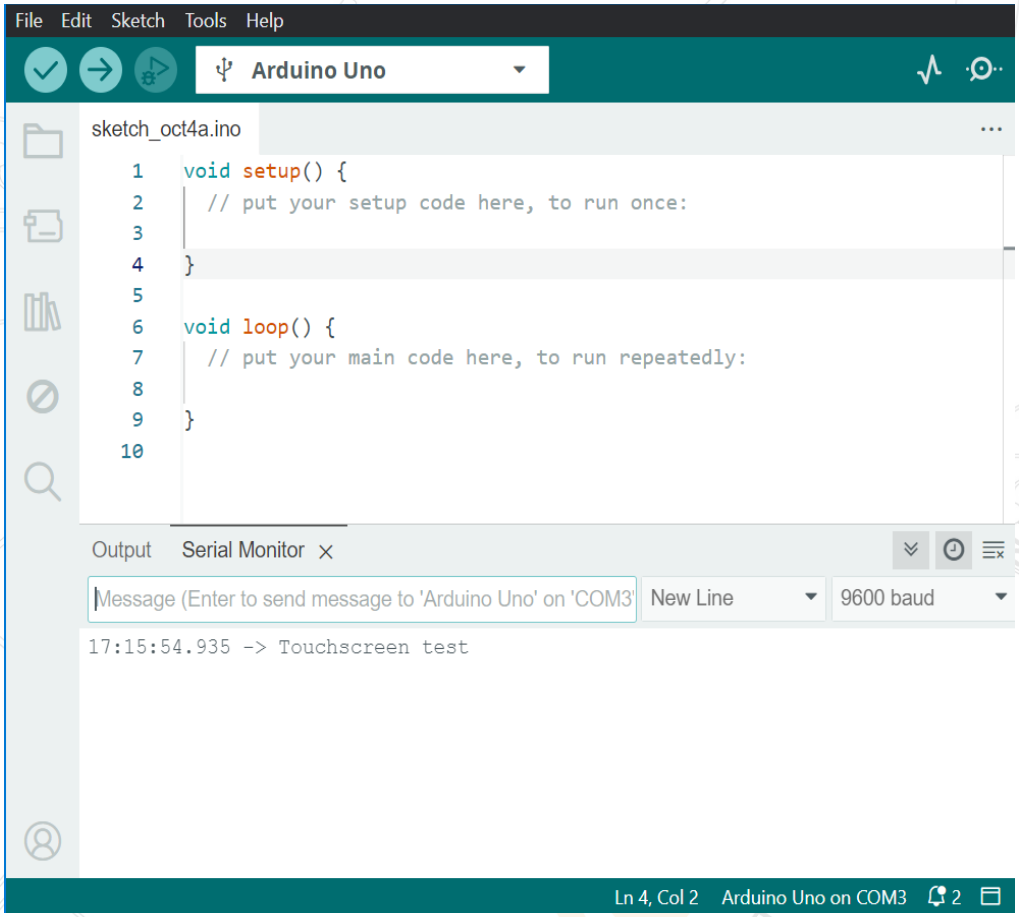
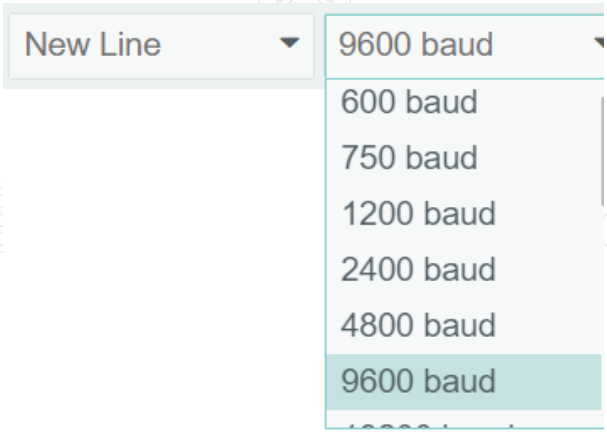


fig 1.9 : ajout d'une bibliothèque_9

Paramètres

Vous pouvez définir le taux de transfert de la connexion de la manière suivante



Et vous pouvez avoir un défilement automatique de l’écran comme suit

☑ Autoscroll

fig 1.10 : ajout d’une bibliothèque_10

Avantages

Le Serial Monitor est un excellent moyen d'établir rapidement une connexion série avec votre Arduino. Si vous travaillez déjà dans l'IDE Arduino, il n'est vraiment pas nécessaire d'ouvrir un terminal distinct pour afficher les données.

Leçon 2

Blink

But de la leçon

Dans cette leçon, vous allez prendre en main l'IDE et apprendre à faire clignoter la LED intégrée à la carte UNO R3.

Matériel nécessaire:

(1) x Arduino Uno R3

Principe

La carte UNO R3 possède deux rangées de connecteurs le long de ses extrémités qui sont utilisés pour brancher une vaste gamme de composants électroniques ou des cartes d'extensions (appelées shields) qui augmentent ses capacités.
Elle est aussi équipée d'une LED intégrée qu'il est possible de commander au travers de vos programmes. Vous pouvez apercevoir cette LED sur l'image ci-dessous, elle est repérable grâce au « L » visible sur la carte.

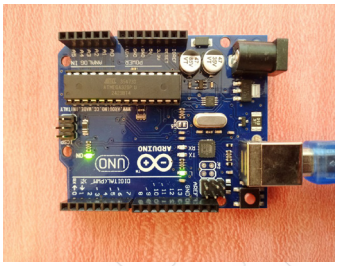


fig 2.1 :schéma blink_1

Lorsque vous connectez votre carte pour la première fois, il est possible que la LED se mette à clignoter. C'est parce que les cartes sont fréquemment expédiées avec le programme « BLINK » pré-installé.

Dans cette leçon, nous allons reprogrammer la carte UNO R3 avec notre propre programme « BLINK », ce qui va nous permettre notamment de changer la fréquence de clignotement de la LED.

Ouvrez le sketch "BLINK" dans l'environnement de programmation via le menu "FICHIER/EXEMPLES/01.BASICS/Blink"

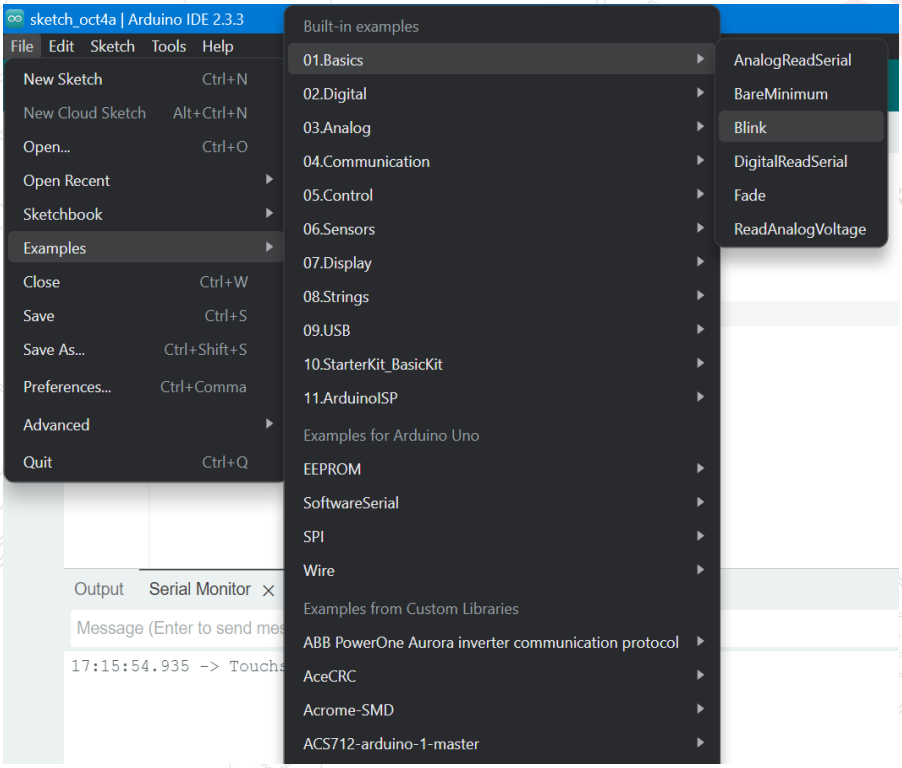


fig 2.2 : blink_2

Voici ce que vous devez obtenir.

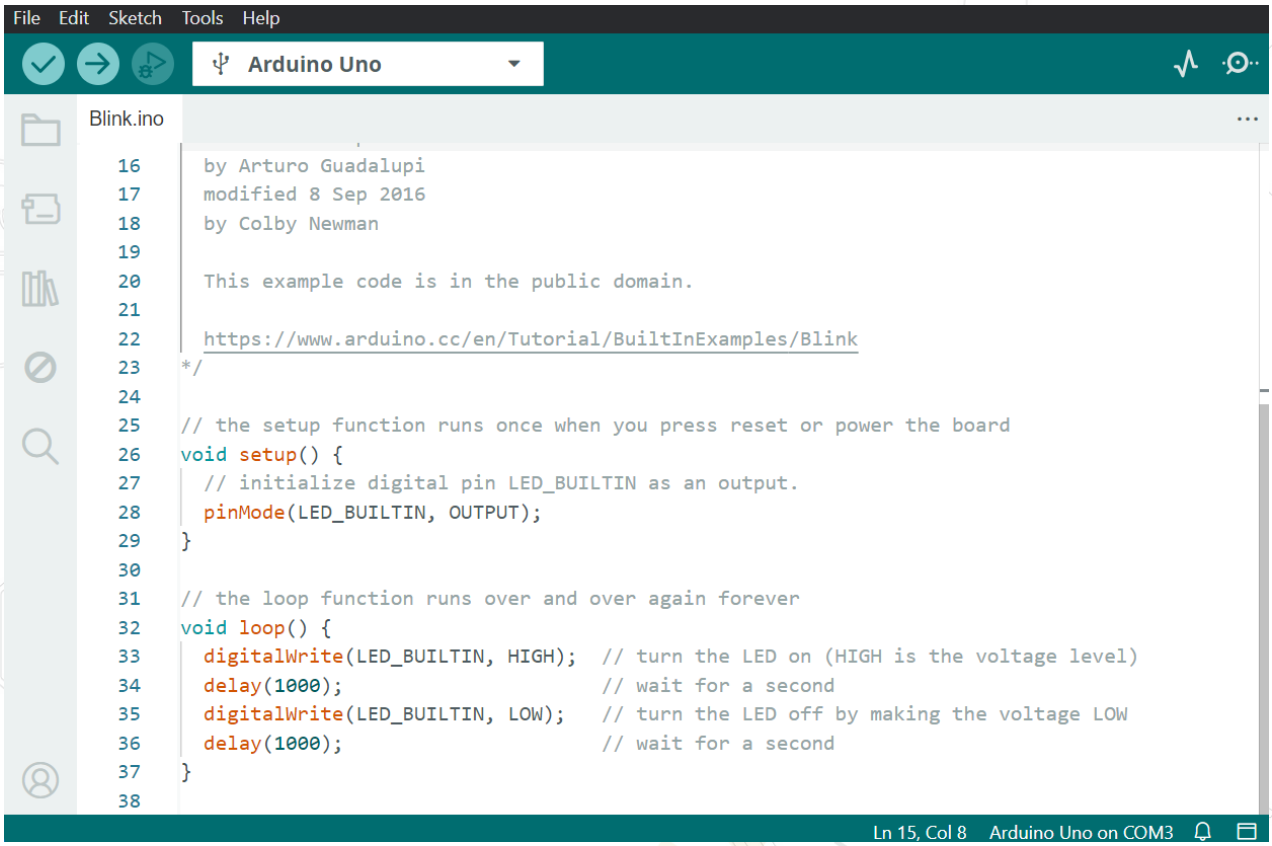


fig 2.3 : blink_3

Comme vous allez apporter des modifications à ce sketch, la première étape consiste à l'enregistrer dans votre répertoire personnel, les sketchs exemples étant en lecture seule.

Utilisez le menu « enregistrer sous... » et définissez le répertoire et nom de fichier que vous souhaitez.

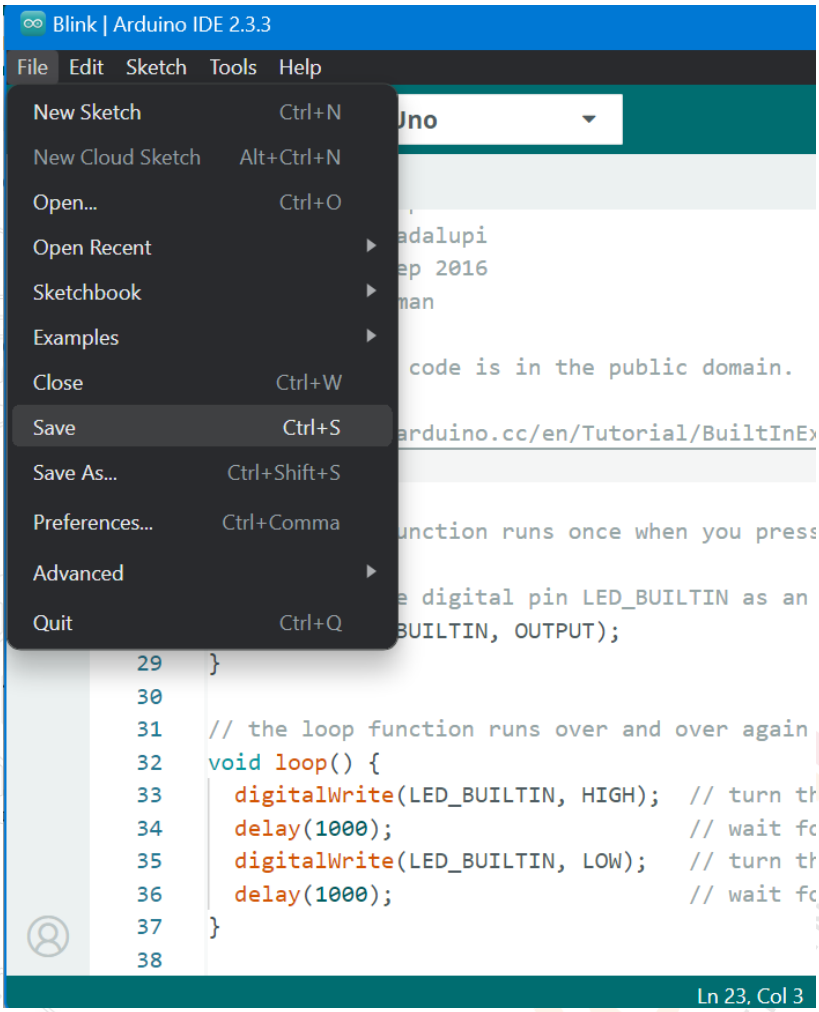


fig 2.4 : blink_4

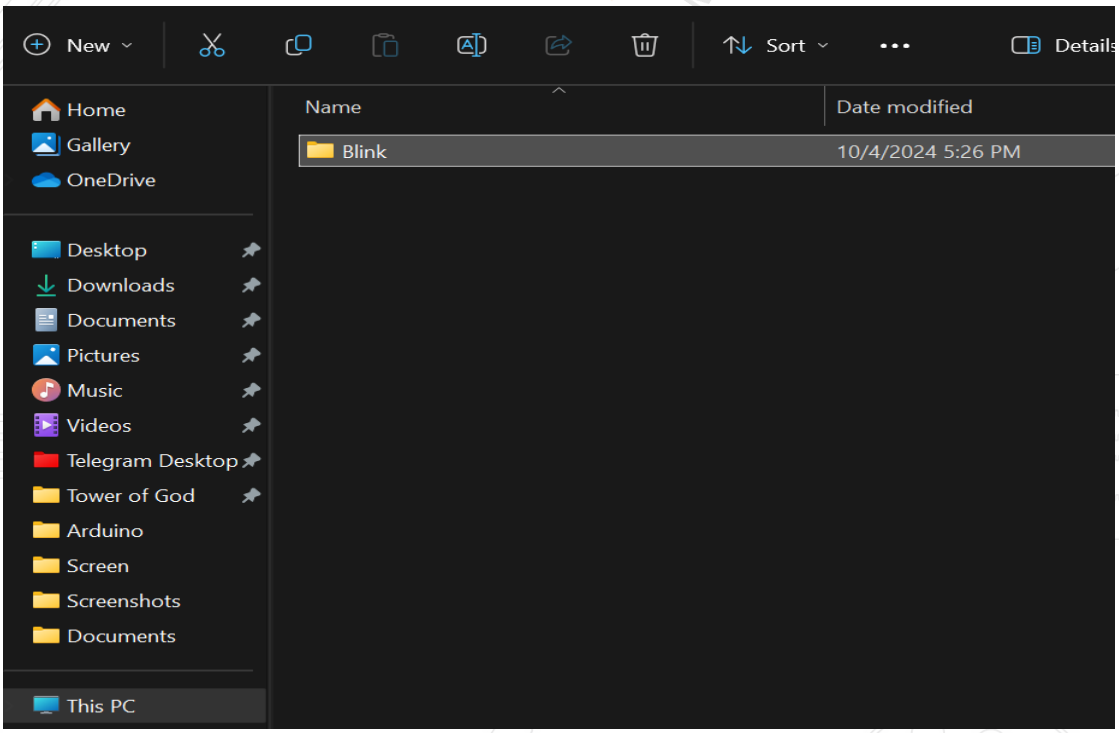


fig 2.5 : blink_5

Vous venez d'enregistrer le fichier dans votre répertoire.

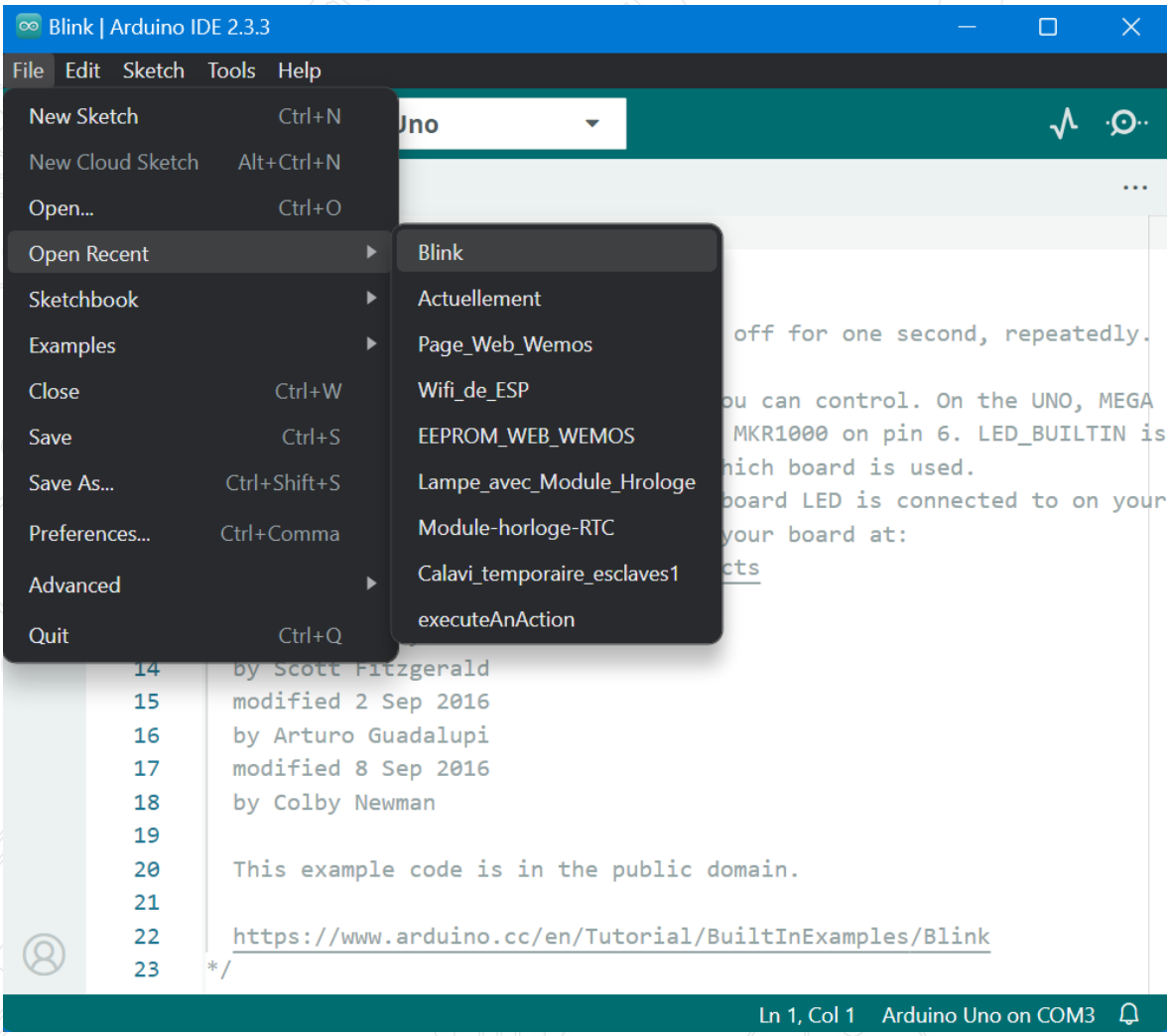


fig 2.5 : blink_5

Vous pouvez accéder à vos fichiers récents de la même manière que la plupart de vos autres logiciels.

Il est temps maintenant de connecter la carte UNO R3 via le port USB de votre ordinateur et de vérifier la bonne reconnaissance de celle-ci.

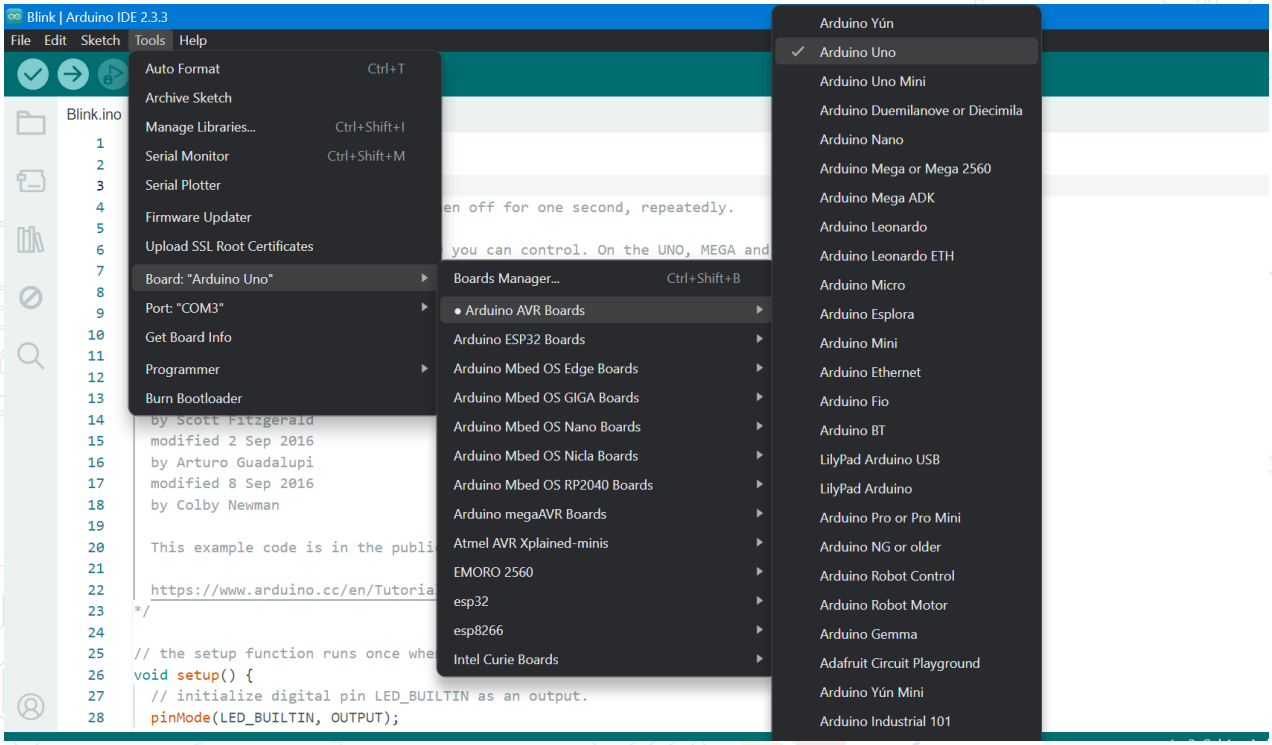


fig 2.6 : blink_6

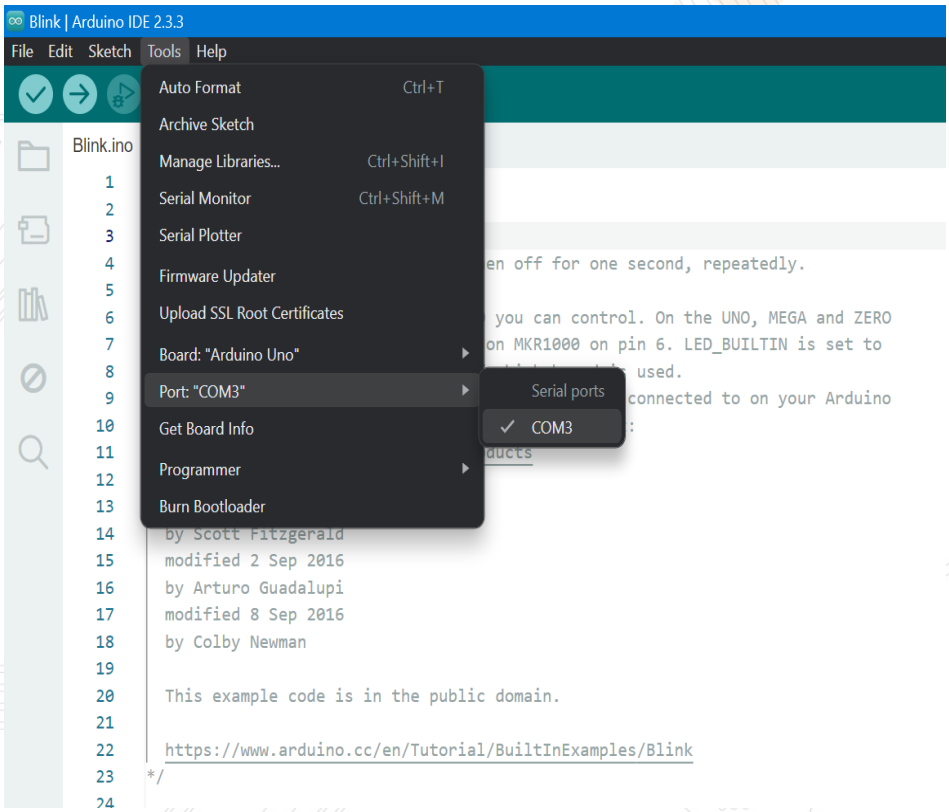


fig 2.7 : blink_7

Note: le numéro de port COM ne sera pas nécessairement celui que vous pouvez apercevoir sur les captures d'écran. En effet, c'est votre ordinateur qui va affecter une référence au moment de la connexion et si vous possédez plusieurs cartes, chacune aura potentiellement un numéro différent.

