Arduino
Factory

Initiez-vous au monde de l'électronique avec la carte Arduino !

ArduinoFactory – Guide de démarrage

Introduction

Merci d'avoir téléchargé notre guide de démarrage sur l'Arduino. Après de nombreux cours écrit sur notre site Arduino Factory, nous avons souhaité vous proposer un petit guide qui rassemblerait les informations essentielles afin que vous puissiez démarrer dans le monde d'Arduino de la meilleure manière possible.

A la fin de ce guide vous saurez quelles cartes Arduino choisir en fonction de votre projet, créer votre propre circuit avec les composants présentés dans le guide et programmer votre projet.

Bonne Lecture,

Arduino Factory

Sommaire

Qu'est-ce que l'Arduino ?

L'Arduino est une **carte électronique** créée pour simplifier l'électronique à ceux qui veulent débuter ou se perfectionner dans l'électronique.

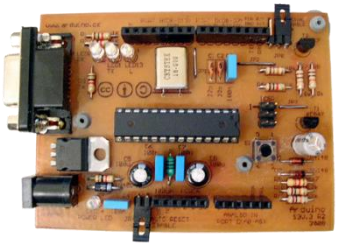
Elle permet de rendre accessible le monde de l'électronique et de créer des projets facilement, notamment la domotique, le pilotage de robot, les systèmes embarquées…

La carte Arduino possède des entrées et des sorties programmables qui permettent de contrôler beaucoup de composants : moteur à courant continu, servomoteur, photorésistance, télécommande, bouton poussoir, capteur de distance…

Un peu d'histoire

L'Arduino est à l'origine un **projet d'étudiants** de l'école de Design en Italie. Dans **les années 2000**, les outils de conception de projets électroniques étaient difficilement accessibles pour ceux qui ne faisaient pas partie du domaine. Maîtriser l'électronique et utiliser des composants demandait beaucoup de temps et d'apprentissage et ralentissait fortement le processus de création pour les jeunes étudiants.

Les étudiants faisant partie de ce projet vont donc créer une plateforme plus abordable et plus simple à utiliser appelée Arduino.



Celle-ci repose sur l'environnement de développement ***Processing*** mis au point en 2001 par le Massachusetts Institute of Technology.

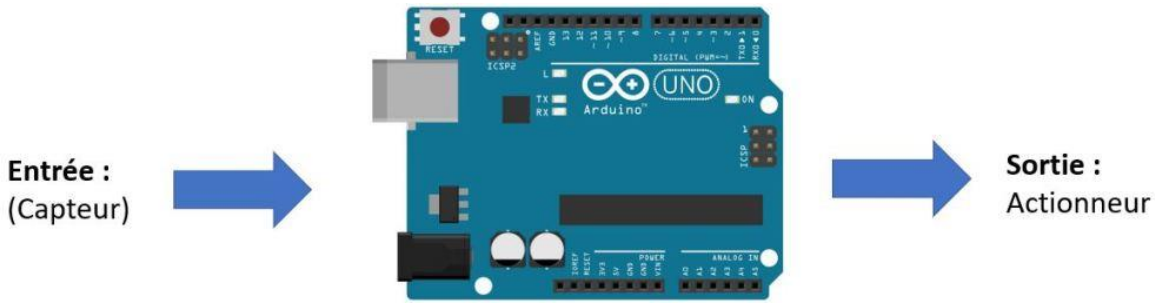
Aidés par leurs professeurs, ils conçoivent la toute première carte Arduino en 2005. Entièrement open source, l'Arduino présente l'avantage d'être multiplateforme et d'être facile à prendre en main. La carte Arduino créée par le groupe d'étudiants s'appelle *Wiring* et ça sera la première commercialisée. Cette carte est programmable avec le logiciel *Processing*. Un logiciel a ensuite été développé spécialement pour les cartes Arduino, appelé ***Arduino IDE***.

La carte Arduino

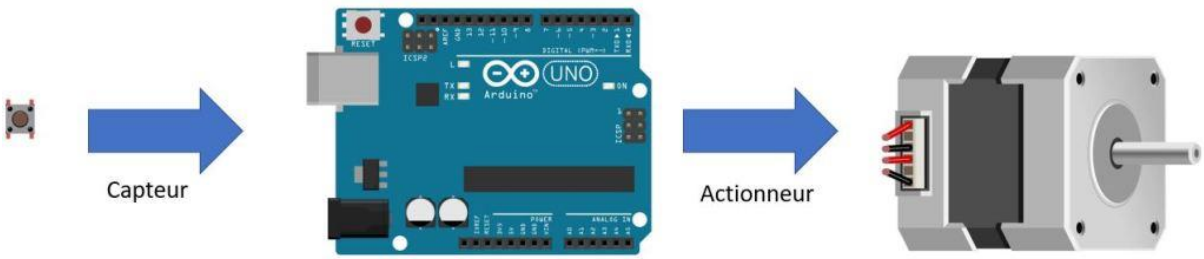
La carte Arduino est une petite carte électronique programmable, dotée d'une puissance de calcul permettant de contrôler des composants et d'en récupérer des valeurs. Elle est destinée aux débutants et amateurs d'électronique désirant réaliser ses propres projets. La marque est déclinée en plusieurs cartes : UNO, MEGA, NANO… La carte Arduino la plus connue et utilisé est l'Arduino UNO. C'est cette carte qui sera utilisée pour les schémas et circuits de notre guide.

La carte Arduino est en **source libre**, cela veut dire que vous pouvez en construire ou en vendre une vous-même. Enfin elle se programme avec le logiciel **Arduino IDE**, correspondant au langage de programmation Arduino, proche du C et C++.

La carte Arduino est équipée d'un microcontrôleur. Celui-ci va permettre de traiter l'information entrante, comme la valeur d'un capteur grâce à un programme et de commander des actionneurs (=sorties de la carte).



Ci-dessous vous avez un exemple de ce que l'on peut faire avec cette carte. Nous avons pris l'exemple d'allumer un moteur en appuyant sur un bouton poussoir avec la carte Arduino :



Pour que ce projet fonctionne, vous aurez besoin de programmer la carte :

- Quand le bouton poussoir est appuyé, on allume le moteur.
- Quand le bouton poussoir n'est pas appuyé, le moteur est éteint.

Le programme devra être téléversé dans la carte pour que le projet fonctionne. Pour cela vous aurez besoin d'un logiciel de programmation, appelé Arduino IDE, qui va transformer votre code en langage machine et le copier dans la carte.

Comment la carte interagit avec les composants électroniques ?

Dans cette partie, nous allons expliquer comment la carte Arduino se rend compte que l'on a appuyé sur le bouton poussoir, par exemple. Comme tout ordinateur, la carte Arduino fonctionne avec des 0 et des 1.

Pour la carte Arduino :

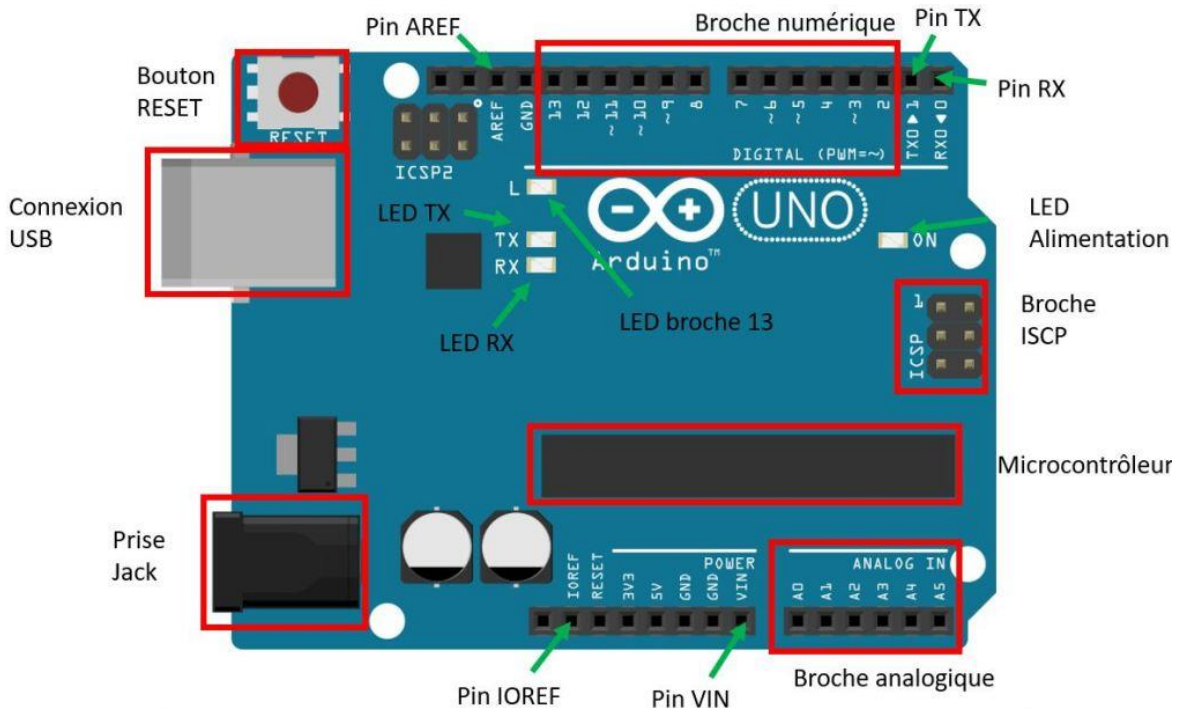
- 0 => état bas : 0V
- 1 => état haut : 5V ou 3,3V

Pour l'exemple du bouton poussoir, nous allons l'alimenter en 5V. Lorsque vous appuyez sur le bouton, cela ferme le circuit et la carte Arduino détecte un changement de tension (de 0V à 5V). Grâce à notre programme, la carte Arduino allume le moteur quand la tension provenant du bouton poussoir est de 5V.

Une fois que le bouton poussoir n'est plus appuyé, le circuit est réouvert et la tension aux bornes du bouton poussoir retombe à 0V. Le moteur est alors éteint par la carte Arduino.

Composants sur la carte Arduino UNO

Nous allons maintenant voir plus en détail les composants situés sur une carte Arduino UNO :



Connexion USB :

- Permet d'alimenter la carte Arduino en 5V
- Permet de téléverser le programme sur la carte Arduino

Prise Jack :

- Donne la possibilité d'alimenter la carte Arduino à l'aide de pile ou de batterie. Cela rend la carte Arduino autonome contrairement à la prise USB où la carte qui doit être reliée à un ordinateur

Broche Analogique :

- Permet de lire ou d'envoyer une tension entre 0V et 5V. Un signal analogique peut prendre une infinité de valeur de 0V et 5V ce qui n'est pas le cas d'un signal numérique. Une valeur analogique peut-être de 1.29V par exemple.

Broche numérique :

- Permet de lire ou d'écrire sur la carte Arduino une tension soit de 0V ou de 5V, ce qui correspond pour la carte à 0 ou 1.
- Permet de contrôler des composants comme des servomoteurs ou même des capteurs nécessitant un signal numérique.

LED broche 13 :

- LED intégrée à la carte Arduino UNO et reliée à la broche 13 de la carte. Cette LED s'allume quand la broche 13 reçoit du courant.

Bouton RESET :

- Permet de redémarrer la carte et de relancer le programme qu'elle contient.

