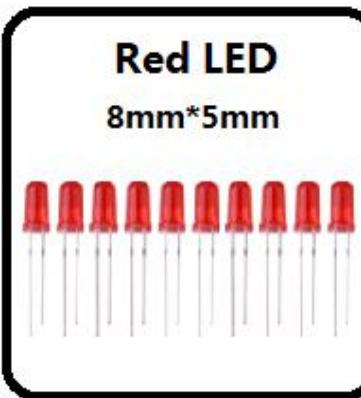
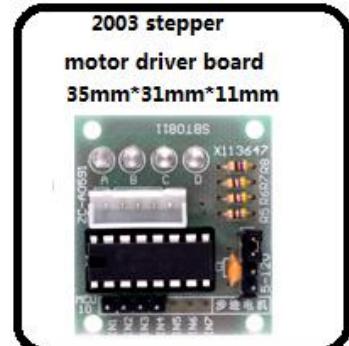
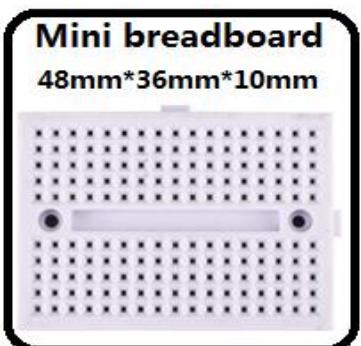
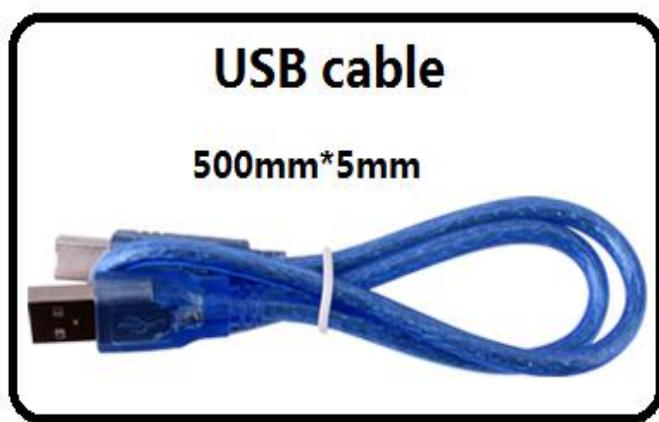
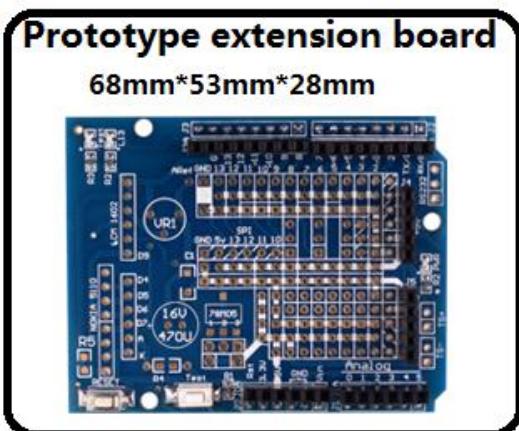


# Kit de démarrage

## Mode d'emploi pour Arduino Uno R3



**Vibration Sensor**  
13mm\*5mm



**Flame sensor**  
8mm\*5mm



**LM35**  
5mm\*5mm



**Infrared receiver**  
13mm\*7mm



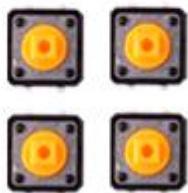
**Photoresistor**  
3mm\*5mm



**Key cap**  
13mm\*12mm



**Key switch**  
12mm\*12mm\*7mm



**Adjustable potentiometer**  
24mm\*16mm



**Passive buzzer**  
8mm\*12mm



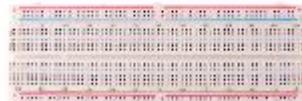
**Active buzzer**  
10mm\*12mm



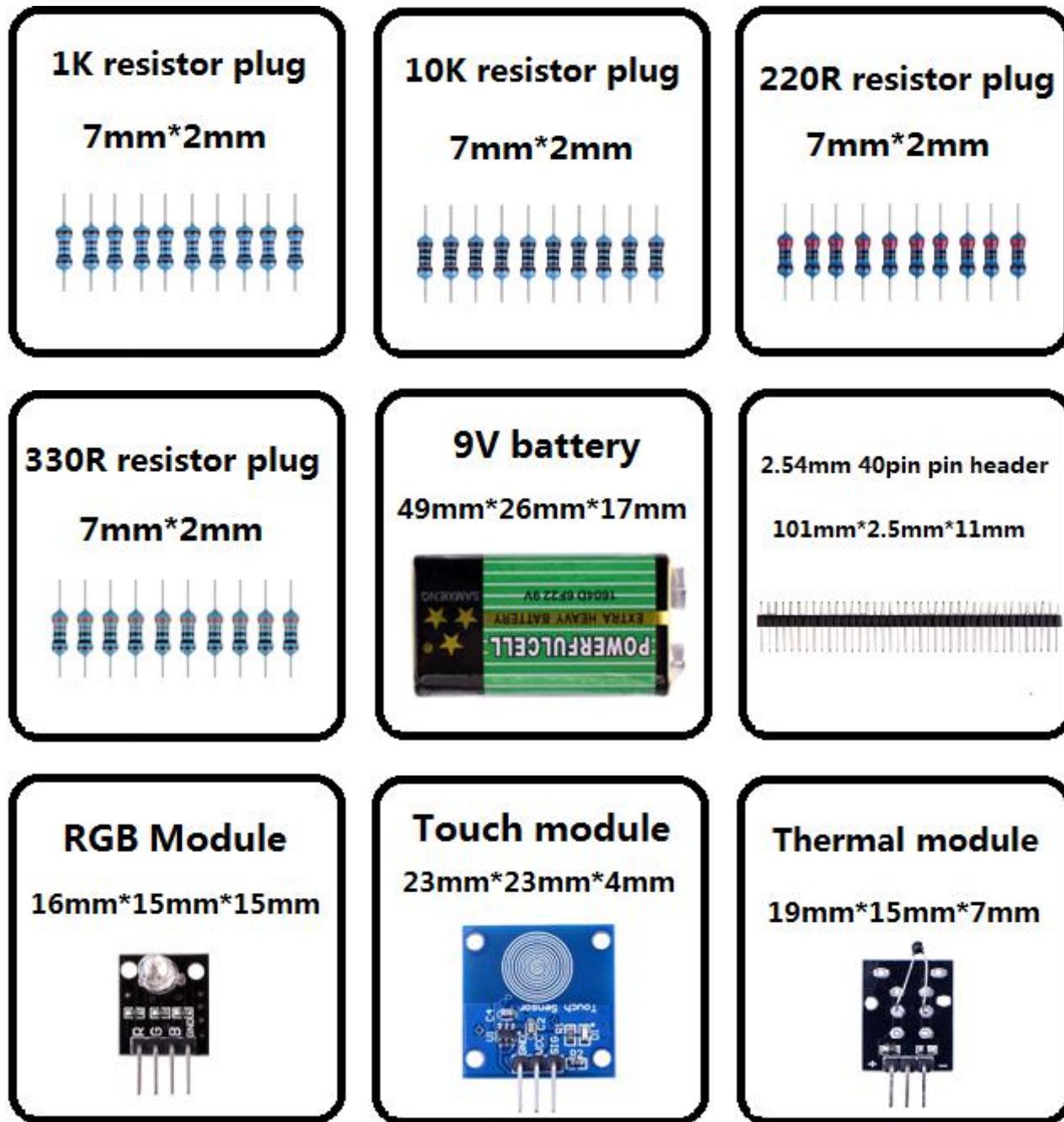
**Jumper cap**  
6mm\*5mm\*2.5mm



**Large breadboard**  
165mm\*55mm\*10mm



<b>Remote Control</b> 86mm*40mm*7mm 	<b>1602 Screen</b> 80mm*36mm*9mm 	<b>Breadboard module</b> 52mm*33mm*22mm 
<b>Hc-SR04 Ultrasonic</b> 45mm*20mm*15mm 	<b>Component box</b> 75mm*32mm*22mm 	<b>DuPont line</b> 200mm*1mm 
<b>Breadboard line</b> 1mm*8mm 1mm*12mm 1mm*16mm 1mm*21mm 	<b>8*8 dot matrix</b> 38mm*38mm*7mm 	<b>One digit eight segment tube</b> 19mm*13mm*8mm 
<b>Four digit eight segment tube</b> 30mm*14mm*7mm 	<b>IC 74HC595</b> 20mm*8mm*7mm 	<b>Battery Holder</b> 150mm*1mm 



## Table des matières.

Leçon 0 : Installation de l'IDE

Leçon 1 : Ajouter les bibliothéques

Leçon 2 : Clignotant.

- Leçon 3 :** LED
- Leçon 4 :** Bouton
- Leçon 5 :** Test alarme incendie
- Leçon 6 :** Essai matrix dot 8x8
- Leçon 7 :** Module buzzer actif
- Leçon 8 :** Buzzer passif
- Leçon 9:** Essai crystal liquide 1602
- Leçon10:** Essai 74HC595
- Leçon 11:** Essai affichage tube digital
- Leçon 12:** Tube digital 4 bits
- Leçon 13:** Module Hit
- Leçon 14:** Interrupteur d'inclinaison
- Leçon 15:** Module ultrasonique de mesure de distances.
- Leçon 16:** Résistance thermique
- Leçon 17:** Essai moteur pas à pas
- Leçon 18:** Essai contrôle de la direction
- Leçon 19:** Essai de la lampe photosensible
- Leçon 20:** Capteur de température LM35
- Leçon 21:** Test scintillement LED

**Leçon 22: Récepteur infrarouge**

**Leçon 23: Potentiomètre**

**Leçon 24 Capteur de température analogue**

**Leçon 25 Module tactile**

**Leçon 26 Module LED full color- 3 couleurs**

## **Leçon 0. Installation de l'IDE**

**Introduction :**

Si nous voulons apprendre et utiliser Arduino, il est nécessaire d'installer le pilote d'abord, avec l'IDE arduino fourni.

## Voici la méthode d'installation :

1: Aller sur le site Arduino pour télécharger ([www.Arduino.cc](http://www.Arduino.cc))



Cliquer sur “télécharger”

Download the Arduino Software



Cliquer sur le lien indiqué par la flèche.



Le télécharger ici



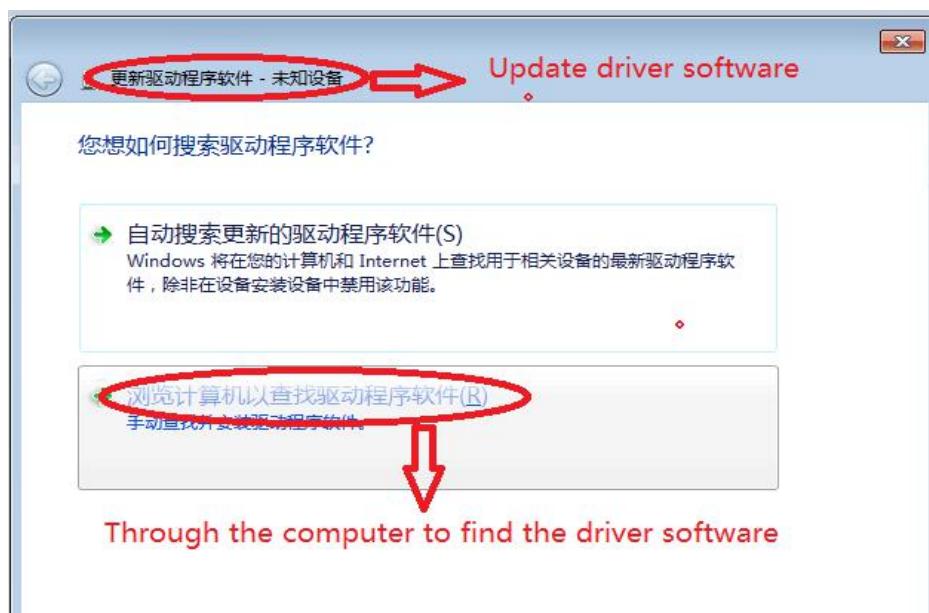
Le dossier téléchargé est compressé il faut l'extraire.

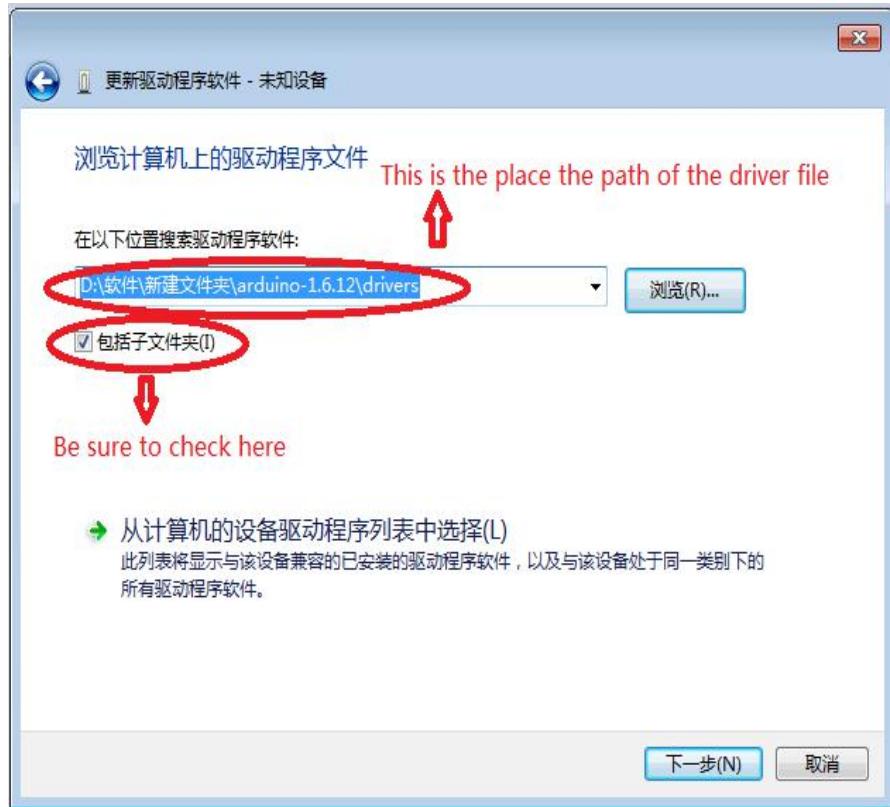


2 : Connecter la carte mère et l'ordinateur, vérifier le « gestionnaire d'applications »



Cliquez droit sur “appareil inconnu” et ensuite sur nouveau pilote.





Cliquez sur suivant, l'installation du pilote est complète

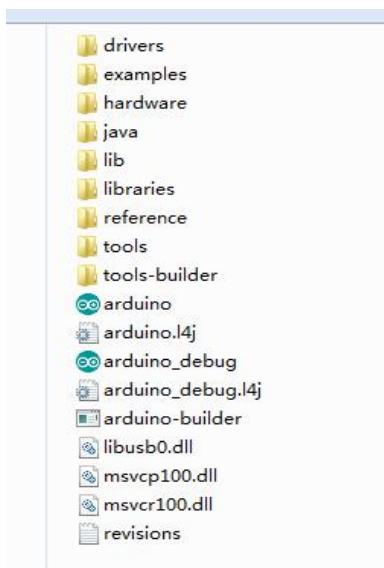
Regarder ensuite à "gestionnaire d'applications"



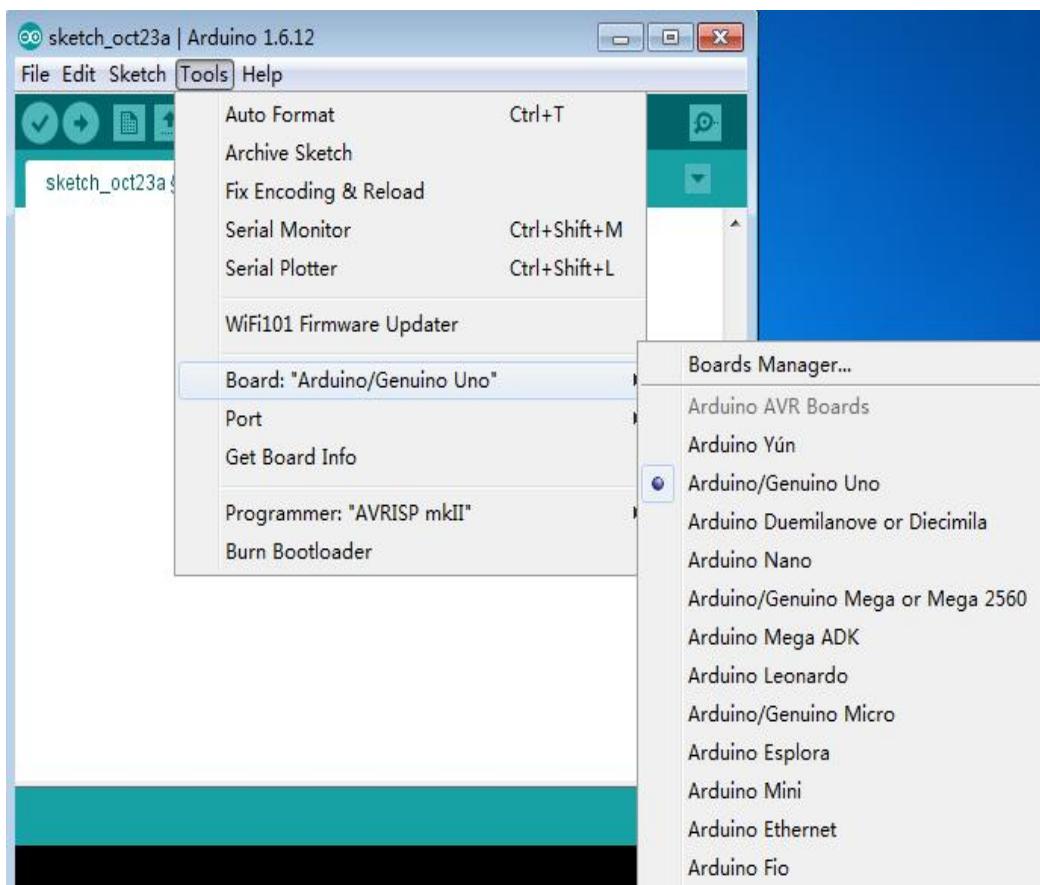
Ici vous pouvez voir “arduino Uno (com3), cela prouve que le pilote a été correctement installé

## Cartes et ports

Vous êtes maintenant prêts à commencer le logiciel arduino, peu importe la plateforme que vous utilisez, ouvrez le dossier et ouvrez l’application arduino qui se trouve dedans.

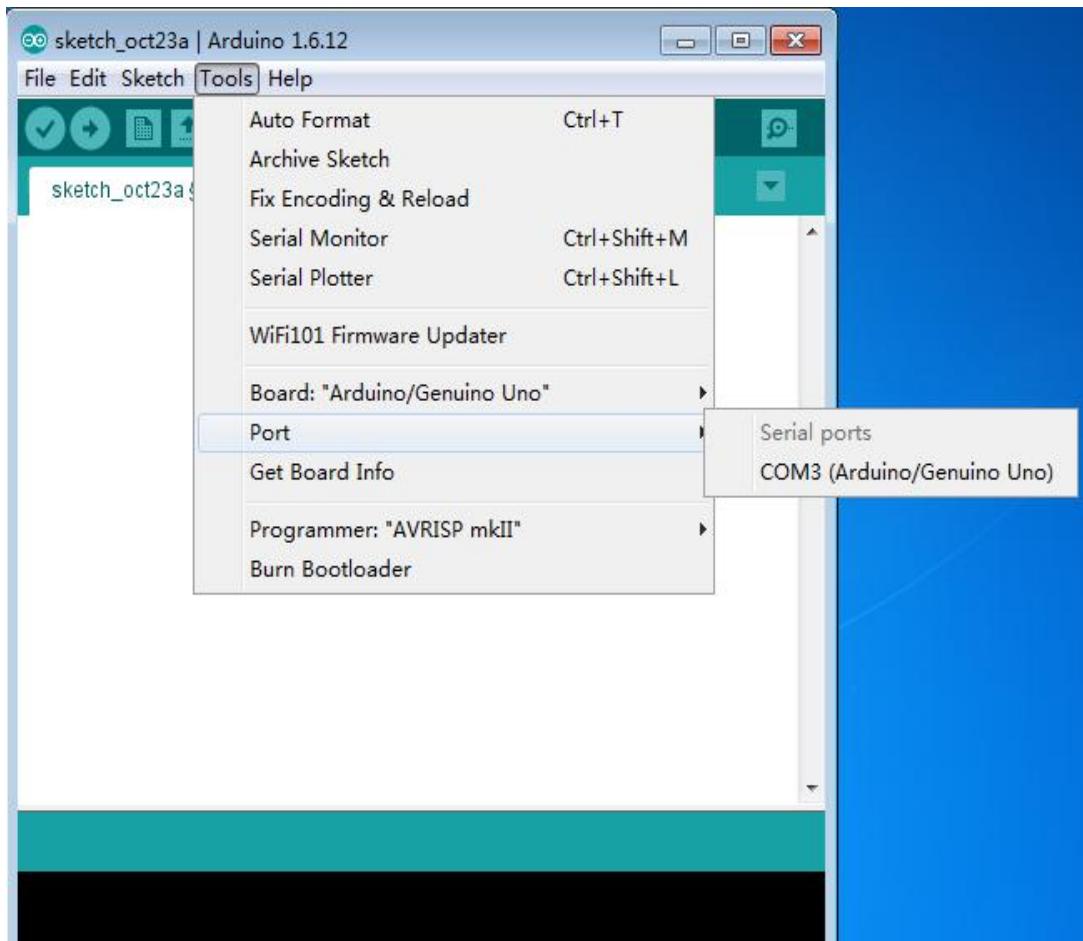


Cela va lancer l'IDE Arduino, mais avant de commencer la programmation, vous devez dire au logiciel Arduino quel type de carte Arduino vous utilisez et aussi sélectionner le port de connexion



Dans le menu “outils”, vous trouverez l’option “ports”. Il faut la sélectionner.

Si vous utilisez Windows, il n’y aura probablement qu’une option, COM3 ou COM4. Même s’il n’y a qu’une option, il faut la sélectionner.



# Leçon 1 : Ajouter les bibliothèques.

Quand vous êtes familier avec le logiciel Arduino et que vous utilisez le fonctions intégrées, vous voudrez sans doute étendre les fonctionnalités en ajoutant des bibliothèques.

## Qu'est-ce qu'une bibliothèque ?

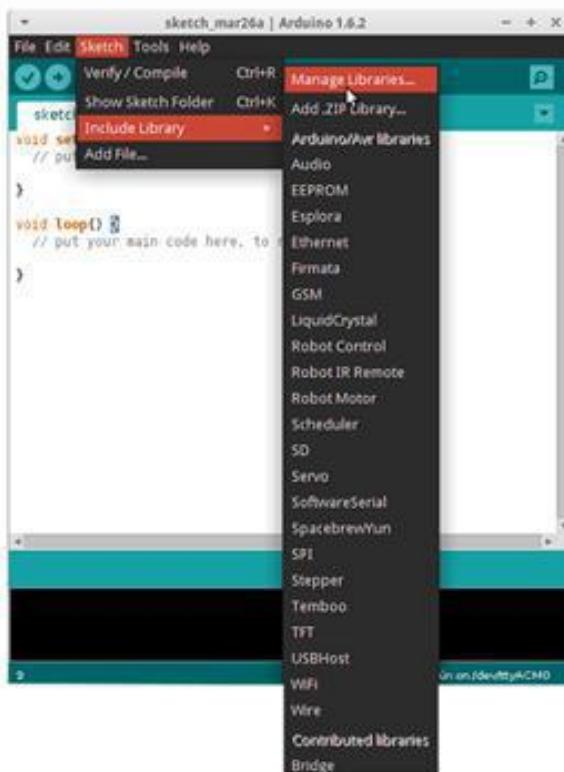
Les bibliothèques sont des collections de codes qui permettent une connexion facile à un capteur, écran, module etc. Par exemple, la bibliothèque intégrée Liquidcrystal permet de faciliter l'utilisation des caractères des écrans LCD. Il y a des centaines de bibliothèques supplémentaires disponibles sur internet en téléchargement. Les bibliothèques intégrées et certaines des bibliothèques additionnelles sont listées en référence. Pour utiliser les bibliothèques additionnelles, il faut d'abord les installer.

## Comment installer une bibliothèque

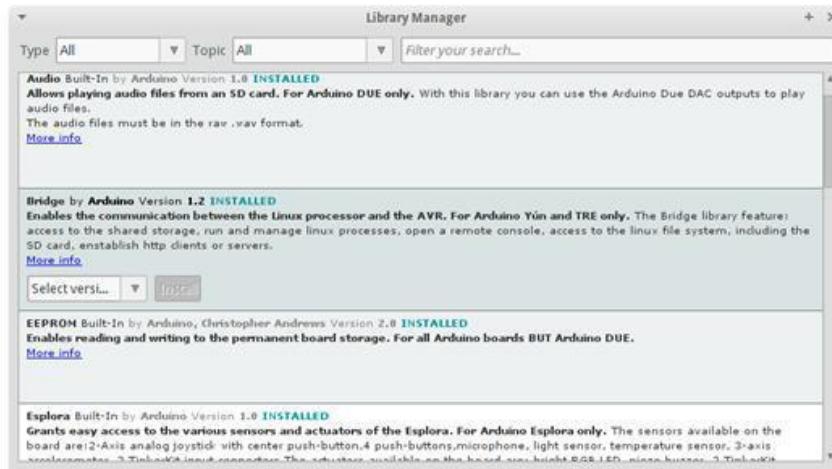
### Utilisation du gestionnaire de bibliothèques.

Pour installer une bibliothèque sur Arduino IDE, vous pouvez utiliser le gestionnaire de bibliothèques (disponible sur la version IDE 1.6.2) Ouvrir l'IDE et cliquer sur Sketch/include/bibliothèque/gérer les bibliothèques

Le gestionnaire de bibliothèques va s'ouvrir et vous trouverez une liste de bibliothèques qui sont déjà installées ou prêtées à être utilisées. Dans cet exemple, nous allons installer la bibliothèque Bridge. Cherchez la dans la liste, et sélectionnez la version de bibliothèque que vous voulez installer. Parfois, seulement une version de bibliothèque est disponible. Si le menu de la sélection de version ne s'affiche pas, ne vous inquiétez pas, c'est normal.



Cliquez finalement sur installer, et attendez que l'IDE installe la bibliothèque. Le téléchargement peut prendre du temps selon la vitesse de votre connexion. Quand c'est fini, une fenêtre d'installation doit apparaître prêt de la bibliothèque Bridge. Vous pouvez fermer le gestionnaire de bibliothèques.

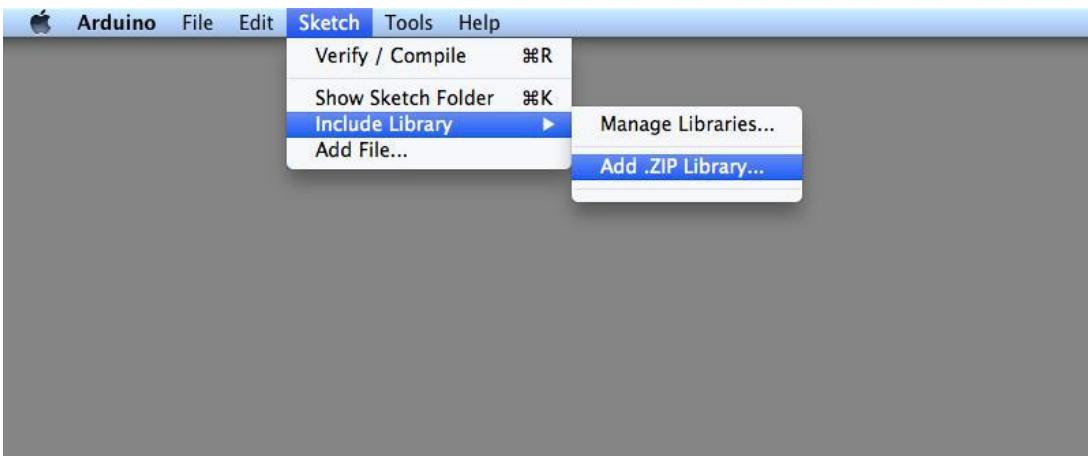


Vous pouvez désormais trouver la nouvelle bibliothèque dans le menu des bibliothèques. Si vous voulez ajouter votre propre bibliothèque, veuillez ouvrir une nouvelle publication dans github.

## Importer une bibliothèque .zip

In the Arduino IDE, navigate to Sketch > Include Library > Add .ZIP Library.