Une fois connecté, vous pouvez apercevoir ce petit texte en bas à droite de votre écran.

fig 2.8 : blink_8

Cliquez sur le bouton avec la flèche pour déclencher le téléversement.

fig 2.9 : blink_9

Vous pouvez constater que la première étape est la compilation (le logiciel vérifie l'intégrité du code).

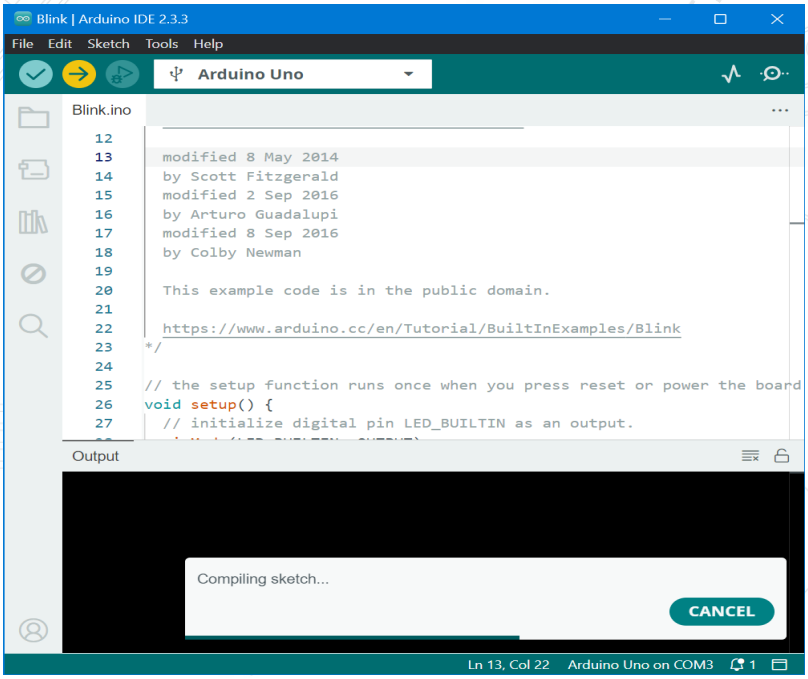


fig 2.10 : blink_10

Un message vous renseigne sur la taille du programme et la quantité de mémoire utilisée sur la carte. Si votre carte n'est pas correctement connectée ou reconnue, vous obtiendrez l'écran suivant. Reportez-vous aux explications données dans les premières leçons pour veiller s'assurer de la bonne installation des drivers.

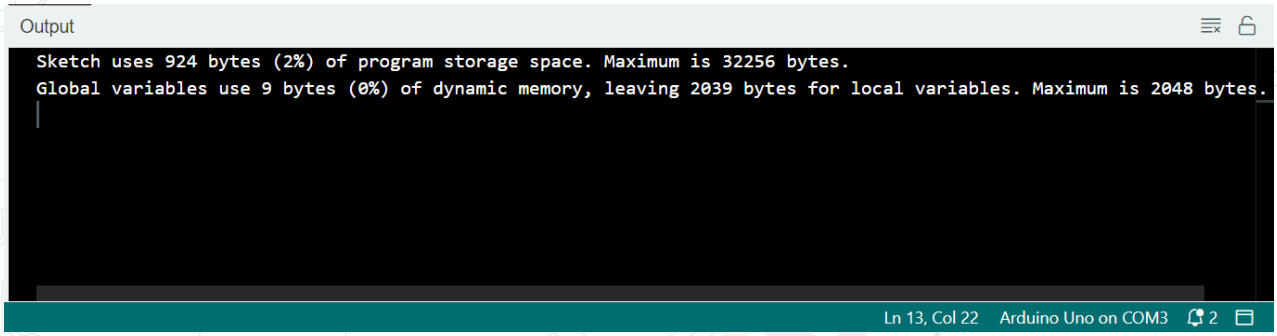


fig 2.11 : blink_11

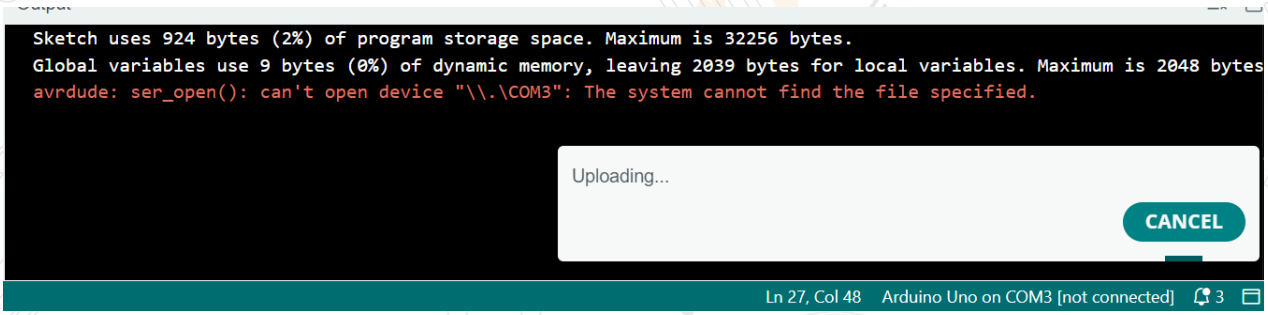


fig 2.12 : blink_12

Vous pouvez constater qu'une part importante du code est consacrée aux commentaires. Les commentaires sont importants ils permettent de bien comprendre le code, surtout quand celui-ci est complexe. Cela permet d'y revenir plus tard lorsqu'il est nécessaire de le faire évoluer.

Il existe deux façons de mettre du texte en commentaire. La première est de commencer la ligne par « // ». La deuxième manière permet de placer une portion de code/texte en commentaire (idéal lorsque l'on veut tester plusieurs alternatives par exemple). Débutez la zone commentaire par « /* » et terminez celle-ci par « */ »

Dans le code « blink », le bloc suivant représente le code qui sera exécuté une unique fois lors de la mise sous tension de la carte.

```
void setup() {
// initialize the digital pin as an output. pinMode(LED_BUILTIN, OUTPUT);
}
```

Tous les sketchs doivent obligatoirement avoir un bloc “setup” dans lequel vous pouvez placer autant d’instructions que nécessaires. Les instructions s’écrivent entre {}.

Dans « BLINK » la seule instruction consiste à affecter le pin 13 en tant que sortie (OUTPUT).

Il est aussi obligatoire dans un sketch d’avoir un bloc “loop”. Contrairement au premier bloc celui-ci s’exécutera en boucle tant que la carte sera sous tension.

```
void loop() {
digitalWrite(LED, HIGH); // turn the LED_BUILTIN on (HIGH is the voltage level)
delay(1000); // wait for a second
digitalWrite(LED, LOW); // turn the LED_BUILTIN off by making the voltage LOW
delay(1000); // wait for a second
}
```

Le bloc « loop » est constitué de 4 instructions. La première déclenche l’allumage de la LED, la deuxième une attente de 1000ms, la troisième l’extinction de la LED. La dernière instruction une attente de 1000ms.

Vous allez maintenant augmenter la fréquence de clignotement de la LED. Vous avez certainement deviné que le paramètre à changer est celui entre () sur les instructions « delay ». Téléversez le programme et observez.

```
30
31    // the loop function runs over and over again forever
32    void loop() {
33      digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
34      delay(1000);                      // wait for a second
35      digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
36      delay(1000);                      // wait for a second
37    }
```

fig 2.13 : blink_13

Leçon 3

LED

But de la leçon

Dans celle leçon, vous allez apprendre à faire varier la luminosité d'une LED en utilisant plusieurs valeurs de résistances.

Matériel nécessaire:

(1) x Arduino Uno R3
(1) x LED rouge de 5mm
(1) x résistance 220 ohm
(1) x résistance 1 kohm
(1) x résistance 10 kohm
(2) x câbles mâle-mâle

fig 3.1 : led rouge

Présentation des composants

PLANCHE DE PROTOTYPAGE MB-102:

Une planche de prototypage vous permet de réaliser des circuits très rapidement, sans avoir besoin de réaliser de soudures.

Exemple :

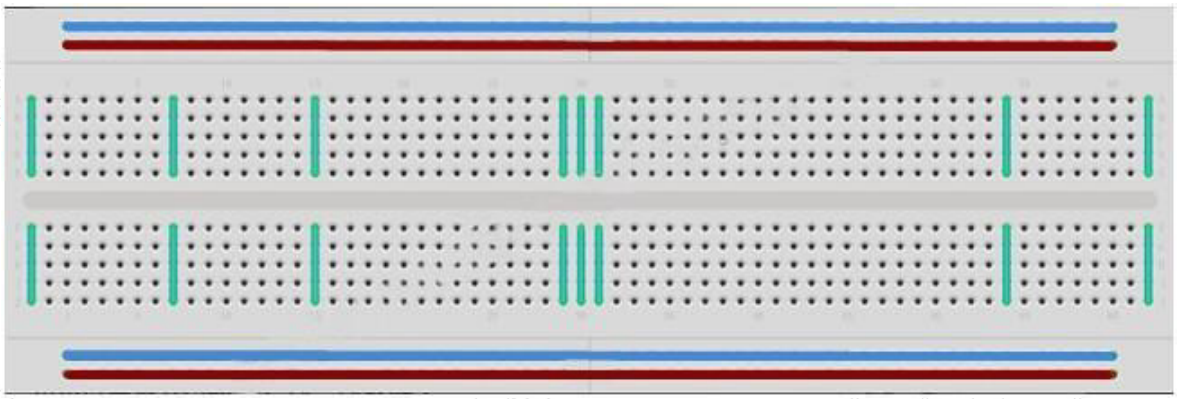


fig 3.2 : planche de prototypage MB-102

Il existe une grande variété de planches de prototypages. La plus simple est une grille de trous dans un bloc de plastique. A l'intérieur se trouvent des lames métal permettant la connexion électrique entre les différents trous d'une même ligne. La ligne creusée au centre de la plaque symbolise une rupture électrique entre la partie haute et la partie basse. Cela signifie aussi que vous pouvez positionner une puce à cheval entre haut et bas. Certaines planches de prototypages ont aussi deux lignes horizontales en haut et en bas. On les utilise généralement pour créer une ligne d'alimentation +5V et masse.

Attention tout de même, les planches de prototypages ont comme limite d'utilisation la qualité des connexions qui ne valent pas une soudure et peuvent entraîner parfois des dysfonctionnements.

LED:

Les LEDs font de parfaits indicateurs lumineux. Elles consomment peu de courant et ont une très bonne durée de vie.

Dans cette leçon, vous allez utiliser certainement le type de LEDs le plus commun, la LED 5mm. Il en existe d'autres de 3 à 10mm.

Attention : il n'est pas possible de brancher directement une LED sur une batterie car l'intensité de courant est trop forte.

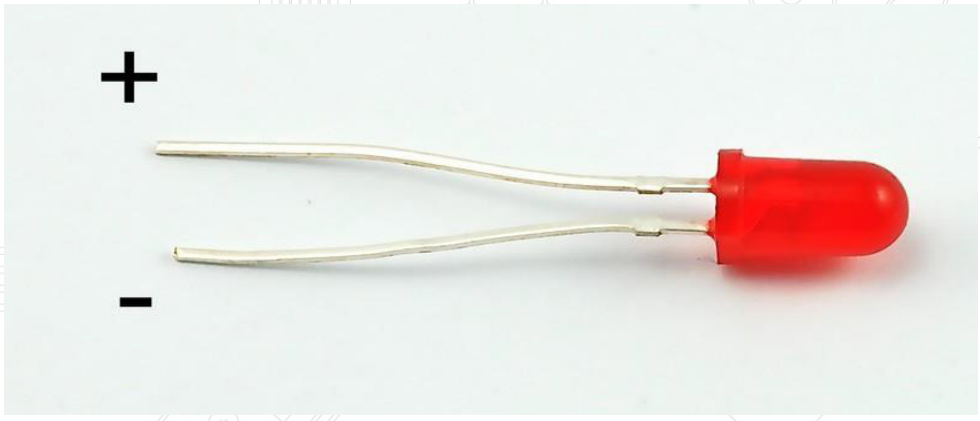


fig 3.3 : led rouge_avec_bornes

Attention aussi au sens de branchement de la LED. La patte la plus longue doit être connectée à la borne + et la patte courte à la borne -.

RÉSISTANCES:

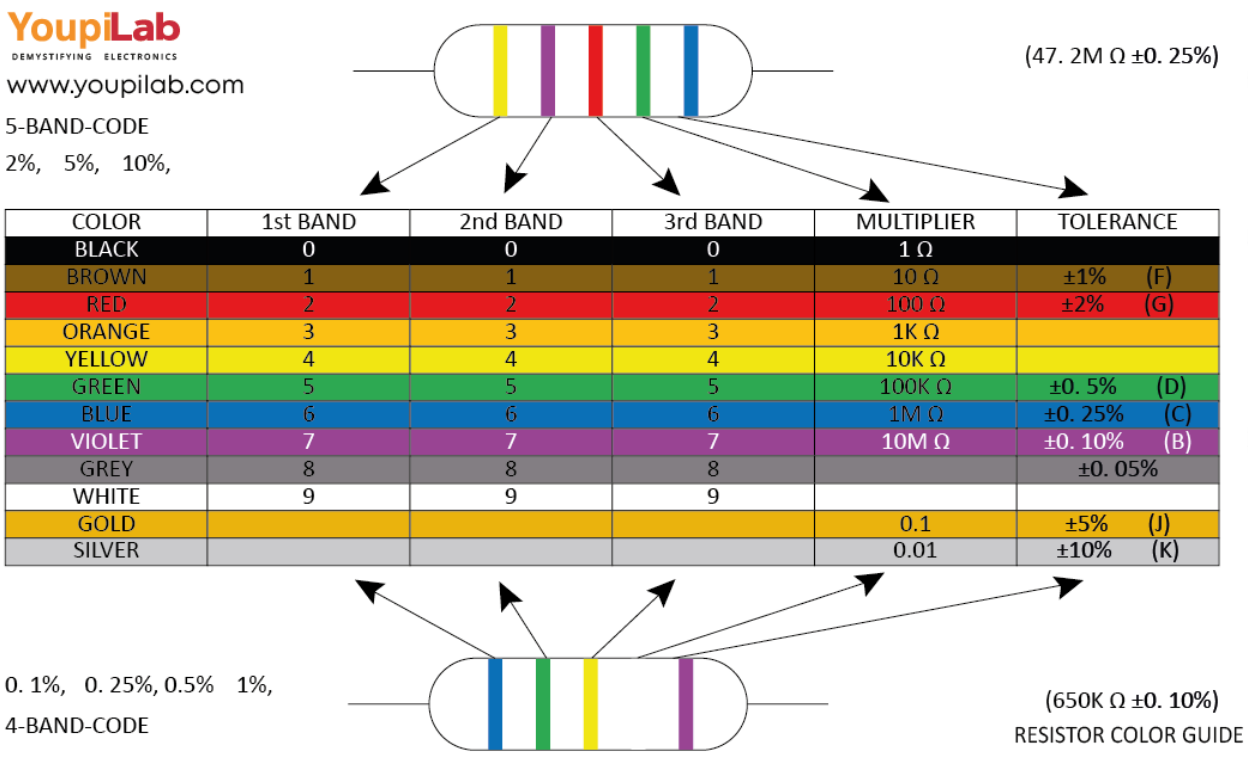
Comme leur nom le suggère, les résistances s'opposent au passage du flux d'électricité. Plus la valeur est grande plus la résistance l'est aussi. Vous allez pouvoir calibrer la brillance de la LED en jouant sur la valeur de la résistance.

fig 3.4 : résistance

Mais tout d'abord, quelques informations supplémentaires...

L'unité des résistances est l'ohm, que l'on représente généralement avec la lettre grecque Ω. Il faut beaucoup d'ohms pour avoir un minimum de résistance, c'est pour cela que l'on retrouve généralement des kilos, voir mégas ohms : kΩ (1,000 Ω) et MΩ (1,000,000 Ω).

Dans cette leçon, vous allez utiliser trois valeurs de résistance : 220Ω, 1kΩ et 10kΩ. C'est grâce aux anneaux de couleurs et au code associé que l'on peut connaitre la valeur d'une résistance.



COLOR	1st BAND	2nd BAND	3rd BAND	MULTIPLIER	TOLERANCE
BLACK	0	0	0	1 Ω	
BROWN	1	1	1	10 Ω	±1% (F)
RED	2	2	2	100 Ω	±2% (G)
ORANGE	3	3	3	1K Ω	
YELLOW	4	4	4	10K Ω	
GREEN	5	5	5	100K Ω	±0. 5% (D)
BLUE	6	6	6	1M Ω	±0. 25% (C)
VIOLET	7	7	7	10M Ω	±0. 10% (B)
GREY	8	8	8		±0. 05%
WHITE	9	9	9		
GOLD				0.1	±5% (J)
SILVER				0.01	±10% (K)

fig 3.5 : transistor

Notez aussi que contrairement aux LEDs, les résistances n'ont pas de sens.

Pour simplifier la tâche, il est aussi possible d'utiliser un appareil de mesure afin de connaître la valeur de la résistance.

Diagramme de câblage

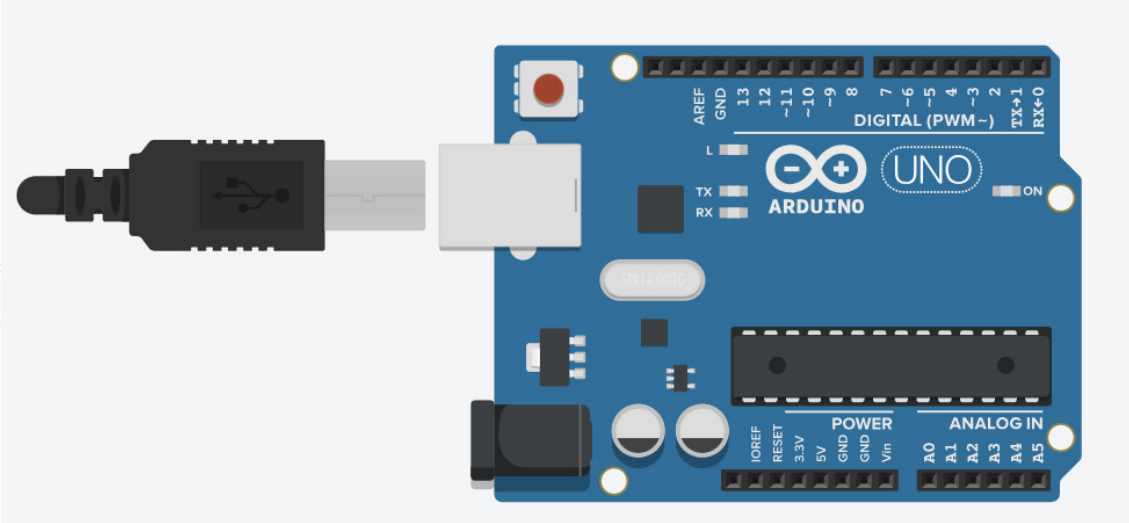


fig 3.6 : led_diagramme de câblage

L'UNO est une bonne source de +5V. Vous allez donc pouvoir l'utiliser pour fournir le bon voltage à la LED et la résistance. Il n'est pas nécessaire de téléverser un code pour cet exemple, simplement connecter la carte à un ordinateur pour celle-ci soit sous tension.

Avec une résistance de 220 Ω, la LED est très brillante. Elle l'est un peu moins avec une résistance de 1kΩ et à peine brillante avec une résistance de 10 kΩ.

Vous pouvez positionner la résistance avant ou après la LED et constater que cela n'a pas d'importance.

Illustration

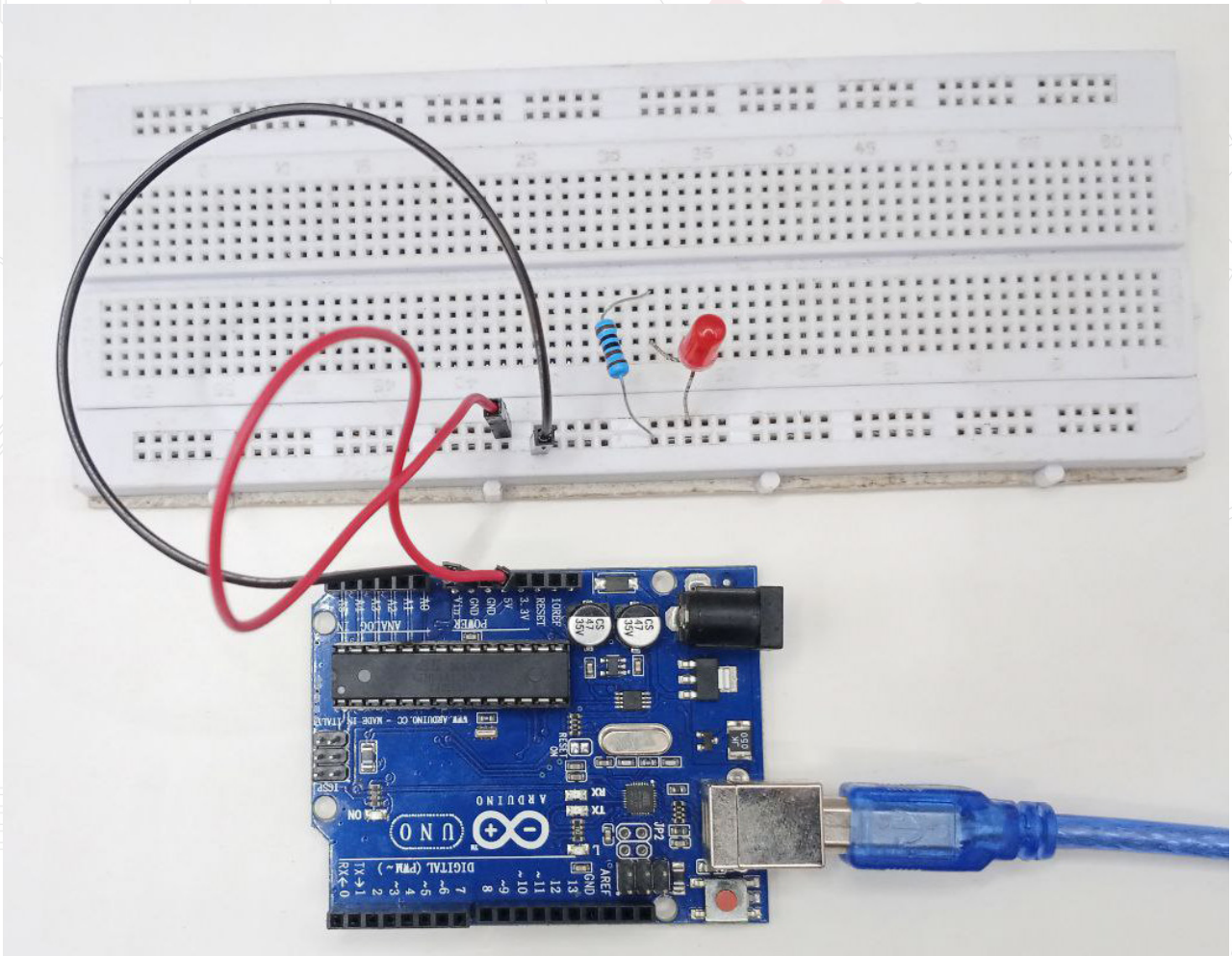


fig 3.7 : led_illustration

Leçon 4

DIGITAL INPUTS

But de la leçon

Dans cette leçon, vous allez apprendre à utiliser les entrées digitales pour allumer/éteindre une LED.
Presser un premier bouton allumera la LED, presser un autre bouton l'éteindra.

Matériel nécessaire:

(1) x Arduino Uno R3
(1) x Planche prototype
(1) x LED rouge 5mm
(1) x résistance 220 ohms
(2) x boutons poussoirs
(7) x Câbles mâle-mâle

Présentation du composant

Boutons poussoirs :

Les boutons poussoirs sont des composants très simples. Lorsque vous pressez le bouton, un contact électrique se fait et laisse passer le courant. Les boutons poussoirs utilisés ont 4 pattes, ce qui peut créer une certaine confusion.

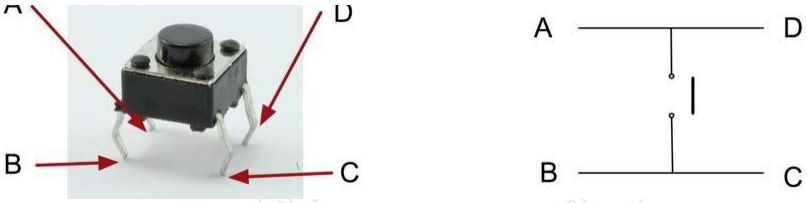


fig 4.1 : digital inputs_ boutons poussoirs

En fait, il n'y a bien que 2 connexions électriques. A et D sont connectées ensemble et B et C aussi. Presser le bouton permet au courant de lier électriquement A-D avec B-C.