Broche TX/RX :

- Les broches TX et RX font partie d'un BUS de données appelé l'UART (Universal Asynchronous Receiver Transmitter). Ils permettent à notre carte Arduino de communiquer avec d'autres appareils comme un ordinateur par exemple. Ces deux broches n'ont pas le même rôle que les broches numériques, donc elles ne peuvent pas contrôler vos composants. Les broches TX/RX sont utilisées pour la communication série, notamment lors de la liaison USB avec votre ordinateur, par exemple pour connecter la carte à Arduino IDE.

LED RX/TX :

- LED qui clignotent quand les liaisons TX ou RX sont utilisées. Elles permettent notamment de savoir que votre programme s'est bien uploadé sur la carte.

LED Alimentation :

- S'allume quand la carte est alimentée. Permet de savoir si celle-ci fonctionne.

Pin AREF :

- Ce pin contient un convertisseur analogique/numérique. Il permet de convertir un signal allant de 0V à 1.5V.

Pin IOREF :

- Permet d'envoyer une tension de référence de la carte. En l'occurrence pour une carte Arduino UNO, du 5V.

Pin VIN :

- Permet d'alimenter la carte Arduino avec du 5V, à la place du câble branché à l'ordinateur par exemple.

Broche ISCP :

- Les broches ISCP sont utilisés pour charger le programme de démarrage de la carte si celui-ci est manquant ou endommagé dans votre carte Arduino.

Microcontrôleur :

- Permet de stocker et exécuter le programme. Sur la carte arduino UNO R3 c'est un Atmega328P.

Les différentes cartes Arduino

Il existe plusieurs modèles de cartes Arduino, développées pour répondre à divers besoins en électronique. Que ce soit une carte Arduino Méga pour un projet de grande envergure ou une carte électronique légère et compacte pour des drones, nous allons vous présenter toutes ces cartes afin de vous aider à choisir celle qui correspond le mieux à votre projet.

- **Arduino UNO**

L'Arduino UNO est la plus connue et utilisée des cartes Arduino. Elle a été créée juste après la carte Arduino Wiring, c'est donc la carte la plus vielle encore vendue sur le marché. Voici un tableau de ses caractéristiques :

Arduino Uno	Dimension	Poids	Alimentation	Broche Analogique	Broche Numérique	Microprosseur
	74 x 53 x 15 mm	25 grammes	Prise USB 5V Prise Jack 7-12V	6 entrées	14 entrées	ATMega328 cadencé : 16 Mhz

Comme vous pouvez voir, la carte Arduino UNO a de nombreux avantages. Elle est légère, pas trop grande pour en faire un système embarqué et son prix est raisonnable. Son plus grand avantage, c'est la carte la plus utilisée donc vous trouverez de nombreuses ressources pour vous aider, nous vous la conseillons si vous débutez !

Néanmoins si vous faites des projets plus complexes, il va peut-être vous falloir plus de pins analogiques ou numériques pour contrôler vos composants. Dans ce cas-là vous aurez besoin d'une carte Arduino Méga !

- **Arduino Méga**

La carte Arduino Méga possède 10 entrées analogiques de plus qu'une carte Arduino UNO, ce qui peut être très utile si vous voulez récupérer la valeur de plusieurs capteurs.

Arduino Méga	Dimension	Poids	Alimentation	Broche Analogique	Broche Numérique	Microprosseur
	101.52 x 53.3 mm	37 grammes	Prise USB 5V Prise Jack 7-12V	16 entrées	15 entrées	ATMega2560 cadencé : 16 Mhz

Un des inconvénients de cette carte est sa taille et son poids. En effet, c'est la carte la plus grande et la plus lourde des cartes Arduino. Elle ne convient donc pas pour les projets de système embarqué comme un drone par exemple. Pour ce type de projet vous aurez plutôt besoin d'une carte légère et petite comme l'Arduino Nano.

- **Arduino Nano**

La carte Arduino Nano a été créé en 2008. Etant la carte la plus petite et la plus légère, elle permet de faire des projets irréalisables avec la carte Arduino UNO et la carte Arduino Méga à cause de leur taille et de leurs poids. Elle est notamment utile pour tous les projets électroniques devant tenir dans une main, comme une radio, une voiture télécommandée, une batterie portable…

Arduino Nano	Dimension	Poids	Alimentation	Broche Analogique	Broche Numérique	Microprosseur
	18 x 45 mm	7 grammes	USB mini-B, de 7 à 12V	8 entrées	14 entrées	ATmega328 : cadencé : 16 Mhz

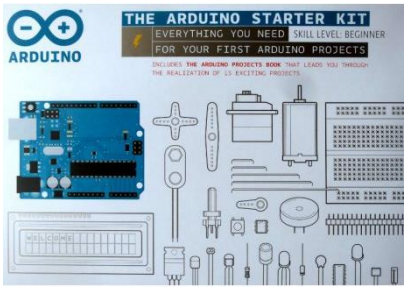
Un des défauts de la carte Arduino Nano et qu'elle ne possède pas de prise jack. Cela peut être compliqué pour faire des projets autonomes parce que vous ne pourrez pas la brancher sur des piles depuis la prise Jack comme sur une carte Arduino.

Choix d'un kit Arduino

Quand on débute sur Arduino, il n'est pas toujours évident de savoir quels composants acheter en plus de la carte Arduino pour créer ses propres circuits. C'est pour cela que des kits pour débutants ont été créés, afin de fournir tous les composants nécessaires pour bien démarrer sur Arduino.

Nous vous proposons deux kits populaires sur Arduino : Le kit officiel Arduino et un kit Elegoo.

Kit Officiel Arduino



Le kit officiel réalisé par Arduino vous permet d'avoir une carte Arduino UNO R3 officiel. Dans ce kit vous aurez plus d'une centaine de composants qui vous permettront de réaliser votre propre projet.

Vous retrouverez notamment une Breadboard, des résistances, des LEDS qui sont essentiels pour la création de circuits électroniques.

Vous disposez également de différents modules, tels qu'un moteur à courant continu, un servomoteur et un petit buzzer. Un afficheur LCD 16x2 permet d'ajouter une interface utilisateur à vos projets. Le kit est livré avec de petites boites à l'intérieur d'une grande, ce qui facilite le rangement de vos composants.

Un livret en français de 170 pages va permettre de bien comprendre le fonctionnement des composants contenu dans le kit. Ce livre contient 15 projets avec schéma de circuit, code et explication. Ce guide est très bien écrit et conçu pour les débutants, expliquant non seulement comment construire chaque projet, mais aussi pourquoi les composants sont utilisés de cette manière.

Voici un exemple de projets contenues dans le livre sur Arduino et qui vont aideront en plus de notre guide :

- **Connaître ses outils :** Introduction aux notions de base.
- **Lampe Coloré :** Produire n'importe quelle couleur avec une lampe.
- **Musique Illuminé :** Créez un instrument de musique dont vous jouez en agitant les mains.
- **Horloge Digital :** Un sablier lumineux qui améliore votre concentration au travail.
- **Roue motorisée :** Une roue colorée qui vous fera tourner la tête.

Le kit Arduino officiel coûte environ 95 euros sur Amazon.

ELEGOO UNO R3 Project Kit

Le kit Elegoo est livré avec une carte ELEGOO UNO R3, qui est une version compatible avec l'Arduino UNO R3. La carte fonctionne parfaitement avec le logiciel Arduino IDE. Elle est accompagnée d'un câble USB pour la connexion à l'ordinateur, ce qui facilite la programmation des projets.

Le kit Elegoo comprend également une large gamme de composants électroniques, tels que des résistances, des condensateurs, des diodes, des transistors et des boutons-poussoirs. Ces composants de base sont essentiels pour la création de circuits électroniques et permettent aux utilisateurs de comprendre les principes fondamentaux de l'électronique.

De plus, le kit est livré avec différents modules tels que des capteurs de lumière, des capteurs de température, des servomoteurs, des écrans LCD, des modules de communication sans fil, etc. Ces modules offrent une grande flexibilité pour la réalisation de projets avancés.

Sur certains composants comme l'afficheur LCD ou encore les moteurs, les broches ont été pré-soudées et les câbles ont été assertis pour vous afin que vos montages soient plus faciles et rapides.

Vous avez aussi un guide d'utilisation français en PDF gratuit incluant 33 projets avec les composants du kit avec librairie, le programme et le schéma du circuit pour chaque composant.

Son prix est actuellement de 70 euros. Vous pouvez le retrouvez sur Amazon.

Arduino IDE

L'environnement de développement intégré Arduino est un logiciel de programmation qui va faire l'interface entre votre carte Arduino et le programme. Arduino IDE possède un compilateur qui va transformer votre programme en langage machine compréhensible par la carte Arduino.

L'Arduino IDE est un dérivé du logiciel Processing. C'est un logiciel Open Source et il peut être utilisé pour programmer des cartes autres qu'Arduino.

Téléchargement d'Arduino IDE

Vous pouvez télécharger le logiciel Arduino IDE sur le site Arduino. Il est disponible sur Windows, Mac et Linux et open Source.

Vous pouvez aussi le trouver sur le magasin d'applications de votre système en recherchant « Arduino » sur le Microsoft Store, Apple Store et Linux App store.

Voici à quoi ressemble le logiciel une fois installé :

Découverte d'Arduino IDE

Le logiciel Arduino IDE possède plusieurs fonctionnalités qui vont vous être utile pour créer vos programmes.

I. Le menu

Nous allons maintenant voir les fonctions les plus utiles du menu d'Arduino IDE.

- **Fichier :** Permet de sauvegarder votre programme, de changer la police du texte dans les préférences, ou d'imprimer votre programme. Cet onglet contient aussi des exemples de programme déjà fait.
- **Editer :** Permet de copier notre programme, le commenter, aller à une ligne particulière ou bien même formater un programme récupérer sur un forum.
- **Croquis :** Regroupe les mêmes fonctions que les boutons en dessous : vérification, téléversement. Vous avez en plus de quoi ajouter un fichier à votre projet et de gérer les librairies.
- **Outils :** Contient le manageur de librairie, le moniteur série, les différentes carte Arduino que vous devez choisir pour téléverser.
- **Help :** Contient toutes les informations sur la version du logiciel et comment utiliser celui-ci.

II. Les boutons

Nous allons maintenant voir les boutons les plus utiles d'Arduino IDE.

a) Bouton Vérifier

Le bouton *vérifier* permet au logiciel de savoir si votre programme a bien été écrit. Il va vérifier si vous avez bien téléchargé la librairie que vous souhaitez utiliser ou bien qu'il ne manque ni parenthèse ni point-virgule à votre programme. Cela n'assure pas que votre programme va fonctionner comme vous le souhaitez.

b) Bouton Transférer

Le bouton *transférer* permet d'envoyer votre programme vers votre carte Arduino.

Le logiciel va d'abord vérifier votre programme comme avec la fonction vérifier, vous aurez donc aussi les problèmes de syntaxe s'affichant. Puis ce programme sera envoyé par un câble sur la carte Arduino.

c) Bouton de Debug

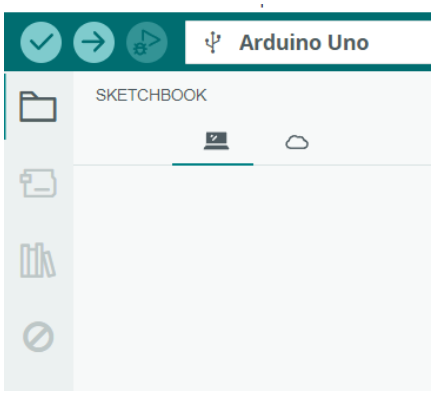
Le bouton *Debug* permet de dérouler le programme de manière contrôlée, avec l'aide d'une interface matérielle qui peut aider à naviguer dans l'exécution du programme.