Les bibliothèques sont souvent sous format Zip ou sous dossier. Le nom du dossier est le nom de la bibliothèque. A l'intérieur du dossier il y aura : fichier .cpp, fichier .h, souvent un fichier keywords.txt, des dossiers d'exemple et d'autres fichiers requis par la bibliothèque. Commencer par la version 1.0.5, vous pouvez installer des bibliothèques tiers dans l'IDE. Ne pas décompresser les bibliothèques téléchargées, laissez les telles quelles.



Vous serez amené à sélectionner la bibliothèque que vous voulez ajouter. Accédez à l'endroit où se trouve le fichier .zip et ouvrez le. Retournez à Sketch/ importer une bibliothèque. Vous devriez maintenant voir la bibliothèque en bas du menu.

C'est prêt pour être utilisé dans votre sketch. Le fichier zip a été étendu dans le dossier des bibliothèques dans le répertoire des sketches Arduino.

Nb : la bibliothèque sera disponible pour être utilisée dans les sketches, mais les exemples pour la bibliothèque ne seront pas montrés dans dossier/exemples avant que l'IDE ne soit redémarré.

## Manuel d'installation

Pour installer la bibliothèque, quittez d'abord l'application Arduino. Décompressez ensuite le fichier Zip contenant la bibliothèque. Par exemple, si vous installez une bibliothèque appelée « Arduino Party », décompressez ArduinoParty.zip. Il devrait contenir un dossier nommé ArduinoParty, avec des fichiers comme ArduinoParty.cpp et ArduinoParty.h (si les fichiers .cpp et .h ne sont pas dans le dossier, vous devrez en créer un. Dans ce cas, vous ferez un dossier nommé « arduinoparty » et déplaceriez tous les fichiers qui se trouvaient dans le fichier Zip à l'intérieur, comme ArduinoParty.cpp et ArduinoParty.h)

Déplacez le dossier ArduinoParty dans ce dossier (le dossier bibliothèques). Sous windows, il sera appelé Mes

Documents/Arduino /bibliothèques. Pour les utilisateurs Mac, il sera appelé documents/arduino/bibliothèques. Sous Linux, ce sera le dossier « bibliothèques » dans votre carnet. Votre dossier bibliothèques Arduino devrait maintenant ressembler à ça (sous windows)

Mes documents/arduino/bibliothèques/arduinoparty/arduinoparty.cpp

Mes documents/arduino/bibliothèques/arduinoparty/arduinoparty.h

Mes documents/arduino/bibliothèques/arduino[arty]/exemples

ou comme ceci (sur Mac et Linux)

documents/arduino/bibliothèques/arduinoparty/arduinoparty.cpp

documents/arduino/bibliothèques/arduinoparty/arduinoparty.h

documents/arduino/bibliothèques/arduino[arty]/exemples

Il peut y avoir plus de fichiers que ceux en .cpp et .h, assurez vous donc qu'ils soient tous là. La bibliothèque ne fonctionnera pas si vous mettez les fichiers .cpp et .h directement dans le dossier des bibliothèques ou si ils sont imbriqués dans un autre dossier. Par exemple, Documents/arduino/bibliothèques/arduinoparty.cpp et documents/arduino/bibliothèques/arduinoparty/arduinoparty.cpp ne fonctionnera pas.

Redémarrez l'application Arduino. Assurez vous que la nouvelle bibliothèque apparaît dans le menu Sketch/Imporation de librairies.

## Résumé

Dans cette leçon, nous allons installer les bibliothèques que nous utiliserons dans le tutoriel.

Ouvrez le dossier bibliothèques et installez les fichiers Zip un par un pour ne pas avoir à faire cette étape dans les prochaines leçons.

Nous connectons simplement le composant selon le schéma et téléchargeons le code fourni. Le kit fonctionnera.

# Leçon 2. Clignotant

## Vue d'ensemble

Dans cette leçon, vous apprendrez à programmer votre carte de contrôle UNO R3 pour faire le clignotant LED intégré Arduino.

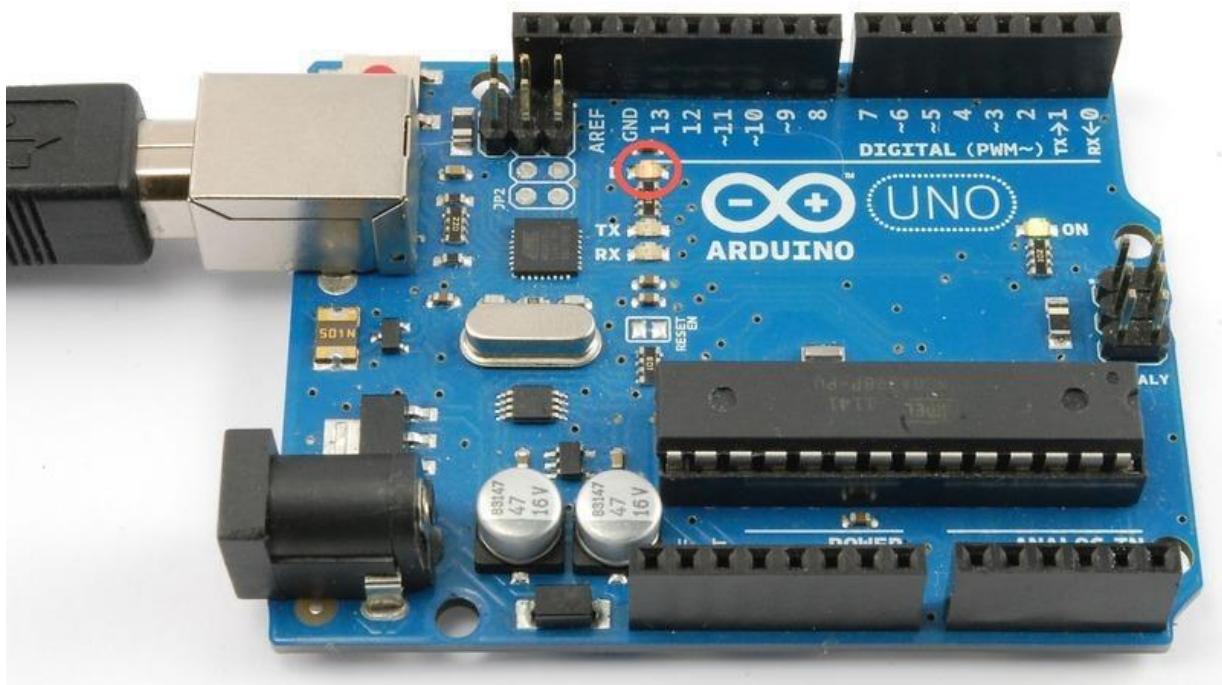
## Composant nécessaire.

(1) x Arduino Uno R3

## Principe

La carte UNO R3 a des rangées de connecteurs le long des 2 côtés qui sont utilisés pour connecter plusieurs appareils électroniques et plug-in “shields” pour étendre sa capacité.

Elle a aussi une LED unique que vous pouvez contrôler à partir de vos sketches. Cette LED est construite sur la carte UNO R3 et est souvent mentionnée comme étant la LED « L » puisqu'elle est ainsi appelée sur la carte.

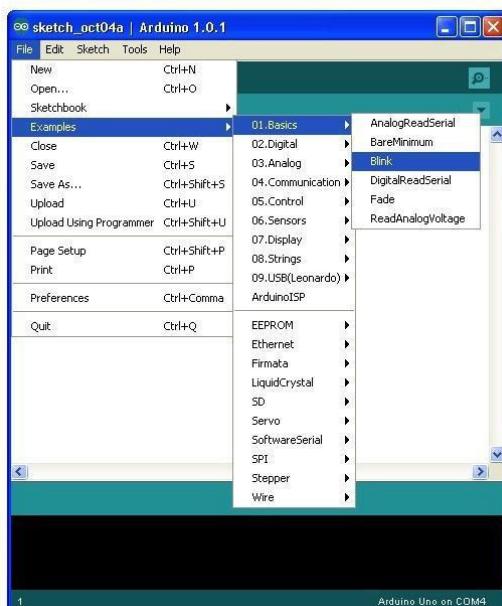


Il se peut que la LED “L” de votre carte UNO R3 clignote déjà quand vous la connectez à une prise USB. C'est parce que les cartes sont en général livrées avec le sketch « clignotant » pré-installé.

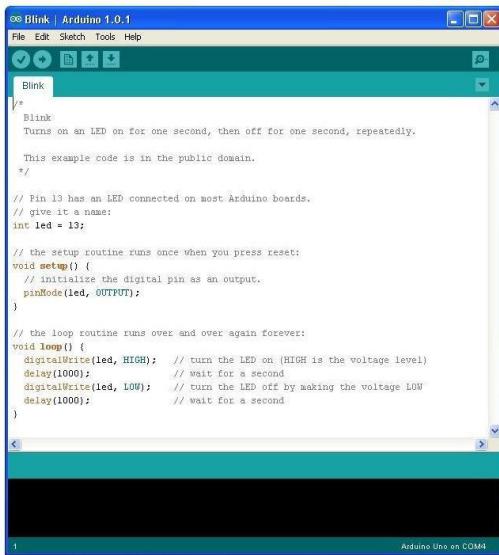
Dans cette leçon, nous allons reprogrammer la carte UNO R3 avec notre propre sketch clignotant, et changer ensuite la vitesse de clignotement.

Dans la leçon 0, vous avez programmé votre IDE arduino et vous êtes assuré de pouvoir trouver le port correct pour le connecter à votre carte UNO R3. Il est maintenant temps de tester cette connection et programmer votre carte UNO R3.

L'IDE Arduino inclut une large collection d'exemples de sketches que vous pouvez charger et utiliser. Cela comprend un sketch exemple pour faire clignoter la LED « L ». Chargez le sketch « clignotant » que vous trouverez dans le menu de l'IDE sous Fichier/exemples/0.1basiques



Quand la fenêtre sketch s'ouvre, élargissez la pour pouvoir visualiser le sketch entier dans la fenêtre.



Les sketches exemples inclus avec l'IDE Arduino sont « read-only ».

Cela signifie que vous pouvez les télécharger sur la carte UNO R3, mais si vous les modifiez, vous ne pouvez pas les enregistrer comme un même fichier.

Dans le menu de l'IDE Arduino, sélectionnez “sauvegarder sous « ” et sauvegardez le sketch sous le nom « monclignotant »



Vous avez sauvegardé votre copie de “clignotant” dans votre carnet. Cela signifie que si vous voulez le retrouver, vous pouvez simplement l’ouvrir dans le menu Fichier/carnet



Branchez votre carte Arduino à votre ordinateur avec un câble USB et vérifiez que le type de carte et le port sont correctement configurés. Vous devrez peut être vous référer à la leçon 0.

```
digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
delay(1000);           // wait for a second
}

< >
Done Saving.
```

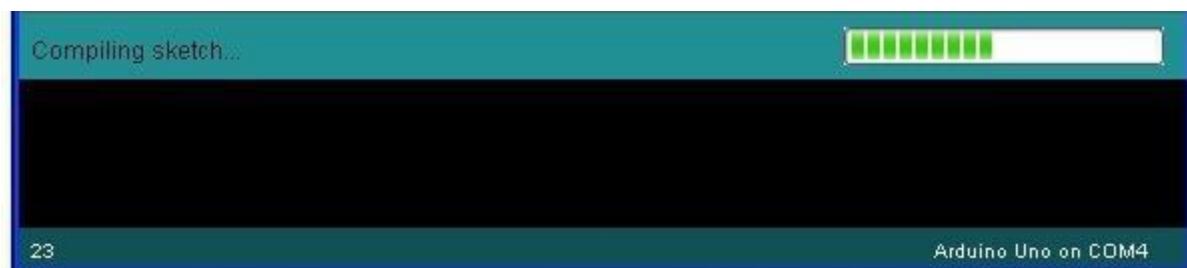
The status bar at the bottom right shows "Arduino Uno on COM4".

Click on the 'Upload' button. The second button from the left on the toolbar.

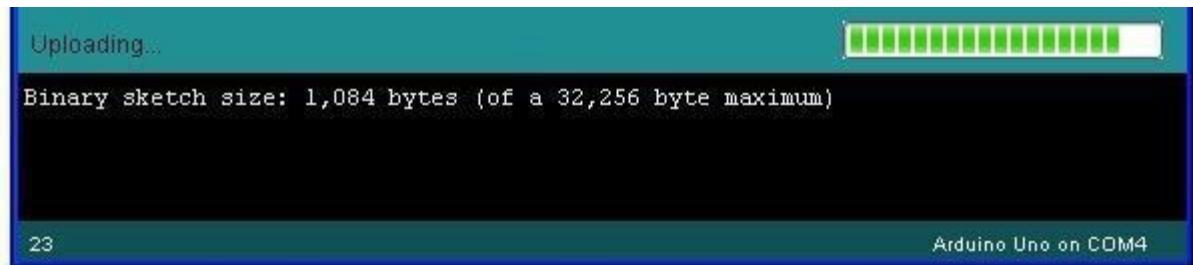
Cliquez sur le bouton “telecharger”. Le second bouton à gauche dans la barre d’outils.



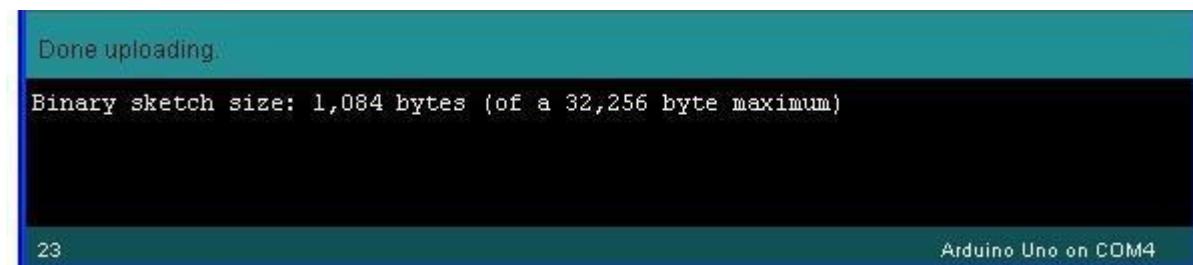
Si vous regardez la zone d’état de l’IDE, vous verrez une barre de progression et une série de messages. D’abord, ce sera « compilation sketch ». Cela convertit le sketch dans un format approprié au téléchargement de la carte.



Ensuite, le statut va se transformer en “téléchargement”. A ce moment, les LED de l’arduino devraient commencer à clignoter comme le sketch est transféré.



Enfin, le statut se transformera en “fini”.



L’autre message nous dit que le sketch utilisé 1,084 bytes sur les 32,256 disponibles. Après l’étape de compilation sketch, vous pourriez avoir ce message d’erreur.



Cela peut signifier que votre carte n’est pas connectée du tout, ou que les pilotes n’ont pas été installés (si nécessaire) ou que le port sélectionné n’est pas le bon.

Une fois que le téléchargement est complet, la carte devrait redémarrer et commencer à clignoter. Ouvrez le code. Notez qu'une large partie du sketch est composée de commentaires.

Ce ne sont pas les instructions actuelles du programme, cela explique juste comment fonctionne le programme. Ils sont là pour vous aider.

Tout ce qui est entre /\* et \*/ en haut du sketch est un commentaire bloc, il explique l'utilisation du sketch. Les commentaires simples commençant par // et tout ce qui suit jusqu'à la fin de la ligne est considéré comme un commentaire.

la première ligne du code est :

```
int led = 13;
```

Comme le commentaire ci-dessus explique, c'est donner un nom à la broche à laquelle la LED est rattachée. C'est 13 sur la plupart des Arduino, y compris Uno et Leonardo.

Ensuite, nous avons la fonction réglages. Encore une fois, comme le commentaire l'explique, c'est exécuté lorsque le bouton reset est enfoncé. C'est également executé chaque fois que la carte se réinitialise pour quelque raison que ce soit, comme pour la première mise en marche, ou après qu'un sketch ait été téléchargé.

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

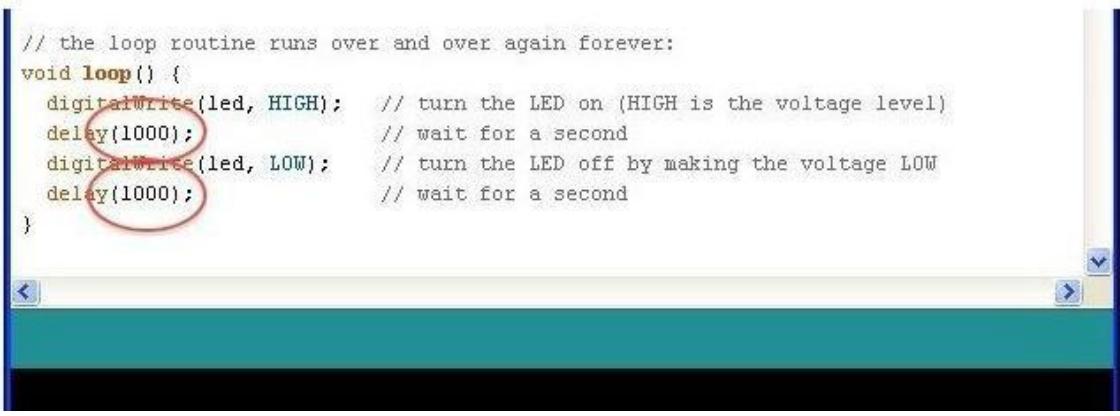
Chaque sketch Arduino doit avoir une fonction 'setup' et la place où vous voulez ajouter les instructions se situe sans doute entre le { et le }

Dans ce cas, il n'y a qu'une commande, qui comme l'indique le commentaire, nous explique que sur la carte Arduino nous allons utiliser la broche LED comme une sortie.

```
void loop()  
{
```

```
digitalWrite(led, HIGH);
delay(1000);
digitalWrite(led, LOW);
delay(1000);
}
```

Dans la fonction ‘loop’, la commande allume d’abord la LED (high) puis la retarde de 1000 millisecondes (1 seconde) puis l’éteint et la met sur pause pendant une seconde supplémentaire. Vous allez maintenant faire clignoter la LED plus rapidement. Comme vous l’avez sûrement deviné, la clé est de changer le paramètre en () pour la commande « retardateur »



```
// the loop routine runs over and over again forever:
void loop() {
    digitalWrite(led, HIGH);      // turn the LED on (HIGH is the voltage level)
    delay(1000);                // wait for a second
    digitalWrite(led, LOW);       // turn the LED off by making the voltage LOW
    delay(1000);                // wait for a second
}
```

Cette période de retard est en milliseconds, donc si vous voulez que la LED clignote 2 fois plus vite, changer la valeur de 1000 à 500.

Cela apportera ensuite une pause d’une demi seconde à chaque retard plutôt qu’une seconde.

# Leçon 3 LED

## Vue d'ensemble

Dans cette leçon, vous allez apprendre comment changer la luminosité d'une LED en utilisant différentes valeurs de résistance.

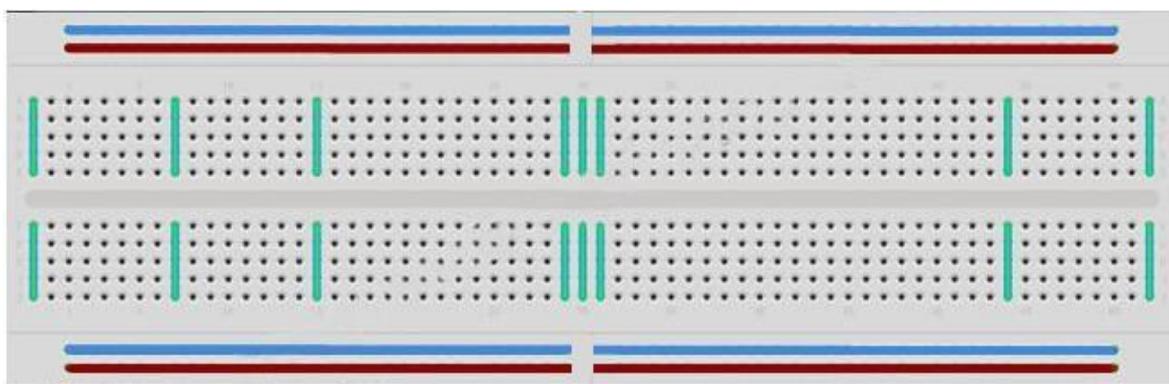
## Composants nécessaires

- (1) 1 x Arduino Uno R3
- (1) 1x 5mm red LED
- (1) 1x 220 ohm résistance
- (1) 1x 1k ohm résistance
- (1) 1x10K ohm résistance
- (2) 2 x M-M câbles

## Introduction des composants.

### Carte de prototypage MB-102

La carte de prototypage vous permet de prototyper des circuits rapidement, sans avoir à souder les connexions. En voici un exemple.



Les cartes de prototypages ont des tailles et configurations variées. La plus simple est juste une grille avec des trous sur un bloc en plastique.

Il y a à l'intérieur des bandes de métal qui fournissent une connection électrique entre les trous des lignes plus courtes.

Insérer les pattes de 2 composants différents dans la même ligne les relie ensemble électriquement. Un canal profond sur le milieu indique qu'il y a une rupture dans les connexions, cela signifie que vous pouvez insérer une puce de chaque côté du canal sans les connecter ensemble.

Certaines cartes de prototypage ont deux bandes de trous le long de deux grands côtés de la carte qui sont séparées de la grille principale. Ils ont des bandes à l'intérieur qui permettent la connexion d'un voltage courant. Ils sont généralement en paires de +5volts et reliés à terre. Ces bandes sont appelées 'rails' et vous permettent d'alimenter différents composants ou points de la carte.

#### **LED:**

Les cartes de prototypage sont parfaites pour créer des prototypes, mais elles ont des limites. Car les connexions sont temporaires, elles ne sont pas aussi fiables que les connexions avec soudure. Si vous avez des problèmes intermittents avec un circuit, cela pourrait être dû à une mauvaise connexion sur une carte de prototypage.

#### **LED :**

Les LED sont de très bons indicateurs lumineux. Ils utilisent très peu de courant et durent très longtemps.

Dans cette leçon, vous utiliserez sans doute la plus commune des LED : une LED 5mm rouge. 5mm est le diamètre de la LED. Les autres tailles courantes sont 3mm et 10mm.

Vous ne pouvez pas connecter directement une LED à une batterie ou une source de courant car : 1/ la LED a une borne positive et négative et ne s'allumera pas si elle est mal installée 2/une LED doit être utilisée avec une résistance pour limiter ou 'étouffer' le courant qui la traverse, sinon elle va exploser.



Si vous n'utilisez pas de résistance avec la LED, elle pourrait être détruite immédiatement, car trop de courant va la traverser, la surchauffer et détruire la junction ou la lumière est produite. Il y a deux façons de différencier la borne positive de la borne négative de la LED.

D'abord, la borne positive est plus longue. Ensuite, où la borne négative entre dans la LED, il y a un bord plat. Si vous avez une LED qui a un côté plat près de la longue borne, vous devez penser que la longue borne est la positive.

## Résistance

Comme le nom le suggère, la résistance résiste au courant. Plus la valeur de la résistance est haute, plus elle résiste et moins le courant la traverse. Nous allons utiliser cela pour contrôler le flux d'électricité qui traverse la LED et par conséquent l'intensité de la lumière.



Tout d'abord, plus d'informations sur les résistances.

L'unité de la résistance est le Ohm, qui est souvent simplifié en  $\Omega$ , la lettre grecque, omega.

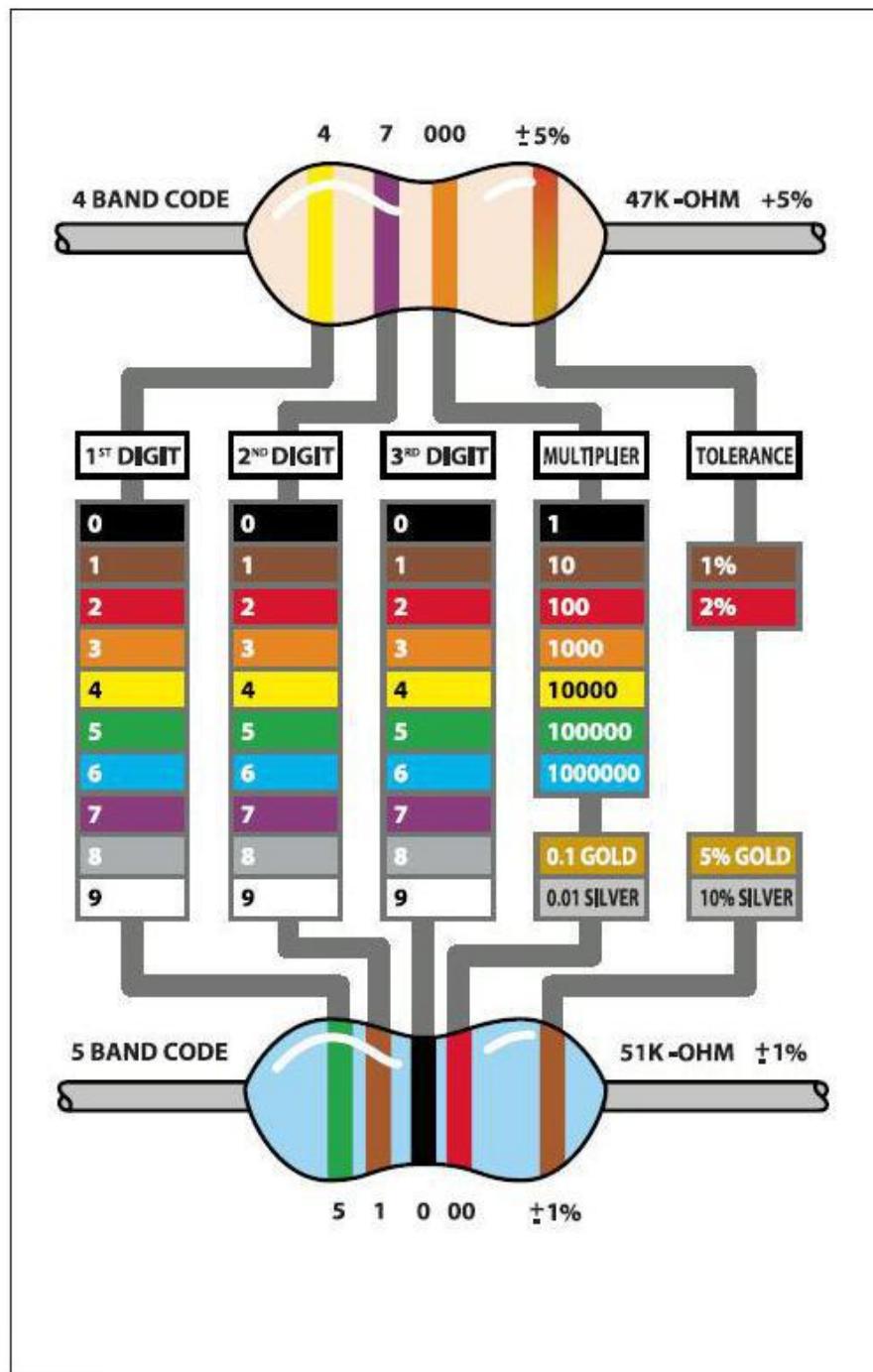
Car le ohm est une valeur basse de la résistance, il ne résiste pas beaucoup, nous identifions aussi les valeurs des résistances en  $k\Omega$  (1,000  $\Omega$ ) et  $M\Omega$  (1,000,000  $\Omega$ ).

Ils sont appelés kilo-ohms et mega-ohms.

Dans cette leçon, nous allons utiliser différentes valeurs de résistance : 220 $\Omega$ , 1k $\Omega$  and 10k $\Omega$ . Ces résistances

se ressemblent toutes, sauf qu'elles ont des bandes colorées différentes.

Ces bandes vous indiquent la valeur de la résistance. Le code de couleurs de résistance est 3 bandes colorées et une bande dorée à une extrémité.



Contrairement aux LED, les résistances n'ont pas de borne négative et positive. Elles peuvent être connectées des 2 côtés.