Diagramme de câblage

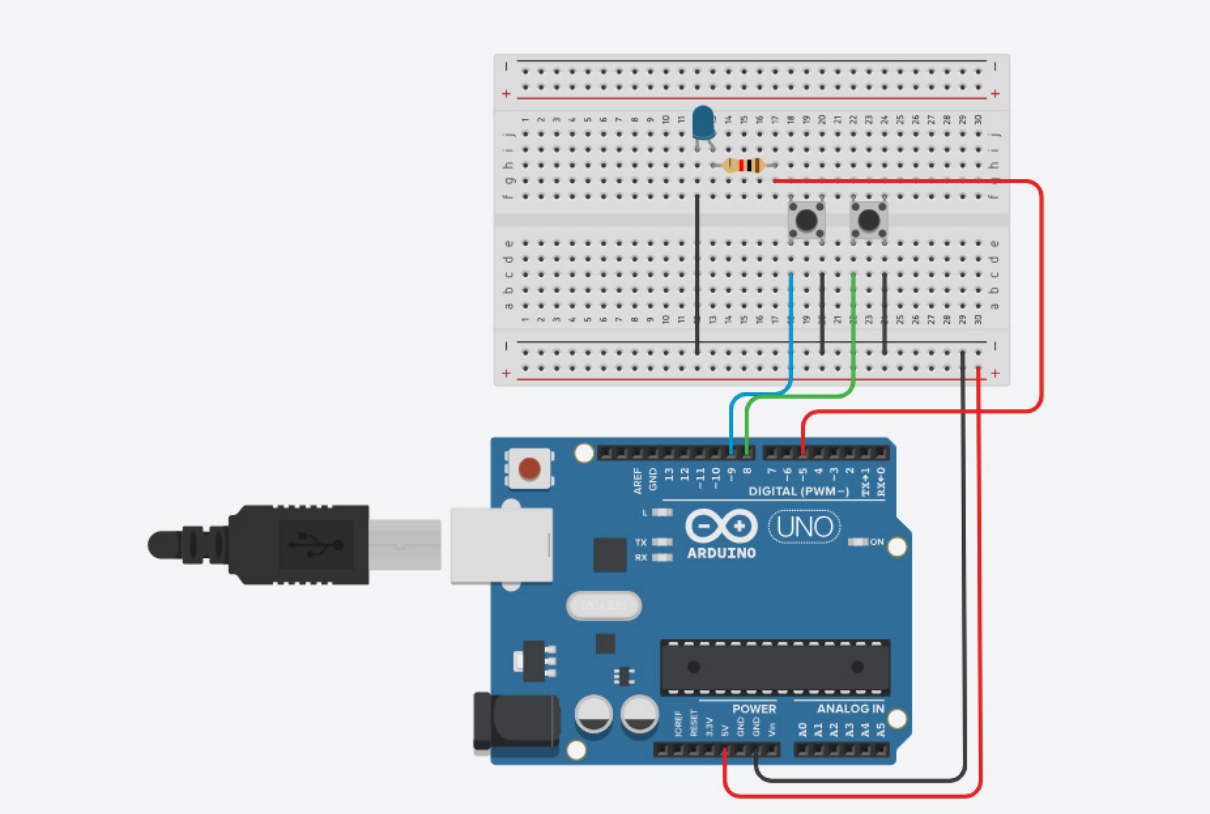


fig 4 .17 : digital inputs_ schéma de câblage

Code

Après avoir réalisé le câblage, ouvrez le sketch “Leçon 4 Digital Inputs” puis téléversez le code sur la carte UNO R3 comme présenté lors de la leçon 2.

Presser le bouton de gauche allumera la LED, presser le bouton de droite l’éteindra.

La première partie du sketch est consacrée à la définition des 3 variables pour les 3 pins qui seront nécessaires au fonctionnement du montage.

Dans le bloc « setup », les pins sont affectées en tant qu’entrées ou sorties. ‘LEDPin’ est défini en tant que sortie (OUTPUT), ‘buttonApin’ et ‘buttonBpin’ en tant qu’entrées (INPUT).

Button.ino

```
1   const int buttonPin = 2;
2   const int ledPin = 13;
3
4   int buttonState = 0;
5
6   void setup() {
7
8     pinMode(ledPin, OUTPUT);
9
10    pinMode(buttonPin, INPUT);
11  }
12
13  void loop() {
14
15    buttonState = digitalRead(buttonPin);
16
17
18    if (buttonState == HIGH) {
19
20      digitalWrite(ledPin, HIGH);
21    } else {
22
23      digitalWrite(ledPin, LOW);
24    }
25  }
```

fig 4.2 : digital inputs_ code câblage

```
pinMode(buttonApin, INPUT_PULLUP);
pinMode(buttonBpin, INPUT_PULLUP);
```

Petite particularité, les entrées sont définies en tant qu'entrées avec résistance de PULLUP. Pourquoi cette subtilité ? Comme vous pouvez le constater sur le schéma de câblage, le fait de presser un bouton met la pin associée à la masse. Mais lorsque le bouton n'est pas pressé, il peut subsister des signaux parasites qui peuvent être interprétés par la carte comme une mise à la masse alors que le bouton n'est pas pressé. Pour éviter ces parasites, on utilise la technique dite de « la résistance de « pullup » qui permet de forcer un état haut via une résistance connectée à une borne +. La carte UNO est capable de simuler ce branchement, pour cela, il suffit de déclarer la pin en INPUT_PULLUP au lieu de faire une déclaration INPUT simple.

Le bloc loop prend ensuite la mesure de la position de chaque bouton:

```
 void loop()
{
if (digitalRead(buttonApin) == LOW)
{
digitalWrite(LEDPin, HIGH);
}
if (digitalRead(buttonBpin) == LOW)
{
digitalWrite(LEDPin, LOW); }
}
```

On regarde si le bouton A est pressé (LOW). Si oui, on allume la LED, si non, on ne fait rien.

On regarde ensuite si le bouton B est pressé. Si oui on éteint la LED, si non, on ne fait rien.

Illustration

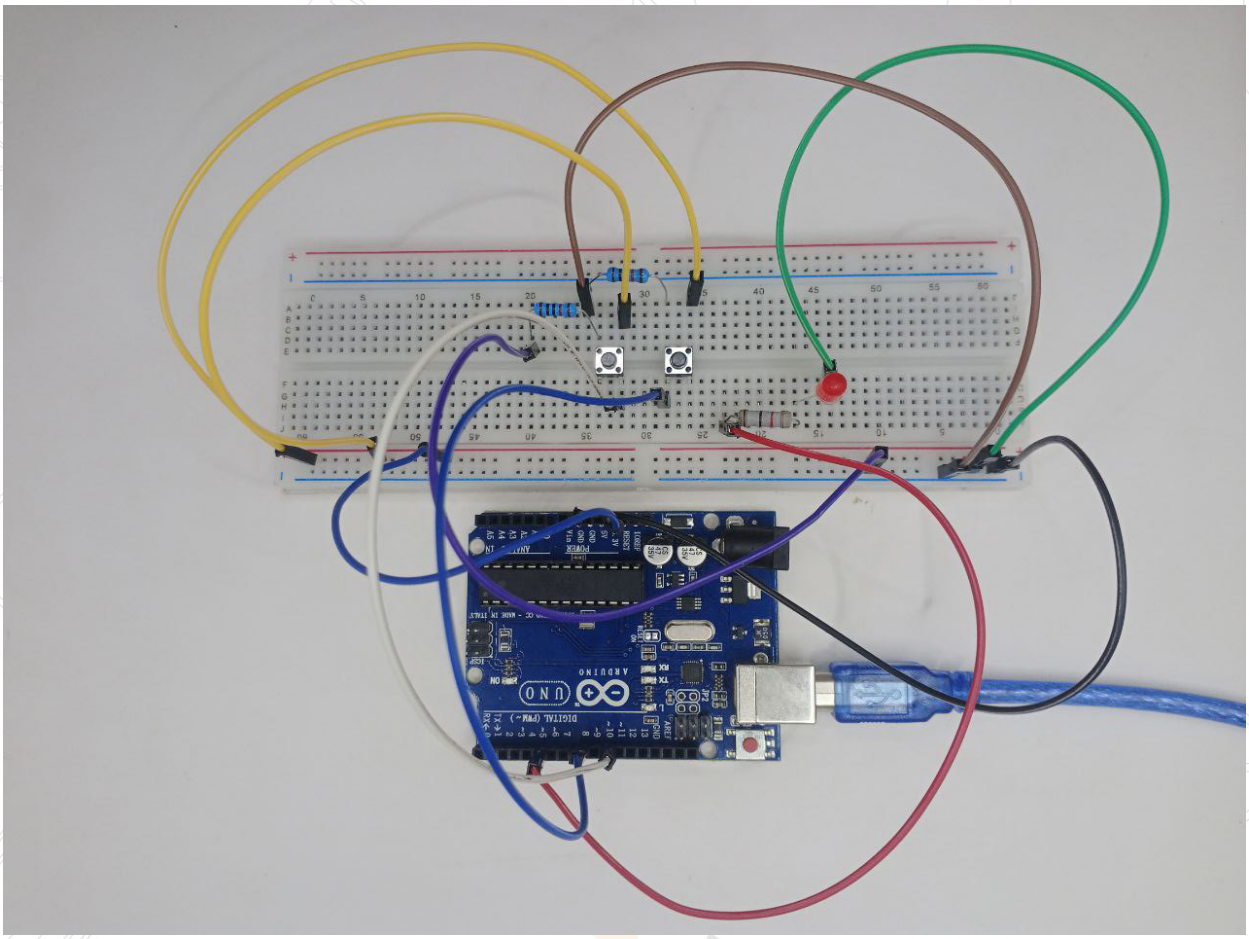


fig 4.3 : digital inputs_ illustration

Leçon 5

Active buzzer

But de la leçon

Dans cette leçon, vous allez apprendre à générer des sons avec un « active buzzer ».

Matériel nécessaire:

(1) x Arduino Uno R3
(1) x Active buzzer
(2) x Câbles Mâle-Femelle

fig 5.1 : Buzzer

Présentation du composant

BUZZER:

Les buzzers électroniques sont alimentés en courant continu et équipés de circuits intégrés. Ils sont largement utilisés dans les ordinateurs, imprimantes, alarmes, jouets etc.... Il en existe deux types : les actifs et les passifs. Placez le buzzer avec les pattes vers le haut. Celui où vous pouvez distinguer un petit circuit généralement vert est un buzzer passif.

La différence entre les deux réside dans le fait qu'un buzzer actif possède un oscillateur intégré, il va donc générer un son lorsque le courant passe. Un buzzer passif utilise un signal en entrée pour générer le son (généralement signal carré en 2kHz et 5kHz).

Diagramme de câblage

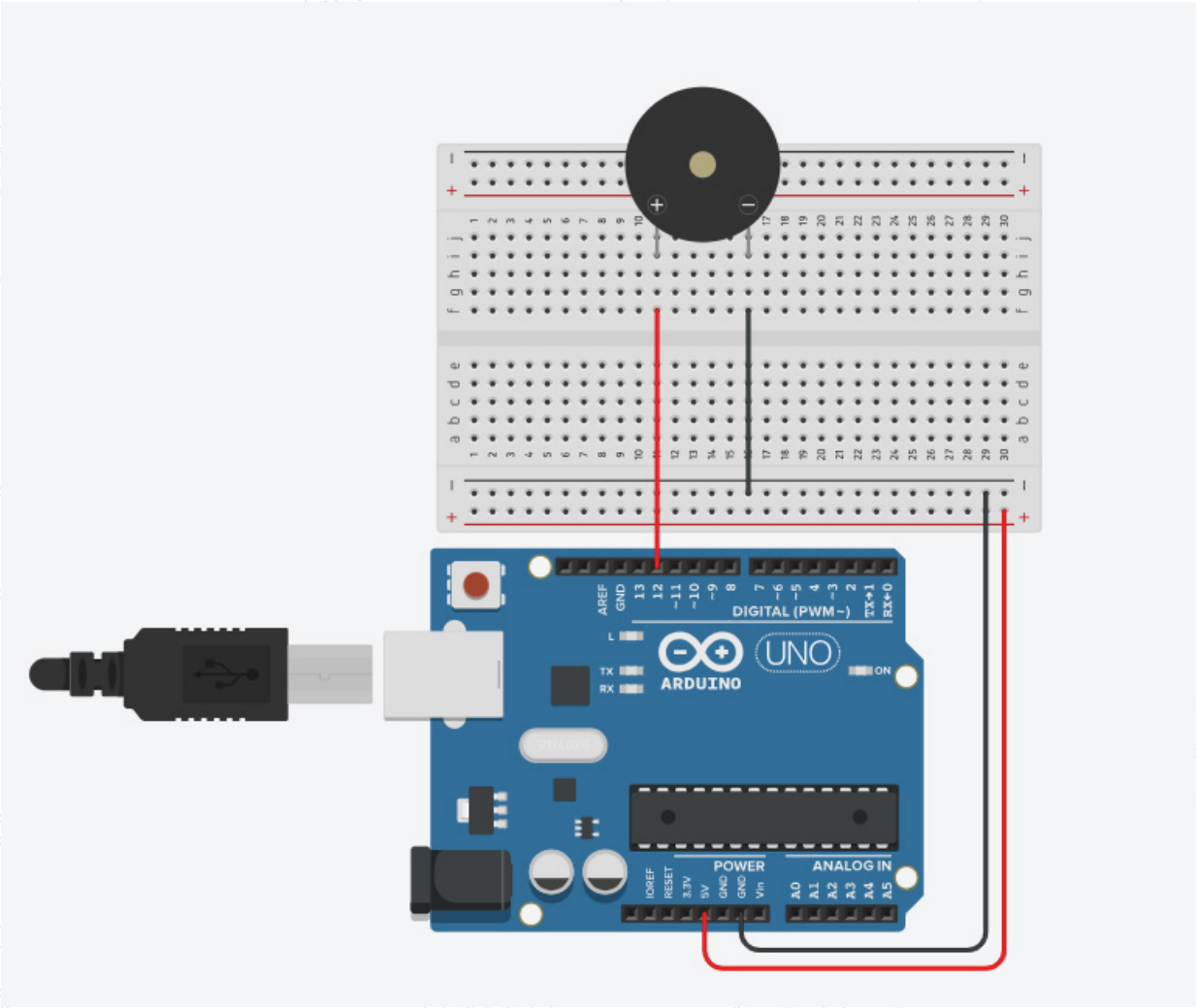


fig 5.2 : Buzzer_diagramme de câblage

Code

Activebuzzer.ino

```
1   const int buzzer = 9; //buzzer pin 9
2   void setup(){
3     pinMode(buzzer, OUTPUT); // mettez buzzer pin en sortie
4   }
5   void loop(){
6     tone(buzzer, 1000); // 1KHz pour le signal du son
7     delay(1000);        // attend 1 sec
8     noTone(buzzer);     // coupe le son
9     delay(1000);        // attend 1sec
10  }
```

fig 5.3 : Buzzer_code

Illustration

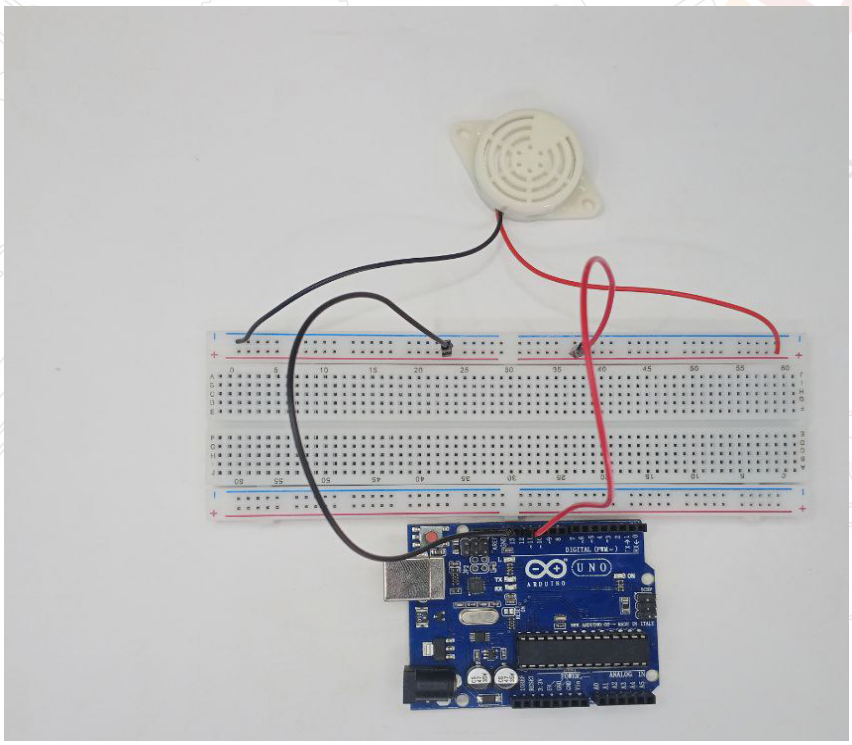


fig 5.4 : Buzzer_illustration

Leçon 6

Tilt Ball Switch

But de la leçon

Dans cette leçon, vous allez apprendre comment il est possible de détecter une inclinaison avec le capteur “tilt ball switch”.

Matériel nécessaire:

(1) x Arduino Uno R3

(1) x capteur Tilt Ball switch

(2) x Câbles Mâle-Femelle

Présentation du composant

fig 6.1 : Tilt ball switch

Capteur « Tilt » :

Ce capteur vous permet de détecter une orientation ou inclinaison. Il est petit, pas cher est consomme très peu d’énergie.

Sa simplicité fait qu’il est très répandu pour les jouets et toutes sortes de gadgets. Vous le trouverez sous différents types et noms : «mercury switches», «tilt switches» ou «rolling ball sensors».

Ces capteurs sont généralement faits d’une cavité cylindrique dans laquelle se trouve une masse libre de mercure par exemple. Au bout de la cavité, deux éléments conducteurs non reliés. Lorsque la cavité change d’orientation ou se retourne, la masse libre vient sur en contact avec les deux éléments conducteur et crée un contact électrique qui laisse passer le courant.

Alors qu’ils ne sont pas aussi précis qu’un accéléromètre, ils sont parfaits pour faire de la détection de mouvement ou d’orientation.

Diagramme de câblage

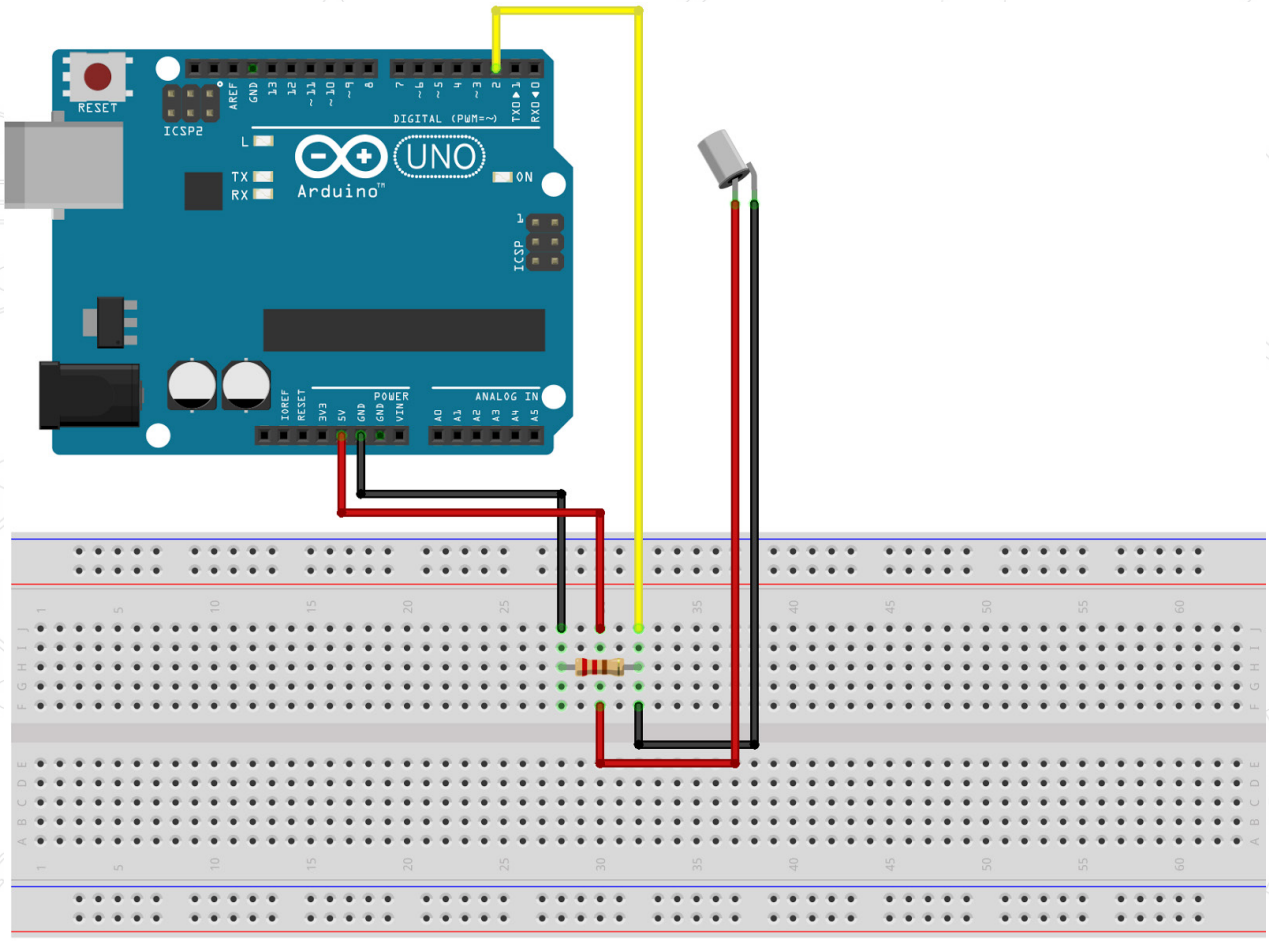


fig 6.2 : Tilt ball switch_ diagramme de câblage

Code

Tilt_Ball_Switch.ino

```ino
// Déclaration de la broche où est connecté le tilt switch
const int tiltSwitchPin = 2;  // Broche numérique D2
// Variable pour stocker l'état du switch
int tiltState = 0;
void setup() {
  // Initialisation de la broche tiltSwitchPin comme entrée
  pinMode(tiltSwitchPin, INPUT);
  // Initialisation de la communication série pour afficher le résultat
  Serial.begin(9600);
}
void loop() {
  // Lecture de l'état du tilt switch (0 ou 1)
  tiltState = digitalRead(tiltSwitchPin);
  // Affichage de l'état sur le moniteur série
  if (tiltState == HIGH) {
    Serial.println("Le capteur est fermé (basculé).");
  } else {
    Serial.println("Le capteur est ouvert (non basculé).");
  }
  // Petite pause pour éviter la saturation du moniteur série
  delay(500);
}

```

fig 6.3 : Tilt ball switch_code

Illustration

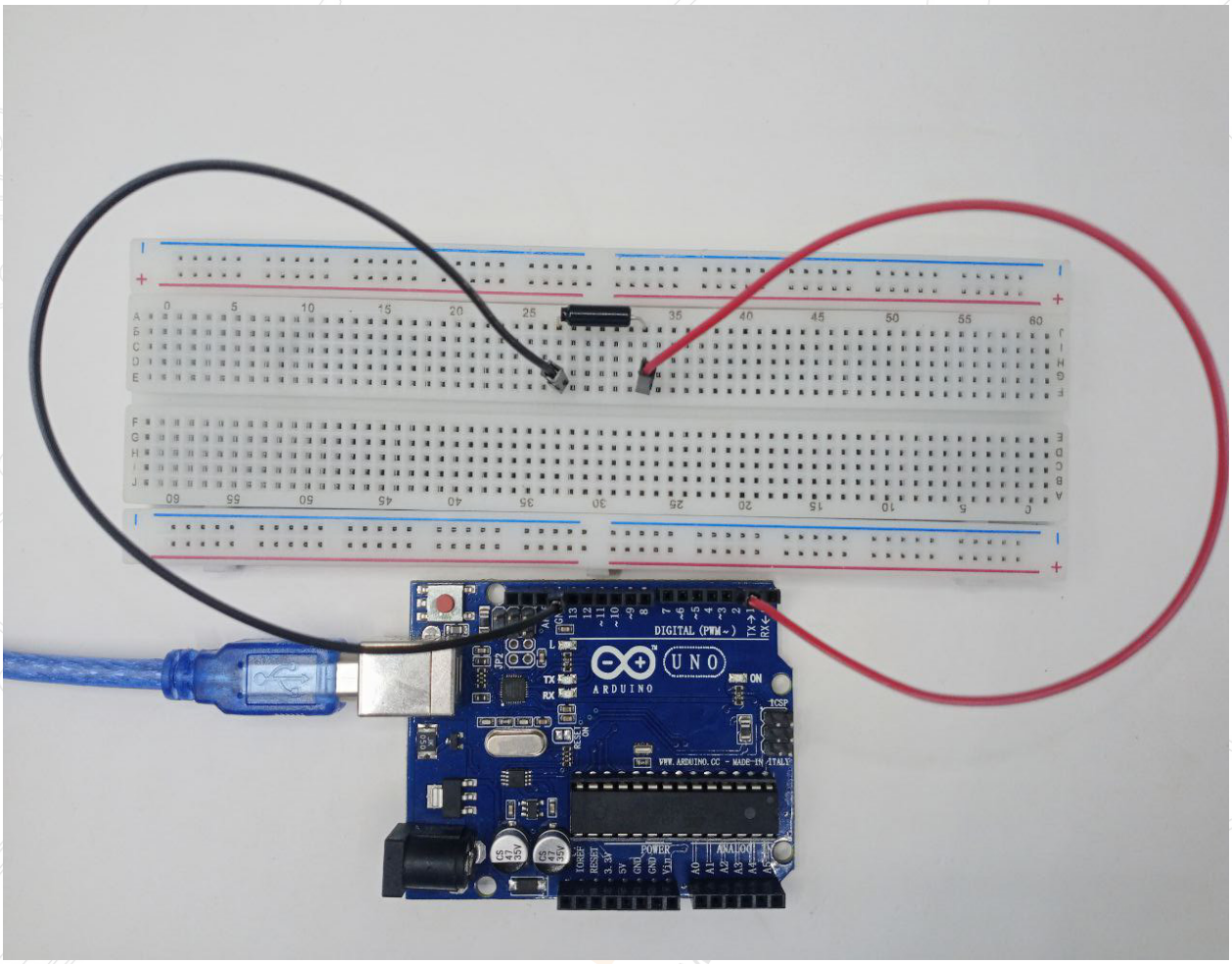


fig 6.4 : Tilt ball switch_ illustration

Leçon 7

Eight LED with 74HC595

But de la leçon

Dans cette leçon, vous allez apprendre comment il est possible de commander indépendamment 8 LEDs sans avoir besoin de mobiliser 8 pins sur la carte UNO R3. Il est bien évidemment possible de connecter 8 LEDs sur 8 pins de votre carte UNO R3 si vous n'avez pas besoin de beaucoup d'autres connexions pour votre projet, cela fonctionnera parfaitement, mais il est vrai que la plupart du temps, il est nécessaire de brancher tout un tas de boutons, capteurs et donc les pins deviennent rapidement pas assez nombreuses. C'est pour cela que nous allons voir comment interfacer les LEDs à la carte via une puce 74HC595 qui est un convertisseur Série / Parallèle. Cette puce dispose de 3 entrées et 8 sorties. Le recours à cette puce ralentira un peu les possibilités de switch on-off (de 800000 à 500000 par seconde), mais cela reste une fréquence plus que raisonnable.

Matériel nécessaire:

(1) x Arduino Uno R3
(1) x planche de prototypage
(8) x LEDs
(8) x résistances 220 ohm
(1) x puce 74hc595
(14) x Câbles mâle-mâle

fig 7.1 : Eight LED with 74HC595

Présentation du composant

74HC595 Shift Register:

Le “shift register” est un type de puce qui contient un nombre de registres mémoire qui peuvent prendre soit la valeur 0, soit la valeur 1. Il est possible de changer chaque valeur indépendamment.