Cela peut aider à mieux comprendre le programme et à repérer les failles potentielles et les erreurs de code. Ceci n'est malheureusement disponible qu'avec les cartes SAMD. La fonction n'est donc pas compatible avec la carte Arduino UNO ni MEGA et seulement avec la NANO version IOT.

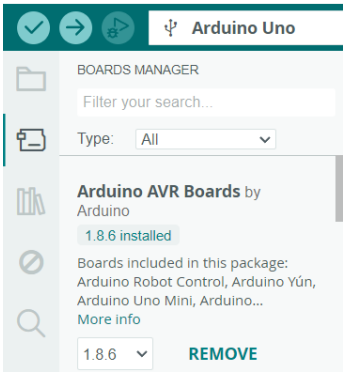
III. Les Onglets

Le logiciel Arduino IDE vous propose différents onglets qui permettent de retrouver vos projets, vos librairies installées et même la gestion des cartes :

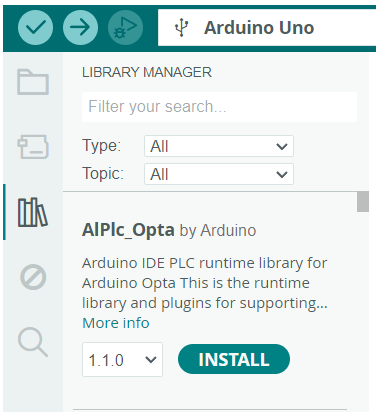
Dans *Sketchbook* vous retrouverez tous vos projets réalisés sur votre ordinateur ou bien même ceux sauvegardé dans le cloud :



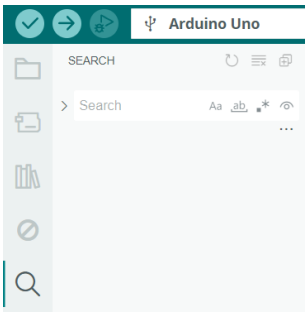
Dans la partie *Board manager* vous aurez la possibilité d'installer de nouvelles cartes Arduino ou non pour les programmer sur Arduino IDE :



Le librairie Manager va vous permettre de gérer vos librairies installées sur Arduino IDE pour programmer votre carte :



Vous avez un *debugger* qui permet de dérouler votre programme étape par étape. Ceci fonctionne seulement pour quelques cartes Arduino :

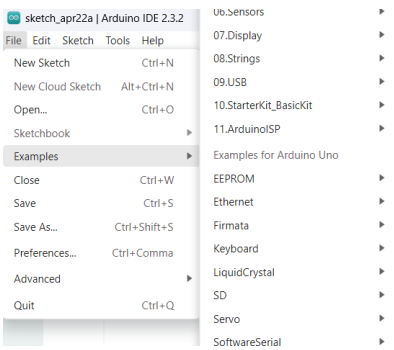


Dans le dernier onglet, *search*, est une barre de recherche.

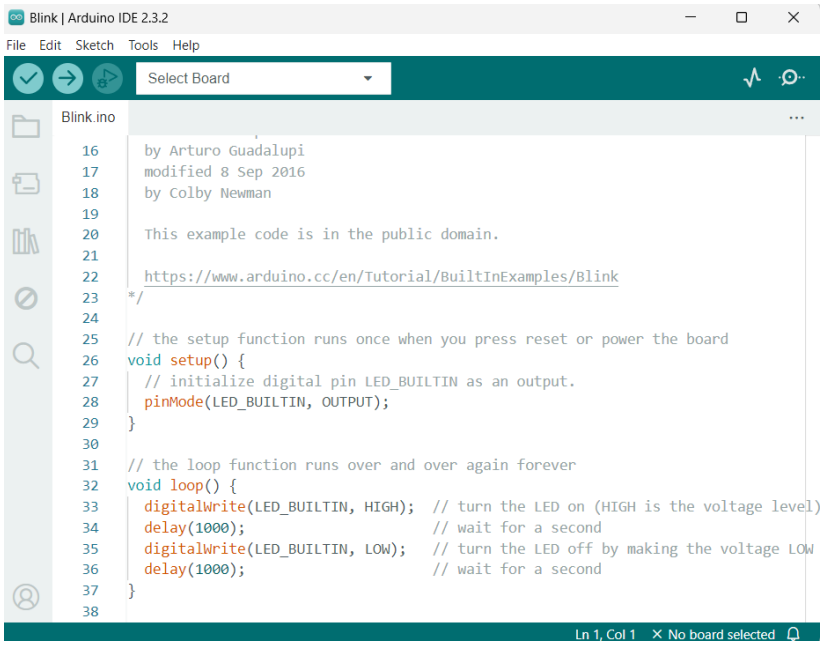
Des programmes inclus dans Arduino IDE

Sur Arduino IDE, vous avez des codes déjà pré-fait. Ils peuvent vous être utile pour voir comment un composant fonctionne et pour le tester rapidement.

Nous allons retrouver ces codes dans *fichier* puis *examples :*



Voici un exemple de programme que vous pouvez retrouver dans la section Example. Il permet de faire clignoter la led integrée sur la broche 13 de la carte Arduino :



Uploader son programme sur votre carte Arduino

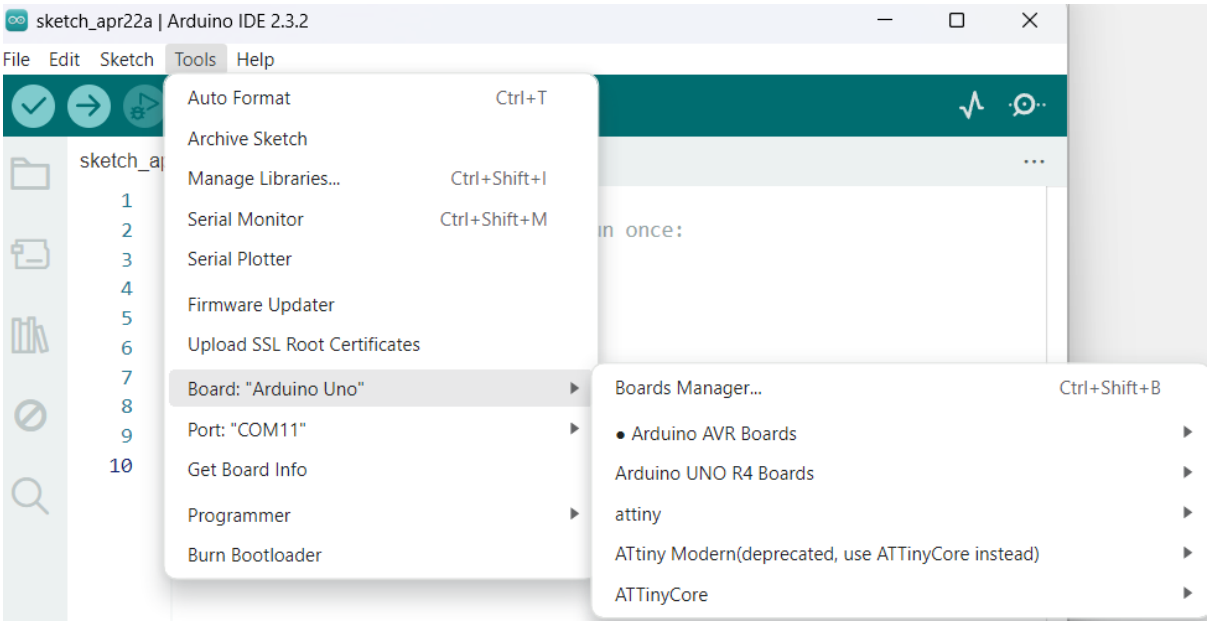
Une fois le programme vérifié et terminé, il est temps de le téléverser sur la carte. Pour cela vous devez brancher votre carte Arduino sur votre ordinateur.

A. Choisir le bon type de carte

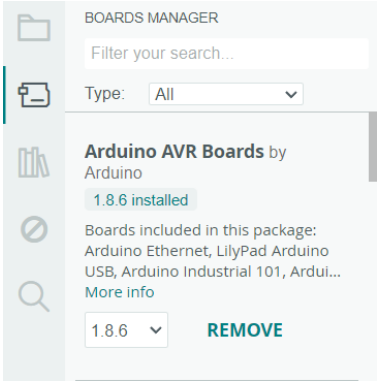
Le logiciel Arduino IDE ne va pas téléverser le programme de la même manière si c'est une carte Arduino Uno ou Nano, car le microprocesseur à l'intérieur n'est pas le même.

Dans la version actuelle d'Arduino IDE, en branchant votre carte Arduino celle-ci est directement reconnue par le logiciel :

Si votre carte n'est pas directement reconnue, vous pouvez la sélectionner dans le menu *Tools* puis *Boards* :

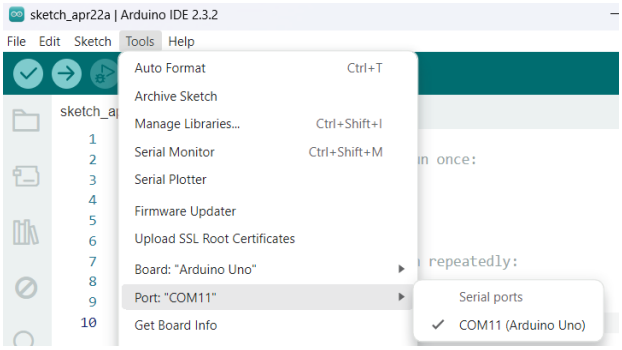


Si vous ne trouvez pas votre carte dans liste, vous pouvez cliquer sur *Board Manager* et télécharger le module correspondant à votre carte.



B. Choisir le Port COM

Si votre carte Arduino n'est pas reconnue directement, vous devez choisir le Port COM sur lequel est branché la carte Arduino. Souvent, le logiciel Arduino IDE vous le propose directement, comme ceci :

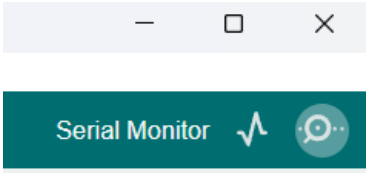


Il vous reste plus qu'à téléverser le programme avec le bouton que l'on a vu plus tôt.

C. Moniteur série

Le moniteur série est une interface entre l'utilisateur et la carte Arduino qui va recevoir des informations de la carte Arduino et les afficher sur Arduino IDE. Ces informations peuvent-être la température en degré donnée par un capteur ou bien l'angle d'un servomoteur.