## connection

### Schéma

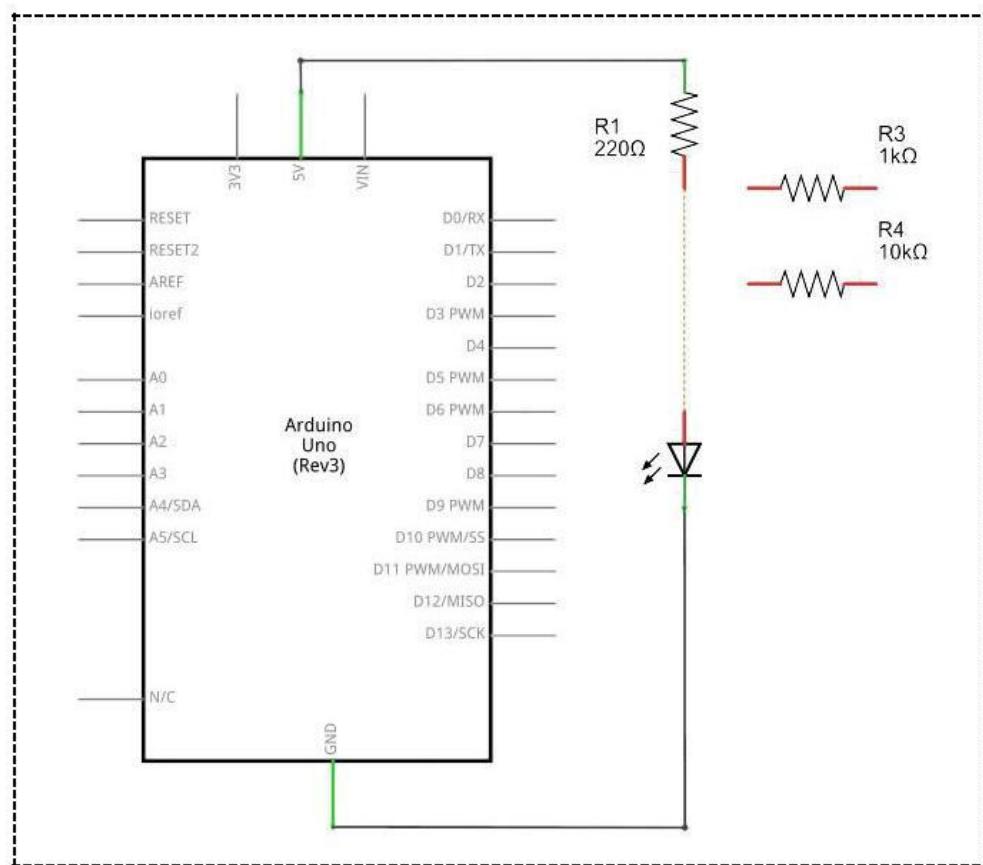
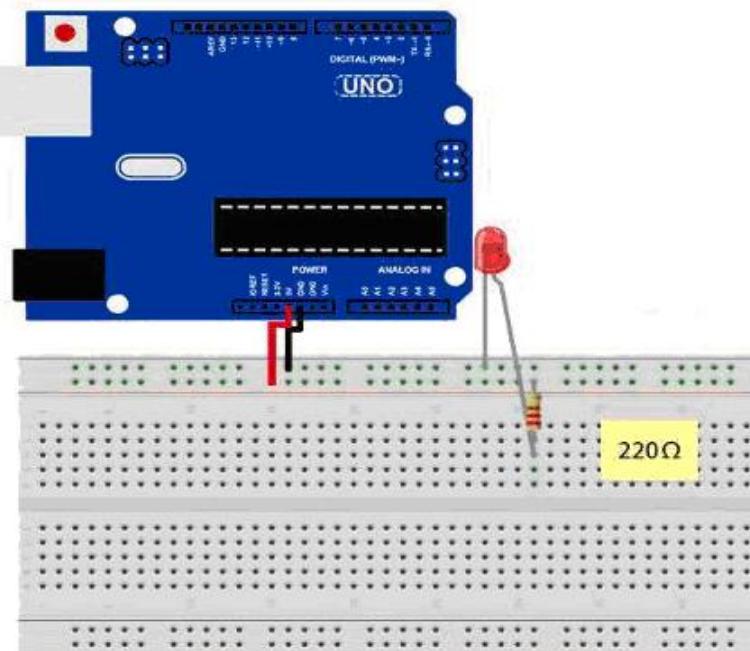


diagramme des câbles



L'UNO est une source pratique de 5 volts, que nous utiliserons pour donner de la puissance à la LED et la résistance.

Vous n'avez pas besoin de faire quoi que ce soit avec votre UNO, sinon de la brancher à un cable USB. Avec la résistance de  $220\Omega$  en place, la LED devrait être assez lumineuse. Si vous permutez la résistance de  $220\Omega$  pour celle de  $1k\Omega$ , la LED paraîtra un peu plus sombre .

Finalement, avec la résistance  $10K\Omega$  en place, la LED sera à peine visible. Tirez le fil rouge de démarrage hors de la carte de prototypage mettez le dans le trou et retirez le, pour que cela fonctionne comme un interrupteur. Vous devriez à peine voir la différence.

Pour le moment, vous avez 5V traversant une patte de la résistance, l'autre patte de la résistance allant vers le coté positif de la LED et l'autre coté de la LED allant vers le GND. Par contre, si nous bougeons la résistance pour la mettre apres la LED, comme montre ci-dessous, la LED s'allumera toujours. Vous voudrez certainement mettre la résistance  $220\Omega$  à sa place d'origine.

# Leçon 4. Bouton

## Introduction

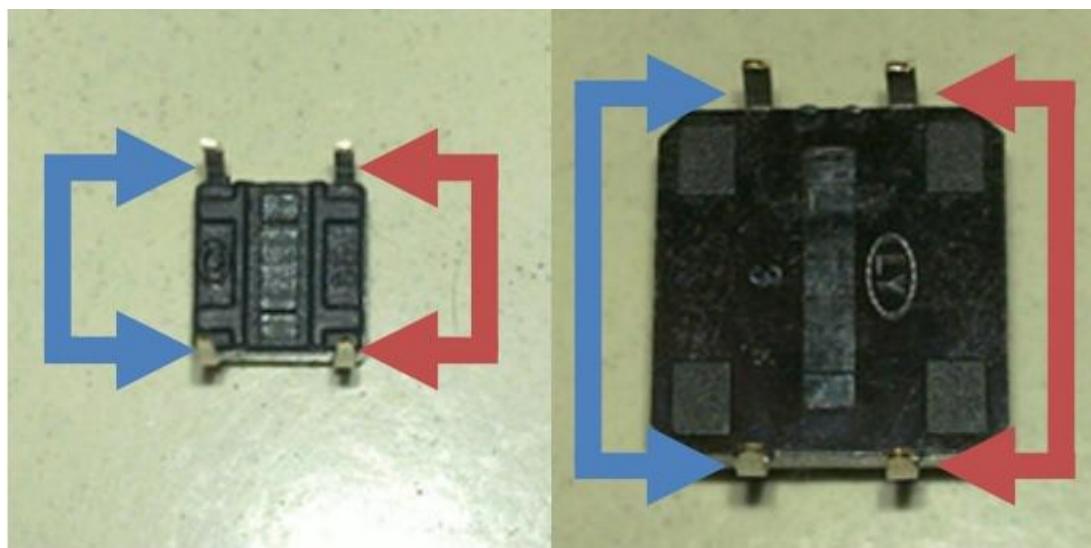
Dans cet essai, vous apprendrez à allumer et éteindre la LED en utilisant un port I/O et un bouton. Le port I/O représent le port Entrée et le port Sortie. Ici, le port ENtrée de la carte Kuman Uno est utilisé pour lire le port Sortie d'un appareil externe. Puisque la carte elle-même a une LED (connectée au pin 13), vous pouvez utiliser cette LED pour le test.

## Composants

- 1 \* kuman Uno board
- 1 \* USB cable
- 1 \* bouton
- 1 \* résistance
- câbles de démarrage
- 1 \* carte de prototypage

## Principe bouton

Les boutons sont des composants communs utilisés pour contrôler les appareils électroniques. Ils sont généralement utilisés comme interrupteurs pour connecter et déconnecter des circuits. Même si les boutons sont de tailles et de formes diverses, celui utilisé ici est un mini-bouton de 6mm, comme montré dans les dessins suivants. Les broches designées par les flèches de la même couleur doivent être connectées.



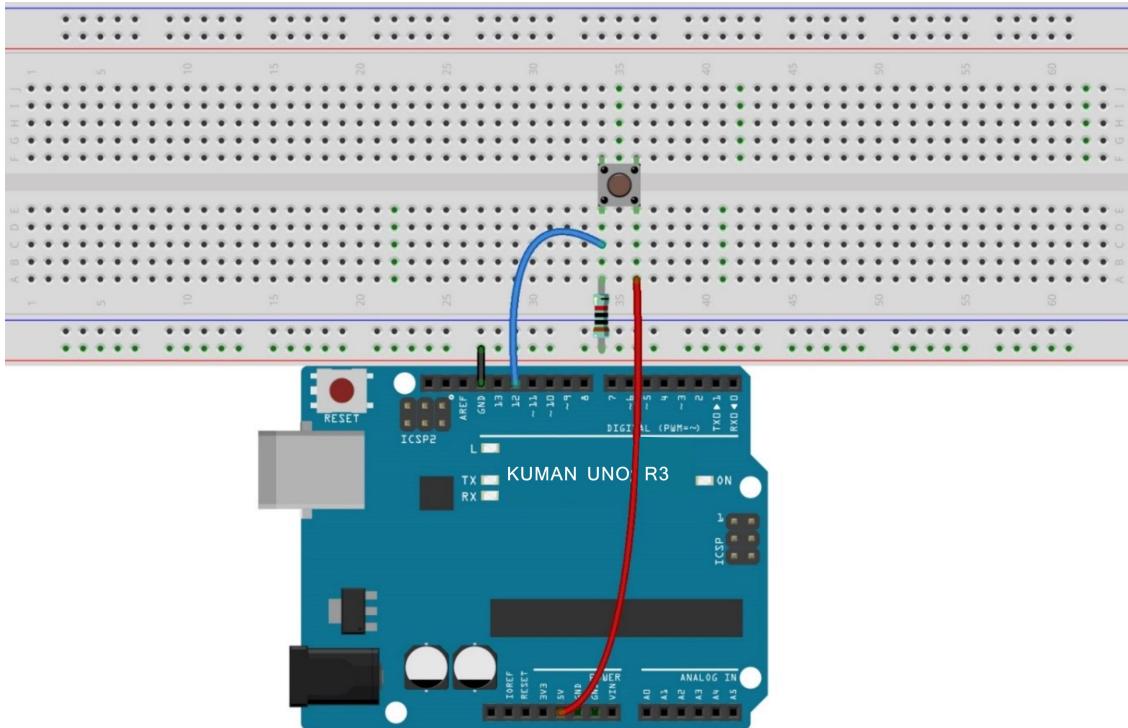
Quand le bouton est pressé, les broches designées par les flèches bleues seront connectées aux broches designées par les flèches rouges.

En général, le bouton est directement connecté au circuit LED afin de pouvoir allumer et éteindre la LED. Cette connexion est relativement simple. Par contre, parfois, la LED va s'allumer automatiquement sans avoir pressé le bouton, ce qui est provoqué par certaines interférences. Afin d'éviter ces interférences externes, une résistance pull down est utilisée, pour connecter la résistance 1K–10KΩ entre le port bouton et le GND. C'est utilisé pour éviter les interférences durant la connection du GND pendant que l'interrupteur est éteint.

Par exemple, si vous pressez n'importe quel bouton de votre téléphone, la lumière de rétroéclairage va s'allumer.

## Procédures d'expérimentation

### Etape 1 : Construire le circuit



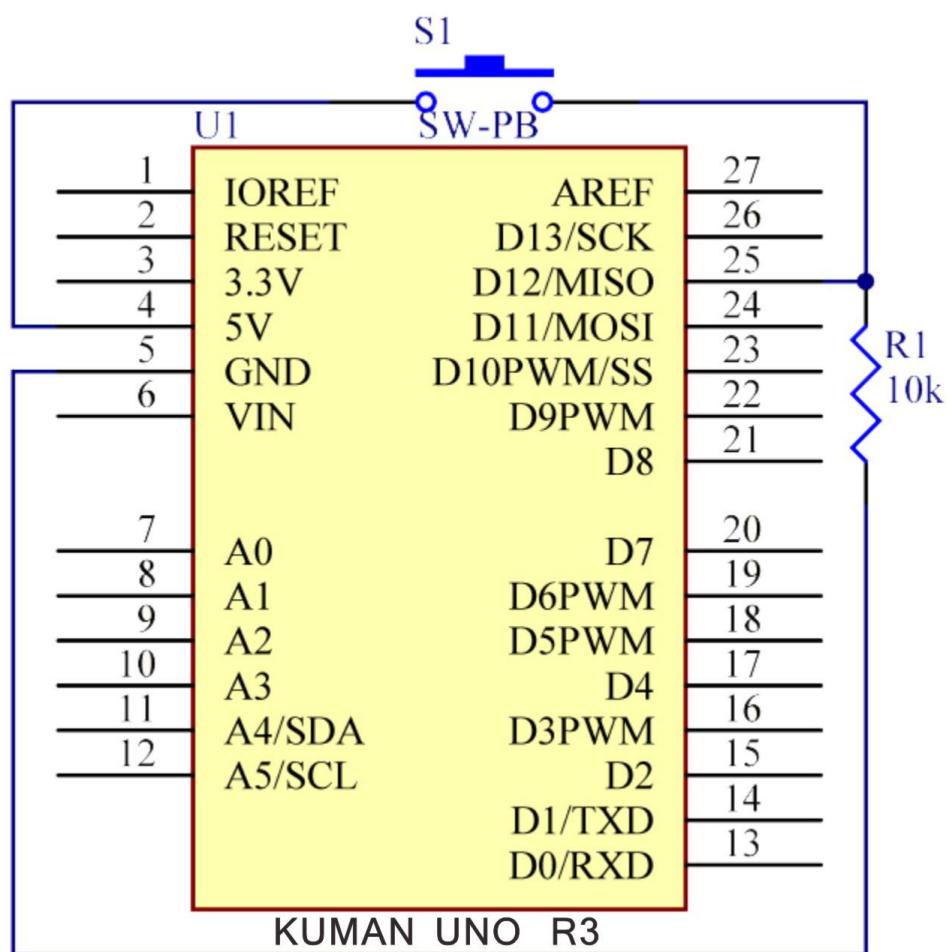


Diagramme schéma

Etape 2 : Compiler le code

Etape 3 : télécharger le sketch sur la carte Kuman Uno

Maintenant, presser le bouton et la LED de la carte Kuman uno va s'allumer

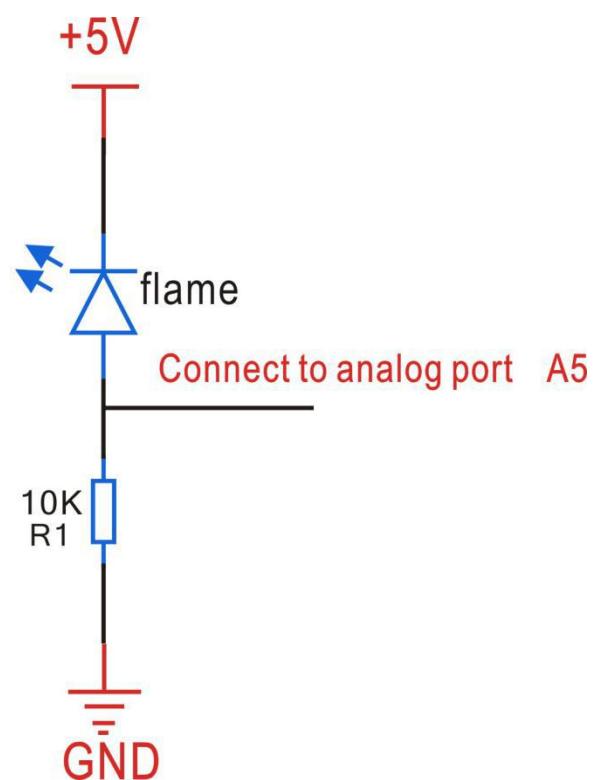
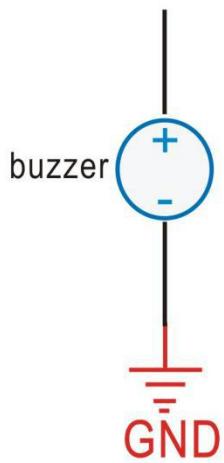
## Code

```
//Controlling Led By Button
//Turns on and off a LED ,when pressings button attach to pin12
//2015.5.7
/*****************/
const int keyPin = 12; //the number of the key pin
const int ledPin = 13;//the number of the led pin
/*****************/
void setup()
{
pinMode(keyPin,INPUT);//initialize the key pin as input
pinMode(ledPin,OUTPUT);//initialize the led pin as output
}
/*****************/
void loop()
{
//read the state of the key value
//and check if the kye is pressed
//if it is,the state is HIGH
if(digitalRead(keyPin) ==HIGH )
{
digitalWrite(ledPin,HIGH);//turn on the led
}
else
{
digitalWrite(ledPin,LOW);//turn off the led
}
}
/*****************/
```

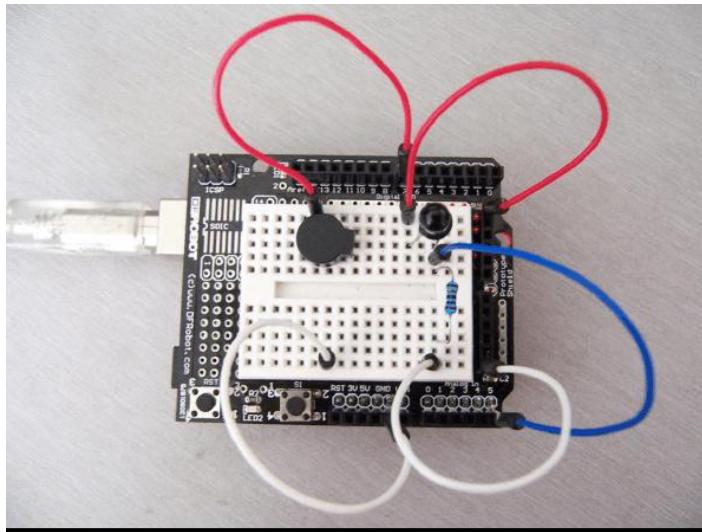
## Leçon 5 : test alarme incendie



Connection interface 8



Longue patte capteur de flamme connexion A5, petite patte connecté 5v, A5 resistance 10K au GND, la connexion positive du buzzer D8, la connexion négative GND, télécharger ensuite le code, vous pouvez utiliser une source de feu près du capteur pour le tester.



### Composants

- 1 \* kuman Uno board
- 1 \* USB cable
- 1 \* Mini bread plate
- cables de démarrage
- 1 \* UNO extension plate
- 1 \* buzzer

-1 \* capteur de flamme

-1 \* résistance 10 k

Code:

```
int flame=A5;//Define flame interface
int Beep=8;//Define buzzer interface
int val=0;//Define numeric variables

val void setup()
{ pinMode(Beep,OUTPUT);//Define Beep as output interface
pinMode(flame,INPUT);//Define flame as input interface
Serial.begin(9600);//Set the baud rate is 9600
}
```

```
void loop()
{
val=analogRead(flame);//Simulated values of the flame sensor
Serial.println(val);//Output analog value, and print it out
if(val>=600)//When the analog value is greater than 600 when the buzzer sounds
{ digitalWrite(Beep,HIGH);
}
else { digitalWrite(Beep,LOW);
}
}
```

## Leçon 6, test matrix 8x8 dot

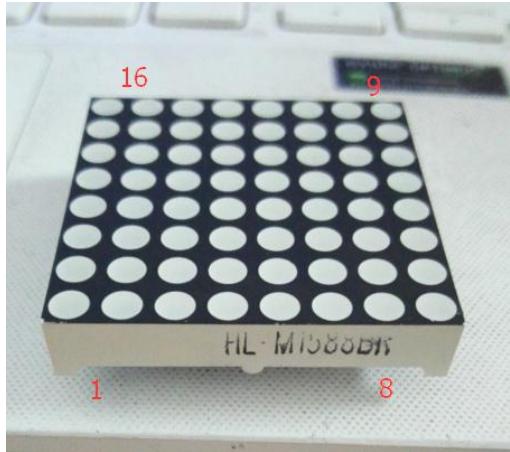
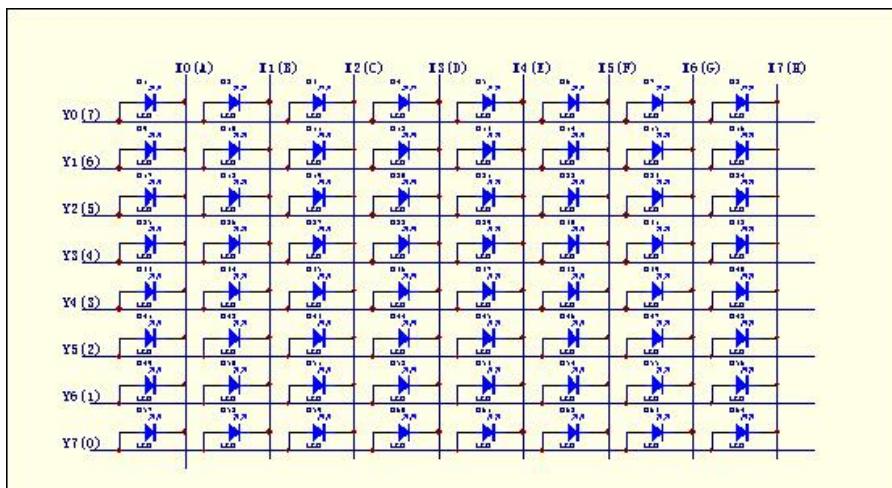


Diagramme dot matrix 8x8



Plan physique matrix dot 8x8

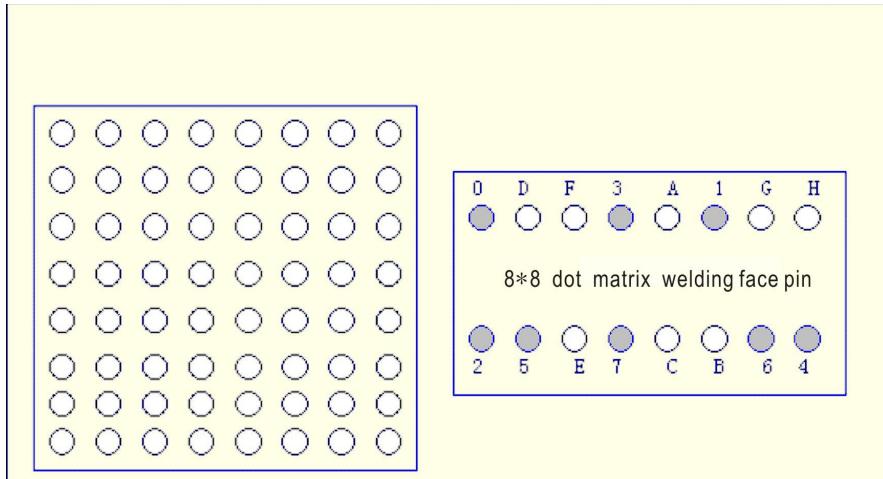


Diagramme de broches et d'apparence LED matrix dot 8x8, c'est équivalent au circuit montré en dessin 2, aussi longtemps qu'il correspond à Y, axe X vers la polarisation, vous pouvez allumer la LED. Par exemple, si vous voulez que le coin supérieur gauche de la LED s'allume, ce sera donc X0=0, Y0=1 peut être placé sur l'axe X ou Y.

#### **Mode scanning matrix dot 8x8**

La LED utilise généralement un affichage de type scan, utilisation actuelle de 3 façons, (1) scan spot, (2) colonne scan,  $16 * 64 = 1024$  Hz, période moins de 1ms. Si vous utilisez le 2ème et la 3ème façon, la fréquence doit être meilleure que  $16 * 8 = 128$  Hz, période moins de 7.8ms conforme aux conditions de la persistance de vision. En plus, une colonne ou ligne (8 LEDs) sur un circuit conducteur externe améliore le courant, sinon le degré de luminosité de la LED ne serait pas suffisant.

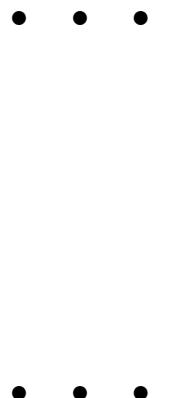
#### **4. Exemple d'applications matrix dot 8x8**

La broche quand la structure en treillis est comme suit : 8x8 dot matrix consiste en une diode émettant 64 lumières, chaque diode émettrice est placée en intersection de colonnes et rangées, correspondant à un niveau fixe, une colonne fixée au niveau 0, la diode correspondante sera lumineuse, quand à la première lumière, 9 niveaux haut et 13 niveaux bas, le premier point est lumineux/ si la première ligne est allumée, la patte 9 est à niveau haut, et les broches 13,3,4,10,6,11,15,16 sont à niveau bas. Donc la première ligne sera allumée avec la première colonne, 13 au niveau bas et 9,14,8,12,1,7,2,5 au niveau haut, la première colonne sera allumée.

En général, nous utilisons la matrix dot en caractères chinois de 16x16, c'est chaque caractère chinois sur la longitude, il croise chaque des 16 points comme montre. C'est en fait quatre 8x8 dot matrix en 16x16 dot matrix.

Comme indiqué dans l'illustration suivante, "you" correspond à la lampe, car le treillis dans la colonne est à niveau bas, et la ligne est activée haute. Envoyer(1111011101111111,0xF7,0x7F),

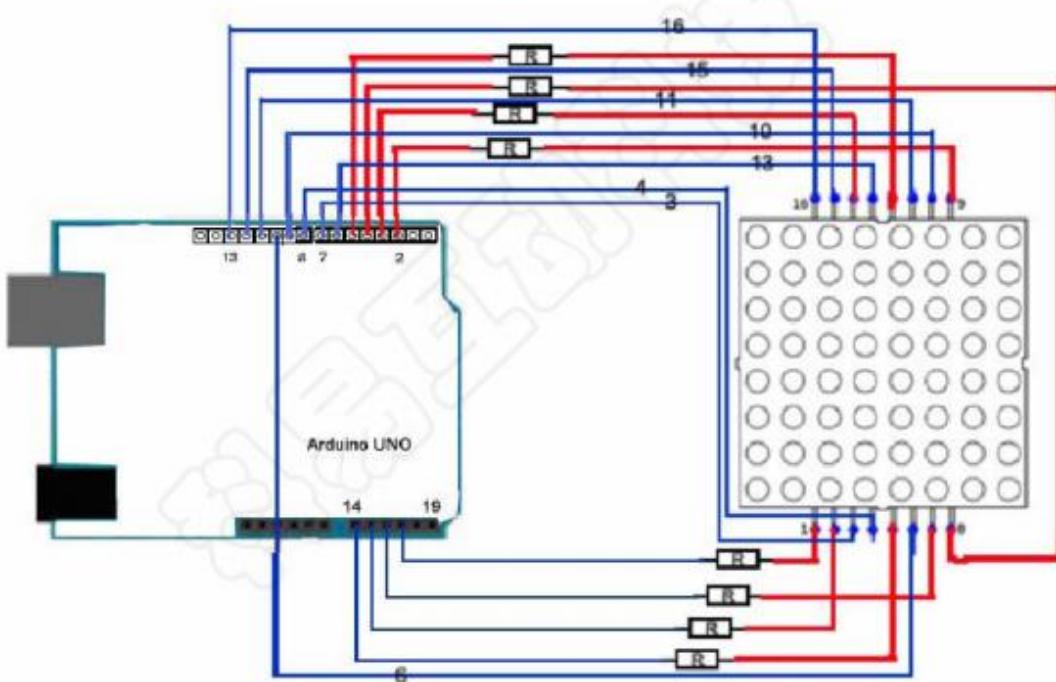
Envoyer la seconde ligne pour afficher les données (13 a 16) pour 1111011101111111,0xF7,0x7F, et la seconde ligne (14) pour envoyer le signal. Par analogie, quand chaque ligne de données montre que le temps d'intervalle est assez court, pour suspendre les effets de la vision humaine, il envoie donc 16 fois la donnée après 16 à scanner, vous verrez le mot « you », la seconde méthode est le signal matrix envoyer dans la colonne à rescaner. Similaire au mot « you » pour illustrer 16 (9,14,8,12 , 1,7,2,5), envoyer (0000000000000000,0x00,0x00) et la première colonne (13) pour envoyer 0. Il scanne la 2ème colonne.



Donc, la formation du code de la colonne 00h, 00h, 3EH, 41h, 41h, 3EH, 00h, 00h;

aussi longtemps que les codes sont envoyés respectivement à la colonne correspondante du dessus, vous pouvez atteindre l'affichage digital « 0 ».

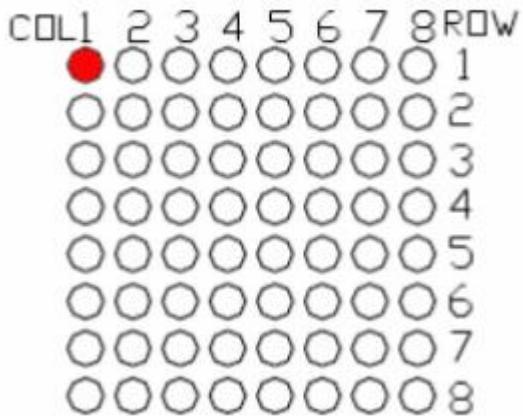
### Diagramme de connexion de ce test.