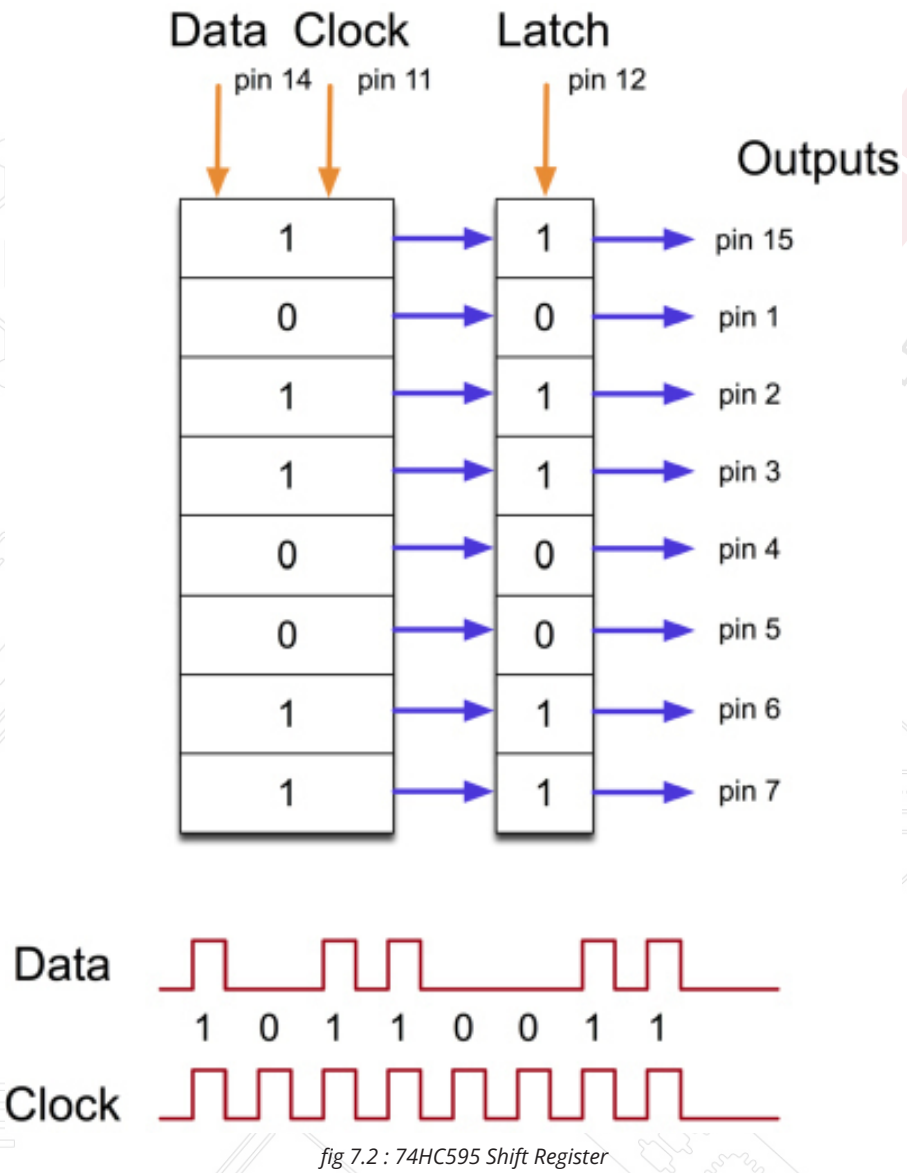


fig 7.2 : 74HC595 Shift Register

Pour cette puce, l’horloge fonctionne avec 8 pulsations. A chaque pulsation correspond une adresse registre. Si la pin est à l’état haut, le registre prendra la valeur 1, s’il est à l’état bas, il prendra la valeur 0.

Diagramme de câblage

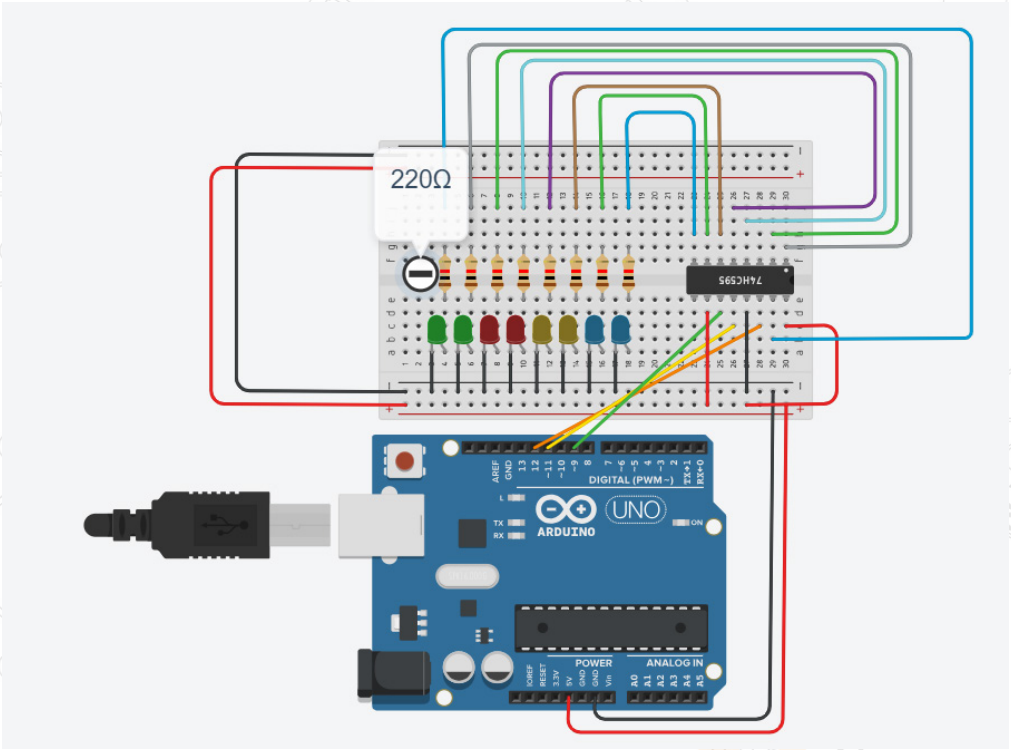


fig 7.3 : Eight LED with 74HC595_ diagramme de câblage

Ce schéma est assez complexe et demande d'être fait avec soin.

Le mieux étant de commencer par la mise en place de la puce 74HC595. Toutes les LEDs sauf une sont sur le même côté de la puce.

Après avoir mis la puce en place installez les résistances.

Placez ensuite les différentes LEDs. Faites bien attention au sens des LEDs la plus longue devait être du côté de la borne positive du montage.

Code

EightLEDwith74HC595.ino

```
// définition des pins de commande du 74HC595
int DATA_pin = 2;   //  définition du pin DATA
int LATCH_pin = 4; // définition du LATCH
int CLOCK_pin = 3;// définition du CLOCK
boolean etats[8]; //c'est un tableau de 8 bits correspondant
                  //aux sorties du 74HC595 à l'état voulu
void setup()
{
    //configuration des pins de commande comme sorties
    pinMode(DATA_pin,OUTPUT);
    pinMode(LATCH_pin,OUTPUT);
    pinMode(CLOCK_pin,OUTPUT);
}
//La fonction appliqueEtat() enregistre l'état des leds  dans le CI 74HC595
//pour chaque état, le LATCH est d'abord déactivé
//ensuite après que le CLOCK est mis à l'état LOW
//on place le DATA à l'état voulu et l'activation  du CLOCK opère le décalage
//à la fin on valide le tout en activant le LATCH
void appliqueEtat()
{
  digitalWrite(LATCH_pin, LOW);
  for (int i = 7; i>=0; i--)
  {
    digitalWrite(CLOCK_pin, LOW);
    digitalWrite(CLOCK_pin, LOW);
    digitalWrite(DATA_pin, etats[i] );
    digitalWrite(CLOCK_pin, HIGH);
  }
  digitalWrite(LATCH_pin, HIGH);
}
void loop()
```

fig 7.3 : Eight LED with 74HC595_ code

Illustration

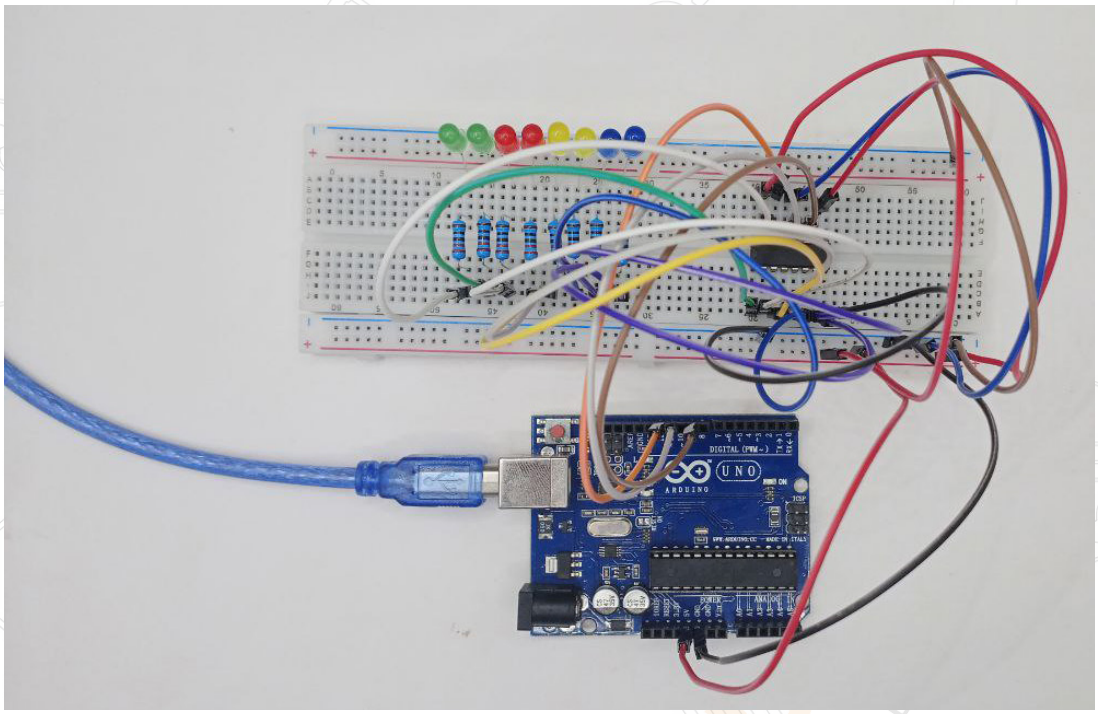


fig 7.4 : Eight LED with 74HC595_illustration

Leçon 8

Photocell

But de la leçon

Dans cette leçon, vous allez apprendre à mesurer l'intensité de la lumière en utilisant une entrée analogique. Vous allez utiliser le montage réalisé à la leçon précédente et utiliser vos nouvelles connaissances pour contrôler le nombre de LEDs allumées.

Matériel nécessaire:

(1) x Arduino Uno R3
(1) x Planche de prototypage
(8) x Résis220 ohm resistors
(1) x 1k ohm resistor
(1) x Photorésistance (Photocell)
(16) x Câbles mâle-mâle

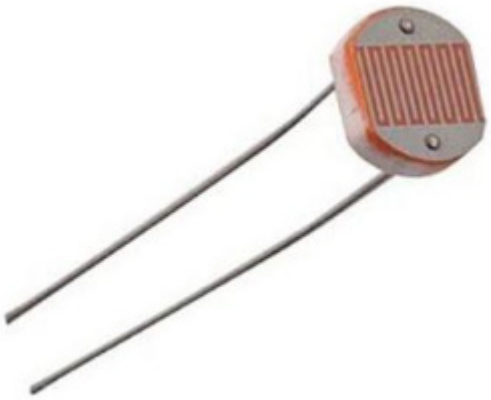


fig 8.1 : Photocell

Présentation du composant

La photocell utilisée est appelée "résistance dépendante de la lumière" ou light dependent resistor (LDR). Comme son nom l'indique la valeur de la résistance dépend de la quantité de lumière reçue par le composant.
Cette résistance prend une valeur de 50 kΩ en obscurité et varie jusqu'à 500 Ω en pleine lumière. Pour convertir cette variation de valeur en variation de courant, il faut bien entendu alimenter la résistance.

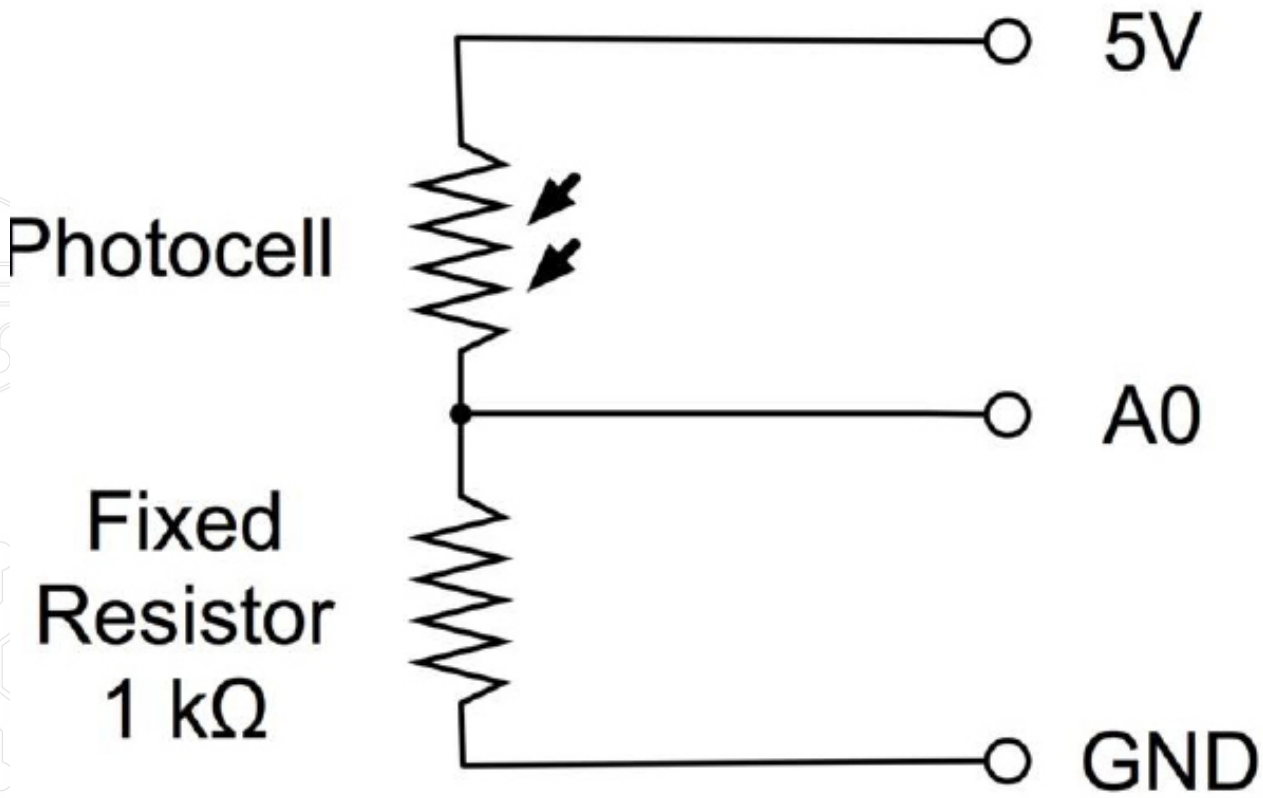


fig 8.2 : Photocell_1

Diagramme de câblage

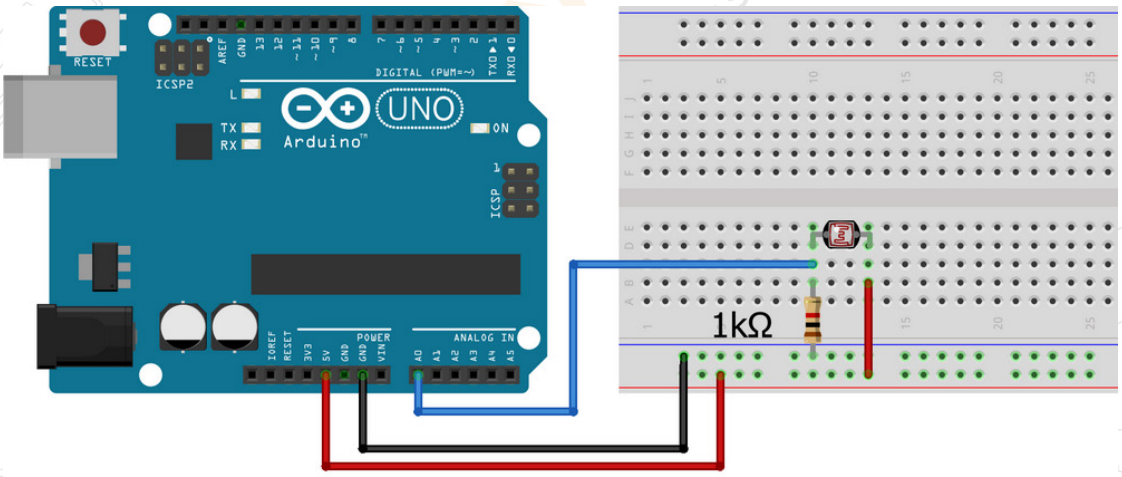


fig 8.3 : Photocell_diagramme de câblage

Illustration

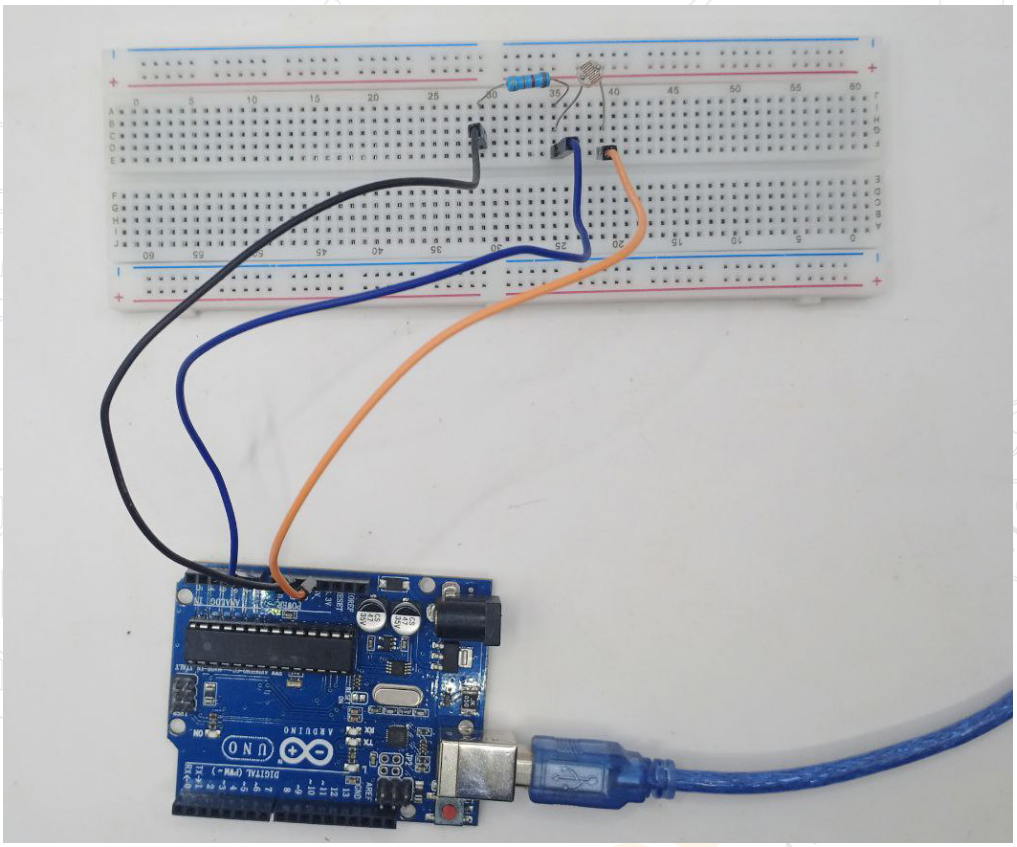


fig 8.4 : Photocell_illustration

Code

Photocell.ino

```
int ldr = A0;// définition de la photorésistance sur le pin A0
int valeur =0;//déclaration de la variable stockant le signal numérique de la photorésistance
const int led = 2;//définition de la led sur le pin 2
void setup() {
  Serial.begin(9600);
  pinMode(led,OUTPUT);//définition de la led comme sortie
  digitalWrite(led, LOW);
}
void loop() {
  valeur = analogRead(ldr);//lecture et stockage du signal numérique
  //Serial.println("la valeur de la LDR est:");
  //Serial.println(valeur);
  delay(1000);
  if(valeur<=300)//allumage de la led dans l'obscurité
  {
    digitalWrite(led,HIGH);
    //Serial.println("LDR DARK, la led s'allume");
  }
  else
  digitalWrite(led, LOW);//extinction de la led en présence de lumière
}
```

fig 8.1 : Photocell_code

Leçon 9

Servomoteur

But de la leçon

Le servomoteur est un type de moteur qui peut seulement tourner de 180 degrés. Il est contrôlé par l'émission de pulsations électriques depuis la carte UNO R3. La pulsation donne au moteur la position. qu'il doit prendre. Le moteur a trois connexions. Le fil brun (masse), le fil rouge (positif), le fil orange (signal à connecter sur la pin 9 de la carte UNO R3)

Matériel nécessaire:

(1) x Arduino Uno R3

(1) x Servo (SG90)

(3)x Câbles mâle-mâle

fig 9.1 : Servomoteur

Présentation du composant

· Universel pour connecteur JR et FP

· Longueur de câble : 25cm

· Pas de charge; Vitesse de fonctionnement: 0,12 s / 60 degrés (4,8 V), 0,10 s / 60 degrés (6,0 V)

· Couple (à 4.8V): 1.6kg/cm

· Température : -30~60'C

· Largeur de bande morte: 5us

· Voltage: 3.5~6V

· Dimensions : 3.2 cm x 3 cm x 1.2 cm

· Poids : 134 g

Diagramme de câblage

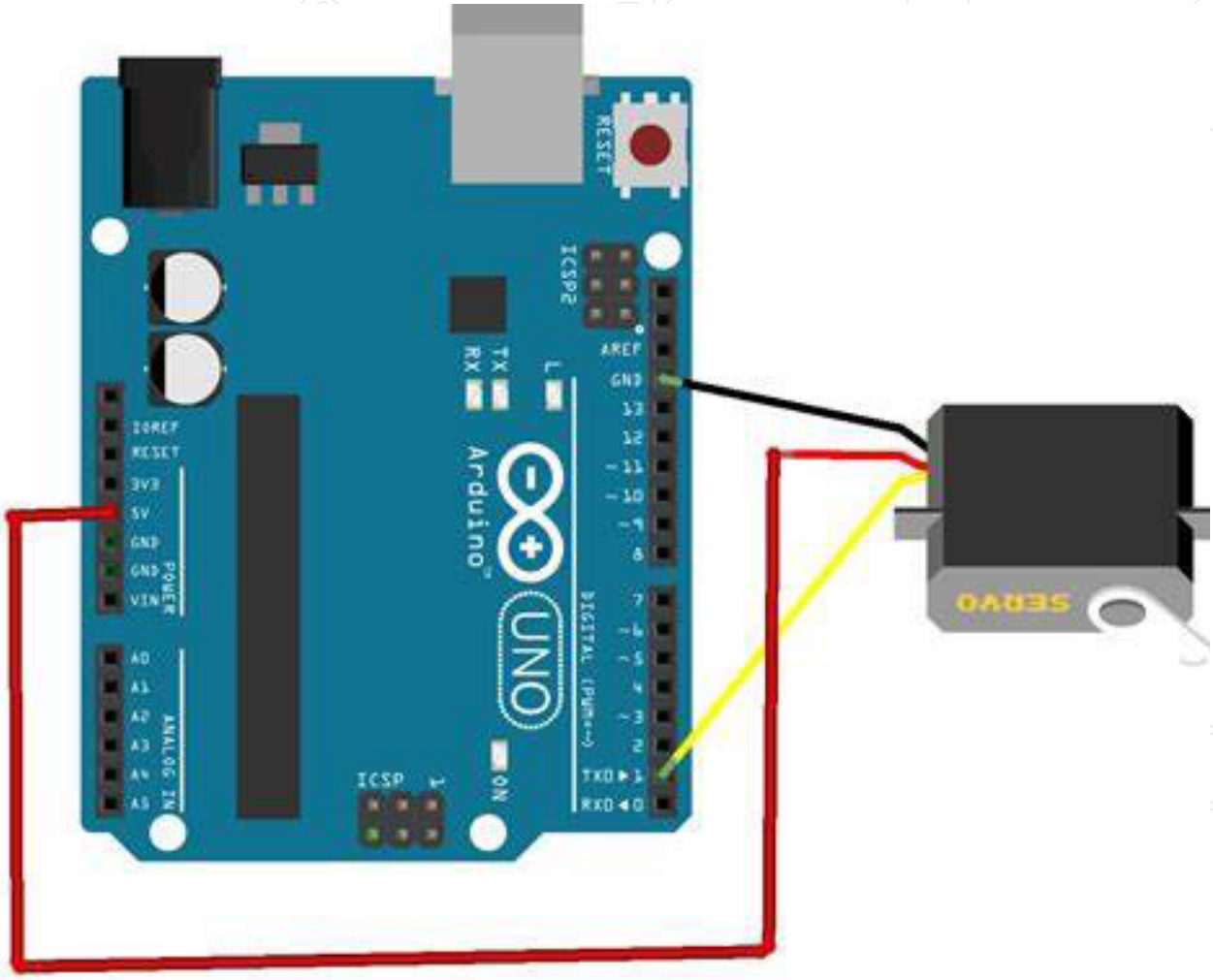


fig 9.2 : Servomoteur_diagramme de câblage

Illustration

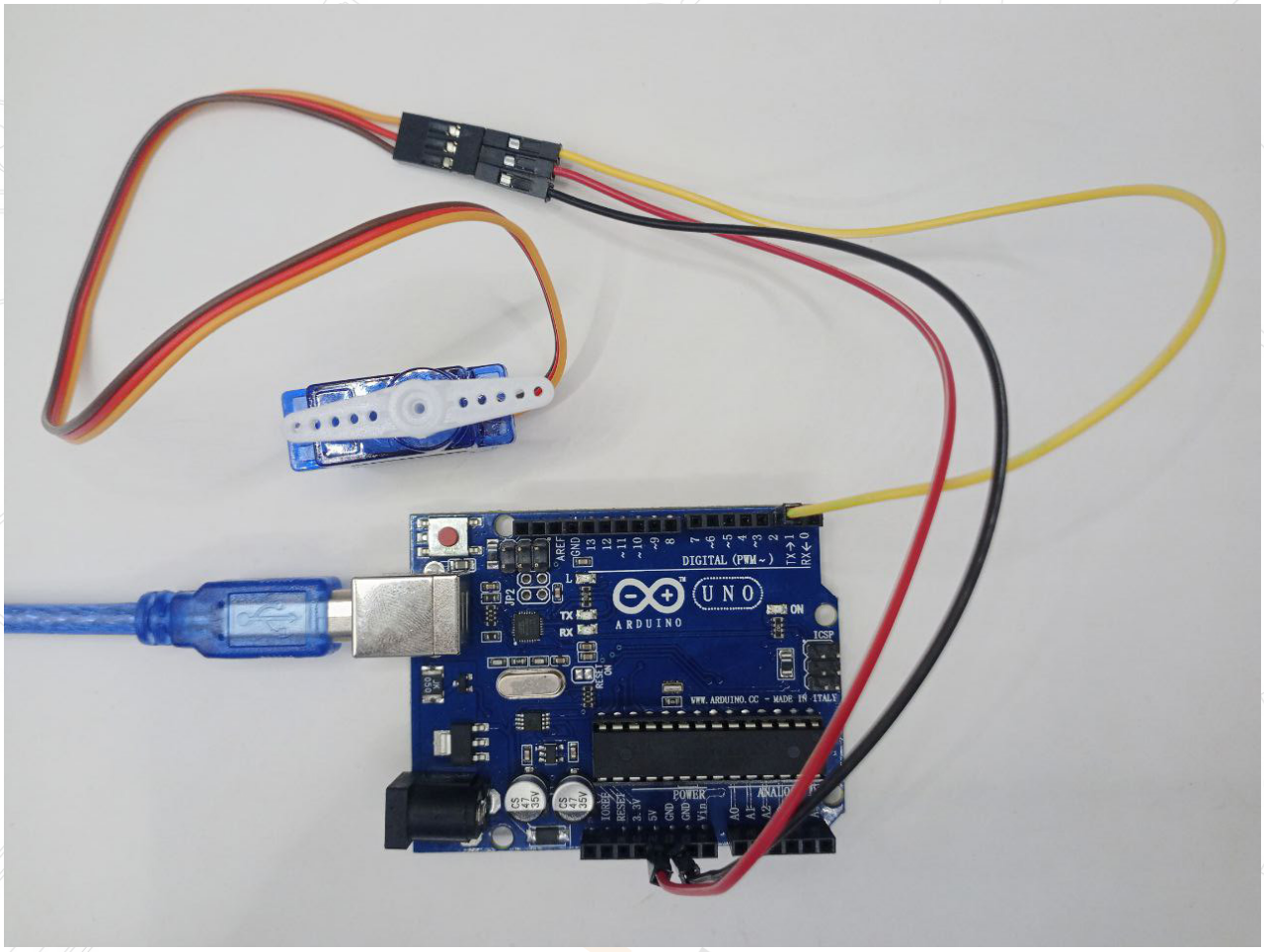


fig 9.3 : Servomoteur_illustration

Code

Servomoteurr.ino

```
/*Guider un servomoteur avec un potetiomètre*/
#include <Servo.h> //on inclut    la bibliothèque servo.h
Servo mon_moteur;  // on crée l'objet mon_moteur
int pot_entree = A0;  // on met l'entrée du potentiomètre sur le pin A0;
int val;    // variable qui lira la valeur envoyée par le potentiomètre
void setup() {
  mon_moteur.attach(8);  //on appelle la fonction attache de l'ojet
  //mon_moteur et on lui envoie en paramètre le pin du servomoteur
}
void loop() {
  val = analogRead(pot_entree);// on lit la valeur du potentiomètre (qui se trouve entre 0 et 1023)
  val = map(val, 0, 1023, 0, 180);// on transforme la valeur en un angle
  mon_moteur.write(val);// on envoie la valeur val à la fonction
  delay(15);// on attend le temps que le servomoteur recoive l'instruction
}
```