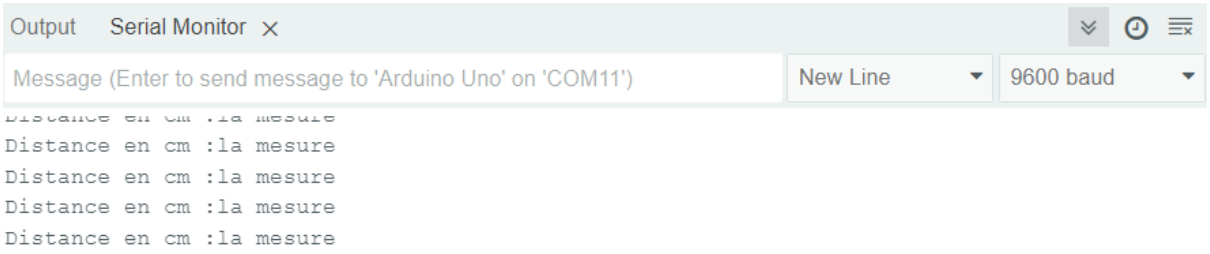
Le moniteur série peut s'ouvrir seulement quand la carte Arduino est connectée et que le programme est téléversé dessus. Vous pouvez l'ouvrir avec le bouton en haut à droite du logiciel. L'utilisateur peut aussi entrer des valeurs depuis le moniteur série pour contrôler un servomoteur par exemple.



Pour avoir des valeurs sur le moniteur série, vous devez le paramétrer dans votre code :

```
void setup() {
    Serial.begin(9600) ; // Initialisation de la communication avec le moniteur série (9600 bits/s)
}
void loop() {
    Serial.print("Distance en cm :"); // Permet d'afficher sur le moniteur série sur la même ligne
    Serial.println("la mesure"); // Permet de sauter une ligne après le texte
}
```

Le moniteur série doit être configuré avec un nombre précis de bit/s, ici 9600. Ceci va être utile pour reçevoir les informations depuis le moniteur série. Assurez-vous que débit du moniteur série correspond à la valeur que vous avez défini sur la carte Arduino. Vous pouvez laisser par défaut 9600 bit/s si votre projet ne nécessite pas de débit spécifique.



Langage Arduino

Le langage Arduino va vous permettre de programmer votre carte Arduino UNO sur Arduino IDE. Le langage Arduino est dérivé du langage C. Ces deux langages sont très utilisés pour programmer des cartes électroniques, c'est logiquement que le langage Arduino s'en inspire.

La plus grande différence entre le langage Arduino et le C est qu'un programme Arduino contient obligatoirement deux fonctions : *void setup()* et *void loop()* alors que le C contient une seule fonction : le *void main().*

Dans ce guide gratuit nous n'allons pas voir en détails le langage Arduino. Néanmoins les programmes présentés seront expliqués en détails afin que puissiez comprendre le circuit et le modifier si besoin.

A. Structure du programme

Il y a deux règles importantes pour commencer à coder en langage Arduino :

- Toutes les actions écrites doivent être terminées par un point-virgule afin que la carte Arduino comprenne que l'action est terminée.
- Toutes les fonctions commencent et se terminent par des accolades pour que la carte Arduino comprenne quand commence et termine la fonction.

Ce sont deux choses que l'on oublie souvent lorsqu'on écrit un programme et qui vont poser problème lors de la compilation de celui-ci.

B. Void setup()

Le *void setup* est une fonction que l'on écrit au début du programme, entre l'initialisation des variables et le *void loop*. Le *void setup* contient l'initialisation des composants comme entrée ou sortie de la carte Arduino, de l'initialisation du moniteur série que l'on va utiliser dans le reste du programme.

La fonction *void setup* est obligatoire dans tous vos programmes Arduino, même s'il n'y a rien écrit dedans, ne pas la mettre va créer une erreur.

Le *void setup* est une fonction qui va s'exécuter qu'une seule fois au début du programme.

C. Void loop()

Contrairement à la fonction *void setup* qui s'exécute une seule fois, la *void loop* s'exécute à l'infinie. Ceci va permettre de contrôler vos composants sans jamais avoir à relancer le programme. La fonction *void loop* contient l'ensemble des fonctions pour lire les mesures de vos capteurs et les afficher sur le moniteur série. Vous pouvez aussi contrôler des composants, comme des LEDS, des servomoteurs…

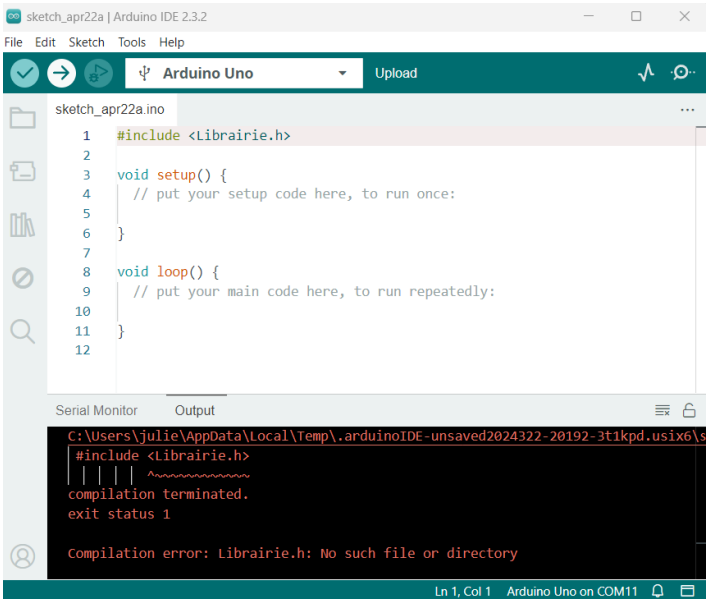
La fonction *void loop* est obligatoire dans tous vos programmes Arduino, même s'il n'y a rien écrit dedans, ne pas la mettre va créer une erreur.

Bibliothèque sur Arduino

Les librairies (ou bibliothèques) sont un ensemble de fonctions permettant de simplifier l'utilisation d'un capteur ou d'une fonctionnalité.

Dès qu'un programme Arduino contient une ligne commence par **#include** alors, il appelle une librairie. Une librairie n'est pas indispensable dans un programme. Néanmoins elle vous permettra de faciliter la manipulation de composant ou encore d'alléger votre code. C'est toujours plus simple de partir d'une base et de ne pas réinventer la roue en partant de zéro !

Quand installer une librairie ?

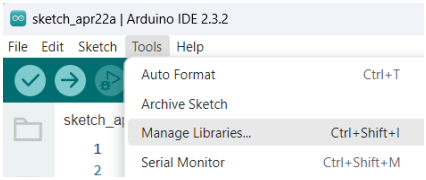


Lors de la compilation d'un programme, Arduino IDE vérifie que la librairie est bien disponible.

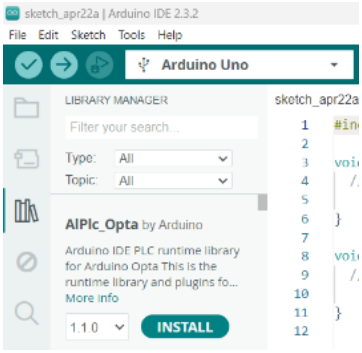
Si la librairie n'est pas incluse le programme affiche alors le message d'erreur suivant : « *Librairie.h: No such file or directory* ».

Ajouter une librairie sur Arduino IDE

A. Gestionnaire de librairie

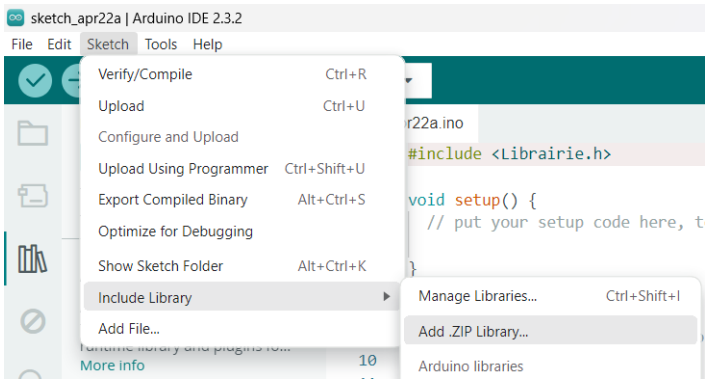


Dans Arduino ide, aller dans *tools* puis *Manage Libraries…* :



Dans le gestionnaire de librairie, tapez le nom de la librairie que vous désirez, puis téléchargez-la.

B. Importer une bibliothèque .zip



Vous pouvez importer directement une librairie depuis votre ordinateur en cliquant sur *sketch*, *include library* et *Add .ZIP Library* :

Où trouver les librairies en zip ?

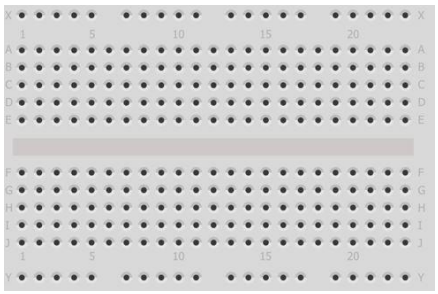
Il y a plusieurs manières de trouver une librairie pour un composant. Quand vous trouvez un code sur internet, il y a de grande chance que la librairie soit donnée avec, comme dans nos cours par exemple.

Néanmoins si vous voulez faire votre propre code, nous vous conseillons un site qui répertorie des librairies.

Breadboard

La breadboard est une platine d'expérimentation permettant de réaliser et tester un circuit électronique. L'avantage de ce système est qu'il est complètement réutilisable puisqu'il ne nécessite pas de soudure.

Le mot breadboard fait référence aux plaques de bois, utilisées pour concevoir des circuits, sur lesquelles étaient fixées les composants électroniques.



La breadboard va vous permettre de créer des prototypes de vos circuits avant d'utiliser une plaque PCB pour les souder.

C'est un élément essentiel pour les débutants parce qu'elle va vous permettre de tester des composants facilement et d'apprendre rapidement comment fonctionne l'Arduino !

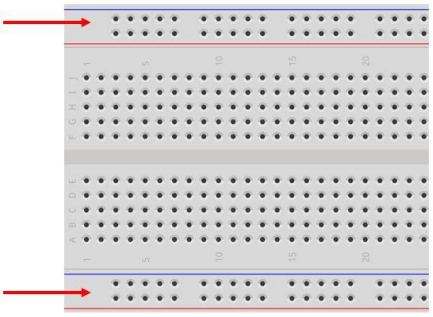
A) Fonctionnement de la Breadboard

Une breadboard est composée de trous de profondeur 0.1 inch (2.54 mm) pour enfoncer vos composants. Ces trous sont interconnectés ensemble horizontalement et verticalement.

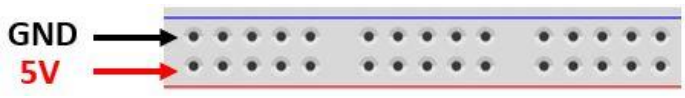
a) Trou connecté à l'horizontale

Les seules lignes connectées ensemble horizontalement se trouvent à l'extrémité de la breadboard.

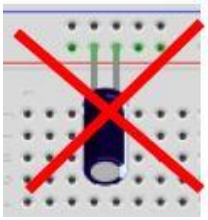
Il y a deux lignes de chaque côté de la breadboard, le + et le – qui sont ici symbolisé par la ligne rouge et la bleue.



En branchant du 5V sur le + et la masse (GND) sur le -, vous pourrez brancher tous vos les composants sur ces mêmes lignes afin de les alimenter. Ces lignes sont connectées ensemble en parallèle.



Attention à bien positionner votre composant !



Ceci est impossible, vous ne pouvez pas mettre un composant dans ce sens si les trous sont connectés horizontalement car vous aller créez un court-circuit et endommager votre composant.