### Composants

- 1 \* KUMAN Uno board
- 1 \* carte de prototypage
- cables de démarrage
- 8x8 dot matrix
- 8 \* (220R) resistance
- 1 \* USB cable

allumage de la LED matrix dot 8x8 comme suit



\*\*\*\*\*

Instance code:

```
//the pin to control ROW
const int row1 = 2; // the number of the row pin 9
const int row2 = 3; // the number of the row pin 14
const int row3 = 4; // the number of the row pin 8
const int row4 = 5; // the number of the row pin 12
const int row5 = 17; // the number of the row pin 1
const int row6 = 16; // the number of the row pin 7
const int row7 = 15; // the number of the row pin 2
const int row8 = 14; // the number of the row pin 5

//the pin to control COI
const int col1 = 6; // the number of the col pin 13
const int col2 = 7; // the number of the col pin 3
const int col3 = 8; // the number of the col pin 4
const int col4 = 9; // the number of the col pin 10
const int col5 = 10; // the number of the col pin 6
const int col6 = 11; // the number of the col pin 11
const int col7 = 12; // the number of the col pin 15
const int col8 = 13; // the number of the col pin 16

void setup(){
int i = 0 ;
for(i=2;i<18;i++)
{
pinMode(i, OUTPUT);
}
pinMode(row5, OUTPUT);
pinMode(row6, OUTPUT);
```

```

pinMode(row7, OUTPUT);
pinMode(row8, OUTPUT);
for(i=2;i<18;i++) {
digitalWrite(i, LOW);
}
digitalWrite(row5, LOW);
digitalWrite(row6, LOW);
digitalWrite(row7, LOW);
digitalWrite(row8, LOW);
}
void loop(){
int i;
//the row # 1 and col # 1 of the LEDs turn on
digitalWrite(row1, HIGH);
digitalWrite(row2, LOW);
digitalWrite(row3, LOW);
digitalWrite(row4, LOW);
digitalWrite(row5, LOW);
digitalWrite(row6, LOW);
digitalWrite(row7, LOW);
digitalWrite(row8, LOW);
digitalWrite(col1, LOW);
digitalWrite(col2, HIGH);
digitalWrite(col3, HIGH);
digitalWrite(col4, HIGH);
digitalWrite(col5, HIGH);
digitalWrite(col6, HIGH);
digitalWrite(col7, HIGH);
digitalWrite(col8, HIGH);
delay(1000);
//turn off all
for(i=2;i<18;i++) {
digitalWrite(i, LOW);
}
delay(1000);
}

```

\*\*\*\*\*

**L'autre code expérimental est le suivant**

**Affichage du A, position dans la dot matrix 1 au travers de l'affichage scan dynamique**

```

#define data_ascii_A 0x02,0x0C,0x18,0x68,0x68,0x18,0x0C,0x02 /*"A",0*/
/***
 **"A"
#define A { //
{0, 0, 0, 0, 0, 0, 1, 0}, //0x02
{0, 0, 0, 0, 1, 1, 0, 0}, //0x0C
{0, 0, 0, 1, 1, 0, 0, 0}, //0x18
{0, 1, 1, 0, 1, 0, 0, 0}, //0x68
{0, 1, 1, 0, 1, 0, 0, 0}, //0x68
{0, 0, 0, 1, 1, 0, 0, 0}, //0x18
{0, 0, 0, 0, 1, 1, 0, 0}, //0x0C
{0, 0, 0, 0, 0, 0, 0} //0x02
}

```

Set the value to 1 , then the Led will be turn on !

code:

```

#define display_array_size 8
// ascii 8x8 dot font
#define data_null 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00 // null char
#define data_ascii_A 0x02,0x0C,0x18,0x68,0x68,0x18,0x0C,0x02 /*"A",0*/
/***
 **"A"
#define A { //
{0, 0, 0, 0, 0, 0, 1, 0}, //0x02
{0, 0, 0, 0, 1, 1, 0, 0}, //0x0C
{0, 0, 0, 1, 1, 0, 0, 0}, //0x18
{0, 1, 1, 0, 1, 0, 0, 0}, //0x68
{0, 1, 1, 0, 1, 0, 0, 0}, //0x68
{0, 0, 0, 1, 1, 0, 0, 0}, //0x18
{0, 0, 0, 0, 1, 1, 0, 0}, //0x0C
{0, 0, 0, 0, 0, 0, 1, 0} //0x02
}
*/
#define data_ascii_B 0x00,0x7E,0x52,0x52,0x52,0x52,0x2C,0x00 /*"B",1*/
#define data_ascii_C 0x00,0x3C,0x66,0x42,0x42,0x42,0x2C,0x00 /*"C",2*/
#define data_ascii_D 0x00,0x7E,0x42,0x42,0x42,0x66,0x3C,0x00 /*"D",3*/
#define data_ascii_E 0x00,0x7E,0x52,0x52,0x52,0x52,0x42 /*"E",4*/
#define data_ascii_F 0x00,0x7E,0x50,0x50,0x50,0x50,0x50,0x40 /*"F",5*/
#define data_ascii_G 0x00,0x3C,0x66,0x42,0x42,0x52,0x16,0x1E /*"G",6*/
#define data_ascii_H 0x00,0x7E,0x10,0x10,0x10,0x10,0x7E,0x00 /*"H",7*/
#define data_ascii_I 0x00,0x00,0x00,0x7E,0x00,0x00,0x00,0x00 /*"I",8*/
// display array
byte data_ascii[][display_array_size] = {
    data_null,

```

```

data_ascii_A, data_ascii_B,
data_ascii_C,
data_ascii_D,
data_ascii_E,
data_ascii_F,
data_ascii_G,
data_ascii_H,
data_ascii_I,
};

//the pin to control ROW
const int row1 = 2; // the number of the row pin 24
const int row2 = 3; // the number of the row pin 23
const int row3 = 4; // the number of the row pin 22
const int row4 = 5; // the number of the row pin 21
const int row5 = 17; // the number of the row pin 4
const int row6 = 16; // the number of the row pin 3
const int row7 = 15; // the number of the row pin 2
const int row8 = 14; // the number of the row pin 1
//the pin to control COI
const int col1 = 6; // the number of the col pin 20
const int col2 = 7; // the number of the col pin 19
const int col3 = 8; // the number of the col pin 18
const int col4 = 9; // the number of the col pin 17
const int col5 = 10; // the number of the col pin 16
const int col6 = 11; // the number of the col pin 15
const int col7 = 12; // the number of the col pin 14
const int col8 = 13; // the number of the col pin 13

void displayNum(byte rowNum,int colNum)
{
    int j;
    byte temp = rowNum;
    for(j=2;j<6;j++)
    {
        digitalWrite(j, LOW);
    }
    digitalWrite(row5, LOW);
    digitalWrite(row6, LOW);
    digitalWrite(row7, LOW);
    digitalWrite(row8, LOW);
    for(j=6;j<14;j++)
    {
        digitalWrite(j, HIGH); }
    switch(colNum)

```

```

{
    case 1: digitalWrite(col1, LOW); break;
    case 2: digitalWrite(col2, LOW); break;
    case 3: digitalWrite(col3, LOW); break;
    case 4: digitalWrite(col4, LOW); break;
    case 5: digitalWrite(col5, LOW); break;
    case 6: digitalWrite(col6, LOW); break;
    case 7: digitalWrite(col7, LOW); break;
    case 8: digitalWrite(col8, LOW); break;
    default: break;
}
for(j = 1 ;j < 9; j++)
{
    temp = (0x80)&(temp) ;
    if(temp==0)
    {
        temp = rowNum<<j;
        continue;
    }
    switch(j)
    {
        case 1: digitalWrite(row1, HIGH); break;
        case 2: digitalWrite(row2, HIGH); break;
        case 3: digitalWrite(row3, HIGH); break;
        case 4: digitalWrite(row4, HIGH); break;
        case 5: digitalWrite(row5, HIGH); break;
        case 6: digitalWrite(row6, HIGH); break;
        case 7: digitalWrite(row7, HIGH); break;
        case 8: digitalWrite(row8, HIGH); break;
        default: break;
    }
    temp = rowNum<<j;
}
}

void setup(){
    int i = 0 ;
    for(i=2;i<18;i++)
    {
        pinMode(i, OUTPUT);
    }

    for(i=2;i<18;i++) {
        digitalWrite(i, LOW);

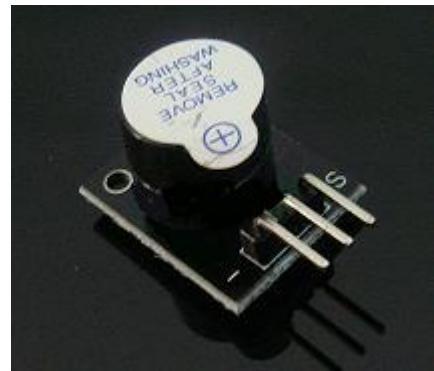
```

```
    }
}

void loop(){
    int t1;
    int l;
    int arrage;
    for(arrage=0;arrage<10;arrage++)
    {
        for(l=0;l<512;l++)
        {
            for(t1=0;t1<8;t1++)
            {
                displayNum(data_ascii[arrage][t1],(t1+1));
            }
        }
    }
}
```

# Buzzer module

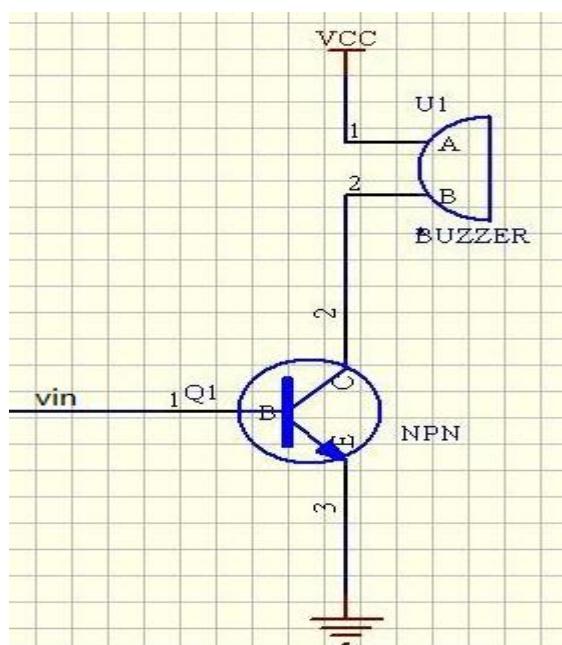
## Leçon 7. Module buzzer actif



Module arduino buzzer  
enceinte active  
Compatible avec PC, imprimante, système audio DIY

### Spécificités

Courant 5V  
Couleur : noir et gris  
Dimensions du colis : 77x42x13mm  
Poids : 5g



Composants requis :

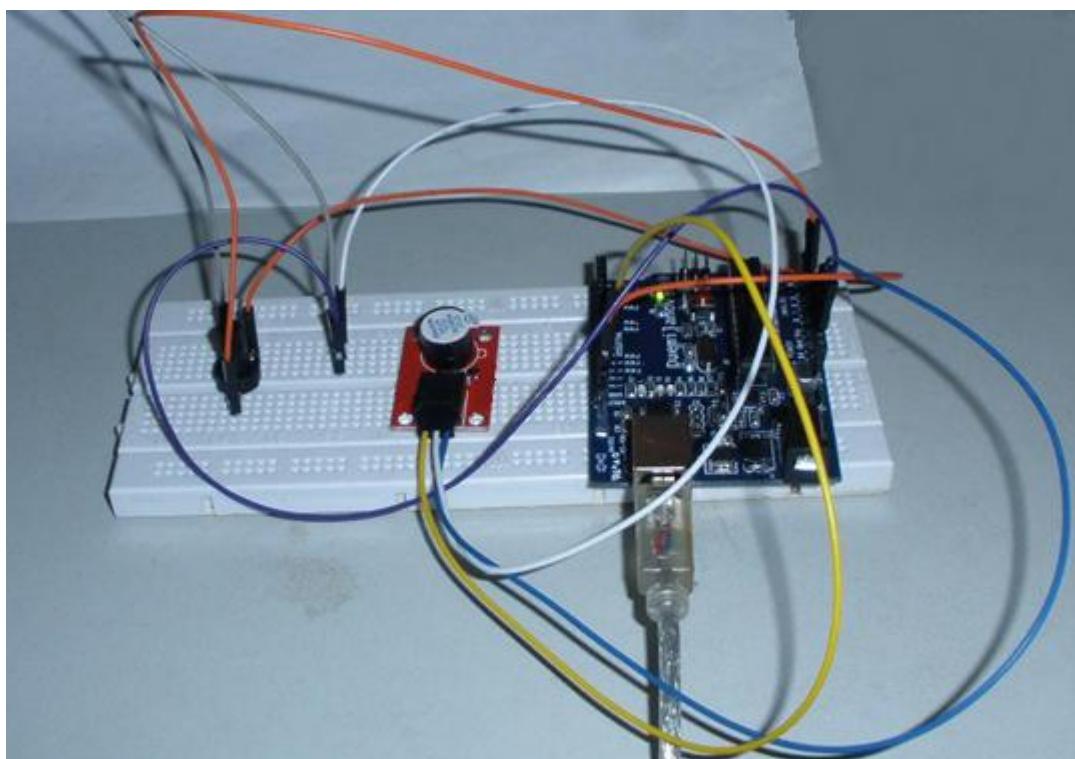
Arduino Controller × 1

USB Cable × 1

Buzzer Module × 1

10K potentiometer (10K) × 1

C



## Connexion

Le test ci-dessous utilisé la fréquence du buzzer

La broche 10 est pour contrôler le buzzer

La broche 3 est un contrôle analogue, et nous utilisons un potentiomètre de 10K

Fonction : ajuster le potentiomètre , nous pouvons entendre le changement de fréquence du buzzer

```
int speakerPin = 8;//Control horn pin
int potPin = 4;//Pin for controlling the adjustable resistor
int value = 0;
void setup() {
    pinMode(speakerPin, OUTPUT);
}
void loop() {
```

```

value = analogRead(potPin);Read the value of the resistor pin
digitalWrite(speakerPin, HIGH);
delay(value);Adjust the time of the horn;
digitalWrite(speakerPin, LOW);
delay(value);Adjust the duration of the horn;
}

```

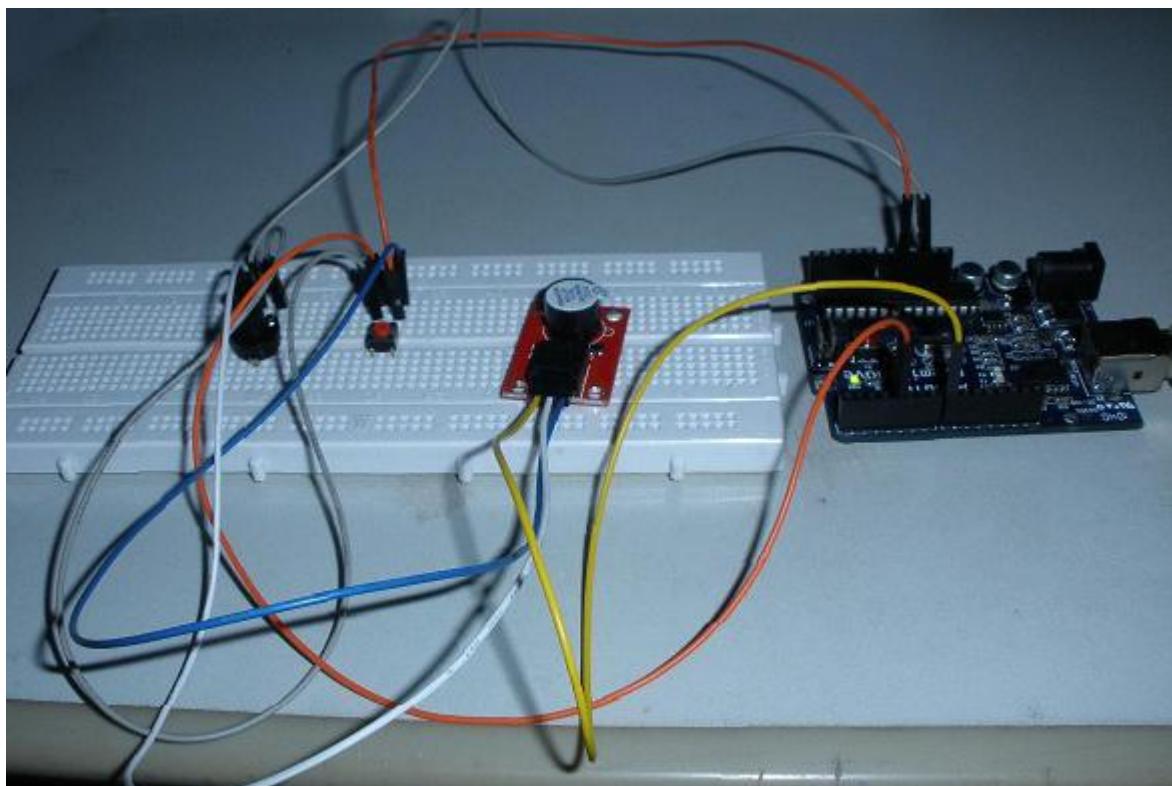
Nous pouvons utiliser le potentiomètre pour changer le temps de retardement, pour pouvoir ensuite modifier la fréquence du buzzer. Nous ajoutons un bouton interrupteur pour contrôler le buzeer, pour imiter une sonnette simple,

Quand le bouton est pressé, l'enceinte émet du bruit.

autre test

Nous ajoutons un bouton pour contrôler le buzzer, nous imitons ensuite une simple sonnette. Quand le bouton est pressé, le buzzer sonne.

moyen de connexion



code

```
const int buttonPin = 4;      // Push pin;
const int speakerPin = 8;      //Buzzer pin;
// variables will change:
int buttonState = 0;          // Read a value of the button pin
void setup()
{
    //Set the button pin for the input mode, the buzzer pin for the output mode;
    pinMode(speakerPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}
void loop(){
    // Read the key to an initial value, here in the circuit, I was in the default high, so the initial value is high;
    buttonState = digitalRead(buttonPin);
    /*If the button is high, then the buzzer does not ring;
    Because I just started in the hardware circuit in the initial value is high, so the if condition was established, the
    buzzer does not ring
    */
    if (buttonState == HIGH) {
        digitalWrite(speakerPin,LOW);

    }
    else {
        //The value of the button here is low (also when the button is pressed).
        digitalWrite(speakerPin,HIGH);
    }
}
```

Below is use PWM to control buzzer:

Code:

```
int speakerPin = 8;
byte song_table[] = { 30, 30, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140,
150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 250, 240, 230, 220, 210, 200, 190, 180,
170, 160, 150, 140, 130, 120, 110, 100, 90, 80, 70, 60, 50, 40, 30, 30, 30 };
int MAX = 50;
int count = 0;
void setup() {
```

```
pinMode(speakerPin, OUTPUT);
}
void loop() {
    analogWrite(speakerPin,song_table[count]);
    count++;
    if (count > MAX) {
        count = 0;
    }
    delay(50);
}
```

# Leçon 8. Buzzer passif

## Introduction

Un buzzer est un outil très utile dans vos expérimentations chaque fois que vous souhaitez émettre des sons.

## Composants

- 1 \* KUMAN Uno board carte kuman uno
- 1 \* carte de prototypage
- 1 \* USB cable de données
- 1 \* Buzzer (Active) buzzer
- câbles de démarrage

## Principe de l'essai

Comme un buzzer électronique avec structure intégrée, les buzzers, qui sont alimentés par un courant DC, sont largement utilisés dans les ordinateurs, les imprimantes, les photocopieurs, les alarmes, les jouets électroniques, les appareils électroniques automobiles, les téléphones, minuteurs, et autres produits électroniques pour les sons. Les buzzers peuvent être actifs ou passifs, (voir dessin ci-dessous). Tourner les broches de deux buzzers vers le haut, celui avec un circuit vert est un buzzer passif, tandis que celui avec un scotch noir est un buzzer actif.



La différence entre un buzzer actif et un buzzer passif.

Un buzzer actif a une source oscillante intégrée, il émettra des bruits lorsqu'il a du courant. Un buzzer passif n'a pas une telle source, il ne fera pas de bruit si le DC est utilisé, il faudra utiliser des vagues de fréquence comprise entre 2K et 5K pour le faire fonctionner. Le buzzer actif est souvent plus cher que le passif, car il a de nombreux circuits oscillants intégrés.

### Procédure de test.

#### Etape 1 : construire le circuit

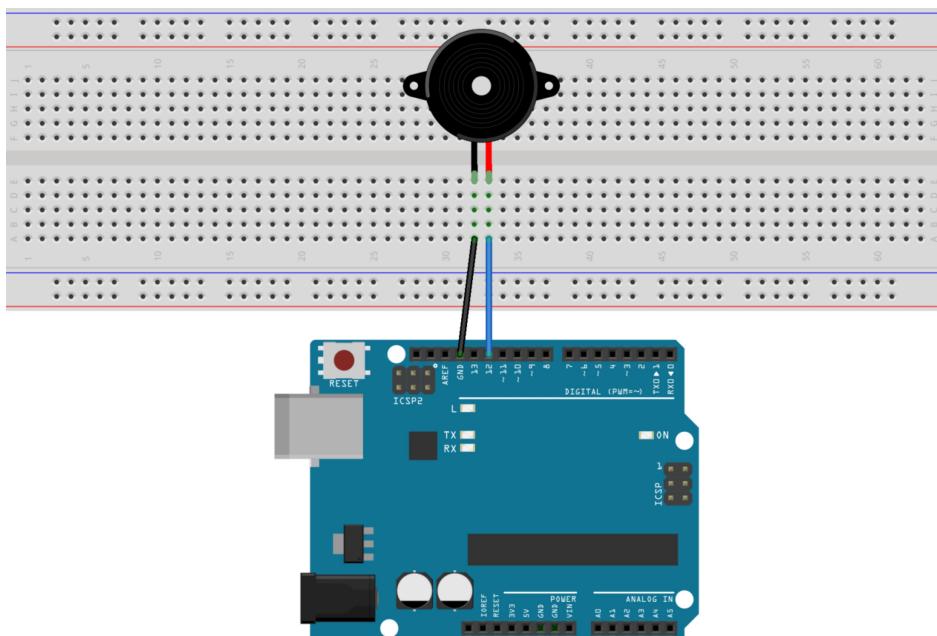
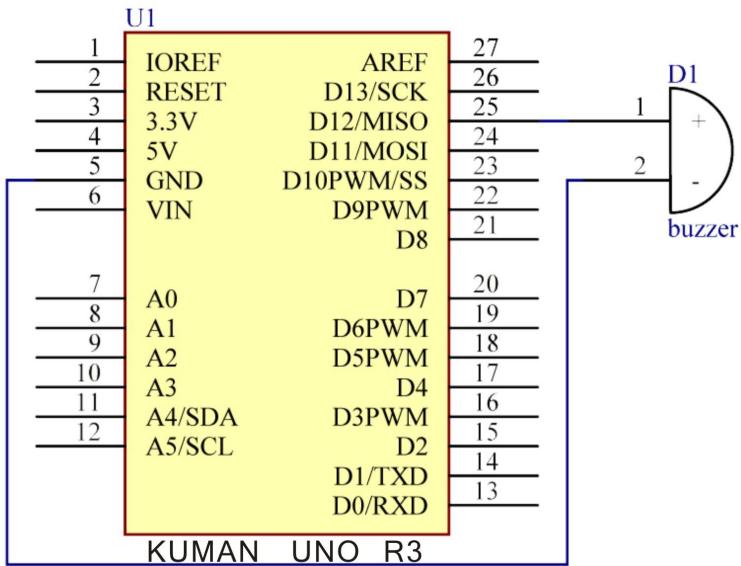


Diagramme schéma



Etape 2 : compiler le code.

Etape 3 : télécharger le sketch sur la carte Kuman uno

#### Code

```
// Buzzer
// buzzer make sounds
//Email:350071174@qq.com
/*****************************************/
int buzzer = 12;//the pin of the active buzzer
void setup()
{
pinMode(buzzer,OUTPUT);//initialize the buzzer pin as an output
}
void loop()
{
unsigned char i; //define a variable
while(1)
{
//output an frequency
for(i=0;i<80;i++)
{
digitalWrite(buzzer,HIGH);
delay(1);//wait for 1ms
digitalWrite(buzzer,LOW);
delay(1);//wait for 1ms
}
//output another frequency
for(i=0;i<100;i++)
{
```

```
digitalWrite(buzzer,HIGH);
delay(2);//wait for 2ms
digitalWrite(buzzer,LOW);
delay(2);//wait for 2ms
}
}
}
*****
/*****
```

## Le buzzer et son principe

### (一) Introduction du buzzer

1 : L'action du buzzer est une structure intégrée au buzzer électronique, alimenté par un courant DC, largement utilisé dans les ordinateurs, imprimantes, photocopieurs, alarmes, jouets électroniques, composants automobiles électroniques, téléphones, minuteurs, etc, produits électroniques pour appareils sonores.

2 La classification du buzzer est séparée en 2 types : le buzzer piezoelectrique et le buzzer électromagnétique

3 symboles du buzzer dans le circuit graphique : lettres 'H' ou 'HA' (l'ancien standard "FM", "LB", "JD", etc.)

### (二) Principe de la structure du buzzer

1 : le buzzer piezoelectrique est principalement composé d'un oscillateur multi harmonique, une pièce buzzing piezoelectrique, un appareil d'impédance, une boîte de résonnance, une coque.

L'oscillateur multiharmonique est composé d'un transistor et d'un circuit intégré, quand le courant est mis en marche, le DC 1.5~15V s'allume, l'oscillateur fonctionne et le signal audio 1.5~2.5kHz est output, l'appareil impédance conduit le buzzer piezoélectrique à émettre un son.

La pièce du buzzer piezoelectrique qui émet le son est faite de zirconium titanate, plomb ou magnésium céramique, plaque avec une électrode grise des deux côtés de la feuille

céramique, et adhère à la feuille en acier inoxydable après polarization et traitement.

2 . The electromagnetic buzzer is composed of an oscillator, an electromagnetic coil, a magnet, a vibrating diaphragm and an outer shell, etc. the audio signal of the oscillator is connected with an electromagnetic coil, and the electromagnetic coil generates a magnetic field.

Le buzzer électromagnétique est composé d'un oscillateur, une bobine, un aimant, un diaphragme vibrant, une coque externe, etc le signal audio de l'oscillateur est connecté à la bobine électromagnétique qui génère un champ magnétique

Quelle est la différence entre le buzzer actif et le buzzer passif

Ici la source n'est pas une source électrique, mais fait référence au choc. C'est la source du buzzer actif à l'intérieur. Si vous utilisez un signal DC, ca ne fera pas sonner. Il faut utiliser les vagues de 2K~5K pour qu'il fonctionne. Le buzzer actif est souvent plus cher que le buzzer passif, car il y a plusieurs circuits.

Les avantages du buzzer passif sont : 1. Bon marché. 2.le contrôle de la fréquence du son permet un effet different.

Programme source de référence Arduino

```
int buzzer=8;//Set the number of IO pins to control the buzzer
void setup()
{
pinMode(buzzer,OUTPUT);//Set the digital IO foot mode, OUTPUT for the output
}
void loop()
{
unsigned char i,j;// defined variable
while(1)
{
for(i=0;i<80;i++)//Output a frequency of sound
{
digitalWrite(buzzer,HIGH);//Hair sound
delay(1);//delay 1ms
digitalWrite(buzzer,LOW);//No sound
delay(1);//delay 1ms
}
}
```

```
for(i=0;i<100;i++)//Output another frequency of sound
{
digitalWrite(buzzer,HIGH);//Hair sound
delay(2);//delay 2ms
digitalWrite(buzzer,LOW);//No sound
delay(2);//delay 2ms
}
}
}
```

Télécharger le programme. L'essai est terminé.

## Leçon 9. Essai 1602 crystal liquide

Ce test utilise le texte d'affichage crystal liquide 1602. L'application crystal liquide 1602 est très large, le 1602 initial est utilisé avec le contrôleur HD44780, et maintenant les modules 1602 des différents fabricants utilisent généralement un IC compatible, dans les caractéristiques sont les mêmes.

Principaux paramètres du LCD 1602

Capacité d'affichage 16x2 caractères

Courant de la puce 4.5 ~ 5.5V;

courant de fonctionnement 2.0mA (5.0V);

le meilleur courant 5.0V;

Taille de caractère 2.95 x 4.35 (W \* H) mm.