fig 9.4 : Servomoteur_code

Leçon 10

Capteur à ultrasons

But de la leçon

Le capteur à ultrasons est idéal pour tous les projets nécessitant des mesures de distance, comme l'évitement d'obstacles. Le HC-SR04 est un produit économique et simple à utiliser..

Matériel nécessaire:

(1) x Arduino Uno R3

(1) x Capteur Ultrasons

(4) x Câbles Mâle-Femelle

fig 10.1 : Capteur à ultrasons

Présentation du composant

Ultrasonic

Le capteur permet une mesure d'un objet situé à une distance allant de 2cm à 4m et fourni une mesure avec une précision de 3mm.

Principe de fonctionnement:

(1) Un signal de déclenchement (trigger) est émis pendant 10µs

(2) Le module se met à l'écoute d'un signal de retour

(3) Le temps entre l'émission et la réception est le temps nécessaire au signal pour faire l'aller et le retour vers l'objet qui a réfléchi celui-ci.

Distance de test = (temps de niveau élevé × vitesse du son (340 m / s) / 2

Le chronogramme est illustré ci-dessous. Il vous suffit de fournir une courte impulsion de 10us au déclencheur entrée pour démarrer la télémétrie, puis le module enverra une rafale d'ultrasons de 8 cycles à 40 kHz et augmenter son écho. La largeur d'impulsion d'écho est proportionnelle à la distance de l'objet, ou gamme. Vous pouvez calculer l'intervalle de temps entre l'envoi du signal de déclenchement et la réception signal d'écho. Formule: us / 58 = centimètres ou us / 148 = inch; ou: la plage = temps de niveau élevé * vitesse (340M / S) / 2; nous suggérons d'utiliser une mesure de plus de 60 ms cycle, afin d'empêcher le signal de déclenchement du signal d'écho.

Diagramme de câblage

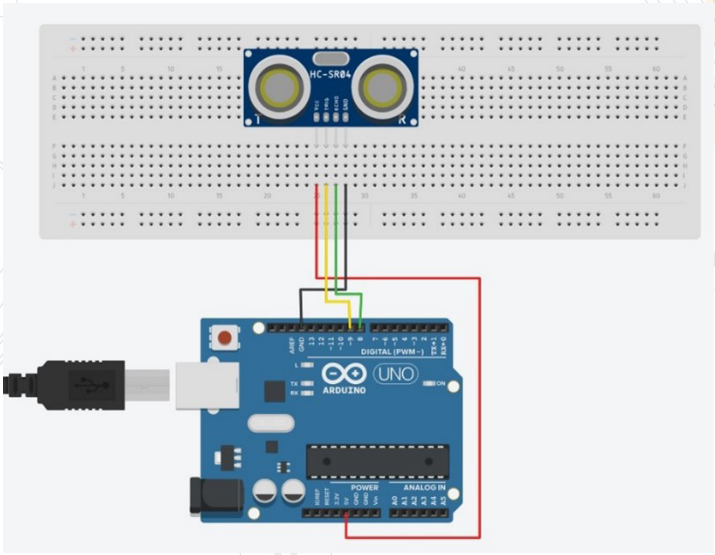


fig 10.2 : Capteur à ultrasons_ schéma de câblage

Illustration

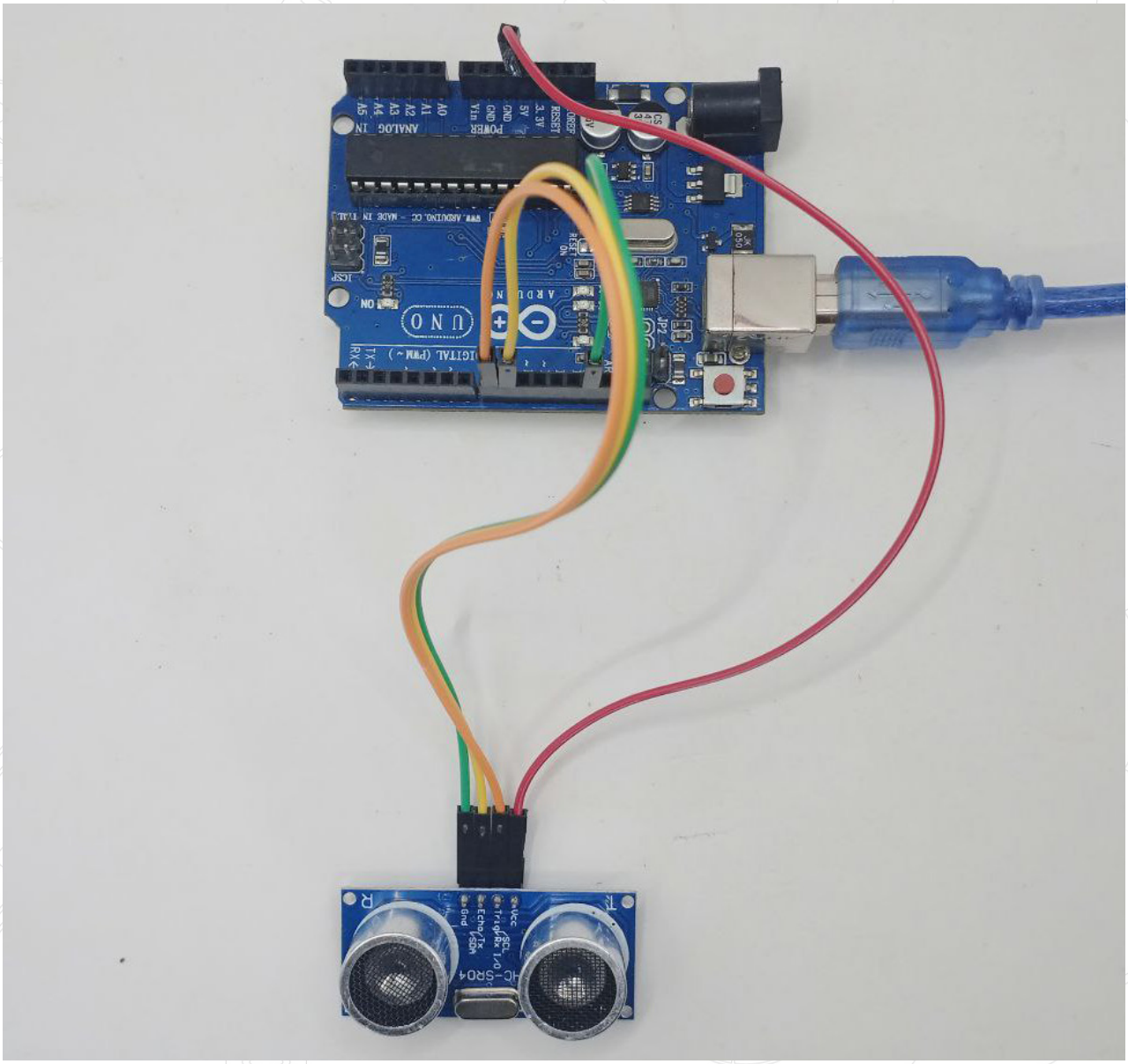


fig 10.3 : Capteur à ultrasons_illustration

Code

CpateurUltrason.ino

```
1   // définition des pins
2   const int trigPin = 9;
3   const int echoPin = 10;
4
5   // definition des variables
6   long duration;
7   int distance;
8   void setup() {
9
10      pinMode(trigPin, OUTPUT); // Paramétrage du trigger comme sortie
11      pinMode(echoPin, INPUT); // Paramétrage de l'écho comme entrée
12      Serial.begin(9600);
13  }
14  void loop() {
15    // Efface le trigger
16    digitalWrite(trigPin, LOW);
17    delayMicroseconds(2);
18    // Mise du trigger sur high pendant 10 microsecondes
19    digitalWrite(trigPin, HIGH);
20    delayMicroseconds(10);
21    digitalWrite(trigPin, LOW);
22    // l'écho renvoie le temps du trajet de l'onde sonore en microsecondes
23    duration = pulseIn(echoPin, HIGH);
24    // Calcul de la distance
25    distance= duration*0.034/2;
26   // Affichage de la distance dans le moniteur série
27   Serial.print("Distance: ");
28   Serial.print(distance);
29   Serial.println("cm");
30  }
```

fig 10.4 : Capteur à ultrasons_code

Leçon 11

Capteur de température et d'humidité DHT11

But de la leçon

Dans cette leçon, vous allez apprendre à utiliser le capteur DHT11 qui permet une mesure de température et d'humidité. C'est un capteur fiable qui permettra de répondre à la plupart des projets nécessitant un suivi de température et/ou d'humidité. L'emploi d'une bibliothèque permet en plus une implémentation simple du code dans vos sketchs.

Matériel nécessaire:

(1) x Arduino Uno R3
(1) x DHT11 Temperature and Humidity module
(3) x Câbles Mâle-Femelle

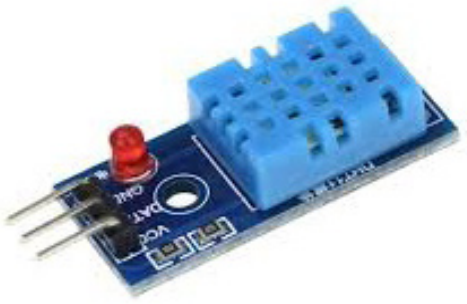


fig 11.1 : DHT11

Présentation du composant

DHT11

Le DHT11 (Digitale Température Humidité 11) est un capteur qui génère un signal digital en sortie codant une valeur de température et d'humidité mesurée en temps réel. Cette technologie est employée car elle permet de produire une mesure fiable sur le long terme. Il contient en plus de ces capteurs un microcontrôleur 8-bit.

Encore une fois, il est utilisé dans un grand nombre d'applications qui nécessitent de moduler le fonctionnement d'un appareil en fonction de conditions de température ou d'humidité.

Paramètres du produit:

Alimentation : 5V

Consommation : 0.5 mA en nominal / 2.5 mA maximum

Etendue de mesure température : 0°C à 50°C ± 2°C

Etendue de mesure humidité : 20-90%RH ±5%RH

Taille :15 mm x 12 mm x 5,5 mm

Poids : 3 g

Description de la broche:

1.l'alimentation VDD 3,5 5,5 V CC.

2.Données série DATA, un seul bus.

3. Masse GND, la puissance négative.

Diagramme de câblage

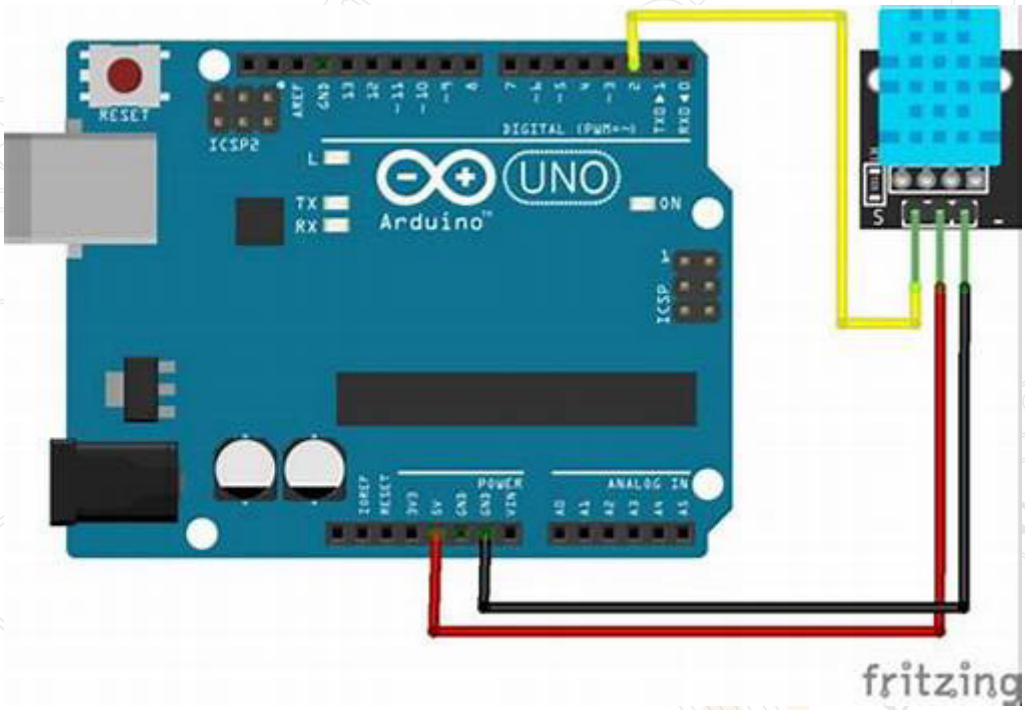


fig 11.2 : DHT11_ diagramme

Illustration

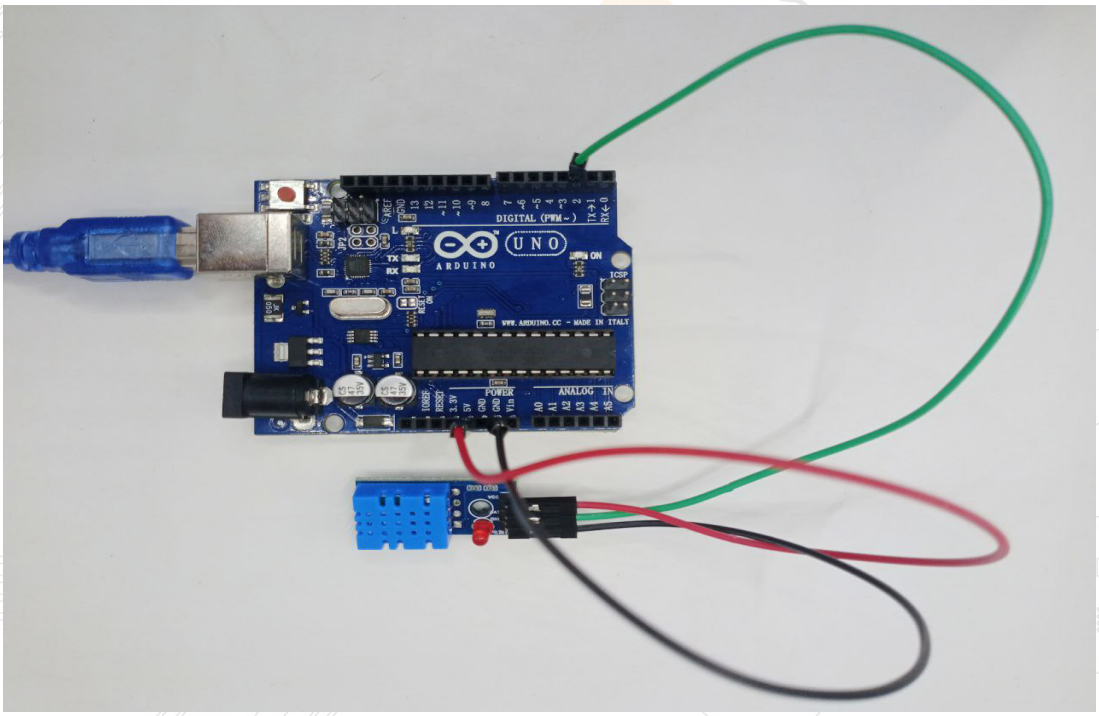


fig 11.3 : DHT11 illustratuio,

Code

DHT11.ino

```
#include <DHT.h>
// Définir le type de capteur (DHT11)
#define DHTTYPE DHT11
// Définir la broche de connexion du capteur (D2)
#define DHTPIN 2
// Créer une instance du capteur DHT
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  // Initialiser la communication série pour l'affichage dans le moniteur série
  Serial.begin(9600);
  // Initialiser le capteur DHT
  dht.begin();
  // Message d'accueil dans le moniteur série
  Serial.println("Démarrage du capteur de température et d'humidité DHT11");
}
void loop() {
  // Attendre un peu avant de lire les données (délai recommandé entre les mesures)
  delay(2000);
  // Lire l'humidité
  float humidity = dht.readHumidity();
  // Lire la température en Celsius
  float temperature = dht.readTemperature();
  // Vérifier si la lecture est valide
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Erreur de lecture du capteur DHT11 !");
    return;
  }
```

fig 11.4 : DHT11_ code

Leçon 12

CLAVIER MATRICIEL 4X4

But de la leçon

Dans cette leçon, vous allez apprendre comment il est possible d'intégrer un clavier à vos projets de sorte que la carte UNO R3 puisse lire les entrées utilisateur faites avec le périphérique.

Les touches de claviers sont utilisées dans un grand nombre d'applications (téléphones / fax / fours / digicodes de portes etc).

Pour ce projet nous allons utiliser un « matrix keypad ». C'est un clavier qui est conçu de telle sorte qu'il nécessite moins de pins de connexion qu'il a de touches. En effet, nous avons 17 touches (0-9, A-D, *, #) et 8 pins. Avec un clavier linéaire il faudrait 17 pins (une par touche) pour fonctionner. L'encodage par bit permet de combiner les pins pour en avoir besoin de beaucoup moins (ici deux fois moins)

Matériel nécessaire:

(1) x Arduino Uno R3

(1) x" Membrane switch module"

(8) x Câbles mâle-mâle

fig 12.1 : clavier matriciel 4x4

Diagramme de câblage

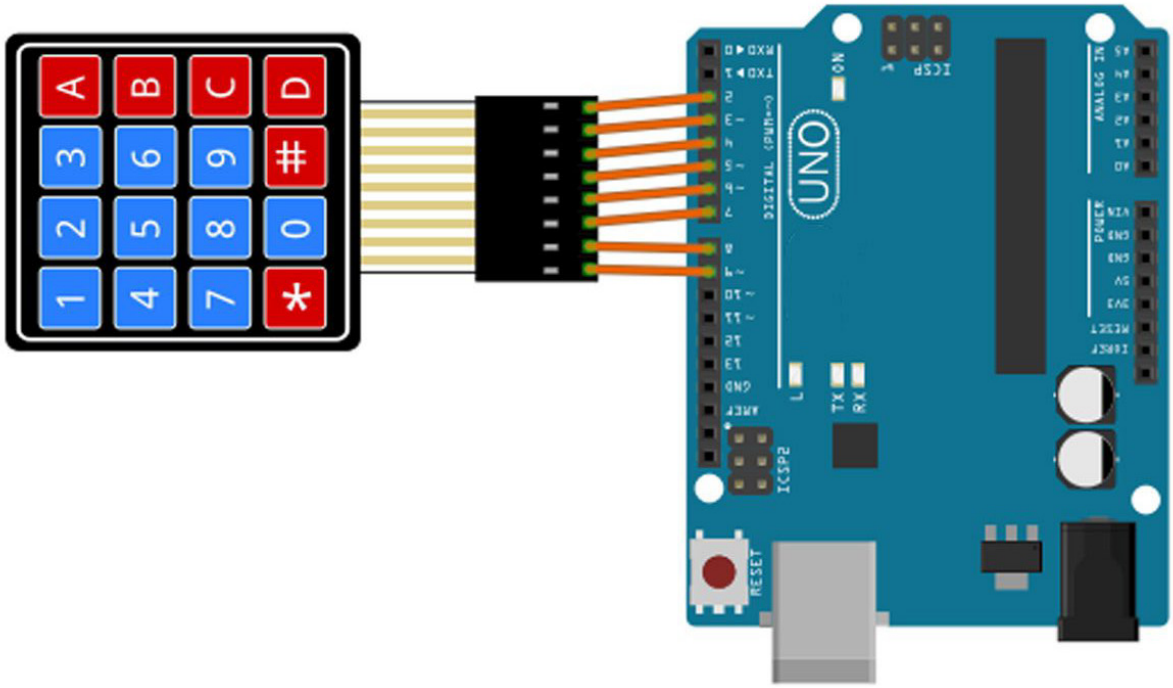


fig 12.2 : clavier matriciel 4x4_diagramme

Illustration

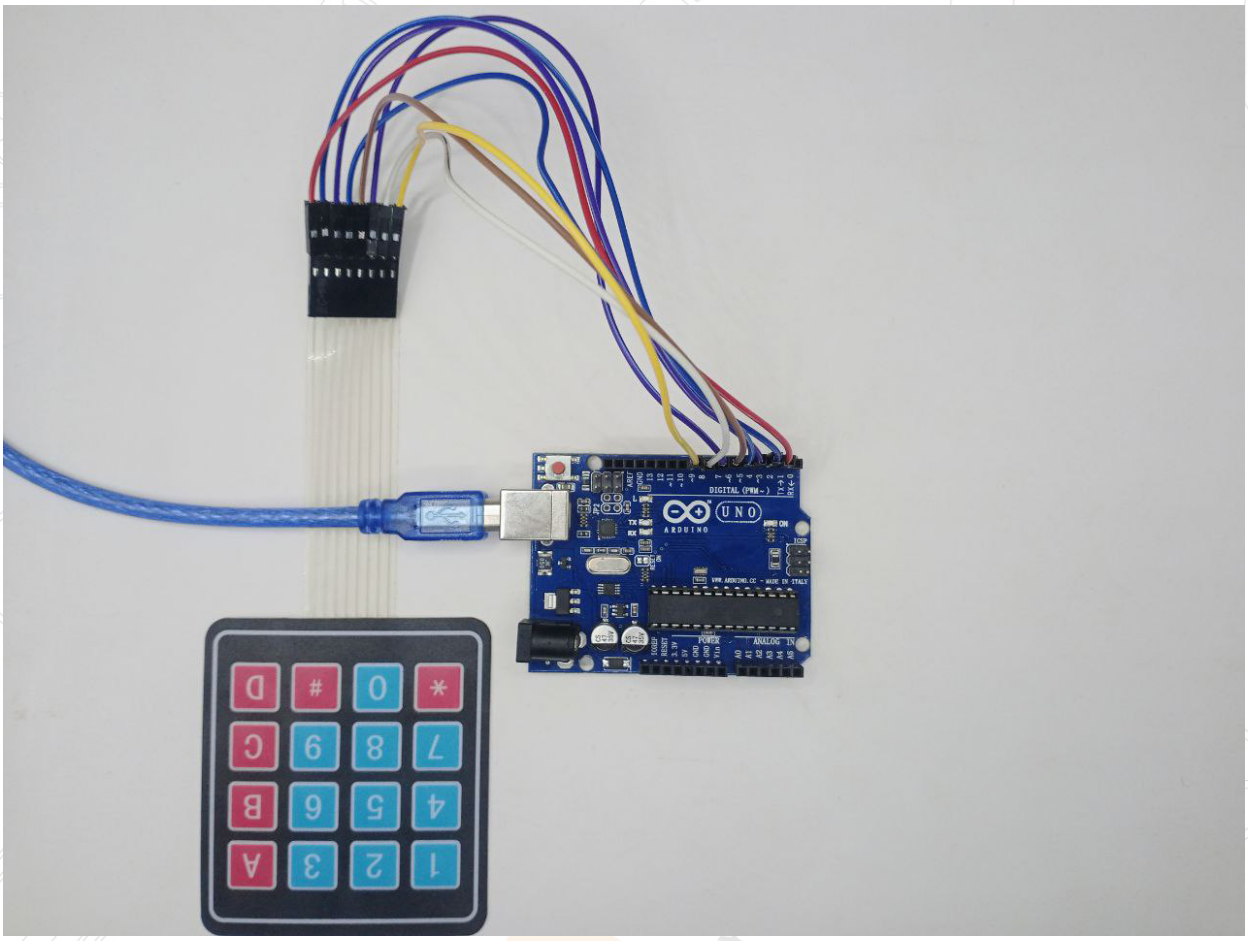


fig 12.3 : clavier matriciel 4x4_illustration

Code

ClavierMatriciel44.ino

```
#include <Keypad.h> // ajout de la bibliothèque au fichier
const byte LIGNES = 4; // nombre de lignes du clavier
const byte COLONNES = 4; // nombre de colonnes du clavier
char touches [LIGNES] [COLONNES] = {{'1','2','3','A'}, // définition
{'4','5','6','B'}, {'7','8','9','C'}, {'*','0','#','D'}}; // des touches du clavier
byte lignPins [LIGNES] = {9, 8, 7,6}; // se connecter aux pins des lignes du clavier dans l'ordre
byte colPins [COLONNES] = {5, 4, 3,2}; // se connecter aux pins des colonnes du clavier dans l'ordre
char touche; // variable stockant la touche pressée
Keypad clavier = Keypad (makeKeymap (touches), lignPins, colPins, LIGNES, COLONNES);
//création d'un objet keypad : initialisation du clavier
void setup () {
Serial.begin(9600);
Serial.println ("Initialisation du clavier");
}
void loop () {
touche = clavier.getKey (); // lecture de la touche appuyé
if (touche != NO_KEY) // gestion d'une touche si elle est pressée
{
  Serial.println (touche); // affichage du caractère dans le moniteur série
  delay(300);
}
}
```

fig 12.1 : clavier matriciel 4x4_code

Leçon 13

Capteur de mouvement PIR

But de la leçon

Dans cette leçon, nous allons créer un circuit simple avec un mouvement Arduino et PIR.
Pour résumer dès que le capteur détecte un mouvement, il le signale en allumant une LED qui restera allumer pendant 1 seconde.