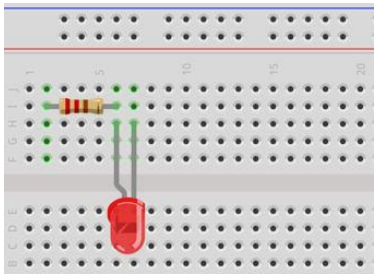
Voici une bonne utilisation de la breadboard :

b) Trou à la verticale

C'est au milieu de la breadboard que vous devez placer vos composants. Les lignes au milieu fonctionnent à l'inverse des lignes aux extrémités, elles sont liées ensemble par la verticale.

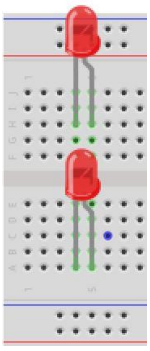
Comment placer nos composants ?

Les composants peuvent être placés à l'horizontale sur la breadboard.



Vous pouvez lier les composants entre eux en connectant un des pins du composant suivant sur un des trous de la ligne verte du dernier composant posé. Sur l'image à droite, la LED et la résistance sont connectées ensemble.

Sur les grandes breadboards, vous pouvez voir une séparation en gris clair au milieu des trous verticaux.



Cette séparation ne permet pas la liaison entre les trous verticaux du haut et du bas.

Comme on peut voir sur l'exemple à droite, les deux LEDS ne sont pas connectées sur la même ligne car la séparation en gris clair ne permet liaison entre les deux composants. Vous n'aurez donc pas de court-circuit.

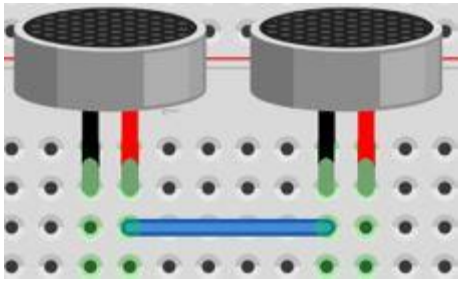
Attention à bien positionner votre composant !

Sur les breadboard, quand vous placez vos composants au milieu, vous devez faire attention que ceux-ci ne soient pas placé en vertical sur les deux mêmes trous. Vous pouvez les placer les uns à la suite des autres avec un trou en commun mais rarement deux.

Ceci est un exemple de ce que l'on ne doit pas faire. Les deux composants sont branchés sur les 2 mêmes trous donc ils sont en court-circuit.

Ceci est un exemple de ce qui fonctionne si vous voulez connecter ces deux composants.

Comment relier des composants éloignés ?



Comme nous venons d'expliquer, on peut relier deux composants proches en branchant une de leurs broches sur la même ligne de trou.

Néanmoins ceci ne fonctionne plus quand les composants n'ont pas de broche dans le même alignement vertical. Pour cela vous aurez besoin d'un fils de liaison entre les deux composants.

Fils de Liaison

Les fils de liaison Arduino sont des fils électriques avec un diamètre très petit, fait pour l'électronique basse tension. Il existe deux types d'embouts pour vos composants : les embouts mâles et les embouts femelles.

A) Les fils mâles

Ayant une pointe, un fils mâle peut être connecté sur la breadboard pour relier des composants contrairement à un fils femelle.

B) Les fils femelles

Un fils femelle est utilisé pour se brancher directement à un composant. Il permet de connecter deux composants ensemble.

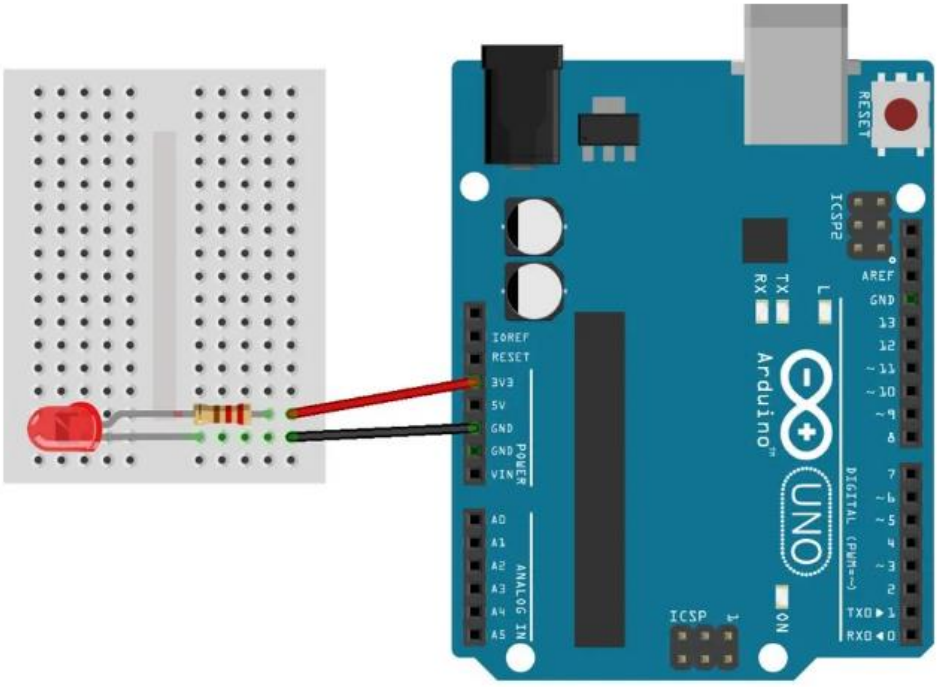
C) Les fils mâles/femelles

Un type de liaison qui est très utile aussi est la liaison mâle/femelle. Ce type de câble possède un embout mâle et un femelle. Cela peut vous servir à brancher directement un composant sur la breadboard ou la carte Arduino.

Votre premier circuit : Allumer une LED

C'est le moment d'utiliser votre Arduino pour la première fois ! Pour ce premier circuit on va voir un exemple de base en électronique : allumer une LED ! Ceci va nous permettre de nous familiariser avec les entrées/sorties de la carte Arduino.

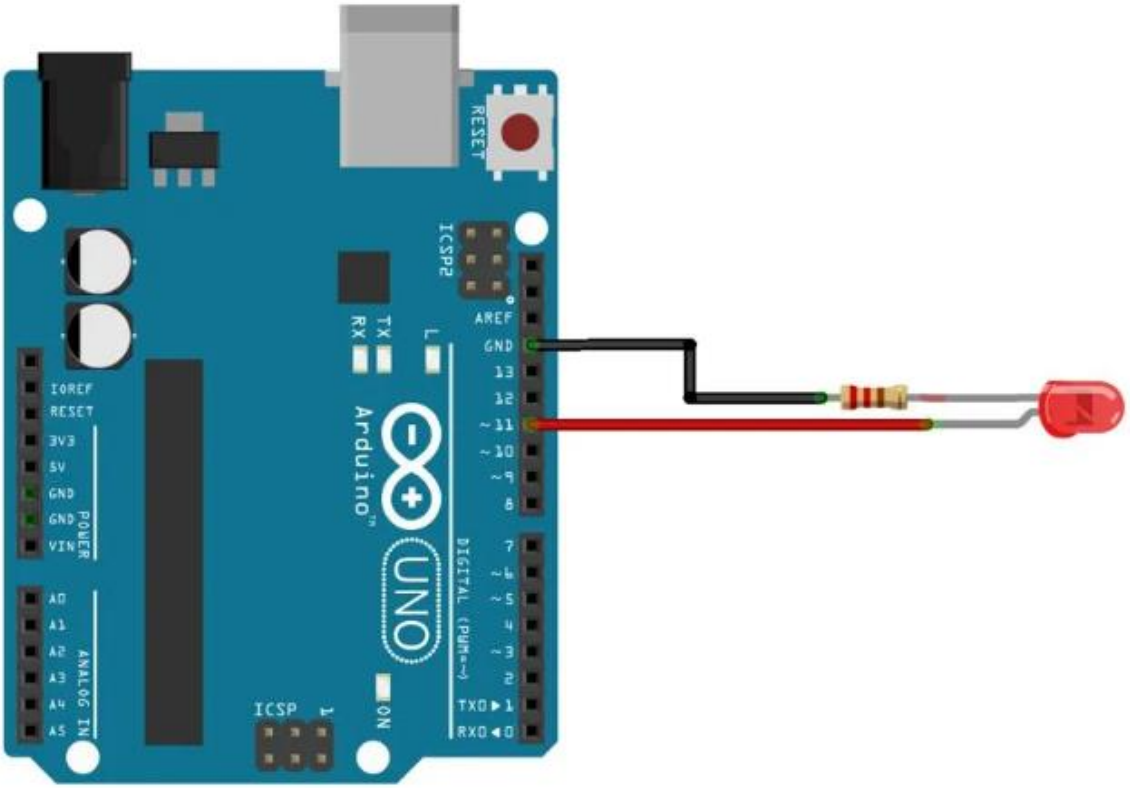
Pour ce premier circuit, nous allons brancher une LED sur le +3.3V et le GND de la carte Arduino. Pour cela nous aurons besoin de deux fils de liaisons et d'une résistance 220 ohms. Vous retrouverez tous ces composants dans les kits que l'on vous a présenté précédemment.



En allumant votre carte Arduino, vous verrez normalement la LED s'allumer. Si celle-ci reste éteinte, vous avez peut-être branché votre LED dans le mauvais sens. Pour cela, débranchez la LED et retournez-la.

Comment faire clignoter votre LED ?

Vous souhaitez peut-être faire des effets visuels avec votre LED, comme la faire clignoter. Pour cela, nous aurons besoin de brancher notre LED sur une entrée/sortie de notre carte Arduino afin de pouvoir contrôler son allumage. Nous avons donc connecté la LED à la broche 11 de la carte Arduino :



Pour pouvoir allumer et éteindre notre LED, il va falloir programmer notre carte avec Arduino IDE. Voici notre programme :

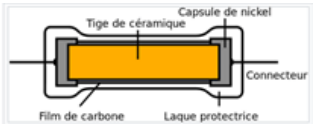
```
int led_Broche = 11;  // On assigne la LED à la broche 11.

void setup() {
    pinMode(led_Broche, OUTPUT); // On assigne la LED en sortie
}

void loop() {
   digitalWrite(led_Broche,HIGH); // On allume la LED
   delay(1000); // On fait une pause pour voir la LED allumée avant de l'éteindre
   digitalWrite(led_Broche,LOW); // On éteint la LED
   delay(1000);
}
```

Dans ce programme, vous pouvez voir que l'on allume une LED pendant 1 seconde puis on l'éteint pendant le même temps.

Les Résistances

Une résistance est un petit composant électrique qui s'oppose au passage du courant électrique afin de le limiter.

Elle est composée d'une tige de céramique. Les deux extrémités permettent de connecter la résistance au circuit à l'aide de capsules de nickel. Enfin, un revêtement de la laque protectrice isole la résistance de l'environnement extérieur et affiche ses anneaux de couleurs.



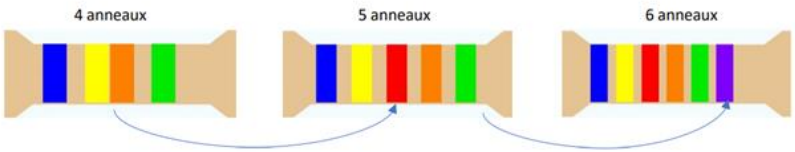
Les résistances sont souvent utilisées sur Arduino pour limiter le courant dans un circuit. Dans les circuits électroniques elles peuvent être utilisée tant que résistance de pull-up ou pull-down. Contrairement à une LED, la résistance est un dipôle non polarisé, c'est-à-dire qu'elle peut être branchée dans n'importe quel sens.