Définition de l'interface LCD

Serial number	symbol	Pin description	Serial number	symbol	Pin description
1	VSS	Power supply cathode -	9	D2	Date I/O
2	VDD	positive +	10	D3	Date I/O
3	VL	Liquid crystal display bias signals	11	D4	Date I/O
4	RS	The data/command select end (V/L)	12	D5	Date I/O
5	R/W	The read/write choice end (H/L)	13	D6	Date I/O
6	E	Can make the signal	14	D7	Date I/O
7	D0	Date I/O	15	BLA	Back light anode
8	D1	Date I/O	16	BLK	Back light anode

#### Spécificités de l'interface

1 : un groupe de 2 groupes d'alimentation électrique est un module de rétroéclairage et utilisé en général une alimentation de 5V. Ce test peut utiliser du 3.3V

2.VL est un régulateur du contraste de broche, peut être ajusté avec un potentiomètre de 5k. ce test utilise une résistance pour fixer le contraste de 1k ohm. Connexion à potentiel haut et bas, l'utilisation de la connexion basse, séries de 1K ohm après le GND

3 RS est a crystal liquid. Le niveau pin est élevé quand la donnee indiquée va etre executée.

RW est aussi a crystal liquid. C'est le choix pour lire et copier l'opération

3. E. aussi un module a crystal liquid, il est généralement sur le bus après le signal d'impulsion positif pour lire la donnée
4. D7-d08 bus parallèle 2ways, utilisé pour transmettre des commandes et des données.

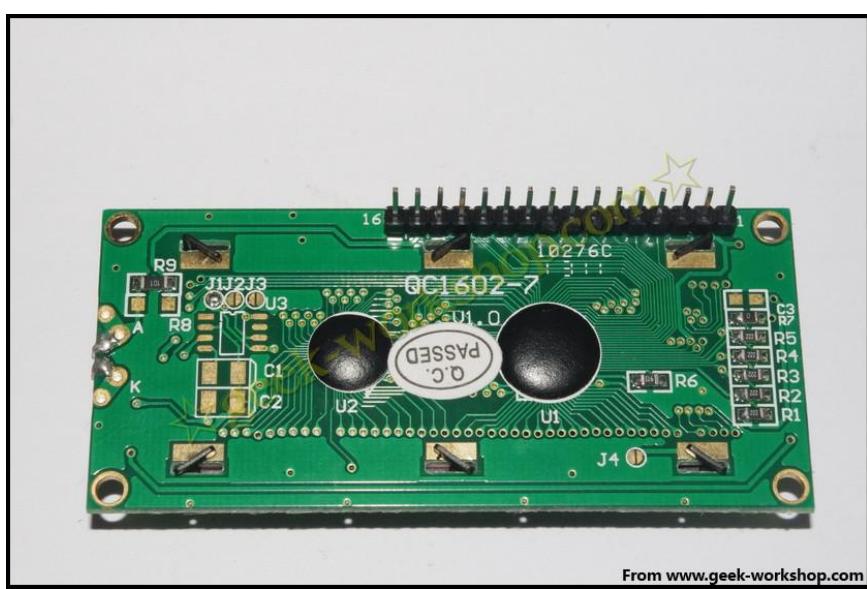
1602 l'opération basique a crystal liquid est divisé en 4 parties :

Read the state	INPUT	RS=L, R/W=H, E=H	OUTPUT	D0~D7= Status word
Written instructions	INPUT	RS=L, R/W=L, D0~D7=Order code E=High pulse	OUTPUT	NC
Read the data	INPUT	RS=H, R/W=H, E=H	OUTPUT	D0~D7= Data
Write the data	INPUT	RS=H, R/W=L, D0~D7=Data E= High pulse	OUTPUT	NC

La figure 1602 LCD physical carte



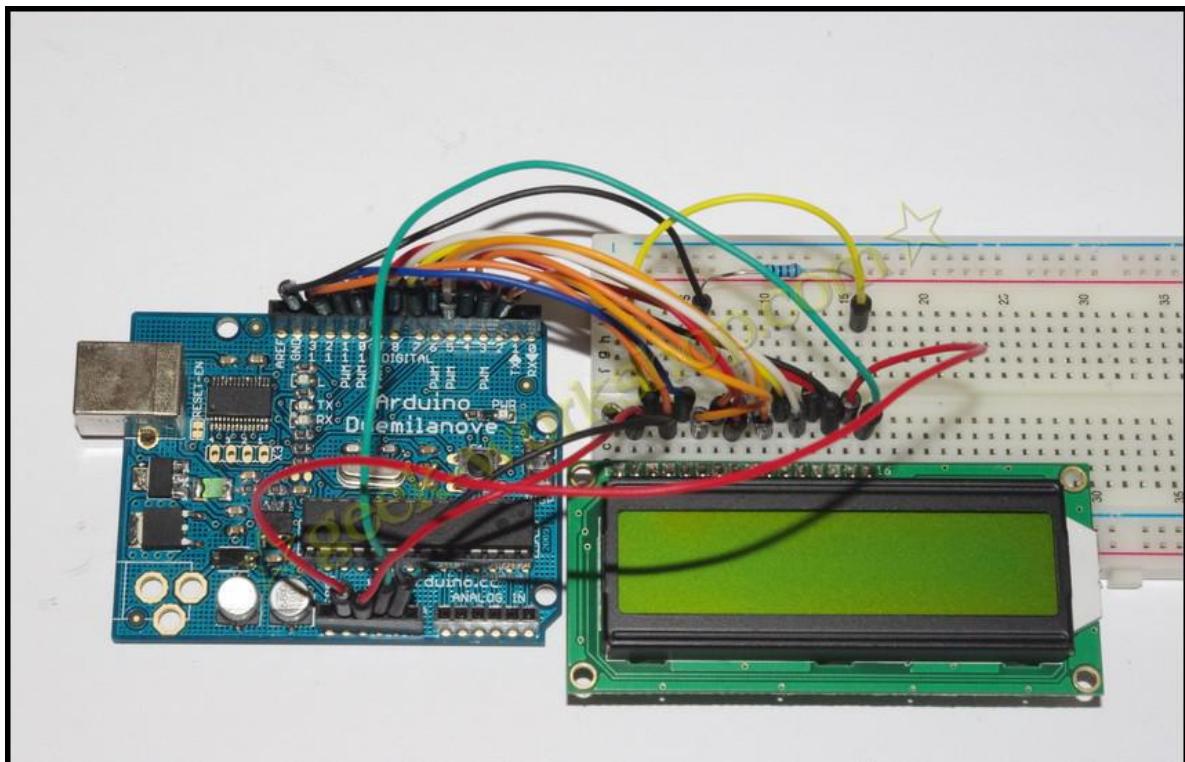
From [www.geek-workshop.com](http://www.geek-workshop.com)



From [www.geek-workshop.com](http://www.geek-workshop.com)

Communication du 1602 directe avec Arduino, selon la description du manuel, il est divisé en 8 méthodes de

connexions ou 4 méthodes, nous utiliserons dans un premier temps la méthode à 8



```
int DI = 12;
int RW = 11;
int DB[] = {3, 4, 5, 6, 7, 8, 9, 10}; //Use an array to define the pins needed by the bus
int Enable = 2;

void LcdCommandWrite(int value) {
// Define all pins
int i = 0;
for (i=DB[0]; i <= DI; i++) //Bus assignment
{
    digitalWrite(i,value & 01); //Because the 1602 liquid crystal signal recognition is D7-D0 (not D0-D7), here is used
    to reverse the signal.
    value >>= 1;
}
digitalWrite(Enable,LOW);
delayMicroseconds(1);
```

```

digitalWrite(Enable,HIGH);
delayMicroseconds(1); //delay 1ms
digitalWrite(Enable,LOW);
delayMicroseconds(1); // delay 1ms
}

void LcdDataWrite(int value) {
// Define all pins
int i = 0;
digitalWrite(DI, HIGH);
digitalWrite(RW, LOW);
for (i=DB[0]; i <= DB[7]; i++) {
    digitalWrite(i,value & 01);
    value >>= 1;
}
digitalWrite(Enable,LOW);
delayMicroseconds(1);
digitalWrite(Enable,HIGH);
delayMicroseconds(1);
digitalWrite(Enable,LOW);
delayMicroseconds(1); //Delay 1ms
}

void setup (void) {
int i = 0;
for (i=Enable; i <= DI; i++) {
    pinMode(i,OUTPUT);
}
delay(100);
// After a short pause, LCD
// For LCD control needs
LcdCommandWrite(0x38); // Set to 8-bit interface, 2 line display, 5x7 text size
delay(64);
LcdCommandWrite(0x38); // Set to 8-bit interface, 2 line display, 5x7 text size
delay(50);
LcdCommandWrite(0x38); //Set to 8-bit interface, 2 line display, 5x7 text size
delay(20);
LcdCommandWrite(0x06); // Input mode setting
                    // Auto increment, no display shift
delay(20);
LcdCommandWrite(0x0E); // Display settings
                    // Turn on the display, the cursor shows, no flicker
delay(20);
LcdCommandWrite(0x01); // The screen is empty, the cursor position is zero

```

```

delay(100);
LcdCommandWrite(0x80); // Display settings
    // Turn on the display, the cursor shows, no flicker
delay(20);
}

void loop (void) {
LcdCommandWrite(0x01); // The screen is empty, the cursor position is zero
delay(10);
LcdCommandWrite(0x80+3);
delay(10);

LcdDataWrite('W');
LcdDataWrite('e');
LcdDataWrite('l');
LcdDataWrite('c');
LcdDataWrite('o');
LcdDataWrite('m');
LcdDataWrite('e');
LcdDataWrite(' ');
LcdDataWrite('t');
LcdDataWrite('o');
delay(10);
LcdCommandWrite(0xc0+1); // Defines the cursor position as second rows and second positions
delay(10);
LcdDataWrite('g');
LcdDataWrite('e');
LcdDataWrite('e');
LcdDataWrite('k');
LcdDataWrite('-');
LcdDataWrite('w');
LcdDataWrite('o');
LcdDataWrite('r');
LcdDataWrite('k');
LcdDataWrite('s');
LcdDataWrite('h');
LcdDataWrite('o');
LcdDataWrite('p');
delay(5000);
LcdCommandWrite(0x01); // The screen is empty, the cursor position is zero
delay(10);
LcdDataWrite('l');
LcdDataWrite(' ');
LcdDataWrite('a');

```

```

LcdDataWrite('m');
LcdDataWrite(' ');
LcdDataWrite('h');
LcdDataWrite('o');
LcdDataWrite('n');
LcdDataWrite('g');
LcdDataWrite('y');
LcdDataWrite('i');
delay(3000);
LcdCommandWrite(0x02); //Set the pattern for the new text to replace the old text, no new text display the same place
delay(10);
LcdCommandWrite(0x80+5); //Defines the cursor position as the first line of the sixth position
delay(10);
LcdDataWrite('t');
LcdDataWrite('h');
LcdDataWrite('e');
LcdDataWrite(' ');
LcdDataWrite('a');
LcdDataWrite('d');
LcdDataWrite('m');
LcdDataWrite('i');
LcdDataWrite('n');
delay(5000);
}

///////////
///////
int DI = 12;
int RW = 11;
int DB[] = {3, 4, 5, 6, 7, 8, 9, 10}; //Use an array to define the pins needed by the bus
int Enable = 2;

```

```

void LcdCommandWrite(int value) {
    // Define all pins
    int i = 0;
    for (i=DB[0]; i <= DI; i++) //Bus assignment
    {
        digitalWrite(i,value & 01); //Because the 1602 liquid crystal signal recognition is D7-D0 (not D0-D7), here is used to reverse the signal.
        value >>= 1;
    }
    digitalWrite(Enable,LOW);
}

```

```

delayMicroseconds(1);
digitalWrite(Enable,HIGH);
delayMicroseconds(1); // delay1ms
digitalWrite(Enable,LOW);
delayMicroseconds(1); // delay 1ms
}

void LcdDataWrite(int value) {
    // Define all pins
    int i = 0;
    digitalWrite(DI, HIGH);
    digitalWrite(RW, LOW);
    for (i=DB[0]; i <= DB[7]; i++) {
        digitalWrite(i,value & 01);
        value >>= 1;
    }
    digitalWrite(Enable,LOW);
    delayMicroseconds(1);
    digitalWrite(Enable,HIGH);
    delayMicroseconds(1);
    digitalWrite(Enable,LOW);
    delayMicroseconds(1); //delay 1ms
}

void setup (void) {
    int i = 0;
    for (i=Enable; i <= DI; i++) {
        pinMode(i,OUTPUT);
    }
    delay(100);
    // After a short pause, LCD
    // For LCD control needs
    LcdCommandWrite(0x38); // Set to 8-bit interface, 2 line display, 5x7 text size
    delay(64);
    LcdCommandWrite(0x38); // Set to 8-bit interface, 2 line display, 5x7 text size
    delay(50);
    LcdCommandWrite(0x38); //Set to 8-bit interface, 2 line display, 5x7 text size
    delay(20);
    LcdCommandWrite(0x06); //Input mode setting
        // Auto increment, no display shift
    delay(20);
    LcdCommandWrite(0x0E); // Display settings
        // Turn on the display, the cursor shows, no flicker
    delay(20);
}

```

```

LcdCommandWrite(0x01); //The screen is empty, the cursor position is zero
delay(100);
LcdCommandWrite(0x80); //Display settings
    // Turn on the display, the cursor shows, no flicker
delay(20);
}

void loop (void) {
    LcdCommandWrite(0x01); // The screen is empty, the cursor position is zero
    delay(10);
    LcdCommandWrite(0x80+3);
    delay(10);
    // write a welcome message
    LcdDataWrite('W');
    LcdDataWrite('e');
    LcdDataWrite('l');
    LcdDataWrite('c');
    LcdDataWrite('o');
    LcdDataWrite('m');
    LcdDataWrite('e');
    LcdDataWrite(' ');
    LcdDataWrite('t');
    LcdDataWrite('o');
    delay(10);
    LcdCommandWrite(0xc0+1); // Defines the cursor position as second rows and second positions.
    delay(10);
    LcdDataWrite('k');
    LcdDataWrite('u');
    LcdDataWrite('m');
    LcdDataWrite('a');
    LcdDataWrite('n');
    LcdDataWrite('A');
    LcdDataWrite('r');
    LcdDataWrite('d');
    LcdDataWrite('u');
    LcdDataWrite('i');
    LcdDataWrite('n');
    LcdDataWrite('o');
    LcdDataWrite('p');
    delay(5000);
    LcdCommandWrite(0x01); // The screen is empty, the cursor position is zero
    delay(10);
    LcdDataWrite('I');
    LcdDataWrite(' ');
}

```

```

LcdDataWrite('a');
LcdDataWrite('m');
LcdDataWrite(' ');
LcdDataWrite('k');
LcdDataWrite('u');
LcdDataWrite('m');
LcdDataWrite('a');
LcdDataWrite('n');
delay(3000);

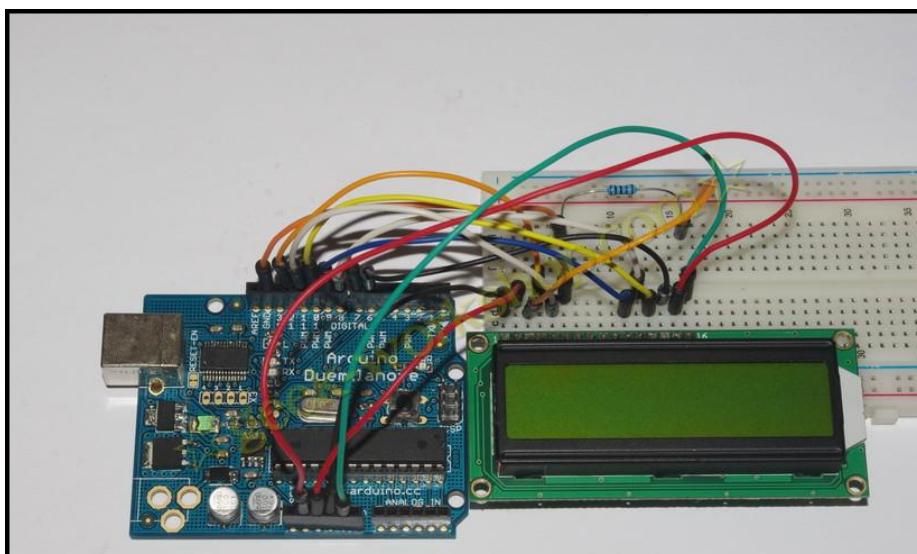
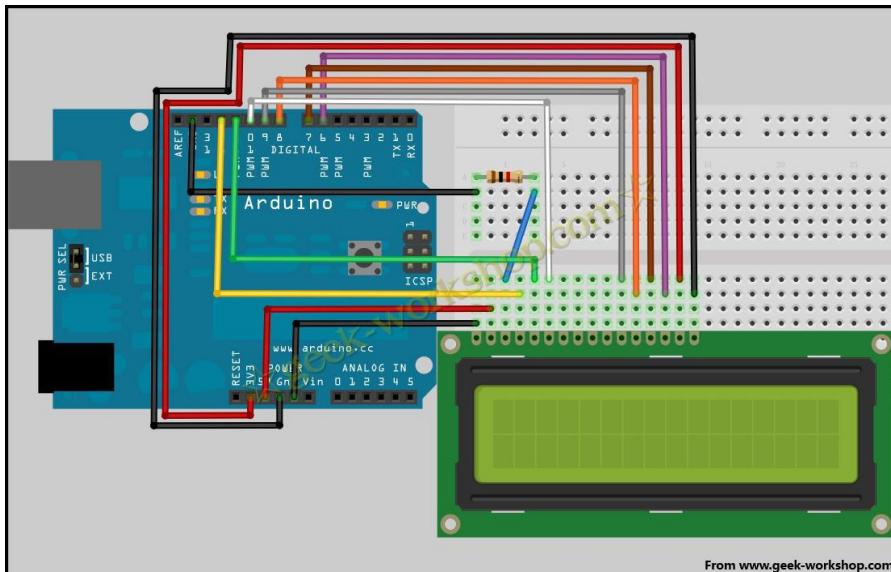
LcdCommandWrite(0x02); /*Set the pattern for the new text to replace the old text, no new text display the same place*/
delay(10);
LcdCommandWrite(0x80+5); //Defines the cursor position as the first line of the sixth position
delay(10);
LcdDataWrite('t');
LcdDataWrite('h');
LcdDataWrite('e');
LcdDataWrite(' ');
LcdDataWrite('a');
LcdDataWrite('d');
LcdDataWrite('m');
LcdDataWrite('i');
LcdDataWrite('n');
delay(5000);

}

```

#### 4 Méthode de connexion a 4bits

En utilisation normale, l'accès 8bit au port digital basique D'arduino est complet, si vous voulez sélectionner quelques capteurs, vous pouvez utiliser l'accès 4bits. La méthode de connexion est la suivante :



Après la connexion avec le hardware, le code suivant est téléchargé sur le panneau de contrôle pour voir le résultat :

```
int LCD1602_RS=12;
```

```

int LCD1602_RW=11;
int LCD1602_EN=10;
int DB[] = { 6, 7, 8, 9};
char str1[]{"Welcome to"};
char str2[]{"geek-workshop"};
char str3[]{"this is the"};
char str4[]{"4-bit interface"};

void LCD_Command_Write(int command)
{
    int i,temp;
    digitalWrite( LCD1602_RS,LOW);
    digitalWrite( LCD1602_RW,LOW);
    digitalWrite( LCD1602_EN,LOW);

    temp=command & 0xf0;
    for (i=DB[0]; i <= 9; i++)
    {
        digitalWrite(i,temp & 0x80);
        temp <<= 1;
    }

    digitalWrite( LCD1602_EN,HIGH);
    delayMicroseconds(1);
    digitalWrite( LCD1602_EN,LOW);

    temp=(command & 0x0f)<<4;
    for (i=DB[0]; i <= 10; i++)
    {
        digitalWrite(i,temp & 0x80);
        temp <<= 1;
    }

    digitalWrite( LCD1602_EN,HIGH);
    delayMicroseconds(1);
    digitalWrite( LCD1602_EN,LOW);
}

void LCD_Data_Write(int dat)
{
    int i=0,temp;
    digitalWrite( LCD1602_RS,HIGH);
    digitalWrite( LCD1602_RW,LOW);
    digitalWrite( LCD1602_EN,LOW);
}

```

```

temp=dat & 0xf0;
for (i=DB[0]; i <= 9; i++)
{
    digitalWrite(i,temp & 0x80);
    temp <<= 1;
}

digitalWrite( LCD1602_EN,HIGH);
delayMicroseconds(1);
digitalWrite( LCD1602_EN,LOW);

temp=(dat & 0x0f)<<4;
for (i=DB[0]; i <= 10; i++)
{
    digitalWrite(i,temp & 0x80);
    temp <<= 1;
}

digitalWrite( LCD1602_EN,HIGH);
delayMicroseconds(1);
digitalWrite( LCD1602_EN,LOW);
}

void LCD_SET_XY( int x, int y )
{
    int address;
    if (y ==0)    address = 0x80 + x;
    else         address = 0xC0 + x;
    LCD_Command_Write(address);
}

void LCD_Write_Char( int x,int y,int dat)
{
    LCD_SET_XY( x, y );
    LCD_Data_Write(dat);
}

void LCD_Write_String(int X,int Y,char *s)
{
    LCD_SET_XY( X, Y );    //Set address
    while (*s)              //Write string
    {
        LCD_Data_Write(*s);
}

```

```

        s++;
    }
}

void setup (void)
{
int i = 0;
for (i=6; i <= 12; i++)
{
    pinMode(i,OUTPUT);
}
delay(100);
LCD_Command_Write(0x28); //4 lines and 2 lines 5x7
delay(50);
LCD_Command_Write(0x06);
delay(50);
LCD_Command_Write(0x0c);
delay(50);
LCD_Command_Write(0x80);
delay(50);
LCD_Command_Write(0x01);
delay(50);

}

void loop (void)
{
    LCD_Command_Write(0x01);
    delay(50);
    LCD_Write_String(3,0,str1); //First lines, fourth addresses
    delay(50);
    LCD_Write_String(1,1,str2); //Second lines, second addresses
    delay(5000);
    LCD_Command_Write(0x01);
    delay(50);
    LCD_Write_String(0,0,str3);
    delay(50);
    LCD_Write_String(0,1,str4);
    delay(5000);
}

Common view copy code to save code print code
int LCD1602_RS=12;
int LCD1602_RW=11;
int LCD1602_EN=10;

```

```

int DB[] = { 6, 7, 8, 9};
char str1[]{"Welcome to"};
char str2[]{"KUMAN Arduino"};
char str3[]{"this is the"};
char str4[]{"4-bit interface"};

void LCD_Command_Write(int command)
{
    int i,temp;
    digitalWrite( LCD1602_RS,LOW);
    digitalWrite( LCD1602_RW,LOW);
    digitalWrite( LCD1602_EN,LOW);

    temp=command & 0xf0;
    for (i=DB[0]; i <= 9; i++)
    {
        digitalWrite(i,temp & 0x80);
        temp <<= 1;
    }

    digitalWrite( LCD1602_EN,HIGH);
    delayMicroseconds(1);
    digitalWrite( LCD1602_EN,LOW);

    temp=(command & 0x0f)<<4;
    for (i=DB[0]; i <= 10; i++)
    {
        digitalWrite(i,temp & 0x80);
        temp <<= 1;
    }

    digitalWrite( LCD1602_EN,HIGH);
    delayMicroseconds(1);
    digitalWrite( LCD1602_EN,LOW);
}

void LCD_Data_Write(int dat)
{
    int i=0,temp;
    digitalWrite( LCD1602_RS,HIGH);
    digitalWrite( LCD1602_RW,LOW);
    digitalWrite( LCD1602_EN,LOW);

    temp=dat & 0xf0;

```

```

for (i=DB[0]; i <= 9; i++)
{
    digitalWrite(i,temp & 0x80);
    temp <<= 1;
}

digitalWrite( LCD1602_EN,HIGH);
delayMicroseconds(1);
digitalWrite( LCD1602_EN,LOW);

temp=(dat & 0x0f)<<4;
for (i=DB[0]; i <= 10; i++)
{
    digitalWrite(i,temp & 0x80);
    temp <<= 1;
}

digitalWrite( LCD1602_EN,HIGH);
delayMicroseconds(1);
digitalWrite( LCD1602_EN,LOW);
}

void LCD_SET_XY( int x, int y )
{
int address;
if (y ==0) address = 0x80 + x;
else address = 0xC0 + x;
LCD_Command_Write(address);
}

void LCD_Write_Char( int x,int y,int dat)
{
LCD_SET_XY( x, y );
LCD_Data_Write(dat);
}

void LCD_Write_String(int X,int Y,char *s)
{
LCD_SET_XY( X, Y ); //Set address
while (*s) / /Write string
{
    LCD_Data_Write(*s);
    s++;
}
}

```

```

}

void setup (void)
{
    int i = 0;
    for (i=6; i <= 12; i++)
    {
        pinMode(i,OUTPUT);
    }
    delay(100);
    LCD_Command_Write(0x28); //4 lines and 2 lines 5x7
    delay(50);
    LCD_Command_Write(0x06);
    delay(50);
    LCD_Command_Write(0x0c);
    delay(50);
    LCD_Command_Write(0x80);
    delay(50);
    LCD_Command_Write(0x01);
    delay(50);

}

void loop (void)
{
    LCD_Command_Write(0x01);
    delay(50);
    LCD_Write_String(3,0,str1); //First lines, fourth addresses from
    delay(50);
    LCD_Write_String(1,1,str2); //Second lines, second addresses from
    delay(5000);
    LCD_Command_Write(0x01);
    delay(50);
    LCD_Write_String(0,0,str3);
    delay(50);
    LCD_Write_String(0,1,str4);
    delay(5000);
}

```

# Leçon 10 Essai 74HC595

74HC595 est le décalage de registres 8bits et mémoire, avec 3 états de fonction output. Nous l'utilisons ici pour contrôler 8LED. Pourquoi devons nous utiliser 74HC595 pour contrôler les lampes ? Beaucoup de personnes se posent cette question. Si nous utilisons simplement un Arduino pour contrôler 8 lampes occupées par le nombre I/O correspondant, les 8 lampes occupent trop de ressources. Nous utilisons 74HC595 pour réduire le nombre de port I/O. Avec 74HC595 nous pouvons utiliser 3 ports digitaux pour contrôler 8 lampes.