Matériel nécessaire:

(1) x Arduino Uno R3

1 Arduino

(1) Capteur de mouvement PIR

(1) LED

(1) Résistance de 1kΩ

fig 13.1 : capteur de mouvement PIR

Circuit

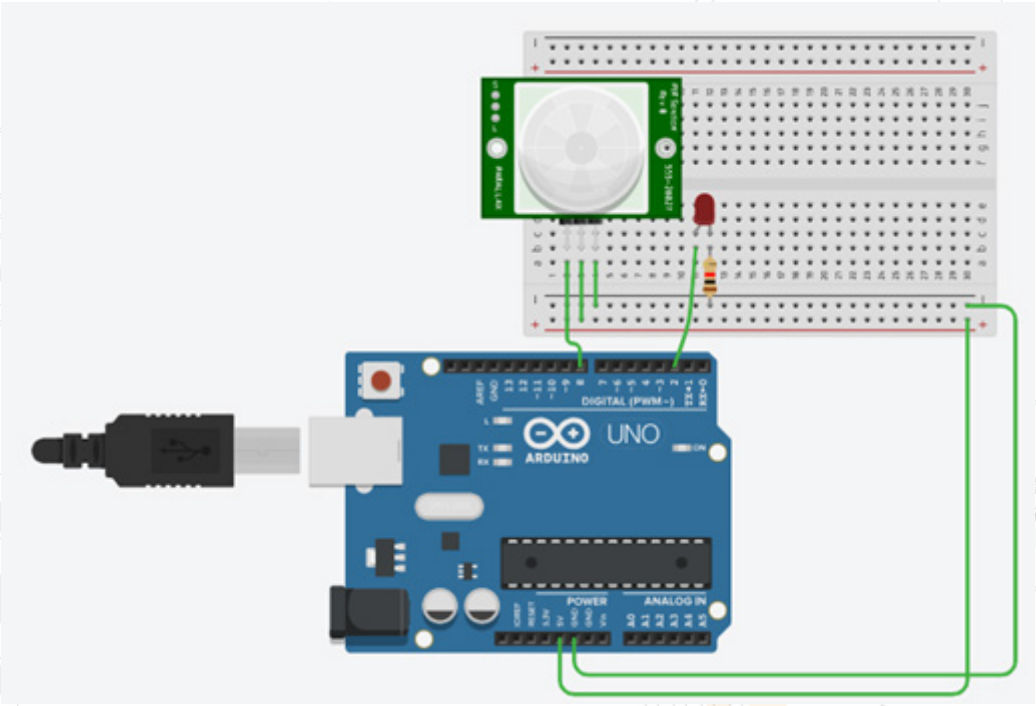


fig 13.2 : capteur de mouvement PIR_diagramme

Illustration

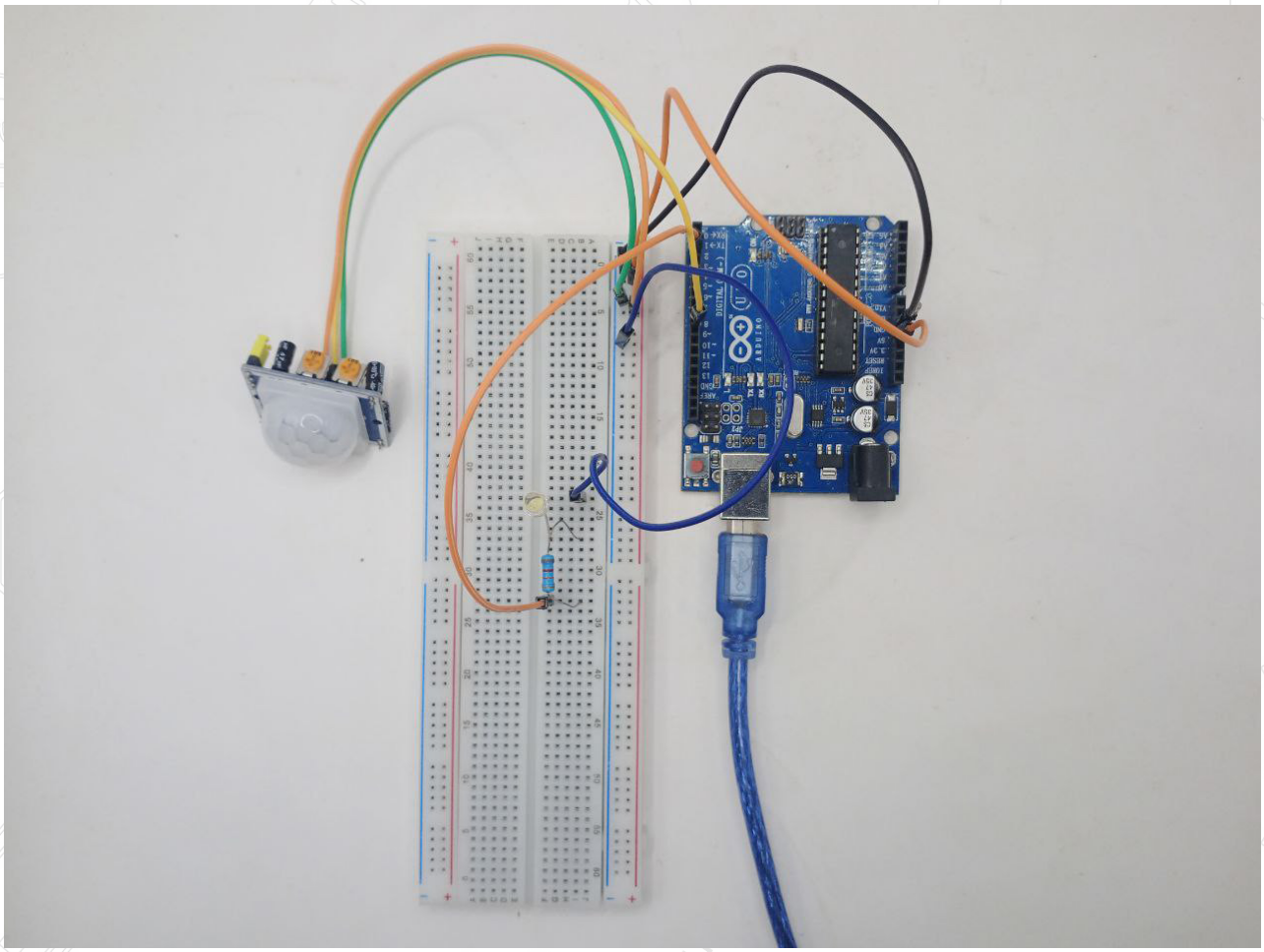


fig 13.3 : capteur de mouvement PIR_illustration

Code

CpateurDeMouvementPIR.ino

```
int Sortie_capteur_PIR=4; //on définie la broche 4 de l'arduino comme la sortie du capteur
int valeur_PIR=0;//on initialise la détection de mouvement à 0
int led=6;//broche de la led
void setup() {
  Serial.begin(9600);
  pinMode(Sortie_capteur_PIR,INPUT);//on définie la sortie du capteur comme une entrée de arduino
  pinMode(led,OUTPUT);//on défine la broche de la LED comme une sortie dans arduino
}
void loop() {
  valeur_PIR=digitalRead(Sortie_capteur_PIR);//Lecture de la broche 4 : la soitie du capteur
  if(valeur_PIR){//si mouvement détecté
    Serial.println("Détecté");
    digitalWrite(led,HIGH);//on allume la LED
    delay(500);//on attend 0,5s avant de l'éteindre
    digitalWrite(led,LOW);//on éteind la LED
    delay(500);//on attend 0,5s avant de la rallumer
    }
    else{
      Serial.println("RAS");
      }
}
```

fig 13.4 : capteur de mouvement PIR_illustration

Leçon 14

Ecran LCD 16 x 2

But de la leçon

Dans cette leçon, vous allez apprendre à utiliser un écran LCD 16x2. Un écran LCD 16x2 est un module d'affichage couramment utilisé dans les projets électroniques et de microcontrôleurs. Le «16x2» signifie que l'écran a 16 colonnes et 2 lignes, permettant ainsi d'afficher jusqu'à 32 caractères à la fois. Ce dernier peut se retrouver sous 2 différentes formes : l'écran LCD 16x2 à rétroéclairage bleu et celui a retro éclairage vert

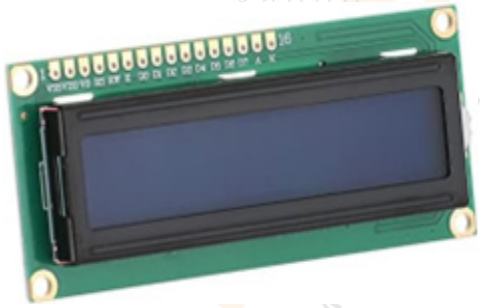


fig 14.1 :Écran LCD

Au-dessus de l'afficheur, 16 marques de soudure sont visibles. Cela reste une bonne façon de repérer les afficheurs compatibles avec la bibliothèque LCD. Ces afficheurs peuvent communiquer avec l'Arduino via c cette bibliothèque
Bibliothèque : La bibliothèque utilisée ici pour son usage est celle-ci : LiquidCrystal (directement incluse dans l'IDE de Arduino)

Caractéristiques:

Taille de l'écran : 16x2
Rétroéclairage : Bleu/Vert
Interface : Compatible avec I2C
Zone d'affichage : 64mm * 16mm
Tension d'alimentation : 5v

Plage de température :

20°C à 70°C température de fonctionnement

30°C à 80°C (température de stockage)

Rétroéclairage : LED blanche

Dimensions : 80mm x 36mm x 12mm.

Consommation de courant : environ 1 mA sans rétroéclairage, et jusqu'à 15-20 mA avec rétroéclairage activé.

Broches principales : VSS (GND), VDD (5V), V0 (contraste), RS (Register Select), RW (Read/Write), E (Enable), D0 à D7 (données), A (anode rétroéclairage), K (cathode rétroéclairage)

Matériel nécessaire

(1) x Arduino Uno ou tout autre modèle compatible

(1) x Écran LCD 16x2

(1) x Résistance de 10kΩ (pour le contraste de l'écran)

(1) x Potentiomètre 10kΩ (facultatif pour ajuster le contraste)

(1) x Fils de connexion (jumpers)

(1) x Breadboard

Brochages de l'écran LCD 16 x 2

-RS (Register Select)

-E (Enable)

-D4, D5, D6, D7 (Data pins)

-VSS, VDD (Power)

-V0 (Contrast)

-RW (Read/Write, souvent connecté à la masse si vous ne lisez pas depuis le LCD)

-BLA, BLK (Backlight, Anode et Cathode)

Diagramme de câblage

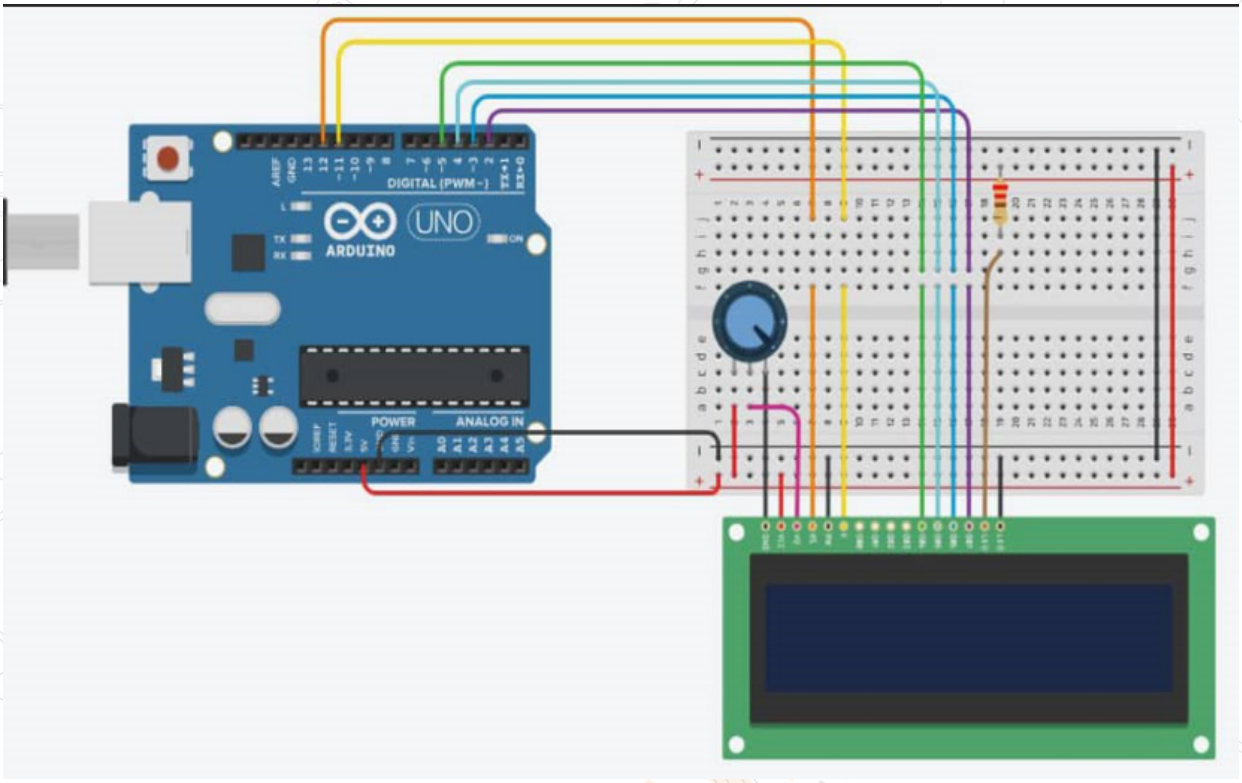


fig 14.2 :Écran LCD_diagramme

Illustration

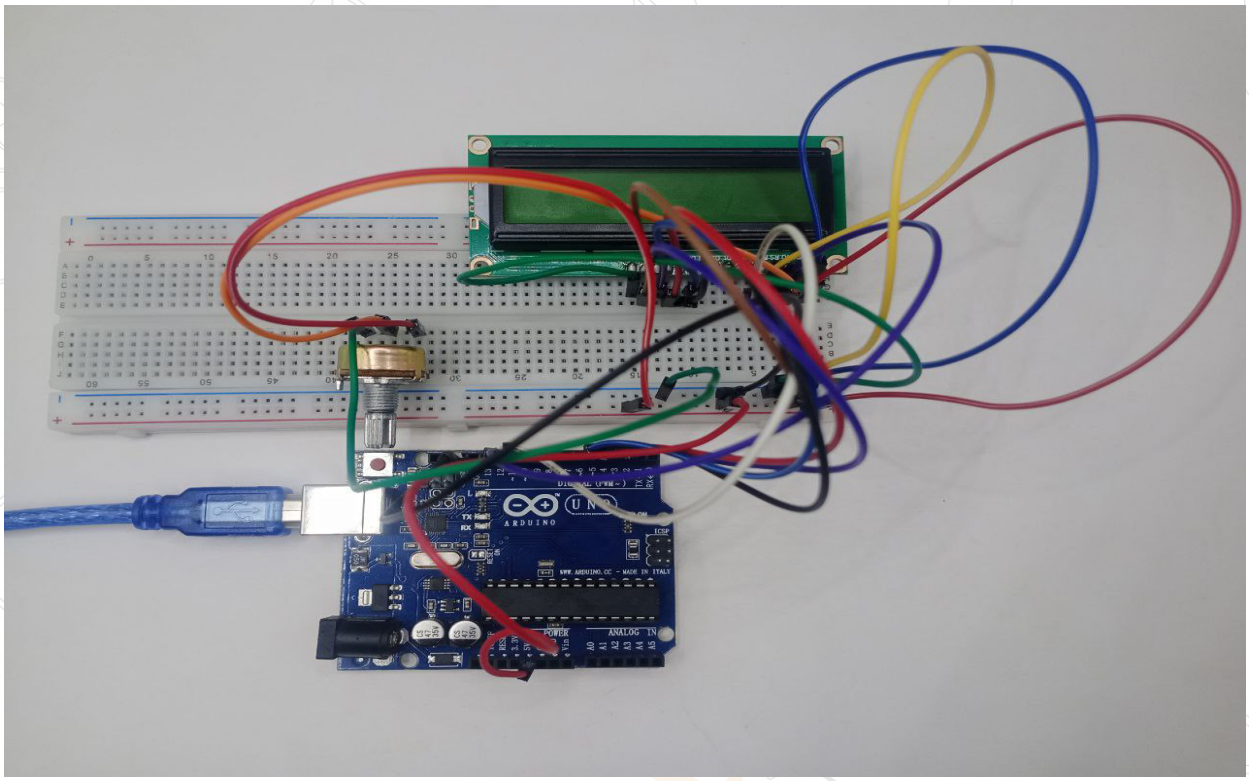


fig 14.3 :Écran LCD_illustration

Code

EcranLCD16.ino

```
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);                                  //Définition de l'adresse du LCD (0x27)
String texte_1 = "Avec YoupiLab:";                                   //texte satique
String texte_2 = "Apprehendez le fonctionnement du module LCD I2C."; //texte à faire défiler
void setup() {
  lcd.init();
  lcd.backlight();
}
void loop() {
  lcd.setCursor(1, 0);
  lcd.print(texte_1);
  defilerTexte(texte_2, 500);  //La fonction qui fera défiler le texte
}
//Définition de la fonction defilerTexte
void defilerTexte(String message, int timing)  // prend deux arguments. Le premier est le texte à faire défiler. Le second est la vitesse de défilement
{
  for (int i = 0; i < 16; i++) {
    message = " " + message;
  }
  message = message + " ";
  for (int pos = 0; pos < message.length(); pos++) {
    lcd.setCursor(0, 1);
    lcd.print(message.substring(pos, pos + 16));
    delay(timing);
  }
}
```

fig 14.4 :Écran LCD_code

Leçon 15

Capteur de niveau d'eau

But de la leçon

Dans cette leçon, vous allez apprendre comment utiliser un détecteur de niveau d'eau.

Ce module peut percevoir la profondeur d'eau dans laquelle il est immergé, grâce à une résistance qui va varier.

Matériel nécessaire:

(1) x Arduino Uno R3

(3) x Câbles Mâle-Femelle

(1) x Water level detection sensor module

fig 15.1 : Capteur de niveau d'eau

Présentation du composant

Le module est composé de trois parties. Une partie avec de l'électronique et des pins de connexion, une résistance de pull-up de 1MΩ et des pistes conductrices. Une série de pistes est reliée à la masse. Entre se trouvent des pistes qui vont permettre de faire la détection et envoyer un signal sur la sortie S. Les pistes de détection ont une faible résistance de pull-up (1 MΩ). La résistance élimine la valeur de traçage du capteur (signal est à l'état bas) jusqu'à ce qu'une goutte d'eau fasse une mise à la terre et passe la sortie signal à une valeur proportionnelle à la quantité d'eau. Il a une faible consommation d'énergie et une grande sensibilité.

Caractéristiques:

1- Voltage: 5V

2 - Courant:<20ma

3 - Interface: Analogique

4 - Largeur de détection: 40mm×16mm

5 - Température: 10°~30°

6 - Signal de sortie: 0~2.4V

Diagramme de câblage

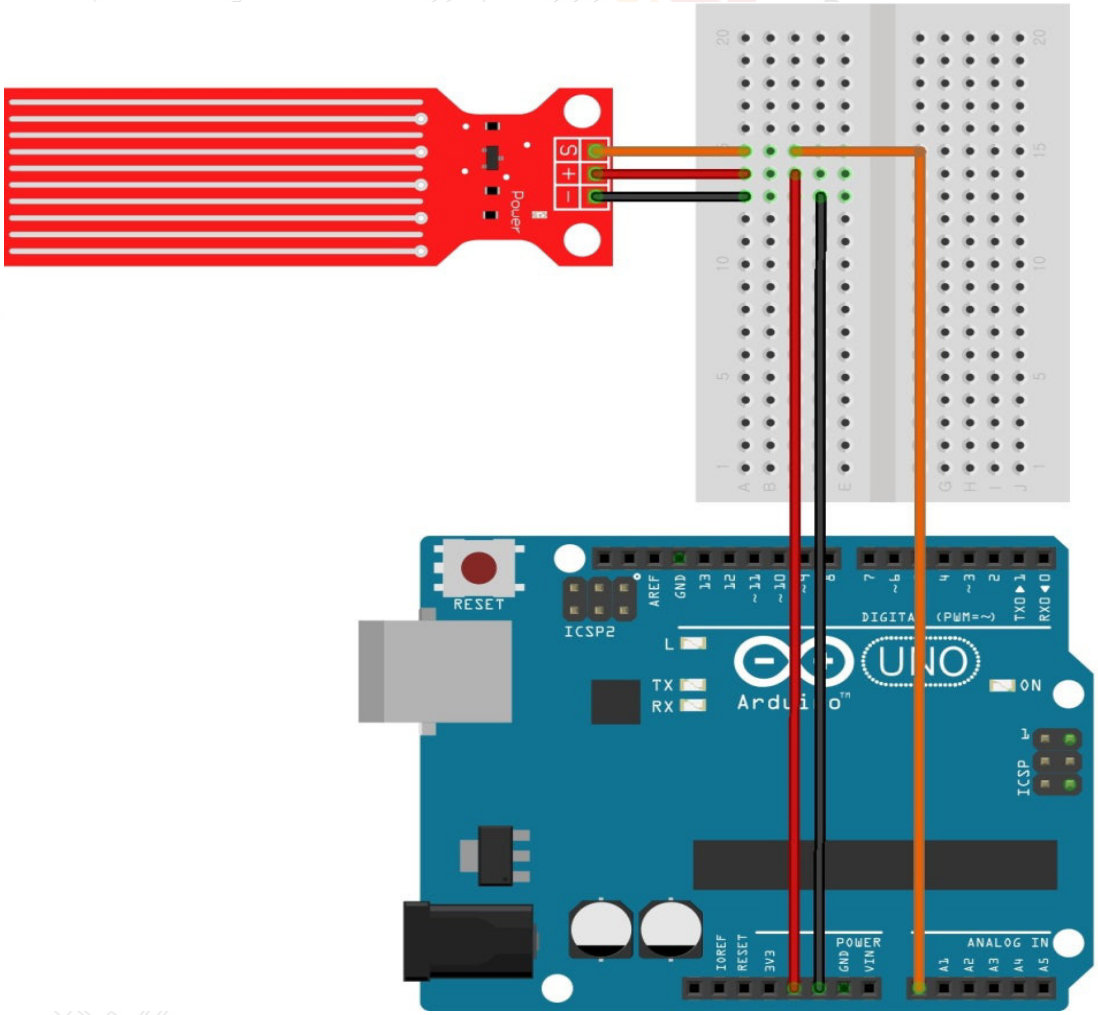


fig 15.2 : Capteur de niveau d'eau_diagramme

Illustration

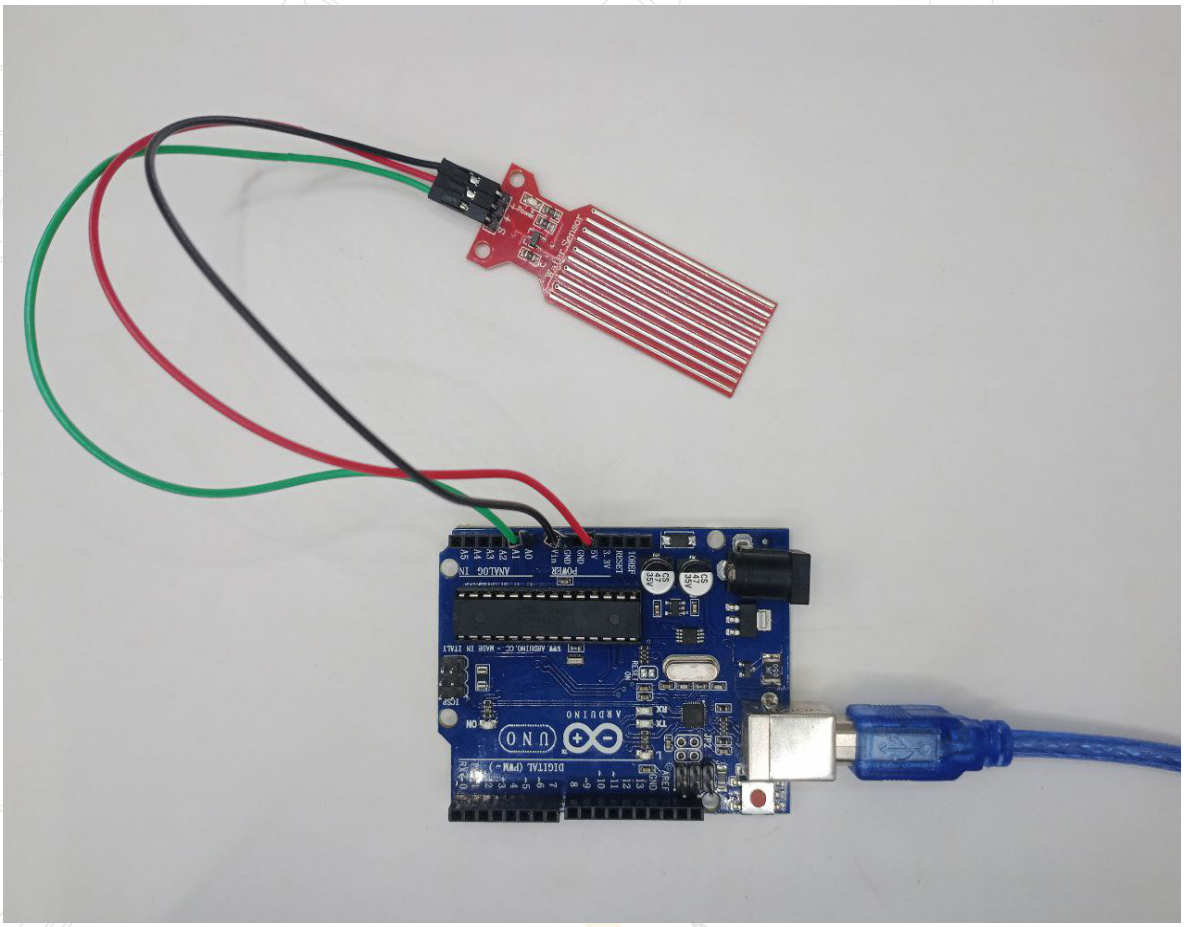


fig 15.3 : Capteur de niveau d'eau_illustration

Code

Capteur_niveau_eau.ino

```
// Déclaration de la broche pour le capteur
const int waterSensorPin = A0;  // Le capteur est connecté à la broche A0

void setup() {
  // Initialisation de la communication série
  Serial.begin(9600);  // Démarrage du moniteur série
}

void loop() {
  // Lecture de la valeur analogique du capteur de niveau d'eau
  int sensorValue = analogRead(waterSensorPin);

  // Affichage de la valeur lue sur le moniteur série
  Serial.print("Valeur du capteur de niveau d'eau: ");
  Serial.println(sensorValue);

  // Délai avant la prochaine lecture
  delay(1000);  // Attendre 1 seconde avant la prochaine mesure
}

```

fig 15.4 : Capteur de niveau d'eau_code

Leçon 16

Module RFID

But de la leçon

Dans cette leçon, vous allez apprendre comment utiliser un lecteur de carte RFID.