L'unité employée pour les résistances est l'Ohm et se note Ω. La loi la plus connue qui utilise la résistance est la loi d'Ohm (U=RI).

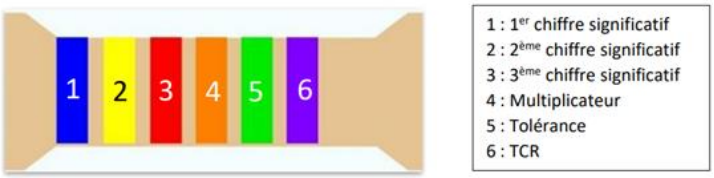
Quelle est la valeur de ma résistance ?

En ouvrant votre kit sur Arduino, vous avez peut-être vu plusieurs résistances avec des anneaux de couleurs différentes. On va voir dans cette partie comment déterminer leurs valeurs en ohms pour savoir laquelle choisir dans votre circuit.

Tout d'abord, plus la valeur de la résistance en ohm est importante et plus celle-ci va limiter le courant. Sur une résistance vous retrouvez des anneaux de couleurs. Vous pouvez avoir des résistances de 4, 5 ou 6 anneaux :



De plus chaque bandeau à une signification particulière :



Chaque anneau est une clé pour définir la valeur de la résistance. Ci-dessous, vous trouverez la signification de chaque anneau dans un tableau :

1er Chiffre Significatif	2ième chiffre significatif	3ième chiffre significatif	Multiplicateur	Tolérance	Coefficient de température (ppm/C°)
0 : Noir	0 : Noir	0 : Noir	10^{-2} Argent	± 20 % Blanc	200 Noir
1 : Marron	1 : Marron	1 : Marron	10^{-1} Or	± 10 % Gris	100 Marron
2 : Rouge	2 : Rouge	2 : Rouge	10^{0} Noir	± 5 % Or	50 Rouge
3 : Orange	3 : Orange	3 : Orange	10^{1} Marron	± 1% Marron	15 Orange
4 : Jaune	4 : Jaune	4 : Jaune	10^{2} Rouge	± 2 % Rouge	25 Jaune
5 : Vert	5 : Vert	5 : Vert	10^{3} Orange	± 0.5 % Vert	10 Bleu
6 : Bleu	6 : Bleu	6 : Bleu	10^{4} Jaune	± 0.25 % Bleu	5 Violet
7 : Violet	7 : Violet	7 : Violet	10^{5} Vert	± 0.10 % Violet	1 Gris
8 : Gris	8 : Gris	8 : Gris	10^{6} Bleu	± 0.05 % Gris	
9 : Blanc	9 : Blanc	9 : Blanc	10^{7} Violet		
			10^{8} Gris		
			10^{9} Blanc		

Le tableau peut sembler complexe à comprendre, c'est pourquoi on vous a prévu un exemple ci-dessous :

Voici une résistance à 3 anneaux de couleurs. Par rapport au tableau ci-dessus, il y a que deux chiffres significatifs, un multiplicateur et tolérance.

- 1er chiffre Significatif : Anneau rouge => 2
- 2ième chiffre Significatif : Anneau rouge => 2
- Multiplicateur : Anneau marron => x 10

La valeur de la résistance est donc de 22*10 = 220 ohms

Qu'est-ce que la tolérance d'une résistance ?

Lorsqu'une résistance est fabriquée, le fabricant tente de lui donner une certaine valeur. Cependant, les variations dans le processus de fabrication signifient que la valeur réelle de la résistance ne correspond pas toujours exactement à la valeur nominale.

La tolérance signifie donc la valeur de résistance au pourcentage près. Par exemple une valeur de 220 ohms à 5% peut avoir comme valeur 219 ohms ou 221 ohms.

Comment déterminer facilement la valeur d'une résistance ?

Il existe d'autre manière de connaître la valeur d'une résistance, on vous en propose deux dans ce guide :

- Sur notre application Arduino Factory disponible sur les téléphones Android vous avez un outil permettant de calculer automatiquement la valeur d'une résistance à partir du nombre de d'anneau et de la couleur de ceux-ci.

- Sur un multimètre vous avez une option ohmmètre qui va vous permettre de mesure la valeur de votre résistance.

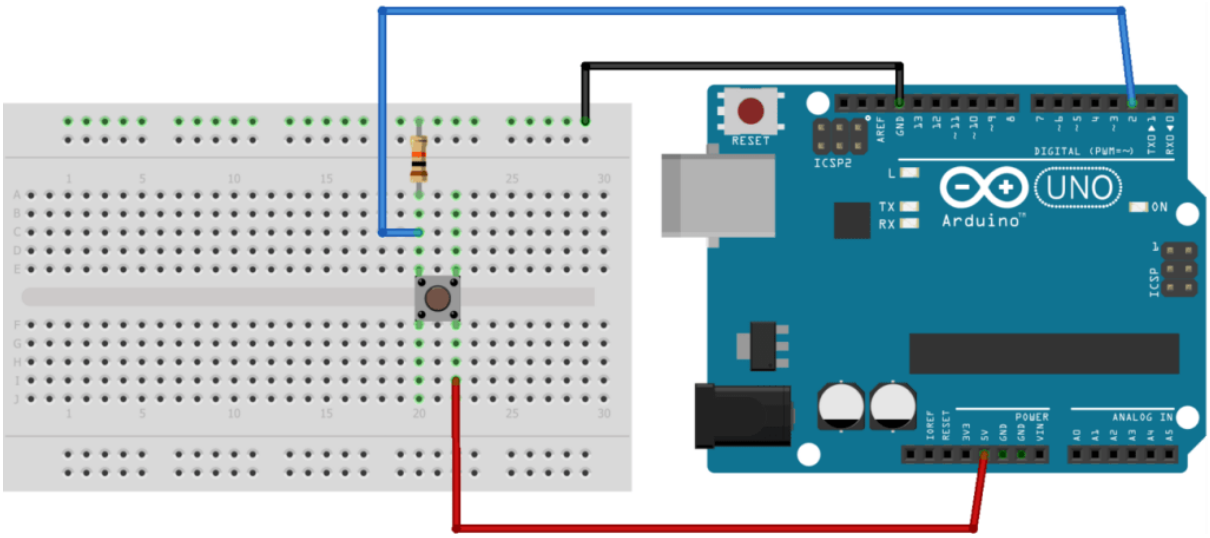
Il est maintenant temps de passer à la pratique avec la découverte de certains composants essentiels du monde d'Arduino.

Bouton Poussoir

Le bouton poussoir est un interrupteur électrique dont le bouton revient à sa position initiale après avoir été actionné. Il est utilisé partout dans la vie de tous les jours, par exemple pour appeler un ascenseur ou bien les touches du clavier de votre ordinateur.

Dans un circuit sur Arduino vous pouvez l'utiliser pour avoir une interaction de l'utilisateur. Pour ce premier circuit on va voir comment lire les valeurs des boutons poussoirs depuis la carte Arduino.

Voici le circuit que vous devez réaliser afin de pouvoir lire les valeurs du bouton poussoir :

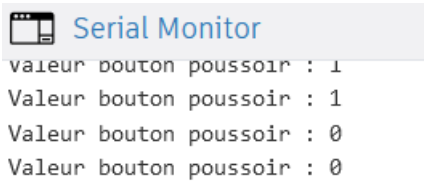


Comme vous pouvez voir sur le schéma nous avons ajouté une résistance de 10 kilo-ohms (couleur orange noir marron) afin que le signal soit bien à 0V lorsque l'on n'appuie pas sur le bouton poussoir.

Voici le programme pour lire la valeur du bouton poussoir sur Arduino IDE :

```
const int pin_INTERRUPTEUR = 2; // L'interrupteur est attaché au pin 2

void setup() {
    Serial.begin(9600);  // Initialisation de la communication avec le moniteur
série
    pinMode(pin_INTERRUPTEUR,INPUT);
}

void loop() {
    delay(1000); // Attente de 1000 ms
    boolean etatBouton = digitalRead(pin_INTERRUPTEUR); // Récupère l'état du
bouton
    Serial.println(etatBouton); // Affiche l'état du bouton sur le moniteur
}
```

Une fois le programme sur votre Arduino Uno, vous devez ouvrir le moniteur série sur Arduino IDE avec la fréquence 9600bd/s afin de lire les valeurs du bouton poussoir :

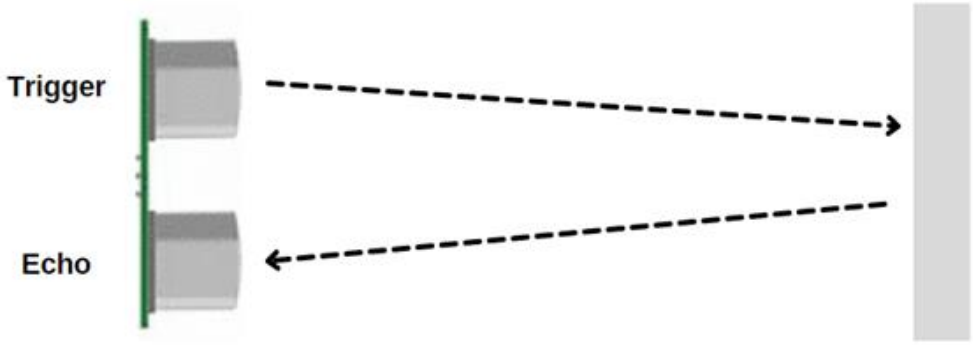


Capteur de Distance

Le capteur de distance (aussi appelé capteur ultrason) permet de réaliser des mesures de distance. Il permet d'estimer une distance allant de 2 cm à 400 cm avec une précision de 3mm. C'est le capteur de distance le plus utilisé et le moins cher. Vous pouvez l'utiliser pour diriger un robot, faire un radar de recul pour voiture et bien plus encore.

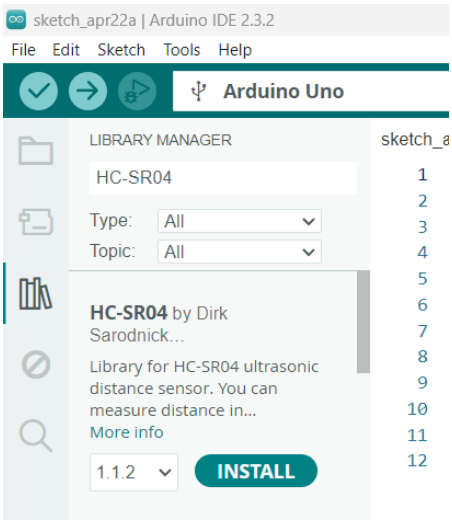
Comment fonctionne le capteur de distance ?

Pour mesurer la distance le capteur ultrason utilise un signal envoyé depuis la borne Trigger (Trig) et reçu par la borne Echo. Le temps que met la borne Echo à recevoir le signal permet de connaître la distance entre le capteur et l'objet.