Voici les composants dont nous avons besoin :

\*1 puce 74HC595

\*4 ligne rouge m5

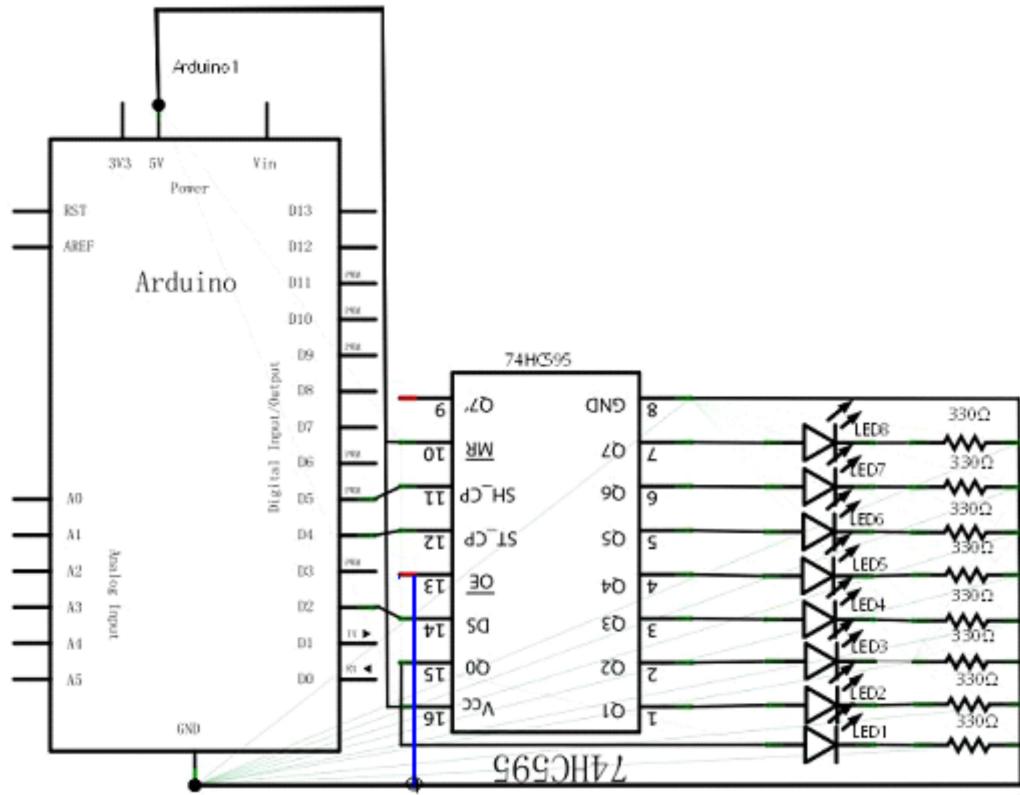
\*4 ligne verte m5

220 ohm line \*8 résistance 220ohm

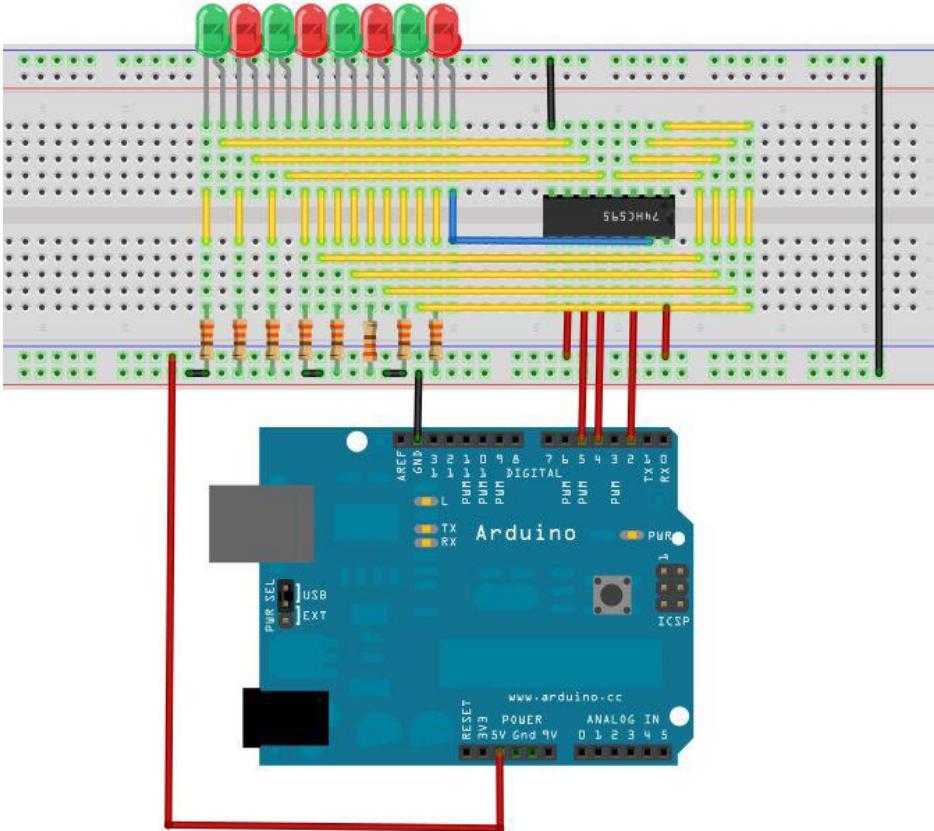
\*1 carte de prototypage

\*1 une bande

Nous sommes prêts à connecter le circuit selon le schéma suivant



Note : la patte OE 74HC595 pour connecter au GND



Le circuit semble complexe, mais nous le trouverons très simple après une étude plus approfondie. Ce qui suit est une référence au programme source :

```

int data = 2;//74HC595 14 pin data input pin SI

int clock = 5;//74HC595 11 pin clock line SCK

int latch = 4;//74hc595 12 pin output memory latch line RCK

int ledState = 0;

const int ON = HIGH;

const int OFF = LOW;

void setup()

{

```

```

pinMode(data, OUTPUT);
pinMode(clock, OUTPUT);
pinMode(latch, OUTPUT);

}

void loop()
{
for(int i = 0; i < 256; i++)
{
updateLEDs(i);
delay(500);
}
}

void updateLEDs(int value)
{
digitalWrite(latch, LOW);//
shiftOut(data, clock, MSBFIRST, ~value);//Serial data output, high priority
digitalWrite(latch, HIGH);//Latch
}

```

Vous pouvez voir les 8 lampes s'allumer. Le phénomène experimental est de voir s'afficher les 8 LED en nombre binaire

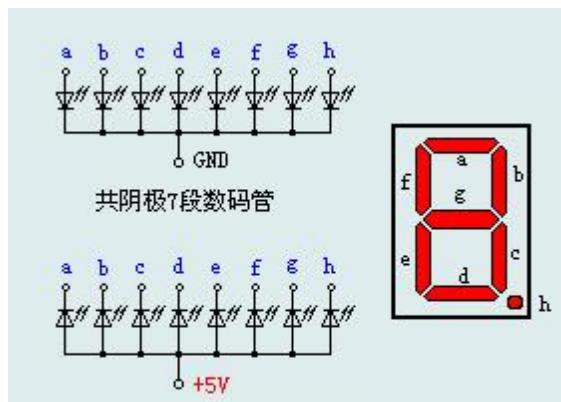
# Leçon 11. Essai affichage tube digital

Le tube digital est un affichage commun, pour la vie de tous les jours, par exemple, four électromagnétique, machine à laver automatique, affichage solaire de la température, horloge électronique...

La cathode du tube digital, est un genre de semiconducteur à émission de lumière, il y a une diode qui émet de la lumière, le tube est divisé en 7 segments et 8 segments,

La connexion est divisée en anode commune et en tube cathode nixie. Le tube digital yang se réfère à la diode émettrice de lumière qui forme une anode (COM). Elle utilise en général +5v, elle utilise peu d'électricité. Quand une cathode utilise beaucoup d'électricité, le champs correspondant n'est pas lumineux.

Le tube digital commun forme une cathode (COM). Dans l'application, elle doit être COM au sol GND, quand une diode anode utilise beaucoup de puissance, le champs correspondant s'allume.



Chaque tube est composé de diodes émettrices de lumière, il faut connecter la résistance sinon le courant va exploser la diode. Ce test avec une cathode de tube digital, doit être GND.

\*1 tube digital 8 segments

\*8 résistance 220ohm

\*1 tie carte de prototypage, bande

On se réfère au diagramme de connection pour connecter le circuit

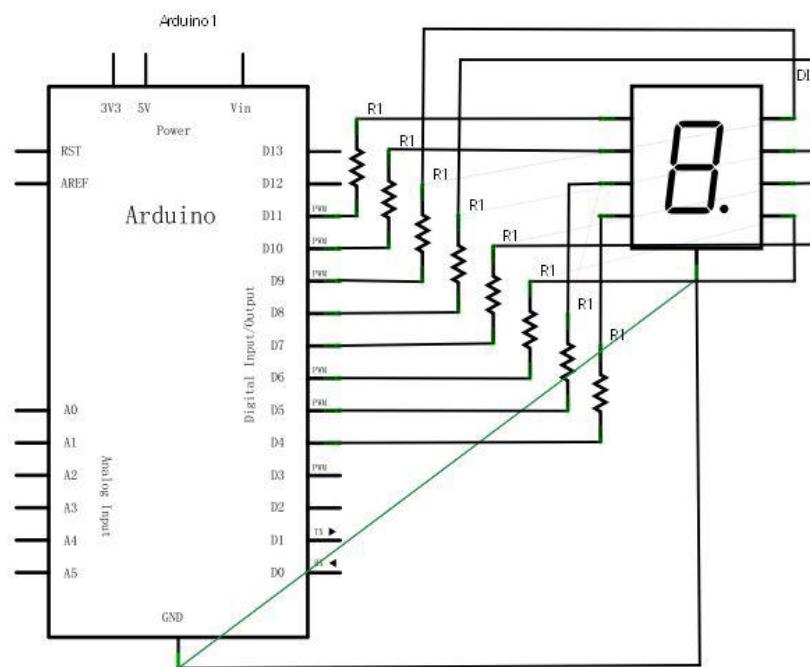
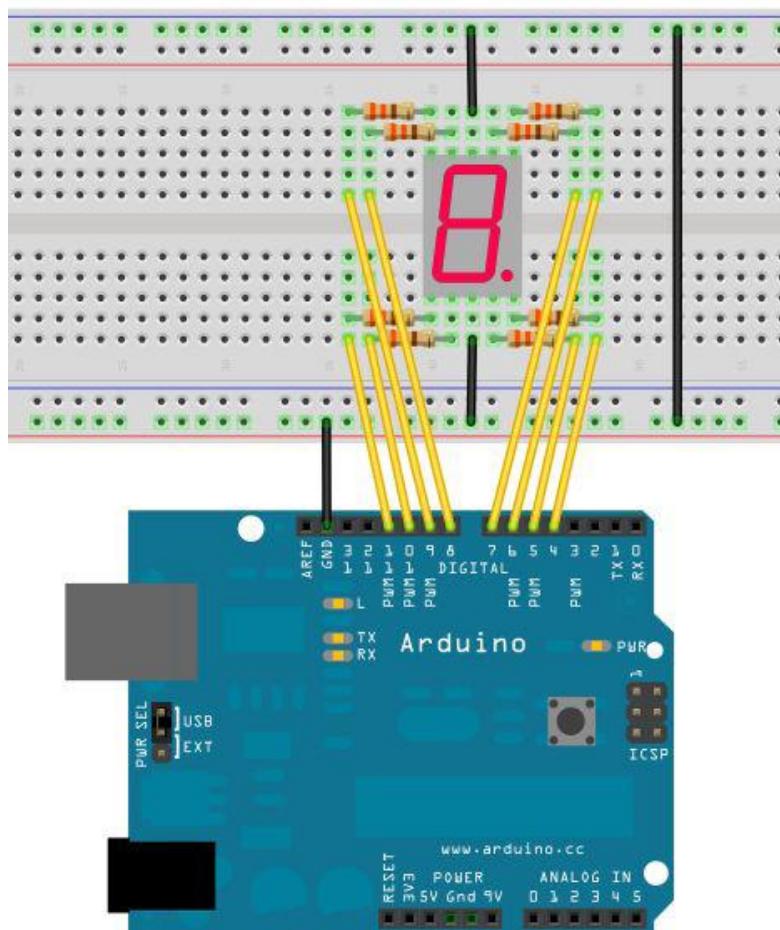


diagram.



7 sections montre le nombre de segments, il y a un affichage du point décimal. Quand le tube digital affiche un digit, il y a une période lumineuse correspondante. Par exemple, le tube affiche 1,C,B la section s'allume. Dans le programme principal, chaque 2s il y a un nombre qui s'affiche, 1 à 8 digits.

```
//Set pin control all digital IO

int a=7;//Definition of digital interface 7 connected a segment digital tube

int b=6;// Definition of digital interface 6 connected B segment digital tubeint c=5;//

Definition of digital interface 5 connected C segment digital tube

int d=10;// Definition of digital interface 10 connected D segment digital tube

int e=11;// Definition of digital interface 11 connected e segment digital tube

int f=8;// Definition of digital interface 8 connected f segment digital tube

int g=9;// Definition of digital interface 9 connected g segment digital tube

int dp=4;// Definition of digital interface 4 connected DP segment digital tube

void digital_0(void) //Display number 5

{

unsigned char j;

digitalWrite(a,HIGH);

digitalWrite(b,HIGH);

digitalWrite(c,HIGH);

digitalWrite(d,HIGH);

digitalWrite(e,HIGH);

digitalWrite(f,HIGH);

digitalWrite(g,LOW);

digitalWrite(dp,LOW);
```

```
}

void digital_1(void) //Display number 1

{
unsigned char j;

digitalWrite(c,HIGH);//Digital interface to the 5 pin high, lit C segment
digitalWrite(b,HIGH);//Light B segment
for(j=7;j<=11;j++)//Extinguish the rest
digitalWrite(j,LOW);
digitalWrite(dp,LOW);//Put out the DP segment of the decimal point
}

void digital_2(void) //Display number 2

{
unsigned char j;

digitalWrite(b,HIGH);
digitalWrite(a,HIGH);
for(j=9;j<=11;j++)
digitalWrite(j,HIGH);
digitalWrite(dp,LOW);
digitalWrite(c,LOW);
digitalWrite(f,LOW);
}

void digital_3(void) //Display number3
```

```
{  
digitalWrite(g,HIGH);  
digitalWrite(a,HIGH);  
digitalWrite(b,HIGH);  
digitalWrite(c,HIGH);  
digitalWrite(d,HIGH);  
digitalWrite(dp,LOW);  
digitalWrite(f,LOW);  
digitalWrite(e,LOW);  
}  
  
void digital_4(void) //Display number 4  
{  
digitalWrite(c,HIGH);  
digitalWrite(b,HIGH);  
digitalWrite(f,HIGH);  
digitalWrite(g,HIGH);  
digitalWrite(dp,LOW);  
digitalWrite(a,LOW);  
digitalWrite(e,LOW);  
digitalWrite(d,LOW);  
}  
  
void digital_5(void) //Display number 5
```

```
{  
unsigned char j;  
digitalWrite(a,HIGH);  
digitalWrite(b, LOW);  
digitalWrite(c,HIGH);  
digitalWrite(d,HIGH);  
digitalWrite(e, LOW);  
digitalWrite(f,HIGH);  
digitalWrite(g,HIGH);  
digitalWrite(dp,LOW);  
}  
  
void digital_6(void) //Display number 6  
{  
unsigned char j;  
for(j=7;j<=11;j++)  
digitalWrite(j,HIGH);  
digitalWrite(c,HIGH);  
digitalWrite(dp,LOW);  
digitalWrite(b,LOW);  
}  
  
void digital_7(void) //Display number7  
{
```

```
unsigned char j;  
  
for(j=5;j<=7;j++)  
  
digitalWrite(j,HIGH);  
  
digitalWrite(dp,LOW);  
  
for(j=8;j<=11;j++)  
  
digitalWrite(j,LOW);  
  
}  
  
void digital_8(void) //Display number 8  
  
{  
  
unsigned char j;  
  
for(j=5;j<=11;j++)  
  
digitalWrite(j,HIGH);  
  
digitalWrite(dp,LOW);  
  
}  
  
void digital_9(void) //Display number 5  
  
{  
  
unsigned char j;  
  
digitalWrite(a,HIGH);  
  
digitalWrite(b,HIGH);  
  
digitalWrite(c,HIGH);  
  
digitalWrite(d,HIGH);  
  
digitalWrite(e, LOW);
```

```
digitalWrite(f,HIGH);

digitalWrite(g,HIGH);

digitalWrite(dp,LOW);

}

void setup()

{

int i;//defined variable

for(i=4;i<=11;i++)

pinMode(i,OUTPUT);//Set the 11 to 4 pin for the output mode

}

void loop()

{

while(1)

{

digital_0();//Display number 1

delay(1000);//delay 1s

digital_1();//Display number 1

delay(1000);//delay 1s

digital_2();//Display number2

delay(1000); //delay 1s

digital_3();//Display number3

delay(1000); //delay 1s
```

```
digital_4();//Display number 4  
delay(1000); //delay 1s  
  
digital_5();//Display number 5  
delay(1000); //delay 1s  
  
digital_6();//Display number 6  
delay(1000); //delay 1s  
  
digital_7();//Display number7  
delay(1000); //delay 1s  
  
digital_8();//Display number 8  
delay(1000); //delay 1s  
  
digital_9();//Display number 9  
delay(1000); //delay 1s  
  
}  
}
```

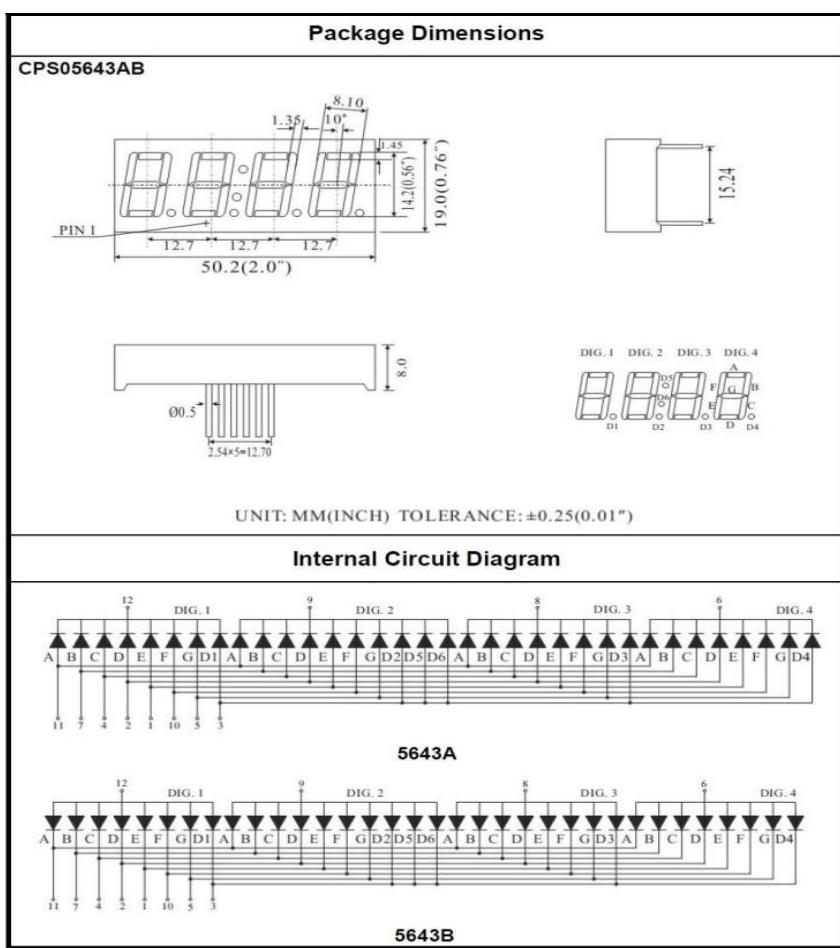
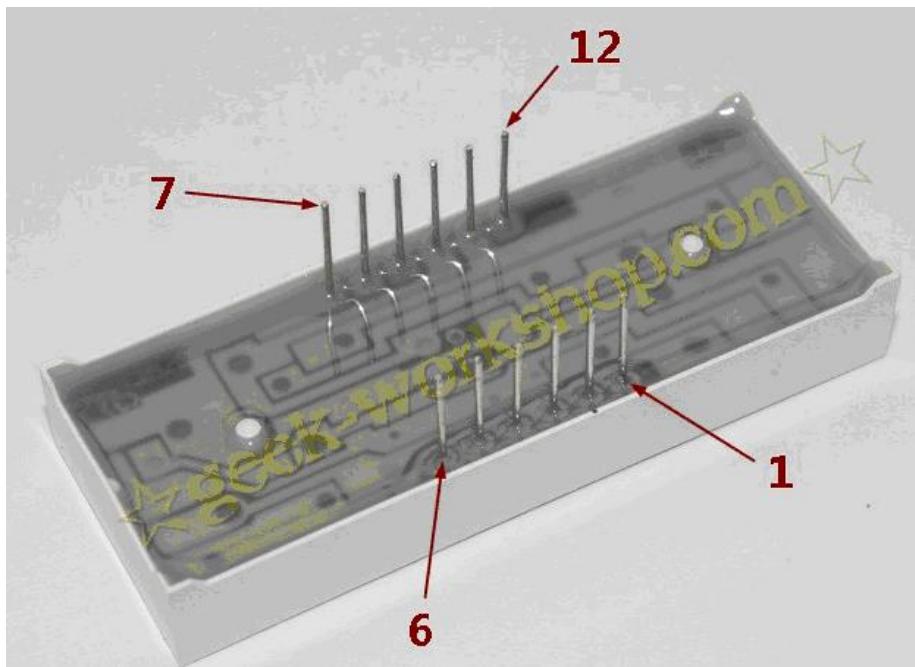
---

## Leçon 12. Tube digital 4bits

Il s'agit d'utiliser Arduino pour une utilisation de 4 tubes numériques. La limite de la résistance du courant est essentielle pour utiliser un tube digital, la résistance de la limite de courant a 2 types de connexion, un total de 4 D1-D4. Cette connexion est moins demandée, mais va produire une luminosité différente, 8 sera le plus foncé. Cette méthode est à 8 broches, la luminosité est la même mais avec plus de résistance, 2201008 ohm, donc utiliser 220 au lieu de 10.0 ohm sera relativement élevé.

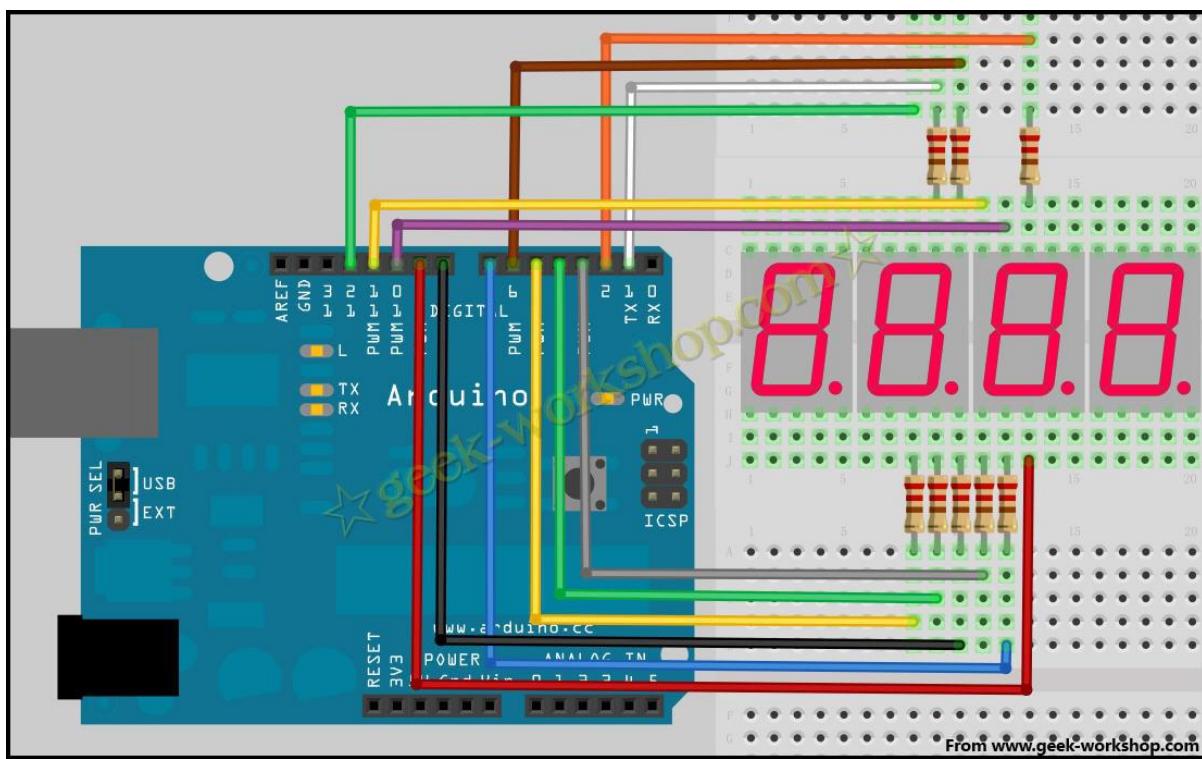


Le tube digital 4 a un total de 12 broches. Le point décimal est placé en face du bas, en bas à gauche du 1, les autres broches sont comme dans le sens des aiguilles d'une montre. Le coin gauche supérieur pour le 12



Four Digits Displays Series

Voici un diagramme de connexion hardware



Voici le program;

```
//display 1234
//Set cathode interface
int a = 1;
int b = 2;
int c = 3;
int d = 4;
int e = 5;
int f = 6;
int g = 7;
int dp = 8;
//Set anode interface
int d4 = 9;
int d3 = 10;
int d2 = 11;
int d1 = 12;
//Set variable
long n = 1230;
int x = 100;
int del = 55; //Here to fine tune the clock
void setup()
{
    pinMode(d1, OUTPUT);
}
```

```

pinMode(d2, OUTPUT);
pinMode(d3, OUTPUT);
pinMode(d4, OUTPUT);
pinMode(a, OUTPUT);
pinMode(b, OUTPUT);
pinMode(c, OUTPUT);
pinMode(d, OUTPUT);
pinMode(e, OUTPUT);
pinMode(f, OUTPUT);
pinMode(g, OUTPUT);
pinMode(dp, OUTPUT);
}

///////////////////////////////
void loop()
{
    Display(1, 1);
    Display(2, 2);
    Display(3, 3);
    Display(4, 4);

}
/////////////////////////////
void WeiXuan(unsigned char n)//
{
    switch(n)
    {
        case 1:
            digitalWrite(d1,LOW);
            digitalWrite(d2,HIGH);
            digitalWrite(d3,HIGH);
            digitalWrite(d4,HIGH);
            break;
        case 2:
            digitalWrite(d1,HIGH);
            digitalWrite(d2,LOW);
            digitalWrite(d3,HIGH);
            digitalWrite(d4,HIGH);
            break;
        case 3:
            digitalWrite(d1,HIGH);
            digitalWrite(d2,HIGH);
            digitalWrite(d3,LOW);
            digitalWrite(d4,HIGH);
            break;
    }
}

```

```

case 4:
    digitalWrite(d1, HIGH);
    digitalWrite(d2, HIGH);
    digitalWrite(d3, HIGH);
    digitalWrite(d4, LOW);
    break;
    default :
        digitalWrite(d1, HIGH);
        digitalWrite(d2, HIGH);
        digitalWrite(d3, HIGH);
        digitalWrite(d4, HIGH);
        break;
    }
}

void Num_0()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
    digitalWrite(g, LOW);
    digitalWrite(dp,LOW);
}

void Num_1()
{
    digitalWrite(a, LOW);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
    digitalWrite(dp,LOW);
}

void Num_2()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, LOW);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, LOW);
}

```

```
digitalWrite(g, HIGH);
digitalWrite(dp,LOW);
}
void Num_3()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);
}
void Num_4()
{
    digitalWrite(a, LOW);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);
}
void Num_5()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, LOW);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);
}
void Num_6()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, LOW);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
```

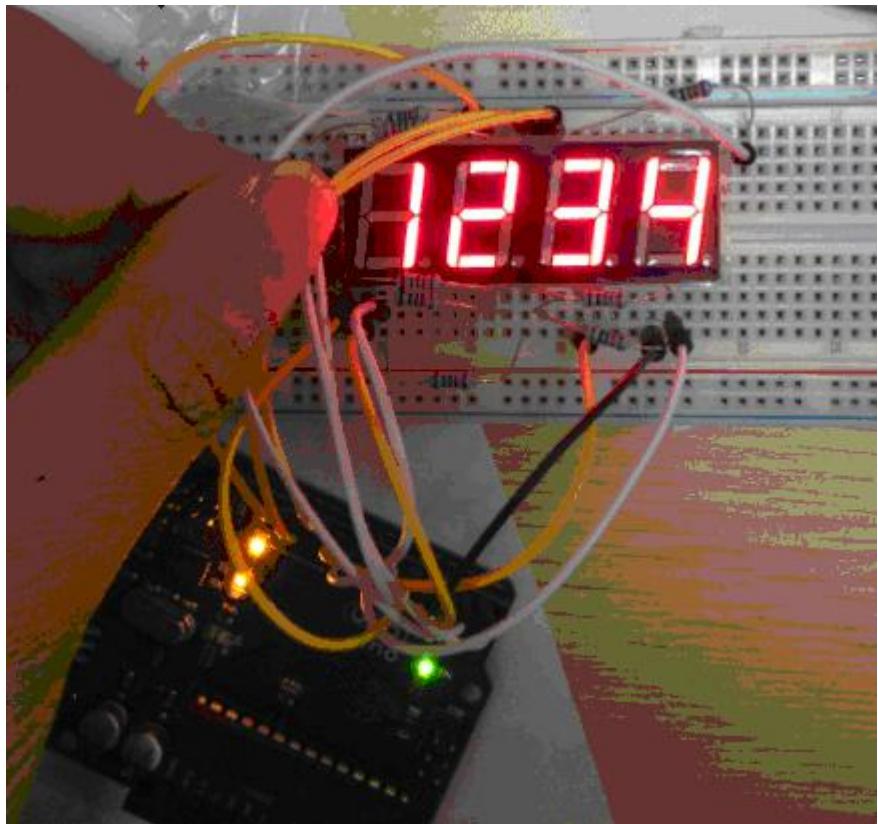
```
digitalWrite(g, HIGH);
digitalWrite(dp,LOW);
}
void Num_7()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
    digitalWrite(dp,LOW);
}
void Num_8()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);
}
void Num_9()
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);
}
void Clear() // Clear the screen
{
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
```

```

digitalWrite(g, LOW);
digitalWrite(dp,LOW);
}
void pickNumber(unsigned char n)//Choose the number of
{
switch(n)
{
case 0:Num_0();
break;
case 1:Num_1();
break;
case 2:Num_2();
break;
case 3:Num_3();
break;
case 4:Num_4();
break;
case 5:Num_5();
break;
case 6:Num_6();
break;
case 7:Num_7();
break;
case 8:Num_8();
break;
case 9:Num_9();
break;
default:Clear();
break;
}
}
void Display(unsigned char x, unsigned char Number)//Show that x is the coordinate, Number is the number
{
WeiXuan(x);
pickNumber(Number);
delay(1);
Clear() ; //Vanishing
}