Matériel nécessaire:

(1) x Arduino Uno R3

(1) x Module RC522 RFID

(7) x Câbles Mâle-Femelle

fig 16.1 : Module RFID

Présentation du composant

Le MFRC522 est un module permettant de faire de la lecture et de l'écriture en RFID qui travaille à 13.56 MHz. Il travaille sur avec la norme ISO 14443A / MIFARE®. Le MFRC522 ne nécessite pas de système de réception/émission supplémentaire. Le récepteur fourni une implémentation robuste et efficace de démodulation et décodage du signal selon la norme ISO/IEC 14443A/MIFARE. Le MFRC522 supporte les communications sans contact utilisant le système de transfert haut débit MIFARE® à 848 kbit/s en flux montant et descendant.

Plusieurs interfaces hôtes sont intégrées:

- Interface SPI
- Interface UART (similaire au RS232)
- Interface I2C.

La figure ci-dessous montre un schéma de circuit typique, utilisant une connexion d'antenne complémentaire le MFRC522.

Diagramme de câblage

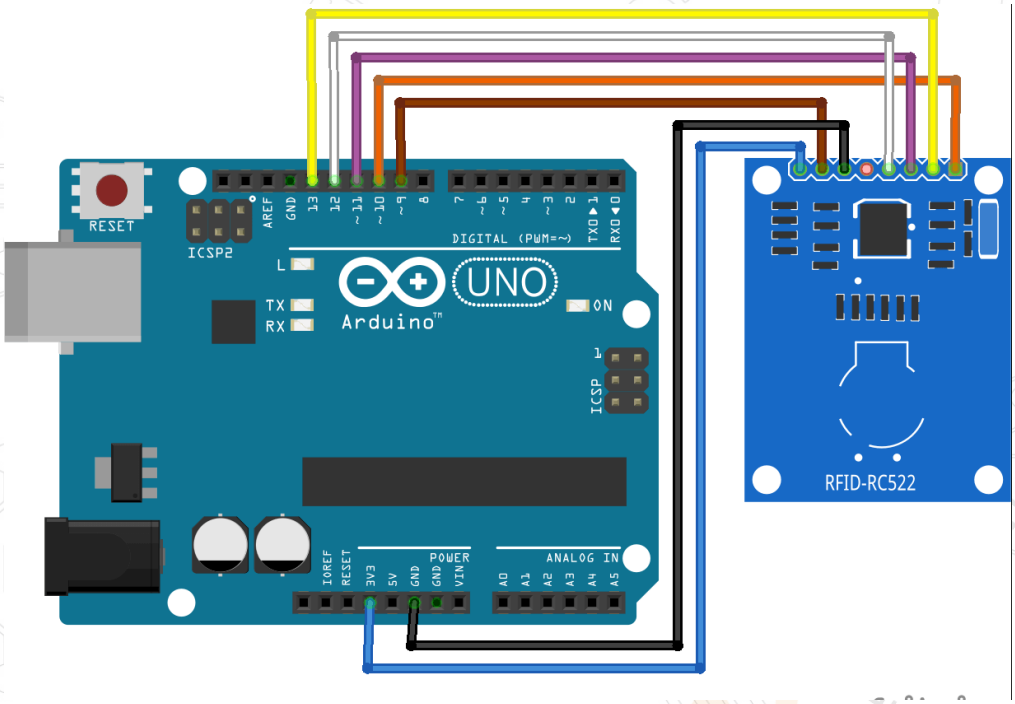


fig 16.2 : Module RFID_diagramme

Illustration

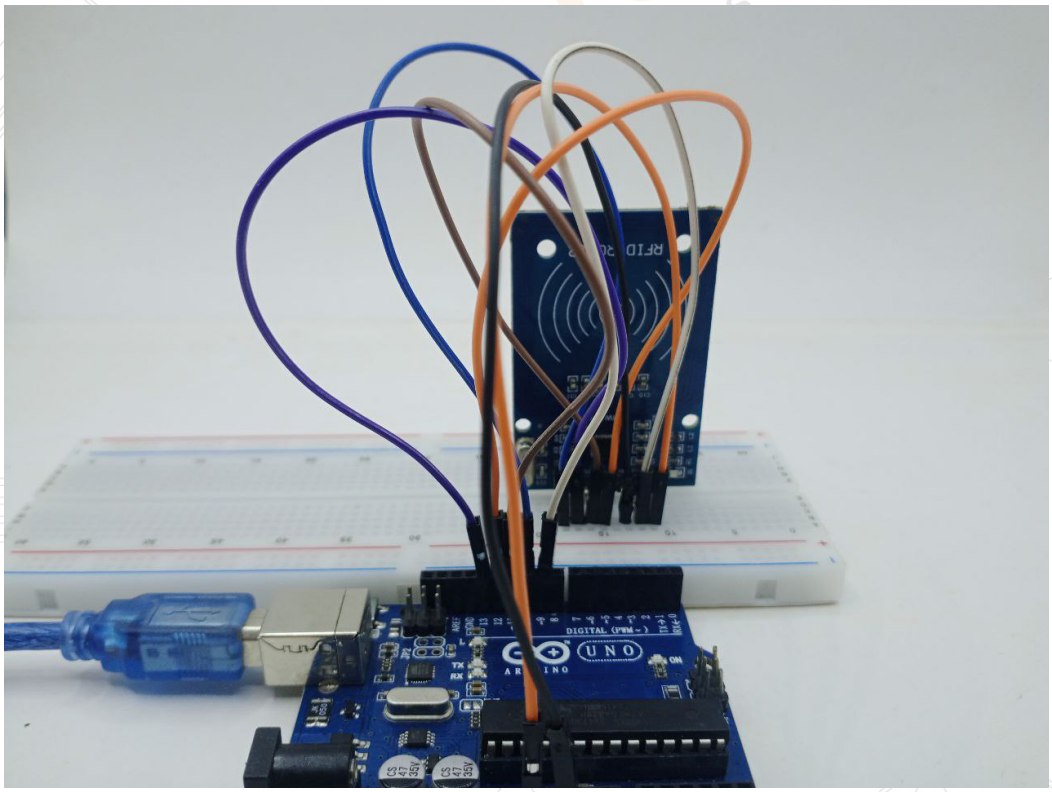


fig 16.3 : Module RFID_illustration

Code

ModuleRFID.ino

```
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 10
#define RST_PIN 9
MFRC522 rfid(SS_PIN, RST_PIN);
RFID RFID(10,9);
int UID[5];
void setup()
{
  Serial.begin(9600);
  SPI.begin();
  RFID.init();
}
void loop()
{
    if (RFID.isCard()) {
        if (RFID.readCardSerial()) {
          Serialprint("« L'UID est: « ");
          for(int i=0;i<=4;i++)
          {
            UID[i]=RFID.serNum[i];
            Serial.print(UID[i],HEX);
            Serial.print(".");
          }
          Serial.println(" »");
        }
        RFID.halt();
    }
    delay(100);
}

```

fig 16.4 : Module RFID_code

Leçon 17

Télécommande et recepteur IR

But de la leçon

L'utilisation de l'infrarouge est un très bon moyen de réaliser un contrôle à distance. Les télécommandes infrarouges ne sont pas compliquées à mettre en œuvre.

Dans cette leçon, vous allez apprendre à brancher un récepteur infrarouge à la carte UNO R3 et utiliser une bibliothèque dédiée à ce capteur. Dans le sketch, vous aurez tout le code hexadécimal disponible avec la télécommande fournie et vous allez pouvoir détecter la touche appuyée sur celle- ci.

Matériel nécessaire:

(1) x Arduino Uno R3

(1) x Capteur IR

(1) x Télécommande IR

(3) x Câbles Mâle-Femelle

fig 17.1 : Télécommande IR

fig 17.2 : capteur IR

Présentation du composant

Les détecteurs IR sont de petites micropuces avec une cellule photoélectrique qui sont réglées pour écouter la lumière infrarouge. Ils sont presque toujours utilisés pour la détection de la télécommande - chaque téléviseur et lecteur DVD en a un à l'avant pour écouter le signal IR du clicker. À l'intérieur de la télécommande se trouve une LED IR correspondante, qui émet des impulsions IR pour indiquer au téléviseur d'allumer, d'éteindre ou de changer de chaîne. La lumière infrarouge n'est pas visible à l'oeil humain, ce qui signifie qu'il faut un peu plus de travail pour tester une configuration. Il y a quelques différences entre celles-ci et disons une photocellule CdS: Les détecteurs IR sont spécialement filtrés pour la lumière IR, ils ne sont pas bons pour détecter la lumière visible. D'un autre côté, les photocellules sont bonnes pour détecter la lumière visible jaune / verte et ne sont pas bonnes pour la lumière infrarouge. Les détecteurs IR ont un démodulateur à l'intérieur qui recherche l'IR modulé à 38 KHz. Brillant juste un IR
La LED ne sera pas détectée, elle doit être PWM clignotante à 38 KHz. Les photocellules n'ont aucune sorte de démodulateur et peut détecter toute fréquence (y compris DC) dans la vitesse de réponse du photocellule (qui est d'environ 1 KHz).

Les détecteurs IR sont à sortie numérique - soit ils détectent un signal IR à 38 KHz et émettent un signal bas (0 V), soit ils ne détectent pas tout et sortie haute (5V). Les photocellules agissent comme des résistances, la résistance change en fonction de la quantité la lumière à laquelle ils sont exposés. Comme vous pouvez le voir sur ces graphiques, le pic de fréquence est à 38kHz e le pic de la LED est à 940nm. Il est possible de travailler entre 35kHz et 41kHz, mais la sensibilité va chuter. De la même manière, vous pouvez utiliser une LED allant de 850 à 1100 nm, mais cela ne marchera pas aussi bien qu'avec celle de 850nm.

Diagramme de câblage

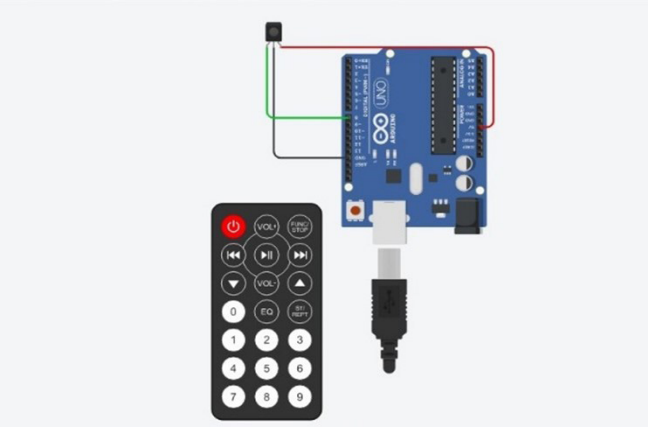


fig 17.3 : Télécommande et récepteur IR_diagramme

Il y a 3 pins sur le capteur :

+Vcc, Masse, Signal

Illustration

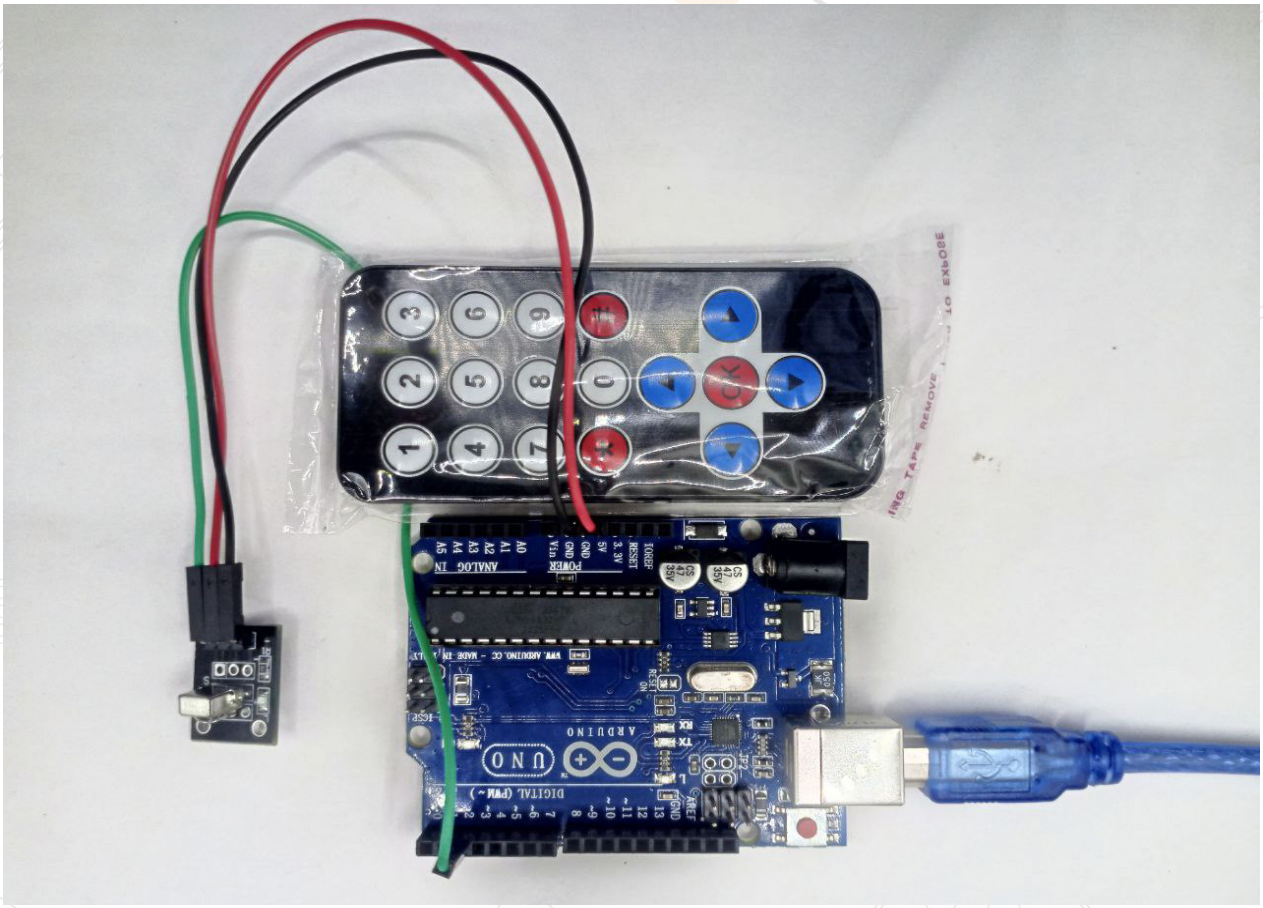


fig 17.4 : Télécommande et récepteur IR_illustration

Code

IR.ino

```
#include <IRremote.h>
// Définir la broche de connexion du récepteur IR
const int IR_PIN = 11;  // Broche où le récepteur IR est connecté
// Création d'un objet IRrecv
IRrecv irrecv(IR_PIN);
// Stocker le résultat du décodage
decode_results results;
void setup() {
  // Initialisation de la communication série
  Serial.begin(9600);
  // Activation du récepteur IR
  irrecv.enableIRIn();
}
void loop() {
  // Vérifier si un signal IR est reçu
  if (irrecv.decode(&results)) {
    // Afficher le code reçu sur le moniteur série
    Serial.print("Code IR reçu : ");
    Serial.println(results.value, HEX); // Afficher le code en hexadécimal
    // Reprendre la réception pour le prochain signal
    irrecv.resume();
  }
  delay(100);  // Petite pause pour éviter une surcharge
}

```

fig 17.5 : Télécommande et récepteur IR_code

Leçon 18

Module de Relais 1 canal

But de la leçon

Dans cette leçon, vous allez apprendre à utiliser un relais.

Matériel nécessaire:

(1) x Arduino Uno R3

(1) x Relais 5v

(1) x Alimentation

(1) x Adaptateur 9v

(8) x Câbles mâle-mâle

fig 18.1 : Relais 1 canal

Présentation du composant

Un relais est un interrupteur commandé électriquement. Beaucoup de relais utilisent l'électromagnétisme pour opérer mécaniquement l'interrupteur, mais il existe d'autres technologies. Les relais sont utilisés notamment lorsque le circuit à commander est isolé électriquement du circuit qui opère le contrôle ou lorsque plusieurs circuit doivent être commandés par un signal unique. Un type de relais qui peut manipuler des circuits à « haute » tension sont appelés les contacteurs. Ils sont faits pour supporter les surcharges électriques. Dans l'électronique moderne, ces fonctions sont produites par des instruments numériques appelés « relais de protection ».

Le schéma ci-dessous montre comment on câble un relais avec une carte UNO R3. Faites bien attention lors de la mise en place du relais.

Diagramme de câblage

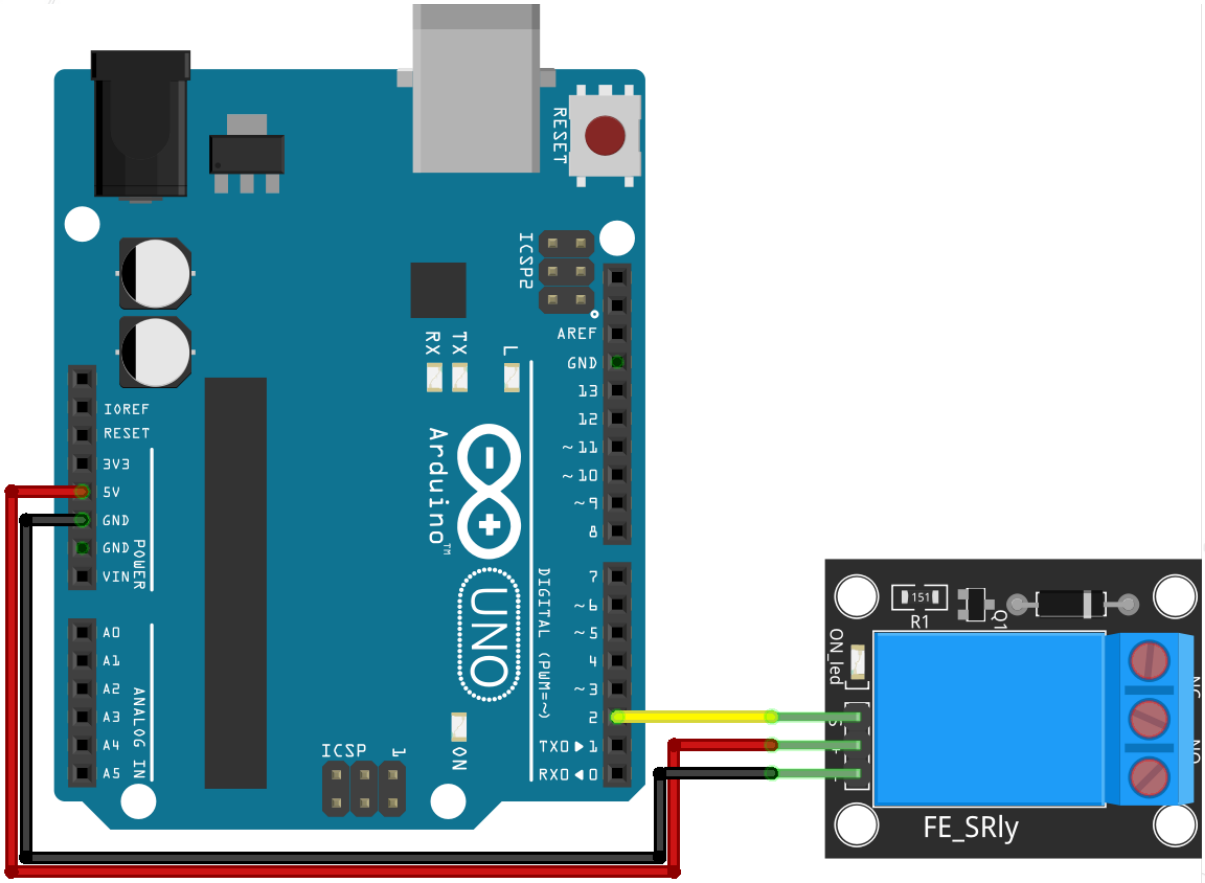


fig 18.2 : Relais 1 canal_diagramme

Illusrarion

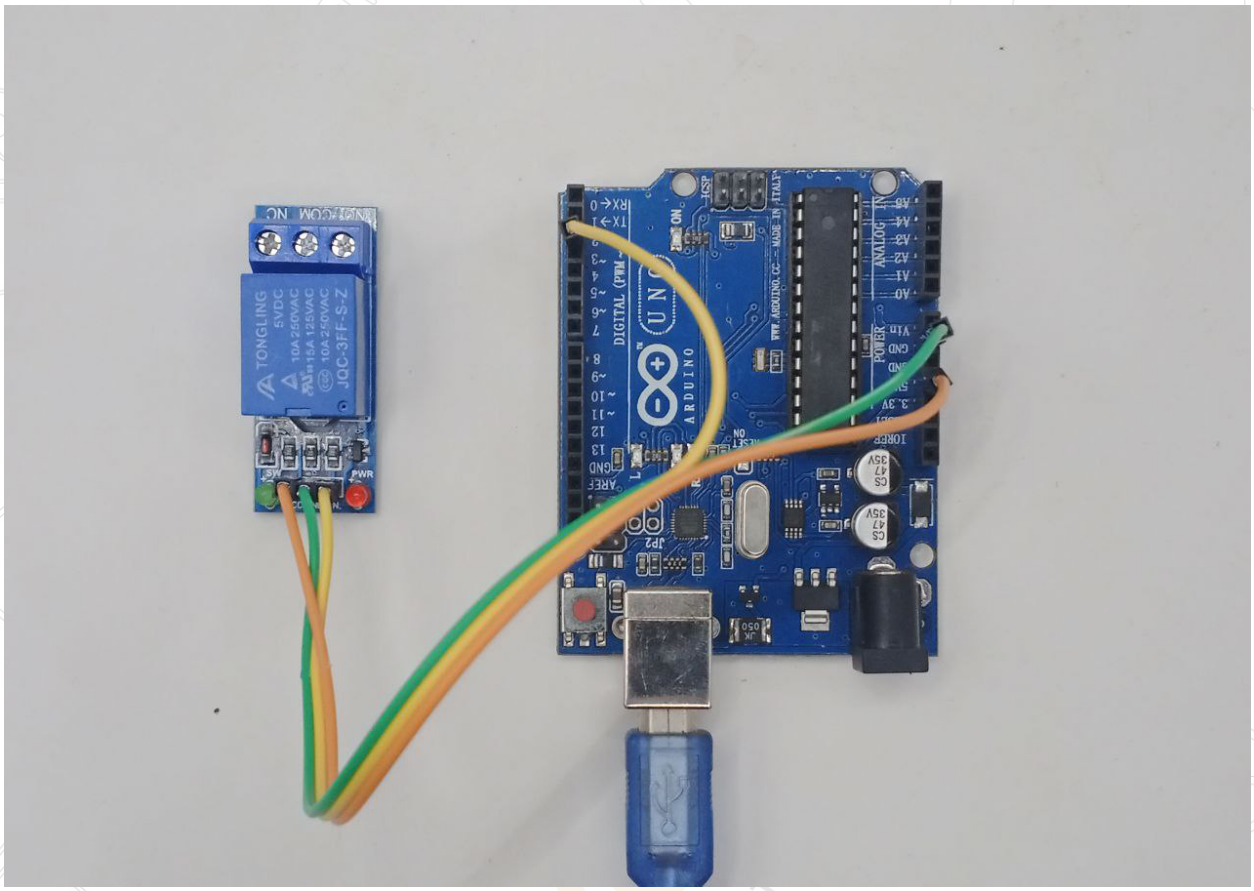


fig 18.3 : Relais 1 canal_illustration

Code

Module_relais.ino

```ino
// Définir le pin connecté au relais
const int relaisPin = 8;

void setup() {
  // Configurer le pin comme sortie
  pinMode(relaisPin, OUTPUT);

  // Initialiser le relais à l'état "OFF"
  digitalWrite(relaisPin, HIGH); // État HIGH signifie que le relais est désactivé dans un module actif LO
}

void loop() {
  // Activer le relais
  digitalWrite(relaisPin, LOW); // LOW active le relais pour un module actif LOW
  delay(2000);                  // Attendre 2 secondes

  // Désactiver le relais
  digitalWrite(relaisPin, HIGH); // HIGH désactive le relais pour un module actif LOW
  delay(2000);                   // Attendre 2 secondes
}
```

fig 18.4: Relais 1 canal_code

Leçon 19

Capteur de flamme

But de la leçon

Le détecteur de flamme est un capteur qui permet de mesurer des longueurs d'onde sur une plage comprise entre 760 nm et 1100 nm. Il peut être utilisé pour détecter une source d'incendie ou d'autres sources lumineuses de longueur d'onde comprise entre 760 nm et 1100 nm. Il est basé sur le capteur YG1006 qui est un phototransistor au silicium NPN à haute vitesse et très sensible. En raison de son époxy noir, le capteur est sensible au rayonnement infrarouge.