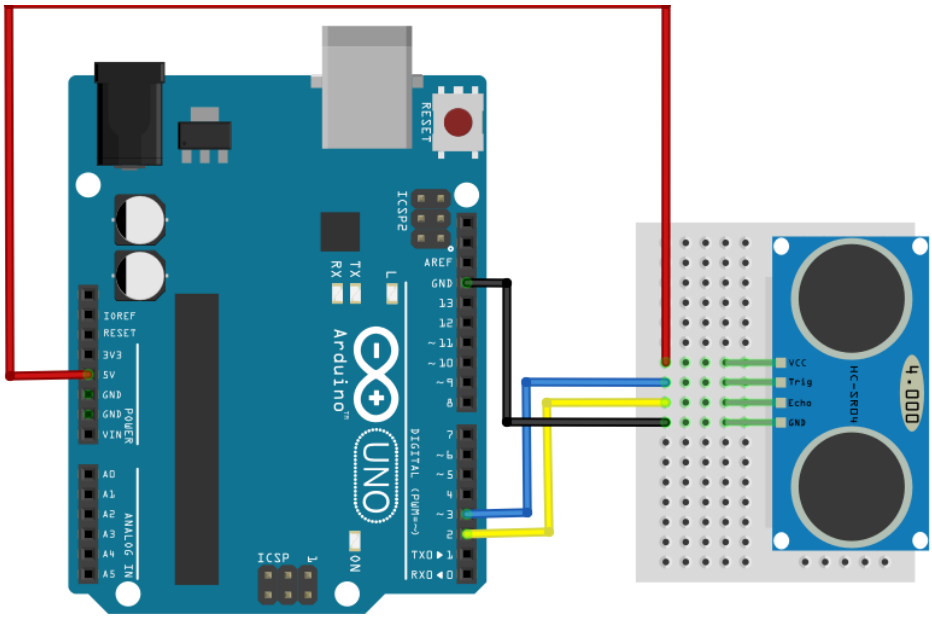


Fonctionnement du capteur de distance avec une librairie

Pour commencer à faire fonctionner le capteur, il faut installer la librairie HC-SR04 sur Arduino :



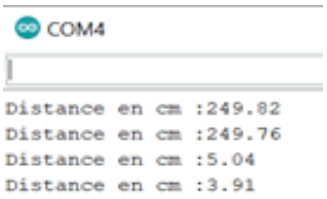
Voici le schéma du circuit pour relier le capteur de distance et la carte Arduino Uno :



Voici un programme permettant de lire une distance grâce au capteur :

```
#include "SR04.h" //Librairie

// définition des broches du capteur
const int trigPin =2;
const int echoPin =3;

UltraSonicDistanceSensor distanceSensor(trigPin, echoPin); // initialisation du
capteur avec les broches utilisées

void setup() {
    Serial.begin(9600); // Initialisation de la communication avec le moniteur
série (9600 bits/s)
}
void loop() {
    // Mesure et affiche la distance en centimètre sur le port série toutes les
500 ms
    Serial.print("Distance en cm :");
    Serial.println(distanceSensor.measureDistanceCm());
    delay (500);
}
```

Une fois le programme installé sur la carte Arduino, on peut voir la distance s'afficher sur le moniteur série :



Buzzer

Les buzzers permettent produire des sons grâce à une lame qui bouge avec l'effet piézoélectriques. Avec une carte Arduino, vous pouvez facilement contrôler un buzzer pour qu'il émette des bips, des mélodies ou des alertes sonores en fonction de vos besoins. Les buzzers peuvent être trouvés dans des dispositifs d'alarme, les ordinateurs, les minuteries…

L'effet piézoélectrique est la propriété que possède certains minéraux de se déformer quand ils sont soumis à un champ électrique. Il permet de transformer l'énergie électrique en vibration. Il permet de jouer des notes et des mélodies simples.

Types de Buzzers

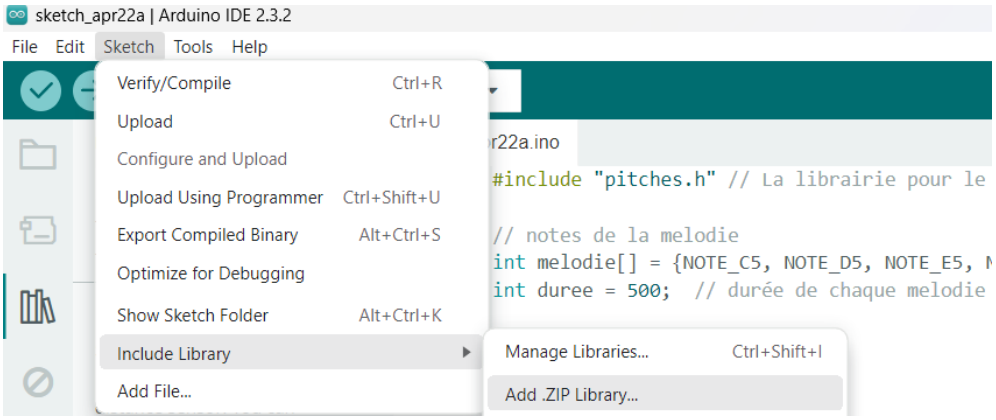
Il existe deux principaux types de buzzers : les buzzers actifs et les buzzers passifs.

- **Buzzer Actif** : Ce type de buzzer dispose d'un oscillateur intégré, ce qui signifie qu'il émet un son dès qu'il est alimenté. Il ne nécessite pas de signal de commande complexe et peut être directement alimenté par une sortie numérique de l'Arduino.

- **Buzzer Passif** : Contrairement au buzzer actif, le buzzer passif nécessite un signal d'entrée pour produire un son. Vous pouvez contrôler la fréquence et la durée du son en utilisant une sortie PWM (Pulse Width Modulation) de l'Arduino.

Librairie

Pour pouvoir faire fonctionner le programme, vous avez besoin d'installer la librairie dédiée au buzzer. Cette librairie contient toutes les notes jouables par le composant. La librairie du buzzer est en .zip, donc pour l'installer il faut cliquer sur ***"Sketch"***, puis ***"Include Library"*** et ***"Add .Zip Library"*** :



Une fois la librairie nommée « pitches.zip » installée, vous aurez un message confirmant sa bonne installation :

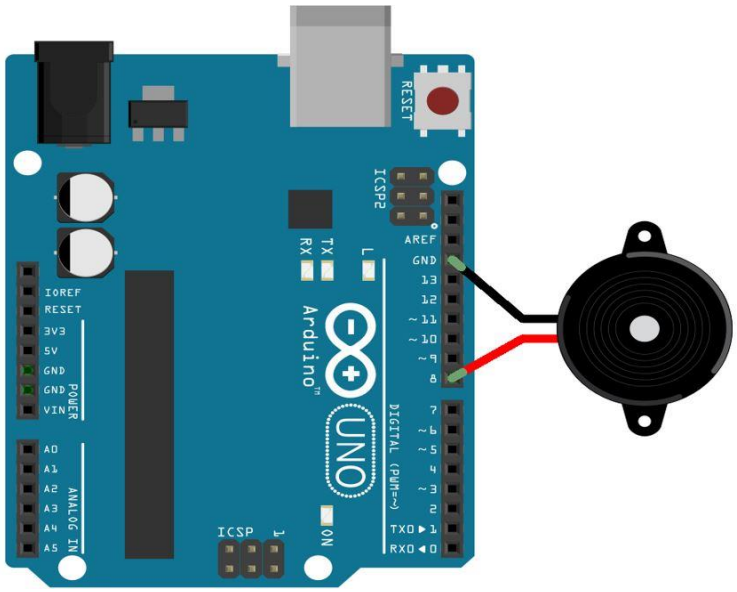
```
Output
Library installed
```

Voici la fonction pour programmer votre buzzer avec la librairie :

```
tone(broche, laNote,Duree);
```

- **Broche :** La broche ou est branché votre buzzer
- **laNote** : La note en fréquence HZ
- **Durée :** La durée que vous voulez que la note se joue

Voici le schéma pour relier votre buzzer à la carte Arduino :

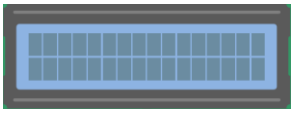


Voici le programme à ajouter à Arduino IDE afin de programmer votre carte Arduino :

```
#include "pitches.h" // La librairie pour le buzzer passif
// notes de la melodie
int melodie[] = {NOTE_C5, NOTE_D5, NOTE_E5, NOTE_F5, NOTE_G5, NOTE_A5, NOTE_B5,
NOTE_C6}; // Voici des notes déjà incluses dans la librairie
int duree = 500;  // durée de chaque melodie

void setup() { }

void loop() {
  for (int thisNote = 0; thisNote < 8; thisNote++) { // Pour toute les notes de la
liste melodie
    tone(13, melodie[thisNote], duree); // On joue la note avec la duree choisit
    delay(1000); // Nouveau son après 1 seconde de pause
  }
  delay(2000); }
```

Ecran LCD

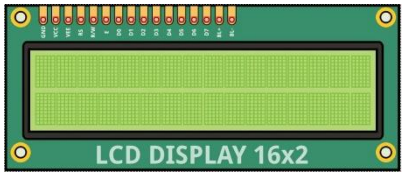
Un des éléments permettant d'afficher des informations le plus utilisé sur Arduino est l'écran à cristaux liquide LCD 16×2.



L'afficheur dispose d'un rétro-éclairage LED et peut afficher deux lignes de 16 caractères. Chaque caractère est un rectangle de pixel. Il est possible de contrôler chaque pixel de chaque rectangle pour créer des caractères spécifiques.

On va voir les différentes broches de l'écran LCD :

- **Vss :** Connecter à la masse
- **Vdd** : Connecter au +5V
- **VEE ou VO :** Connecter à un potentiomètre pour ajuster le contraste
- **Rs :** Contrôler le registre de mémoire
- **R/W :** Sélectionner écriture ou lecture
- **E :** Lorsqu'elle est à l'état bas, provoque l'exécution des instructions par le module LCD.
- **D0-D7** : Lire et Ecrire des données
- **A and K:** Contrôler le rétro-éclairage

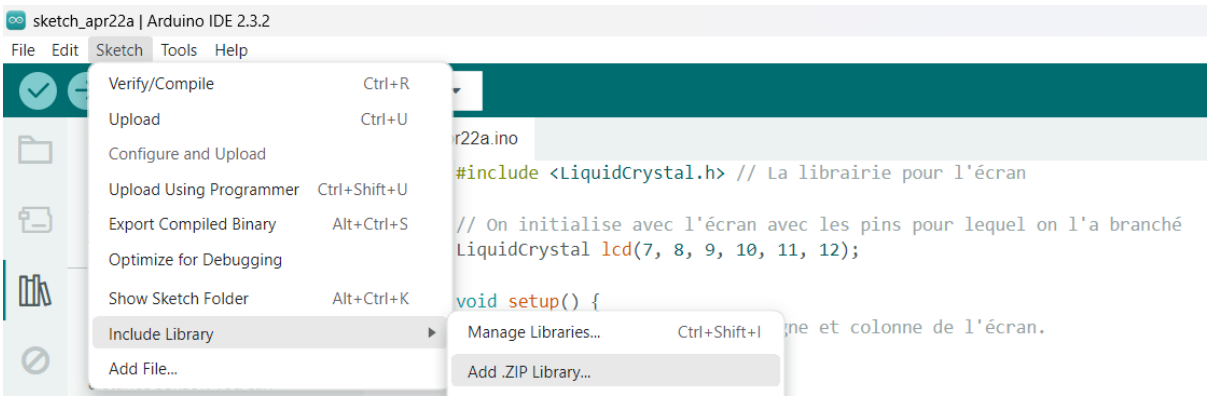


Librairie Liquid Crystal

Pour utiliser l'écran LCD, vous avez besoin d'une librairie : LiquidCrystal.zip. Il y a un ensemble de fonction dedans qui vont simplifier votre code et l'utilisation de l'écran 16×2.

A) Installation

Pour pouvoir l'installer, il vous faut ouvrir *Arduino IDE* et cliquer sur *sketch* puis *include librairie* et *add .ZIP Library* comme ci-dessous :



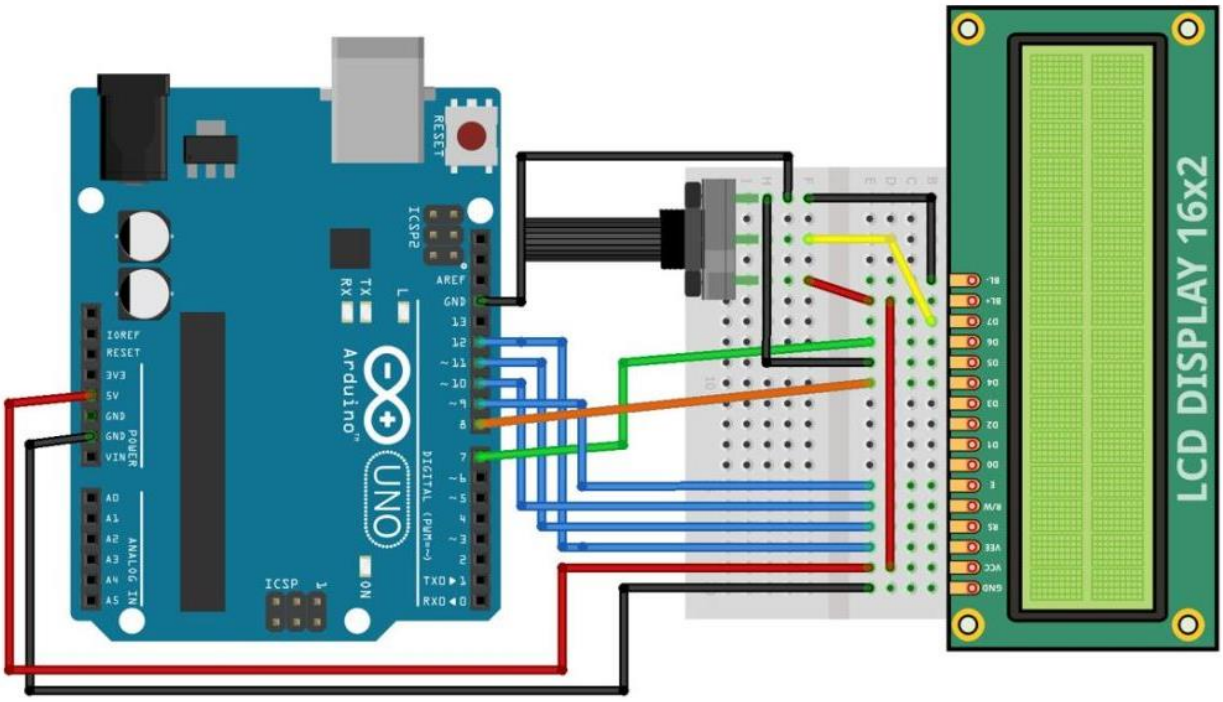
B) Les fonctions de la librairie

La librairie *Liquid Crystal* est composée de plusieurs fonctions pour vous faciliter l'utilisation de celle-ci :

- **LiquidCrystal () :** Permet d'initialiser les broches associées à l'écran LCD.
- **begin(colonne,ligne)** : Initialise l'écran LCD avec les dimension de l'écran
- **clear() :** Efface l'écran et fait revenir le curseur en haut à gauche.
- **home()** : Fait revenir le curseur en haut à gauche sans effacer l'écran
- **setCursor(colonne, ligne) :** Positionne le curseur sur l'écran.
- **write() :** Ecrit un caractère sur l'écran.
- **Print() :** Ecrit une chaîne de caractère sur l'écran
- **createChar() :** Permet de créer son propre caractère.

C) Afficher du texte à l'écran

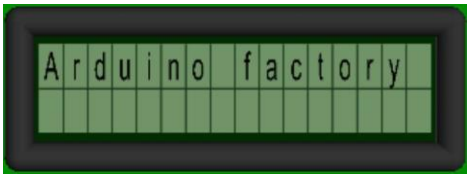
On va maintenant voir comment afficher du texte sur l'écran 16×2. Vous pourrez l'utiliser pour afficher des informations comme l'heure, la réponse d'un capteur. Le circuit est un peu complexe à réaliser… Néanmoins pour compenser le code est très simple et compréhensible grâce à la librairie.



Pour simplifier le circuit si vous le souhaiter, vous pouvez enlever le potentiomètre du circuit et remplacer le fils jaune sur le schéma par un fils allant vers le 5V. Vous ne pourrez alors plus régler la luminosité de l'écran, elle sera à son maximum, mais vous aurez moins de composant et de fils. Voici le programme associé au projet :

```
#include <LiquidCrystal.h> // La librairie pour l'écran

// On initialise avec l'écran avec les pins pour lequel on l'a branché
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

void setup() {
  // On définit le nombre de ligne et colonne de l'écran.
  lcd.begin(16, 2);
  // On affiche le message
  lcd.print("Arduino Factory");
}

void loop() {
  // On met le curseur à la colonne 0 ligne 1
  lcd.setCursor(0, 1);
}
```

Voici le résultat que l'on obtient :



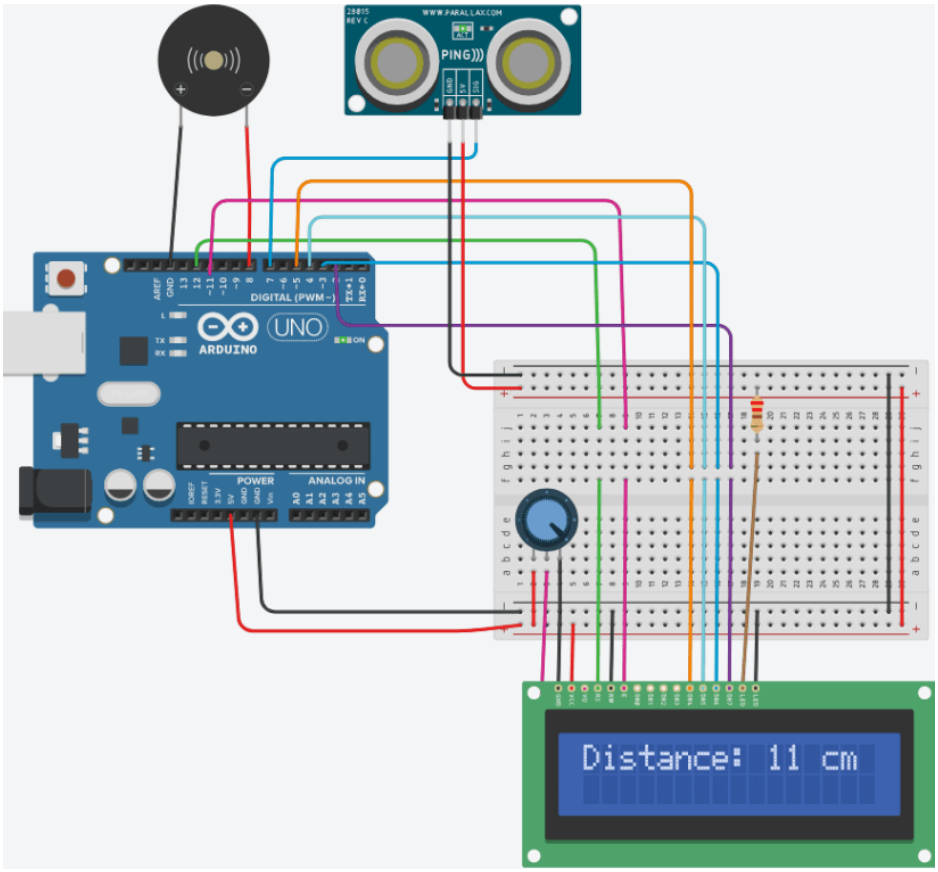
Projet : Radar de distance sur Arduino

Nous approchons de à la fin de ce guide. Avec toutes les informations que vous avez assimilées, nous allons pouvoir passer à la réalisation d'un projet concret pour mettre en pratique nos connaissances.

L'objectif de ce projet est d'ajouter un radar à l'arrière de votre pare-chocs de véhicule, afin de vous indiquer la distance entre votre voiture et un objet quand vous vous garez. Cette distance sera indiquée sur un écran LCD en centimètres et un buzzer sonnera si vous vous approchez de trop près de l'objet.

Voici le matériel nécessaire pour le projet :

- Une carte Arduino Uno
- Un Ecran LCD 16×2 à Liquide Crystal
- Un potentiomètre
- Une résistance 220 ohms
- Capteur de distance
- HC-SR04
- Un buzzer
- Fils de liaisons (une quinzaine environ !)



Dans le programme, la distance maximum entre l'objet et la voiture pour laquelle le buzzer va se mettre à sonner est de 15 centimètres. Vous pouvez le modifier selon votre convenance :

```
#include <LiquidCrystal.h> //libraire pour l'écran LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int buzzer_pin = 8; // pin du buzzer
int cm = 0; // On initialise la valeur à 0

//fonction pour avoir la distance venant du capteur de distance
long readUltrasonicDistance(int triggerPin, int echoPin) {
    pinMode(triggerPin, OUTPUT);
    digitalWrite(triggerPin, LOW);
    delayMicroseconds(2);
    digitalWrite(triggerPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(triggerPin, LOW);
    pinMode(echoPin, INPUT);
    return pulseIn(echoPin, HIGH);
}
void setup() {
    lcd.begin(16, 2); // On initialise l'écran
    pinMode(buzzer_pin, OUTPUT); //on met le buzzer en sortie
}

void loop() {
    // On écrit la distance sur l'écran LCD
    lcd.print("Distance: ");
    lcd.print(cm);
    lcd.print(" cm");
    delay(10);

    cm = 0.01723 * readUltrasonicDistance(7, 7); // On convertie la valeur en
centimètre
    delay(100);

    if (cm < 15){ // Si la voiture est à moins de 15 cm de l'objet on fait sonner
le buzzer
        tone(buzzer_pin, 1000, 1000);
    }

    lcd.clear(); // On efface ce qui est écrit sur l'écran LCD
}
```

Conclusion

Merci d’avoir parcouru notre guide gratuit sur Arduino. Nous espérons que les informations et les projets présentés vous ont été utiles et ont renforcé vos connaissances sur l'utilisation de la carte Arduino. Si vous avez apprécié réaliser les projets dans ce guide, sachez que ce n'est que le début de votre aventure avec Arduino.

Pour aller plus loin, nous vous invitons à poursuivre votre apprentissage sur notre site internet Arduino Factory. Vous y trouverez une multitude de ressources supplémentaires, de tutoriels avancés, ainsi que de nombreux autres projets pour enrichir vos compétences en électronique et en programmation.

Encore une fois, merci pour votre intérêt et votre enthousiasme.

Arduino Factory