Le module de détection de flamme YG 1006 pour Arduino mesure l'intensité de la lumière infrarouge émise par le feu sur une plage de longueur d'onde comprise entre 760 à 1100 nm. Le module dispose de sortie digitale D0 et d'un potentiomètre pour régler la sensibilité.

Caractéristiques:

Module didactique basé sur un récepteur IR permettant la détection d'une flamme ou d'autres sources lumineuses.

- Alimentation: 3.3V à 5 V
- Plage de mesure: 760 à 1100 nm
- Broches: Gnd, Vcc et D0
- Température de service: -40°C à +85 °C
- Humidité de service: 30 à 90 % RH
- Dimensions: 42 x 16 x 15mm
- L'angle de détection est d'environ 60 degrés

Présentation du composant

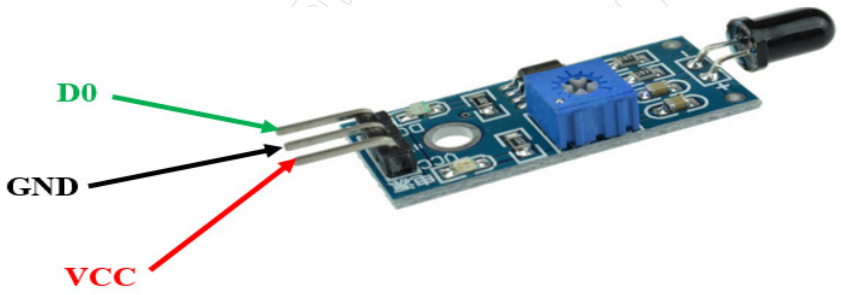


fig 19.1: capteur de flemme

VCC : à connecter au VCC de la carte Arduino (Sortie 5V)

GND : à connecter au GND de la carte Arduino

D0 : à connecter à une broche digitale de la carte Arduino

Matériel nécessaire

(1) x Une carte Arduino

(1) x Un capteur de flamme

(1) x Une LED

(1) x Une résistance de 220ohms et

(1) x Un buzzer

(1) x Un breadboard

Plusieurs fils de connexion.

Diagramme de câblage

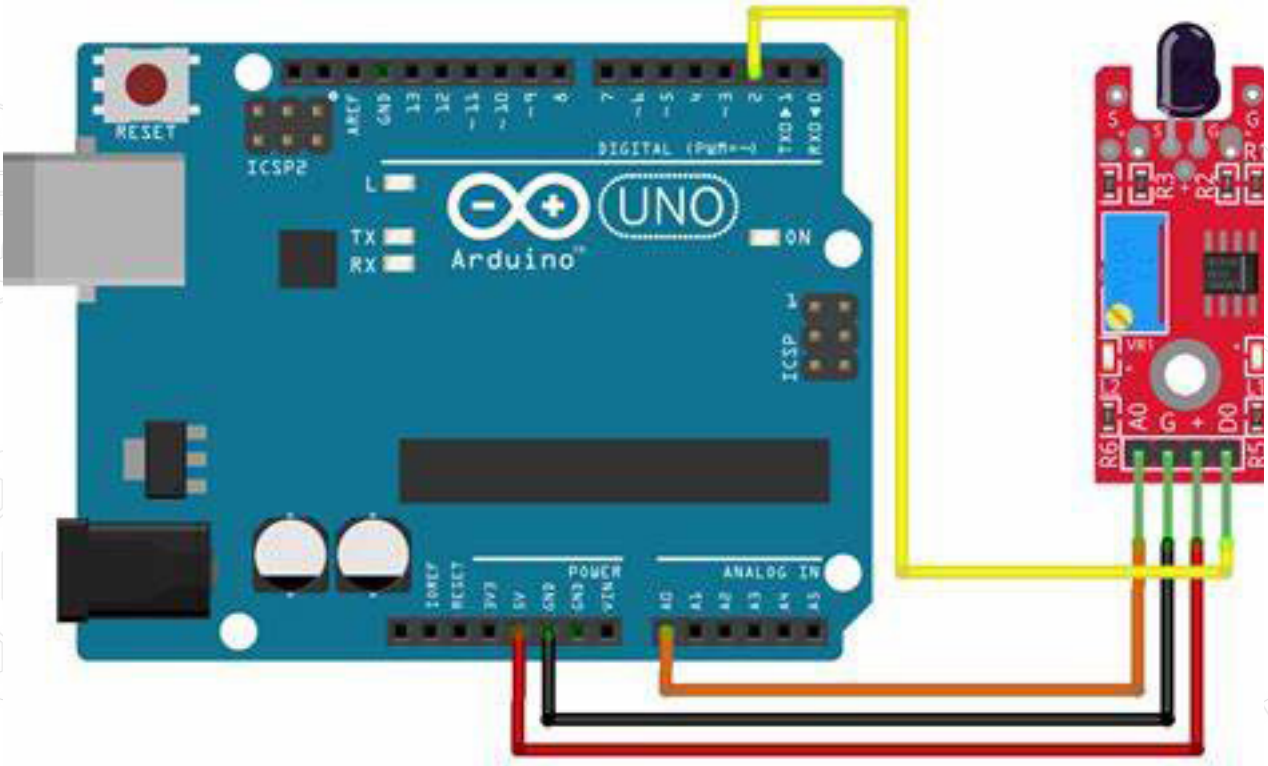


fig 19.2: capteur de flemme_diagramme

Illustration

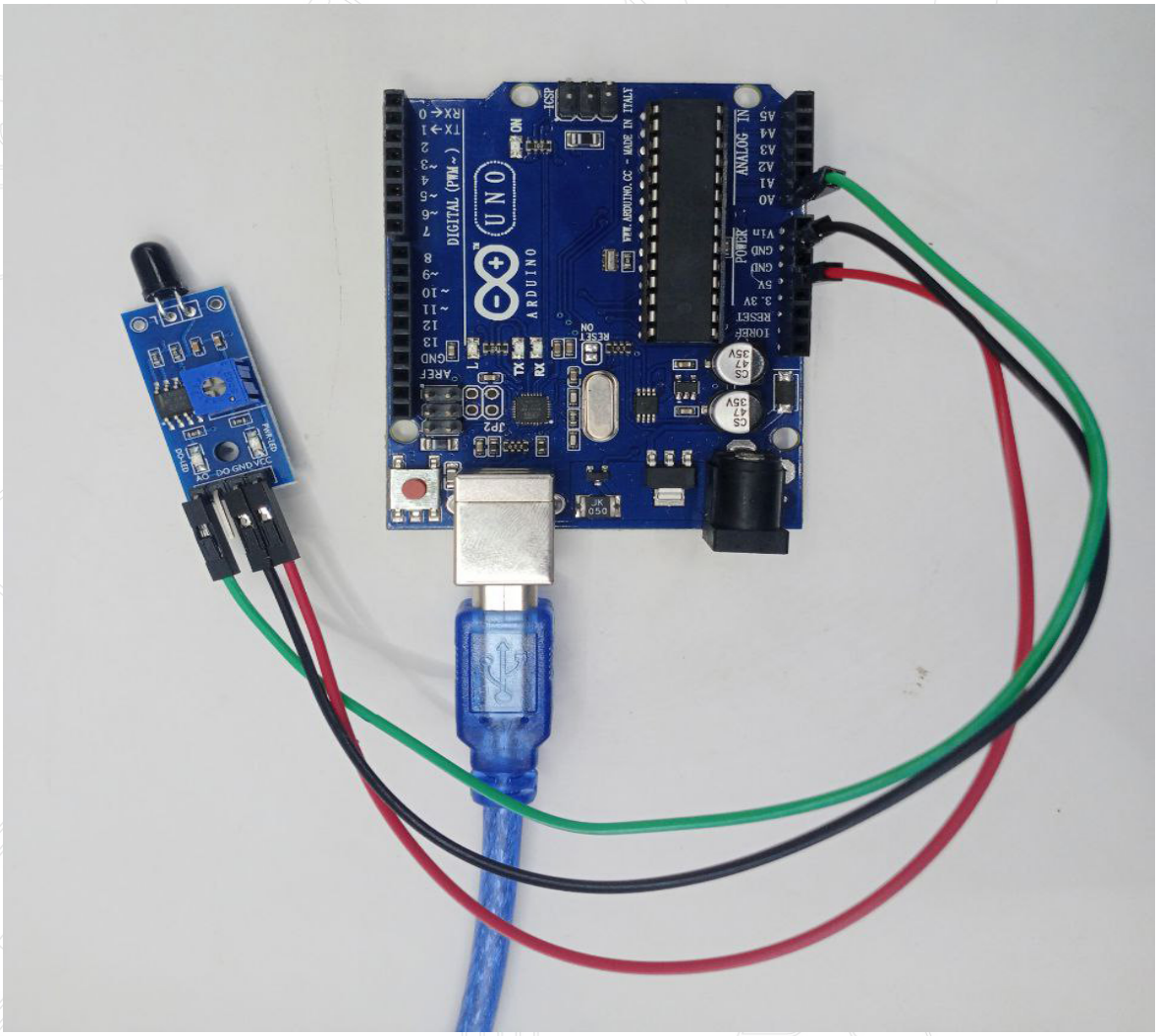


fig 19.3: capteur de flemme_illustration

Code

CapteurDeFlamme.ino

```
int D0 =2; // Broche D0 du capteur
int led=3; // Broche de la LED
int buzzer=4; //Broche du buzzer

void setup()
{
  Serial.begin(9600);// initialisation du moniteur série
  pinMode(led,OUTPUT);
  pinMode(buzzer,OUTPUT);
  Serial.println("**********Programme de détection de feu***************");
}

void loop()
{
if(digitalRead(D0)==0)
{
                    Serial.println("Feu!!!!!!!!!!!!");
                    digitalWrite(led,HIGH); // on allume la LED
                    digitalWrite(buzzer,HIGH);// on allume le buzzer
                }else{
                    digitalWrite(led,LOW);//on éteind la LED
                    digitalWrite(buzzer,LOW); // on éteind le buzzer
                }
        delay(100);
}

```

fig 19.4: capteur de flemme_code

Leçon 20

Afficheur 7 segments

But de la leçon

Dans cette leçon, vous apprendrez à utiliser un afficheur à 7 segments à 1 chiffre. Pour un **afficheur à anode commune**, la broche commune se connecte à la source d'alimentation. Pour un **afficheur à cathode commune**, la broche commune se connecte à la masse (GND).

Ce type d'afficheur est souvent utilisé pour afficher des nombres ou des caractères simples dans des projets électroniques, grâce à son fonctionnement intuitif et sa faible consommation d'énergie.

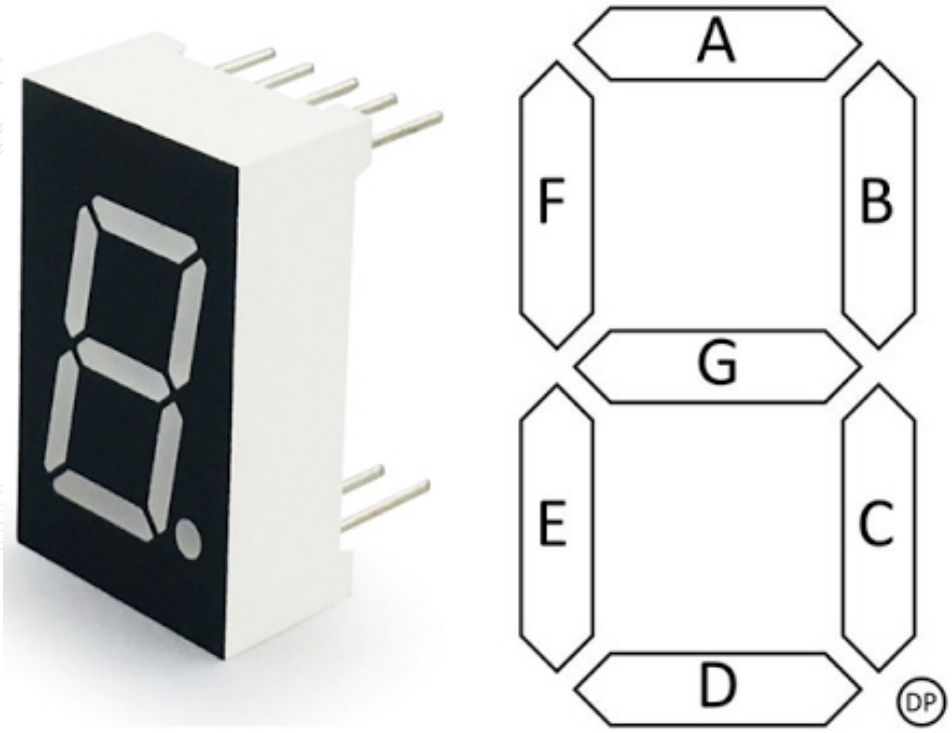


fig 20.1: afficheur 7 segments

Les afficheurs 7 segments sont constitués de 7 segments, d'où leur nom. Ces segments sont nommés A, B, C, D, E et F par convention, et ils se présentent dans l'ordre illustré ci-dessus.

Chaque segment correspond à une LED qu'il est possible d'allumer ou d'éteindre pour former des chiffres, des lettres et même des caractères spéciaux rudimentaires. En général, les afficheurs disposent de 7 segments et d'un « point décimal » qui peut être utilisé pour afficher des nombres à virgule ou des sous-unités (dixième de seconde par exemple).

Matériel nécessaire:

(1) x Une carte Arduino UNO (et son câble USB),
(1) x Un afficheur 7 segments à cathode commune,
(8) x résistances de 1K ohms
(1) x Une plaque d'essai
Des fils pour câbler notre montage.

Diagramme de câblage

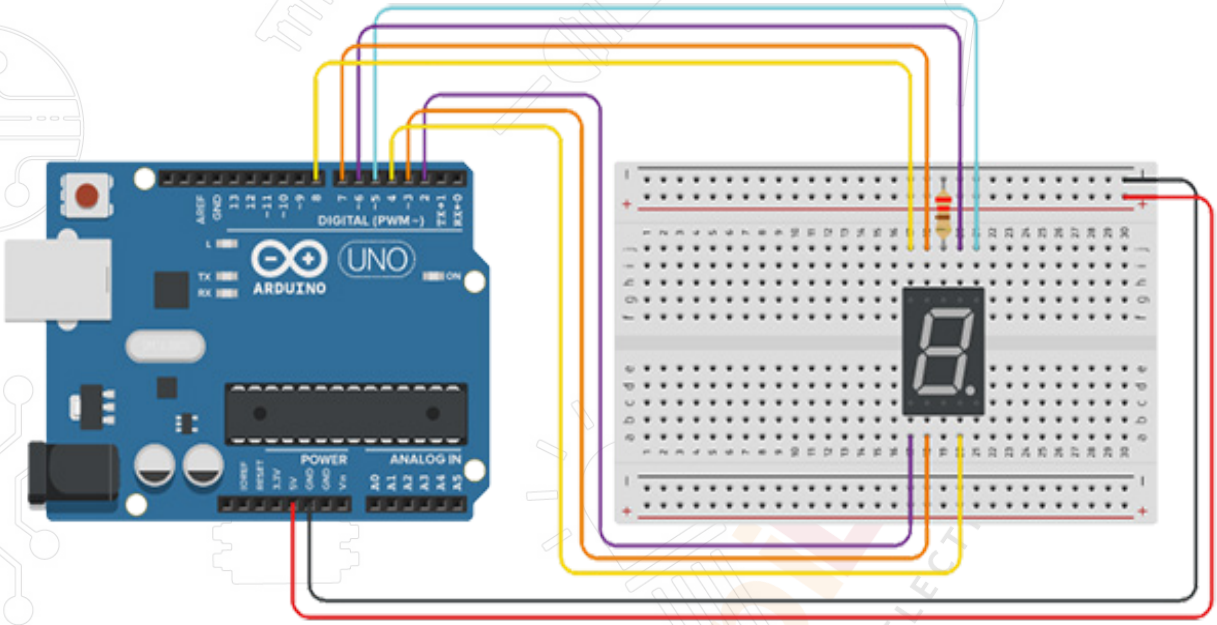


fig 20.2: afficheur 7 segments_diagramme

Code

Afficheur7segments.ino

```
/* Broches des différents segments de l'afficheur */
const byte PIN_SEGMENT_A = 2;
const byte PIN_SEGMENT_B = 3;
const byte PIN_SEGMENT_C = 4;
const byte PIN_SEGMENT_D = 5;
const byte PIN_SEGMENT_E = 6;
const byte PIN_SEGMENT_F = 7;
const byte PIN_SEGMENT_G = 8;
const byte PIN_SEGMENT_DP = 9;
/* Décommenter si utilisation d'un afficheur 7 segments à ANODE commune */
//#define _7SEG_COMMON_ANODE
/* Table de correspondance valeur -> états des segments de l'afficheur */
const byte LUT_ETATS_SEGMENTS[] = {
  0b00111111,
  0b00000110,
  0b01011011,
  0b01001111,
  0b01100110,
  0b01101101,
  0b01111101,
  0b00000111,
  0b01111111,
  0b01101111,
  0b01110111,
  0b01111100,
  0b00111001,
  0b01011110,
  0b01111001,
  0b01110001
};
```

fig 20.1: afficheur 7 segments_code

Leçon 21

Thermistance

But de la leçon

Dans cette leçon, vous utiliserez une thermistance pour afficher la température ambiante.

Matériel nécessaire:

(1) x Arduino Uno R3

(1) x 10k ohm resistor

(1) x Thermistance

(1) x Planche prototype

Des Câbles mâle-mâle

fig 21.1: Thermistance

Présentation du composant

Thermistance

Une thermistance est une “ résistance thermique” : sa valeur varie en fonction de la température. Une résistance standard voit aussi sa température varier, mais très faiblement, une thermistance est faite pour cette variation soit importante et donc facile à mesurer. (100 ohms par degré)

Il en existe deux types : NTC (negative temperature coefficient) et PTC (positive temperature coefficient). En général, on utilise des NTC.

Diagramme de câblage

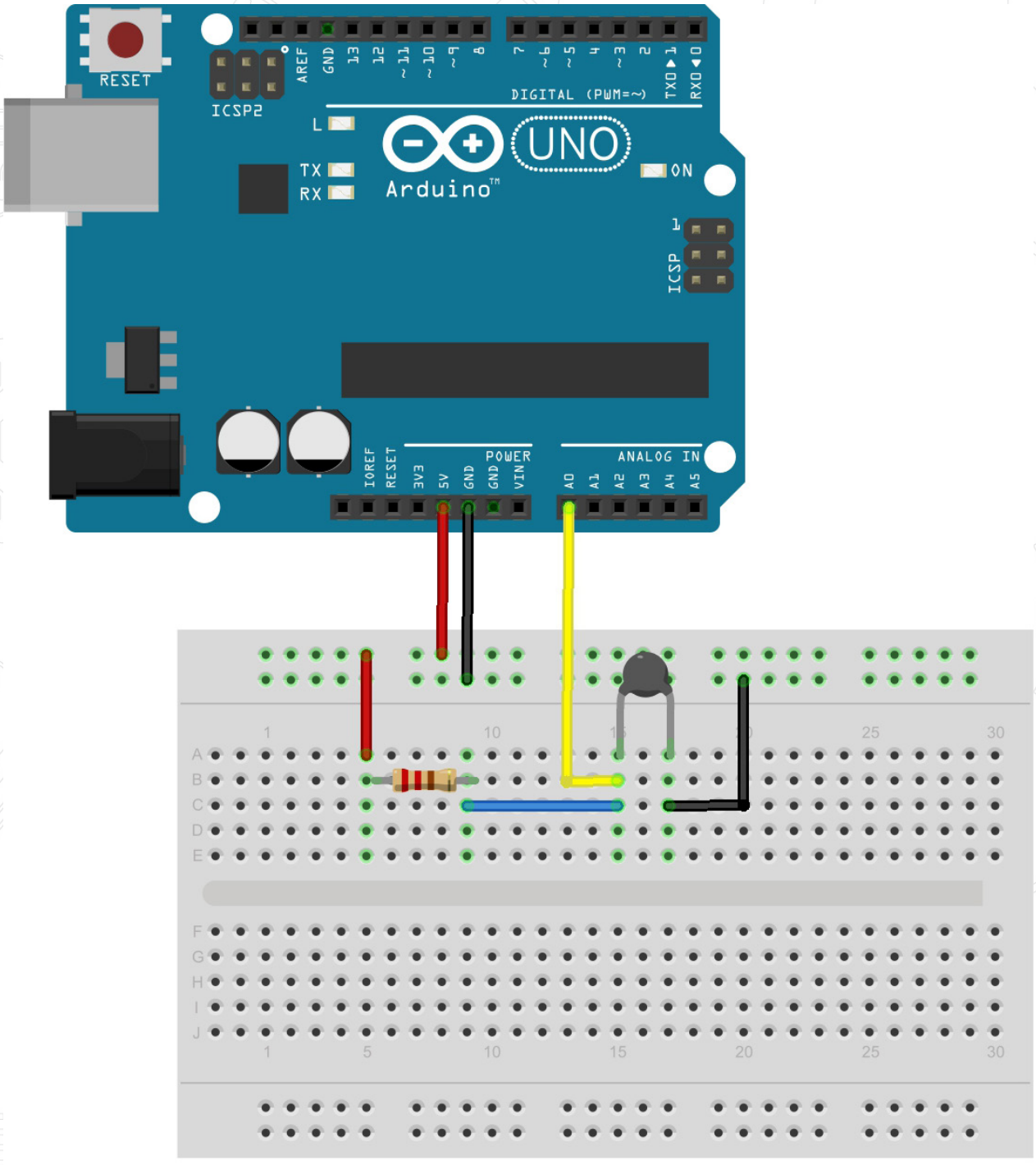


fig 21.2: Thermistance_diagramme

Illustration

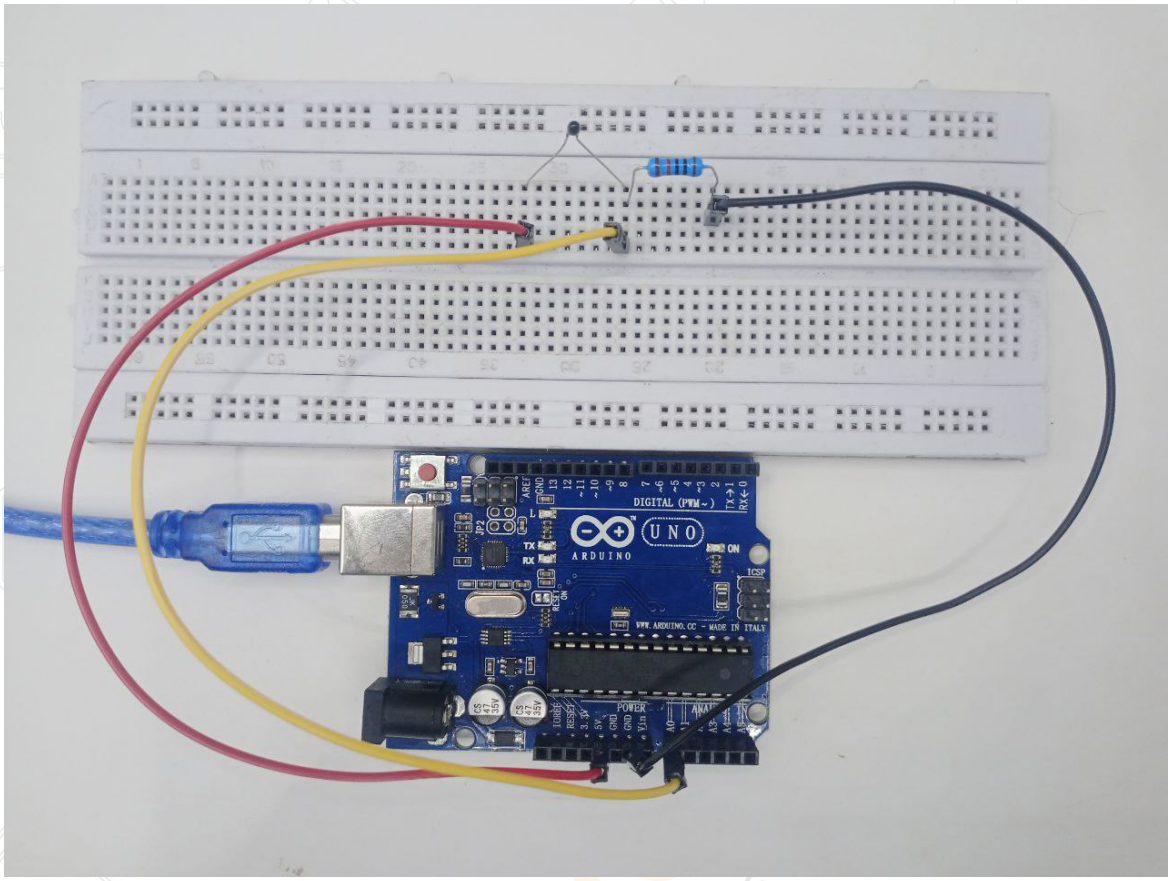


fig 21.3: Thermistance_illustration

Code

```
#include <math.h>

double Thermister (int RawADC) {

double Temp;

Temp = log (((10240000/RawADC) - 10000));

Temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * Temp * Temp ))* Temp );

Temp = Temp - 273.15;

return Temp;

}

void setup () {

Serial.begin (9600);

}

void loop()

{Serial.print (Thermister (analogRead (A0)));

Serial.println ("c");

delay (1000); }
```

fig 21.4: Thermistance_code

REMERCIEMENTS

Félicitations d'avoir exploré ce guide d'utilisation !

Nous espérons que ce voyage au cœur de l'électronique et de la programmation Arduino a éveillé votre curiosité et enrichi vos compétences. Chaque projet que vous réalisez est une pierre posée dans l'univers de l'innovation technologique.

Chez YoupiLab, nous croyons que chaque idée peut devenir réalité, et nous sommes ravis d'avoir pu vous accompagner dans vos premiers pas ou dans votre progression.

N'oubliez jamais : l'apprentissage est un processus continu, et les possibilités avec Arduino sont infinies. Alors, expérimentez, créez et partagez vos réalisations avec la communauté YoupiLab !

Pour toute question ou assistance, notre équipe est là pour vous aider : sales@youpilab.com
Retrouvez-nous également sur nos réseaux sociaux pour découvrir encore plus d'idées et de projets inspirants.

Encore merci d'avoir choisi YoupiLab. Continuez à explorer, et surtout... amusez-vous !

Avec toute notre passion pour l'innovation,
L'équipe YoupiLab