```

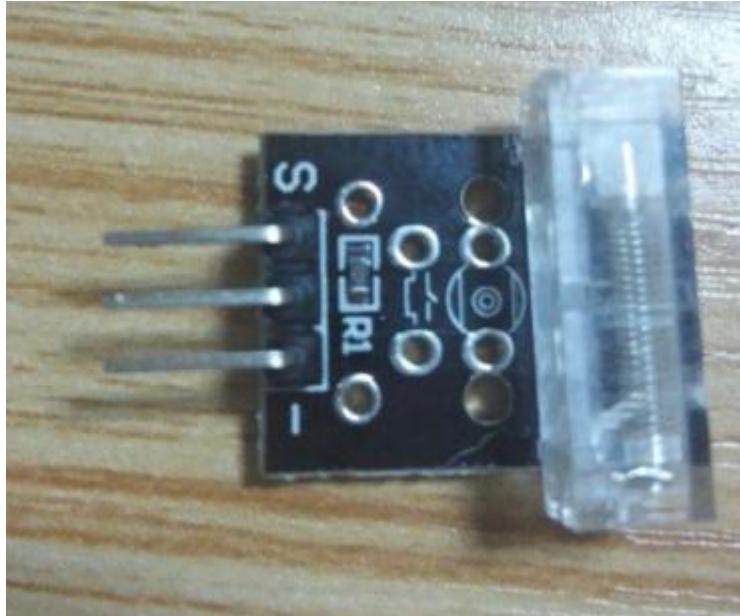
Copier le code suivant pour le télécharger sur le panneau de contrôle pour voir l'effet.



digital tube display 1234,

Note : la connexion demande beaucoup de patience et de précision, aucune erreur n'est possible

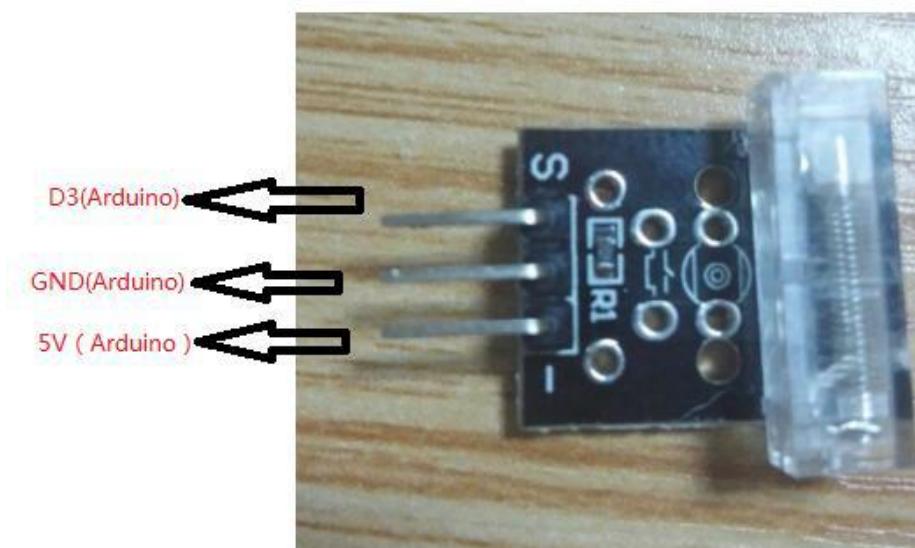
## Leçon 13. Module hit



Le module de température Hit peu connecter un UNO R3 PIN13

La LED peut s'allumer quand elle est connectée au hit module sur la broche 3

### Wiring diagram:



## Composants

- 1 \*carte uno
- 1 \*cable de données
- 1 \* module
- 1 \*câbles de démarrage

Code:

```
int Led=13;//Define LED interface
int Shock=3//Define vibration sensor interface
int val;//Define numeric variable val
void setup()
{
pinMode(Led,OUTPUT);//Define LED as output interface
pinMode(Shock,INPUT);//Define vibration sensor as output interface
}
void loop()
{
val=digitalRead(Shock);//Read the value of the value of the digital interface 3 to val
if(val==HIGH)//When the vibration sensor detects the signal, the LED flashes
{
digitalWrite(Led,LOW);
}
else
{
digitalWrite(Led,HIGH);
}
}
```

## Leçon 14. Interrupteur d'inclinaison

### introduction

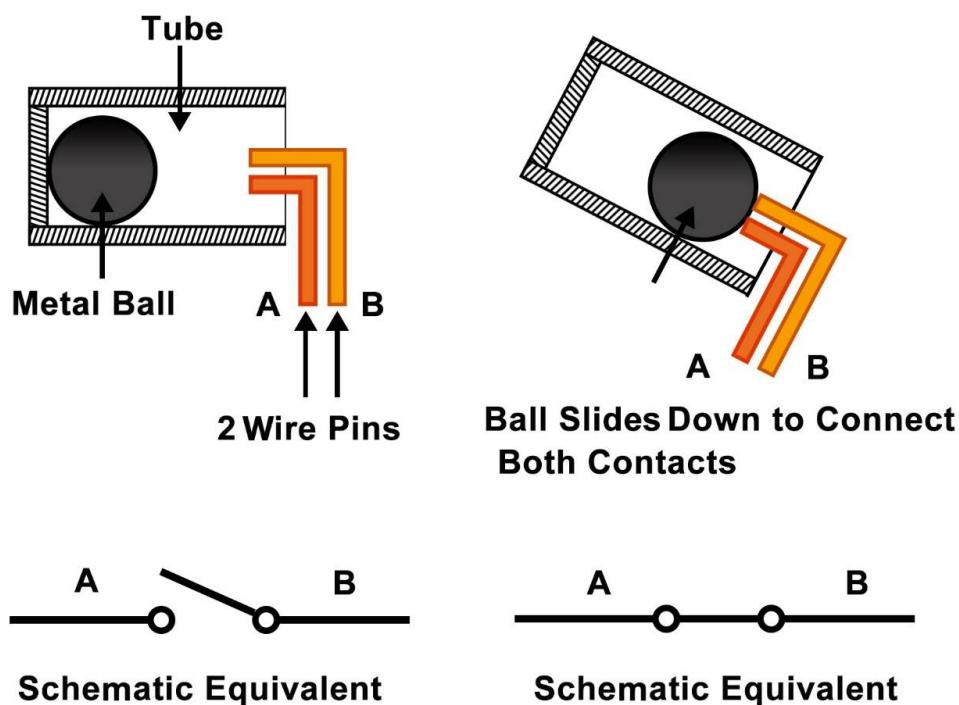
L'interrupteur utilisé ici est une boule avec du métal à l'intérieur. Il est utilisé pour détecter les petits angles inclinés.

### Composants

- 1 \* Kuman Uno board
- 1 \* USB cable de données
- 1 \* interrupteur tilt
- câbles de démarrage

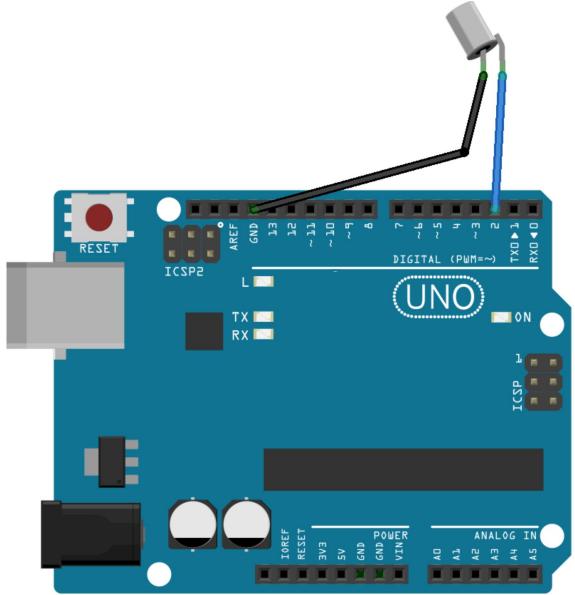
### Principe du test

Le principe est très simple. Quand l'interrupteur s'incline dans un certain angle, la boule à l'intérieur touche les 2 contacts connectés aux broches externes, déclenchant ainsi les circuits. Sinon, la boule restera à l'écart des contacts, cassant ainsi les circuits.



## Procédure du test.

Etape 1 : construire le circuit



U1			
1	IREF	AREF	27
2	RESET	D13/SCK	26
3	3.3V	D12/MISO	25
4	5V	D11/MOSI	24
5	GND	D10PWM/SS	23
6	VIN	D9PWM	22
		D8	21
7	A0	D7	20
8	A1	D6PWM	19
9	A2	D5PWM	18
10	A3	D4	17
11	A4/SDA	D3PWM	16
12	A5/SCL	D2	15
		D1/TXD	14
		D0/RXD	13

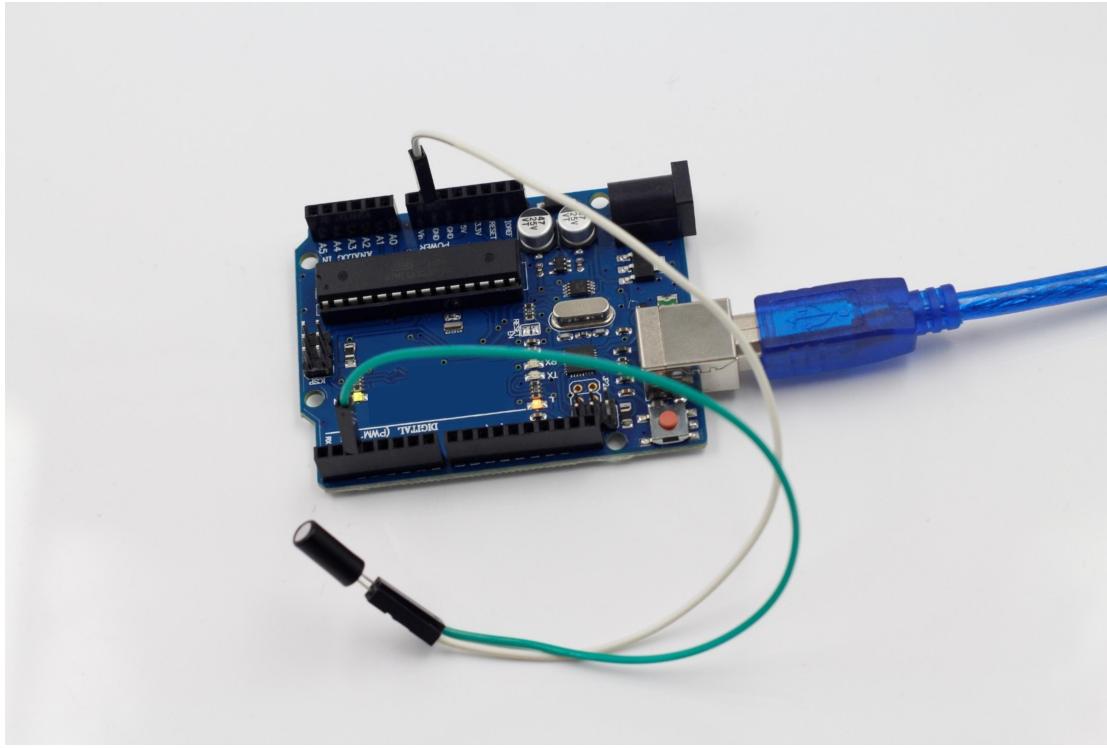
KUMAN UNO R3

tilt\_switch

Etape 2 : Compiler le code.

Etape 3. Télécharger le sketch sur la carte Uno kuman

Maintenant, incliner l'interrupteur et la LED attaché à la broche 13 va s'allumer.



## Code

```
//tilt switch  
//tilt the switch, and the LED attached to pin 13 on SunFounder Uno board will light up.  
//tilt switch attach to pin2  
//Email: support@sunfounder.com  
//Website: www.sunfounder.com  
*****  
const int ledPin = 13;//the led attach to  
void setup()  
{  
pinMode(ledPin,OUTPUT);//initialize the ledPin as an output  
pinMode(2,INPUT);//set pin2 as INPUT  
digitalWrite(2, HIGH);//set pin2 as HIGH  
}  
*****  
void loop()  
{  
int digitalVal = digitalRead(2);//Read the value of pin2
```

```
if(HIGH == digitalVal)//if tilt switch is not breakover
{
  digitalWrite(ledPin,LOW);//turn the led off
}
else //if tilt switch breakover
{
  digitalWrite(ledPin,HIGH);//turn the led on
}
}
*****
/*****
```

## Leçon 15. Module de mesure de distance ultrasonique.

Ce module a une performance stable, une mesure précise, et peut être comparé au SRF05 et SRF02 ou autres modules ultrasoniques. Module de haute précision, stable, angle mort de 2cm, c'est un produit très populaire sur le marché !

Composants nécessaires

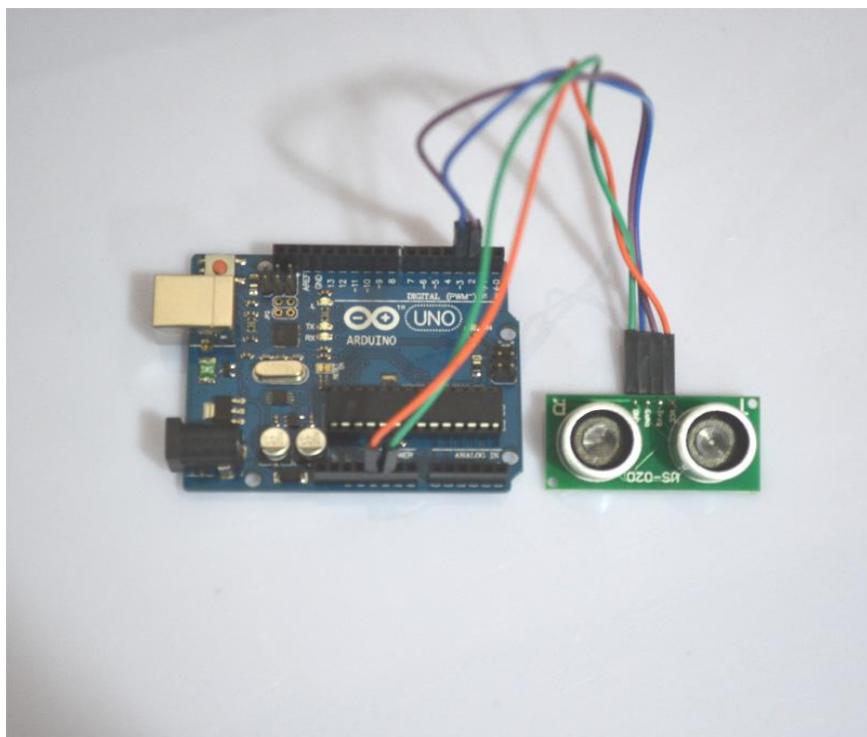
\*1 carte de controle arduino

USB line \*1

Ultrasonic \*1

Principaux paramètres techniques : courant DC5V, courant statique, moins de 2mA, niveau output : haut 5V, bas 0V, angle d'induction : pas plus de 15 degrés, portée de détection : de 2cm à 450cm, haute précision jusqu'à 0.2cm

Mode de connexion, VCC, trig (contrôle), echo (récepteur), GND



Note : le module ne doit pas être chargé, si vous voulez garder la connexion, laisser le premier module de connexion GND ou cela va affecter l'opération normale du module.

Program source code:

```

const int TrigPin = 2;
const int EchoPin = 3;
float cm;
void setup()
{
Serial.begin(9600);
pinMode(TrigPin, OUTPUT);
pinMode(EchoPin, INPUT);
}
void loop()
{
digitalWrite(TrigPin, LOW); //Low high and low level to send a short pulse to TrigPin
delayMicroseconds(2);
digitalWrite(TrigPin, HIGH);
delayMicroseconds(10);
digitalWrite(TrigPin, LOW);

cm = pulseIn(EchoPin, HIGH) / 58.0; //The echo time is converted into cm
cm = (int(cm * 100.0)) / 100.0; //Keep two decimal places
Serial.print(cm);
Serial.print("cm");
Serial.println();
delay(1000);
}

```

## Introduction

Module portée ultrasonique HC-SR04 permet une mesure de 2cm a 700cm sans contact, la précision peut aller jusqu'a 3mm. Signal stable sous 5m, le signal s'amenuise apres 5m jusqu'à disparaître apres 7m

Le module inclut des transmetteurs ultrasons, un récepteur et un circuit de contrôle. Le principe de base du fonctionnement est :

1: Utiliser la gachette IO pour le signal de haut niveau de 10us

2 : Le module envoie automatiquement huit 40kHz et détecté si il y a un signal reçu

3 : le signal reçu, de haut niveau, la durée de output IO est le temps d'envoyer les ultrasons et de les recevoir.

Test de distance : (temps haut niveaux) / vitesse du son (340m/s)/2

### Fonction

**TRIG:** Rythme trigger input

**ECHO:** rythme echo output

**GND:** terre

**VCC:** courant 5V

### Caractéristiques

Tension de fonctionnement : DC5V

Courant : 16mA

Fréquence 40Hz

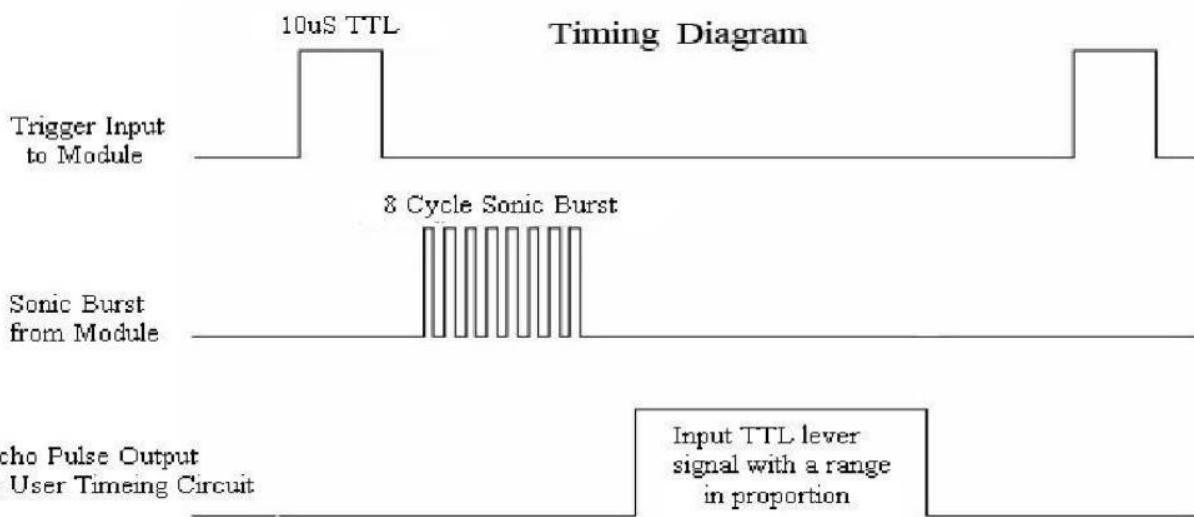
Portée maximale : 700cm. signal stable jusque 5m, perte progressive du signal après 5 jusque disparition après 7m  
Portée minimale : 2cm

Rythme trigger input 10uS TTL

Dimension 46x20.5x15 mm

### Principe

Le diagramme de temps est montré ci-après. Vous devez simplement fournir une impulsion courte de 10uS sur le trigger pour commencer la mesure, le module ensuite va envoyer un cycle d'ultrasons à 40kHz et obtenir un écho. Vous pouvez calculer la portée par rapport au temps entre envoyer le signal et le recevoir. La formule est  $US / 58 = \text{centimètres}$  ou  $US / 148 = \text{inch}$ ; où: portée = high level time \* velocity (340M/S) / 2; nous suggérons d'utiliser un cycle supérieur à 60ms.



### Test Code:

```

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#define Echo 11 //Echo connnect to pin11
#define Trig 12 //Trig connect to pin12
unsigned long rxTime; //define a variable
float distance;
LiquidCrystal_I2C lcd(0x27,16,2);
void setup()
{
    // put your setup code here, to run once:
    Serial.begin(115200); //set the baud rate of serial monitor
    pinMode(Echo,INPUT);
    pinMode(Trig,OUTPUT);
    lcd.init();           // initialize the lcd

    // Print a message to the LCD.
    lcd.backlight(); //turn o the backlight
    lcd.setCursor(0, 0); //set the cursor on 0 row,0 col
    lcd.print("Ping:"); //print the "Ping: "on the LCD
}

void loop()
{
    // Generates a pulse
    digitalWrite(Trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(Trig, LOW);
    rxTime = pulseIn(Echo, HIGH); //read the Receive time
    // Serial.print("rxTime:");
    // Serial.println(rxTime);
    distance = (float)rxTime * 34 / 2000.0; //Converted into a distance ,cm
    if(distance < 800 ) //filter interference signal
    {
        Serial.print("distance:");
        Serial.print(distance); //print it the distance in serial monitor
        Serial.println("CM");
        lcd.setCursor(6, 0);
        lcd.print(distance);//print it in LCD1602
        lcd.print("CM");
        delay(100);
    }
}

```

## **Leçon 16. Résistance thermique**

### **1: Introduction de la thermistance.**

La résistance de la thermistance à la température est comme une résistance, aussi appelée thermistance semi-conductrice. Cela peut être par simple cristal, polycristallin et verre, fait de matériaux semi-conducteurs comme du plastique. Cette résistance a une série de propriétés électriques spéciales, les caractéristiques de base est que le changement de résistance à la température a un changement très significatif, et la courbe d'ampère volt est non linéaire.

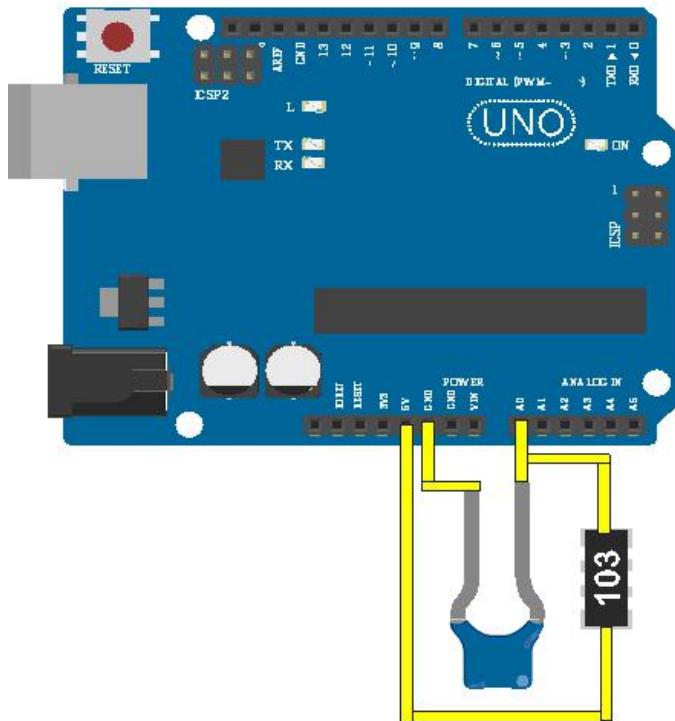
### **2 : Principales caractéristiques**

Sensibilité à la haute température, inertie thermique, longue vie, petit volume, structure simple, de forme et de structure diverses. Avec le développement de la production industrielle et agriculture, la science et la technologie, cet élément a été largement utilisé, comme pour mesurer la température, la contrôler, mesurer le niveau de liquide, la pression, alarme incendie, sondage météorologique, circuit switch, protection de surcharge, fluctuation de tension, retardateur, stabilité d'amplitude, ajustement automatique de hausse, mesure des micro ondes et laser, etc

### **3: Paramètres**

Les paramètres caractéristiques principaux de la thermistance sont la résistance à la température, la tension, et la constante de temps thermique

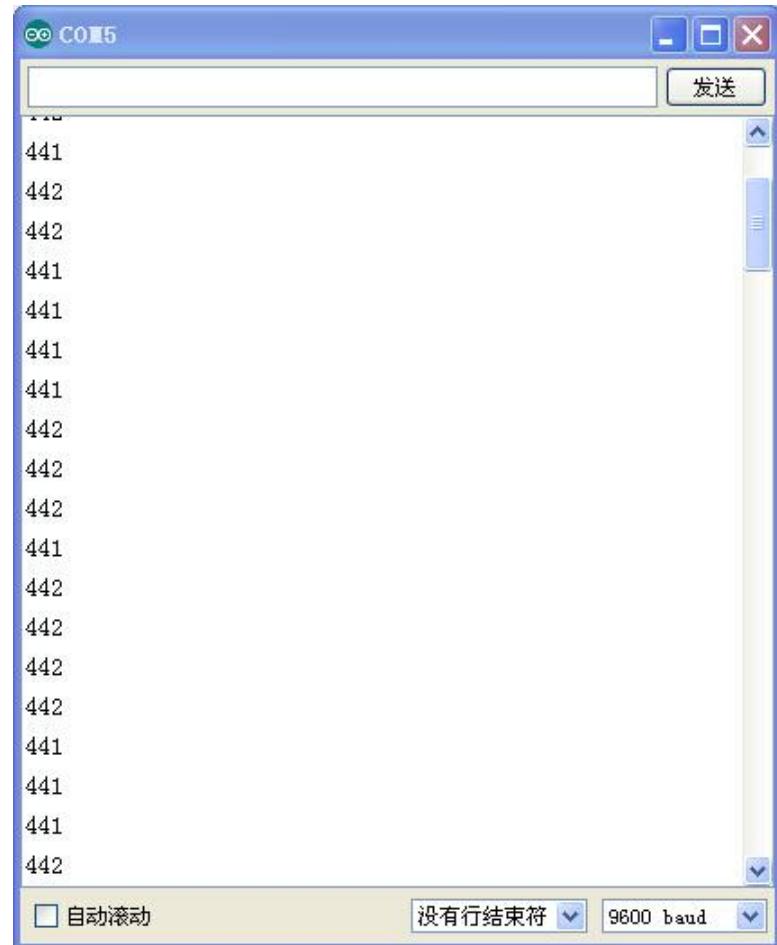
### **4: Méthode de connexion**



## 5: Test code

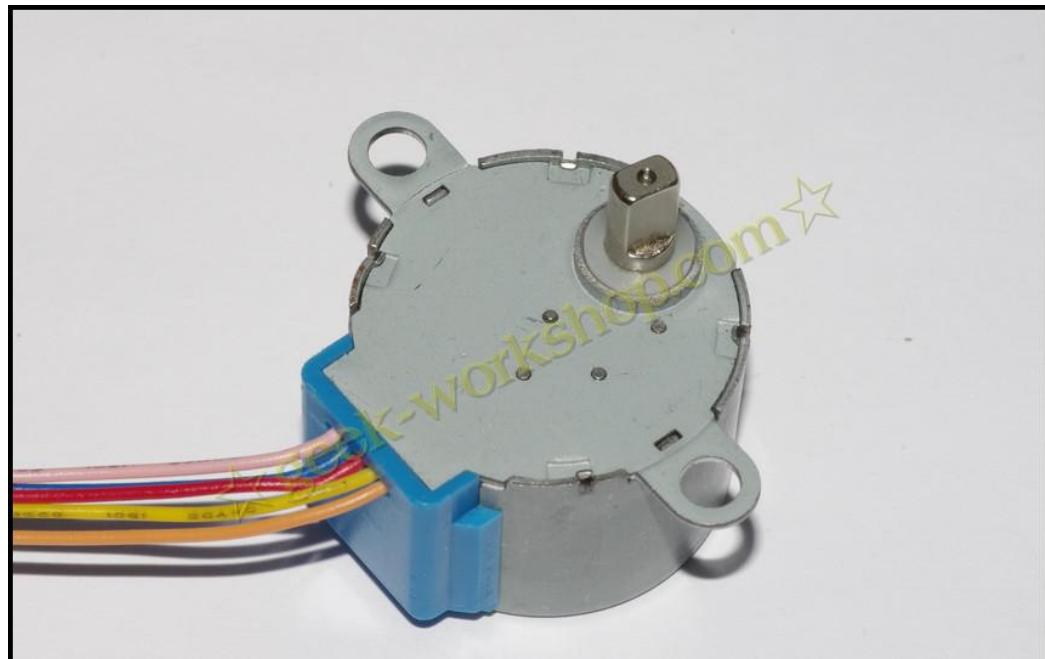
```
void setup() {  
    // initialize serial communication at 9600 bits per second:  
    Serial.begin(9600);  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
    // read the input on analog pin 0:  
    int sensorValue = analogRead(A0);  
    // print out the value you read:  
    Serial.println(sensorValue);  
    delay(1);           // delay in between reads for stability  
}
```

The above code onto your board, open the serial port window, you can see the following:



## Leçon 17. Test du moteur pas à pas.

Le moteur pas à pas est un genre d'impulsion électrique dans un déplacement angulaire du corps exécutif. Point de vue populaire : quand le stepper reçoit un signal, il marche en fonction d'un angle fixe de rotation. Vous pouvez contrôler le déplacement angulaire en contrôlant le nombre d'impulsions et vous pouvez contrôler la vitesse de rotation et l'accélération en contrôlant la fréquence d'impulsion pour atteindre votre but. Ce qui suit est une utilisation expérimentale du moteur pas à pas.



picture:

Avant d'utiliser le moteur pas à pas, vérifiez attentivement les spécificités, pour confirmer que les 4 phases ou 2 phases, comment connecter les différentes lignes, cet essai utilise le moteur pas à pas en 4 phases, les différentes couleurs des lignes sont définies comme suit :

Driving mode : (4-1-2Phase drive)

Wire color	1	2	3	4	5	6	7	8
5	+	+	+	+	+	+	+	+
4	-	-						-
3		-	-	-				
2				-	-	-		
1						-	-	-

→ CCW (depending on the direction of rotation axis of SEC)

Slow smoteur pas à pas

Diameter: 28mm

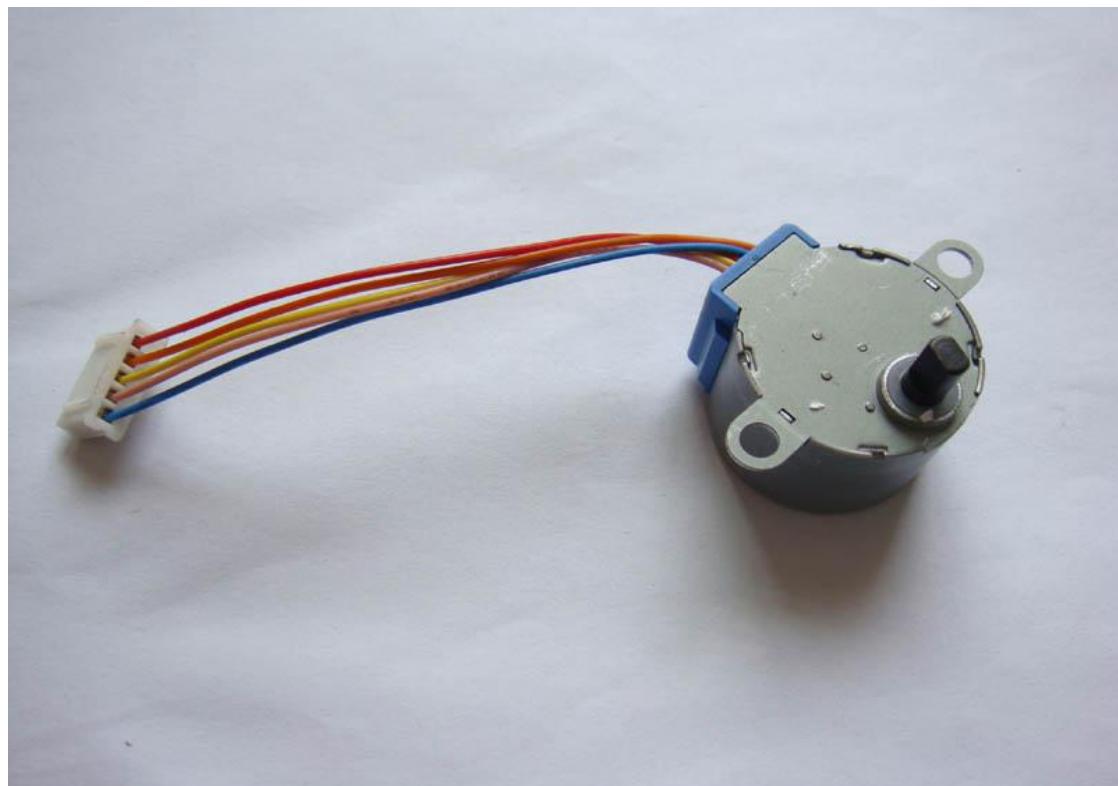
Voltage: 5V

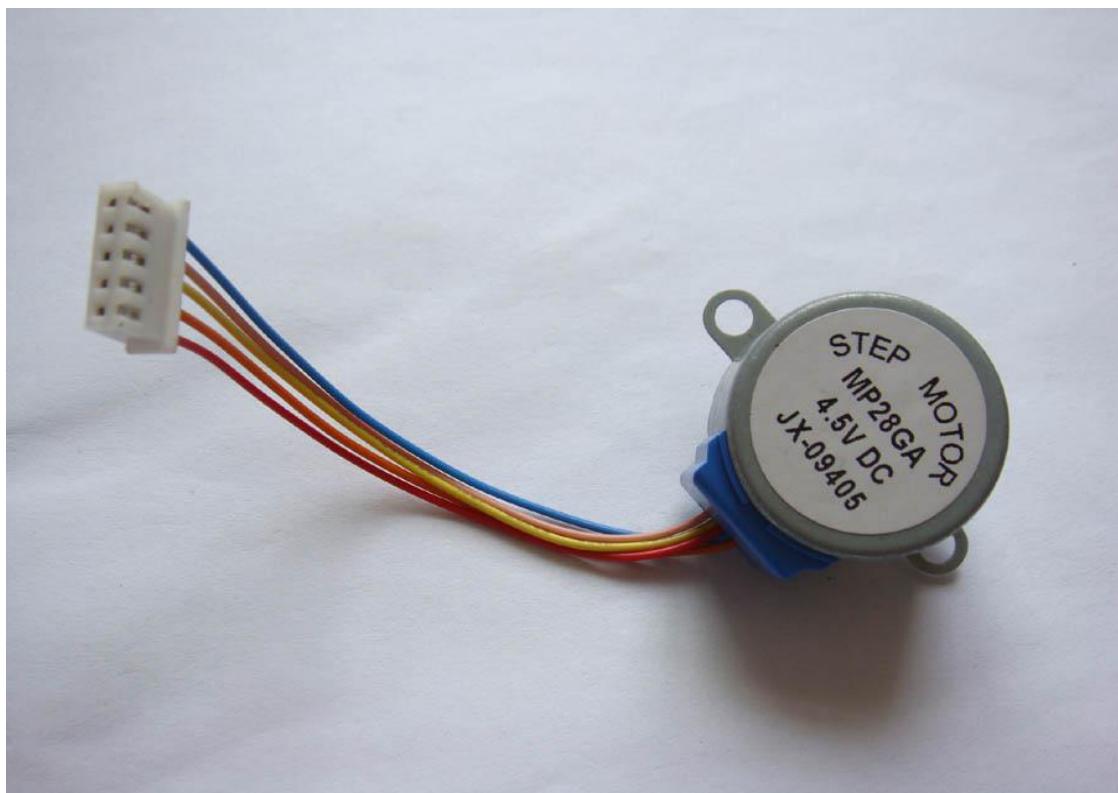
Step angle: 5.625 1/64 x

Reduction ratio: 1/64

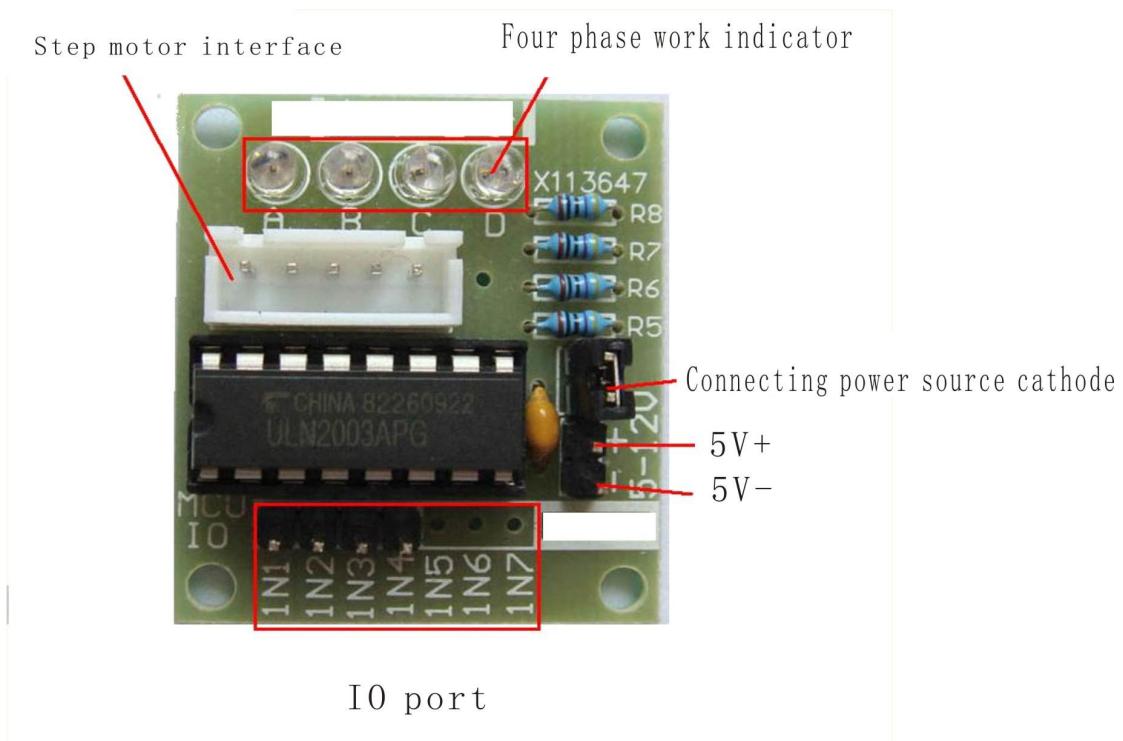
5 line 4 phase can be used with ordinary ULN2003 chip driver, can also be connected to the use of 2 phase

Le moteur utilise une puissance sous 50mA avec 64 fois la vitesse du réducteur, c'est très pratique pour le développement des cartes.



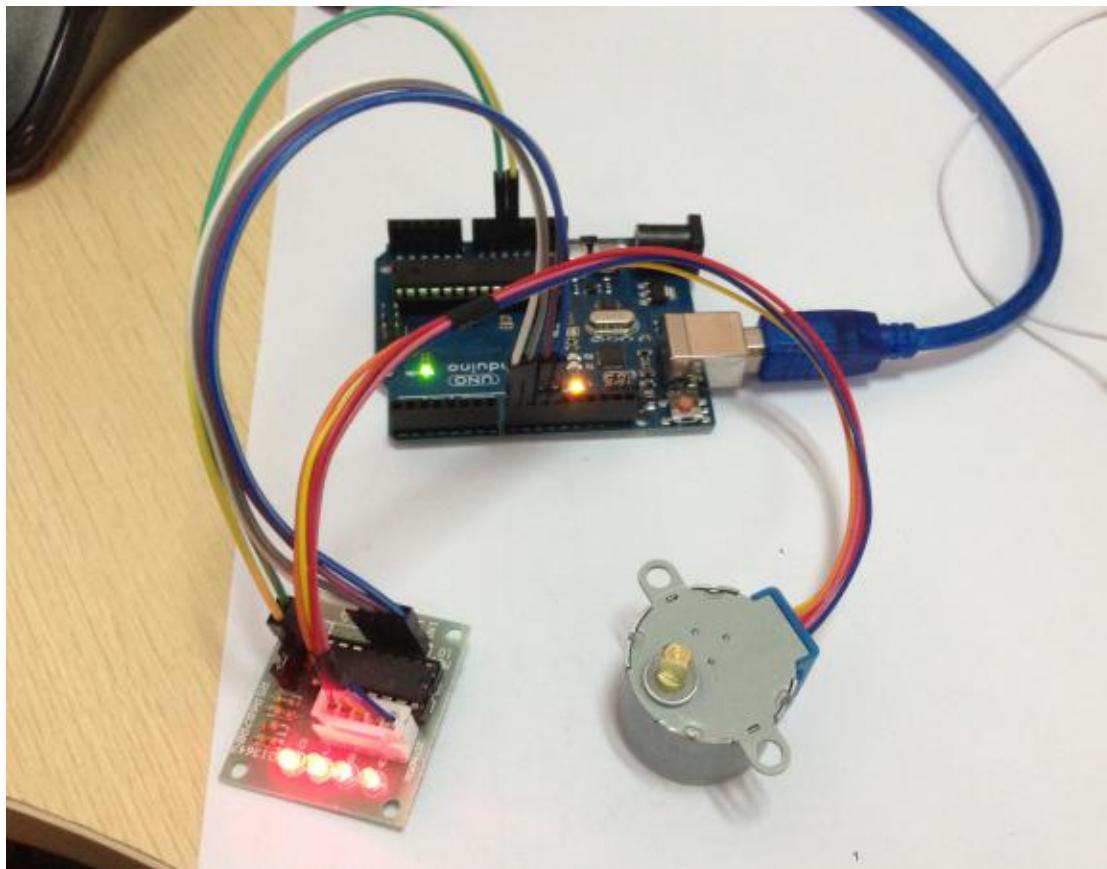


### Moteur pas à pas (5 lignes, 4 phases) carte (UL2003)



external SIZE: 31×35mm

Le diagramme de connexion hardware est comme suit :



Télécharger le code sur le panneau de contrôle Arduino pour voir le résultat.

```
/*
 * Stepper motor with potentiometer rotation
 * (or other sensors) using 0 analog inputs
 * Use IDE Stepper.h comes with the Arduino library file
#include <Stepper.h>
// Here to set the stepper motor rotation is how many steps
#define STEPS 100
// attached toSet the step number and pin of the stepper motor
Stepper stepper(STEPS, 8, 9, 10, 11);
```

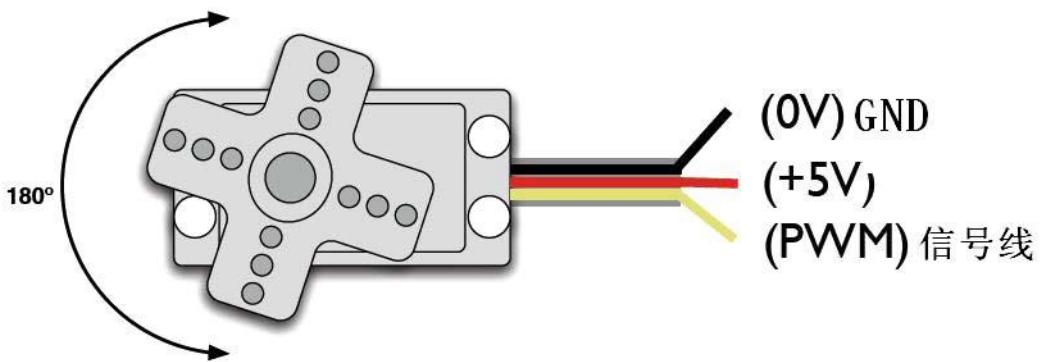
```
// Define variables used to store historical readings
int previous = 0;
void setup()
{
// Set the motor at a speed of 90 steps per minute
stepper.setSpeed(90);
}
void loop()
{
int val = analogRead(0); // Get sensor readings
stepper.step(val - previous);// Move the number of steps for the current readings less historical readings
previous = val;// Save historical readings }
```

## Leçon 18. Contrôle de la direction

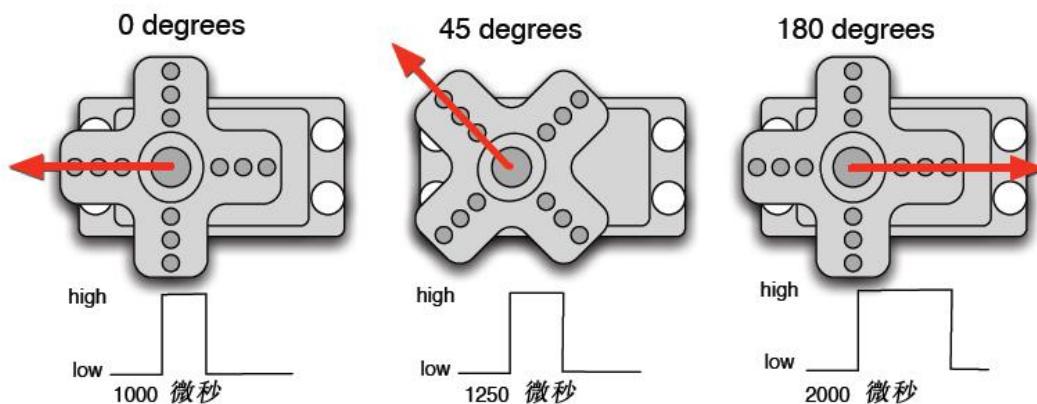
Le mécanisme de direction est un genre de servo drive, composé principalement d'une boite, un circuit, un moteur non core, un levier, un détecteur de position. Le principe de fonctionnement est par le récepteur ou le SCM envoie des signaux au servomoteur et l'intérieur a un circuit benchmark, générant des cycles de 20ms, largeur 1.5ms comme signal de référence, ils obtiennent une tension DC et la tension du potentiomètre est comparée avec la différence de tension de l'output. Au travers du circuit IC, il juge de la direction de rotation, le moteur commence à tourner, avec le levier de réduction, le courant est transmis par un balancier, aussi par un détecteur de position, qui juge si l'endroit est atteint. C'est convenable pour le contrôle du système qui a besoin d'être changé et qui doit être gardé à un certain degré. Quand la vitesse du moteur est sûre, le potentiomètre est mis en rotation par un flot provenant du levier de réduction, la différence de tension est 0, le moteur arrête de tourner, et la portée de l'angle est de 0 à 180 degrés.



Le servo a de nombreuses spécificités, chaque levier de direction a 3 lignes externes, marron, rouge, orange, pour les distinguer, si la marque du levier est différente, la couleur sera différente, marron pour le fil de terre, rouge pour le positif, et orange pour le signal.



L'angle de direction est ajusté PWM (modulation à impulsion), le cycle PWM standard est fixé à 20ms (50hz), la largeur de distribution d'impulsion devrait être entre 1ms et 2ms, mais en fait, la largeur peut être de 0.5ms à 2.5ms, et l'angle de 0 à 180degres. Il y a un point qui vaut la peine d'être noté, si la marque du levier est différente, l'angle peut être différent.



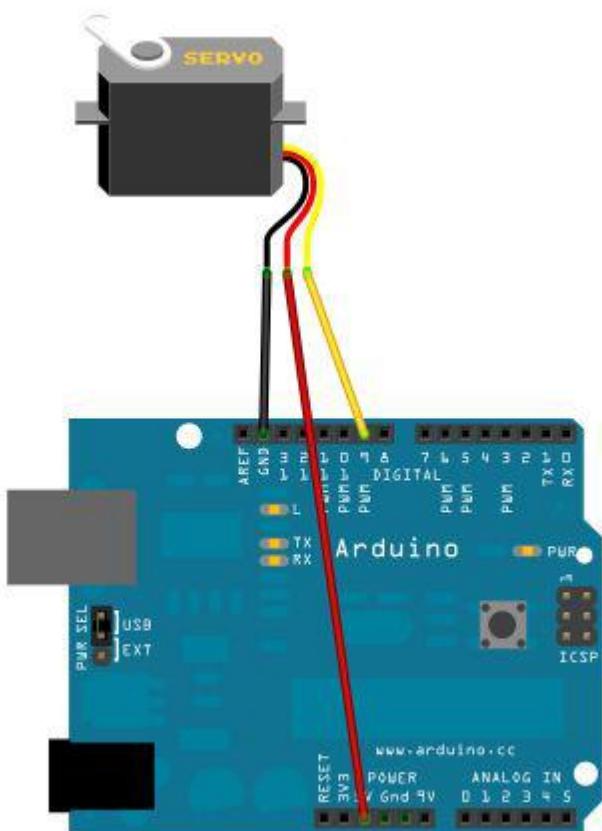
En sachant les principes de base, nous pouvons apprendre à contrôler un moteur servo, les composants sont simplement un levier de direction A et une bar jumper.

RB - 412 steering gear \*1

Bread board jumper \*1 tie

- Le levier de direction avec Arduino a 2 genres de méthodes, une est au travers de l'interface pour produire un cycle de Fang bo, un signal analogue PWM, la seconde méthod est l'utilisation directe de la fonction servo Arduino, l'avantage de cette méthode de contrôle est la direction en 2 sens, car Arduino a une fonction unique, interface 9,10, sa capacité est limitée.

One method



Le levier de direction est connecté à l'interface digitale 9

Copier un programme, pour que le nombre input du levier de direction corresponde au nombre de degrés de la position et l'angle d'affichage à l'écran.

Reference avec source program A:、

**Int servopin=9; // define digital interface 9 connected servo signal line**

**Int myangle; // define the angle variable**

**Int pulsewidth; // define variable pulse widthint val;**

```

void servopulse(int servopin,int myangle)//Define a pulse function

{
pulsewidth=(myangle*11)+500;//The pulse width of 500-2480 is converted into the
pulse width.

digitalWrite(servopin,HIGH);//The rudder interface level is high
delayMicroseconds(pulsewidth);//Time delay pulse width
digitalWrite(servopin,LOW);//Servo interface level to low
delay(20-pulsewidth/1000);

}

void setup()

{
pinMode(servopin,OUTPUT);//Set steering interface for output interface
Serial.begin(9600);//Connected to the serial port, the baud rate is 9600
Serial.println("servo=o_serai_simple ready" );
}

void loop()//The number of 0 to 0 is converted to a 9 - to 180 angle, and the number
of the corresponding number of LED flashes

{
val=Serial.read();//Read the value of the serial port
if(val>'0'&&val<='9')

{
Val=val-'0'; / / the features into numerical variables

```

```

Val=val* (180/9); // digital into perspective

Serial.print("moving servo to ");

Serial.print(val,DEC);

Serial.println();

for(int i=0;i<=50;i++) //Give the steering gear enough time to let it go to the specified
angle.

{

servopulse(servopin,val);//Reference pulse function

}

}

}

```

## Méthode 2

La première analyse détaillée d'Arduino vient avec la fonction Servo et ses déclarations, pour présenter la fonction gouvernail

1. Régler l'interface de direction, seuls les nombres 9 et 10 peuvent être utilisés.
2. copier (angle) pour régler l'angle de direction entre 0 et 180 degrés.
3. Lire : une commande pour l'angle de direction peut être comprise comme une valeur de la dernière commande copiée.
4. Déterminer si les paramètres ont été envoyés à l'interface..
5. détacher. Le levier de direction est séparé de son interface, et l'interface (nombre 9 et 10) peut continuer à être utilisée comme une interface PWM.

Note : la déclaration ci-dessus est écrite par l'opérateur au nom variable. La déclaration spécifique (), par exemple myservo.attach (9).

<Servo.h>// #include header files, there is a point to pay attention, you can directly in the Arduino software menu bar click Sketch>Importlibrary>Servo, call the Servo function, you can directly enter the <Servo.h> #include, but in the input should pay attention to between #include and <Servo.h> to have a space, or compile time will be reported to the wrong.

```
Servo myservo; // define servo variable name
```

```
void setup()
```

```
{
```

```
myservo.attach(9);//Definition of steering gear interface (9,10 can, shortcomings can only control 2)
```

```
}
```

```
void loop()
```

```
{
```

```
myservo.write(90);//Set the steering angle of the steering gear.
```

```
}
```

Ci-dessus est pour contrôler la direction avec 2 méthodes, chacun a ses propres avantages et inconvénients selon les besoins de l'utilisateur.

## Leçon 19. Test lampe photosensible

Avec l'achèvement des tests ci-dessus, nous devrions avoir plus de compréhension sur l'application Arduino, avec input et output digital basique, input analogue, et génération PWM, nous pouvons maintenant commencer l'application des capteurs. La résistance photosensible appelée perception de lumière, est faite d'un semiconducteur photoélectrique avec un effet de résistance avec l'intensité de la lumière.

Les résistances photosensibles sont généralement utilisées pour les mesures optiques, le contrôle de lumière photoélectrique (la variation de la lumière est convertie en variation électrique)

La résistance photosensible peut être utilisée dans des circuits optiques variés, comme le contrôle de la lumière, la régulation et peut être utilisé aussi dans les interrupteurs de lampes. Dans ce test, nous regardons d'abord l'utilisation d'une résistance photosensible simple. La résistance photosensible, puisqu'elle varie en fonction de l'intensité de la lumière, il faut simuler le port pour lire la valeur analogue. Il peut se référer à l'interface PWM, le potentiomètre de la résistance quand l'intensité de la lumière et la clarté des LED auront un changement qui correspond.

\*1 résistance photosensible

M5 LED\*1 ligne rouge

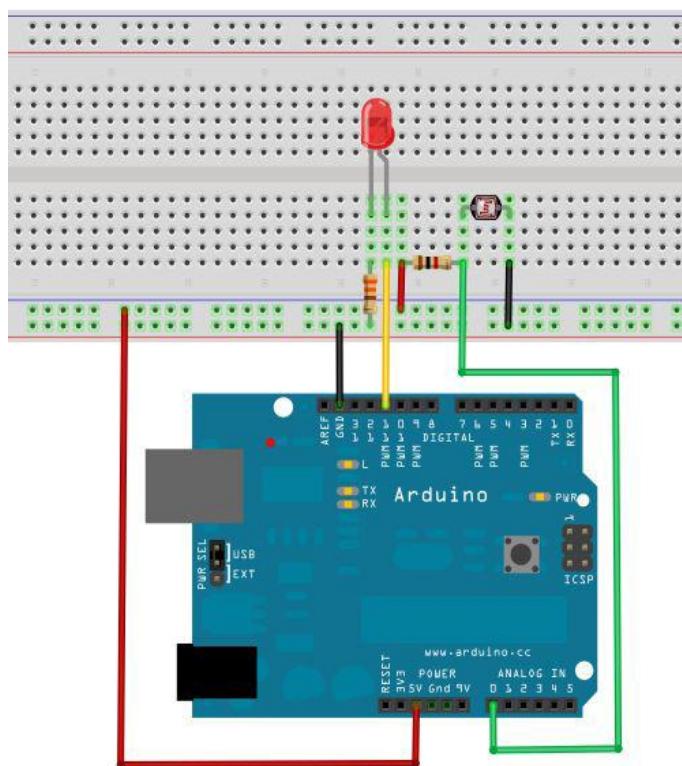
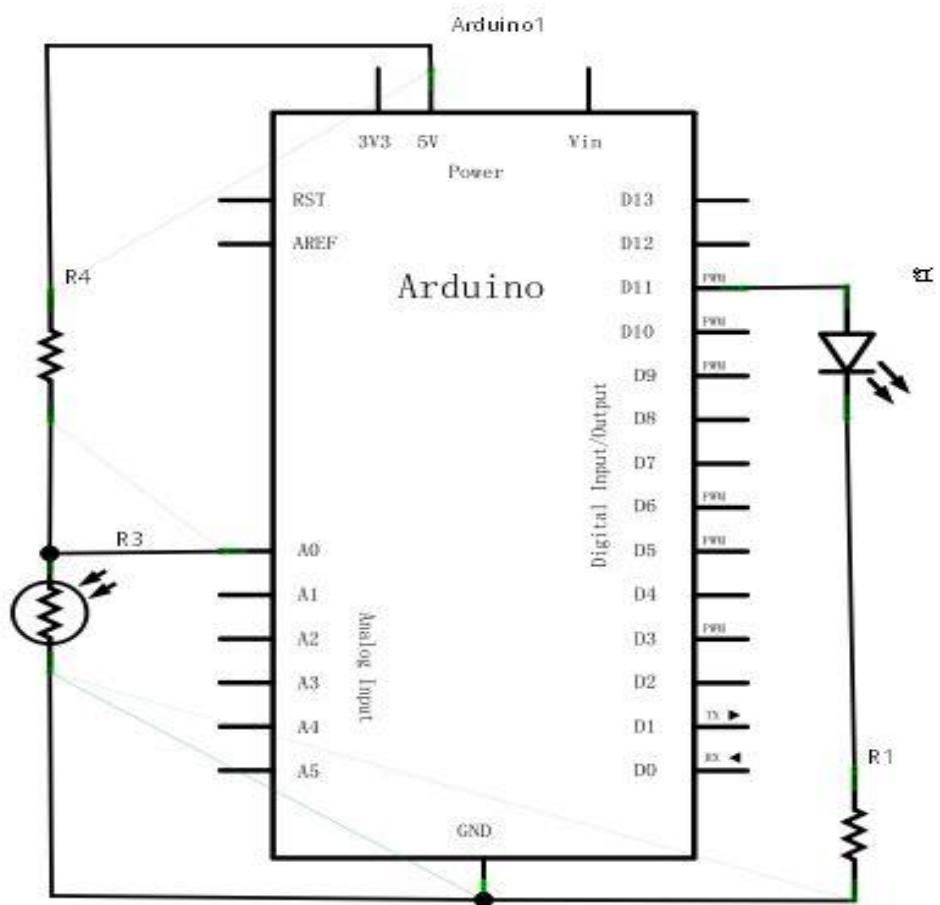
\*1 résistance 10k

\*1 résistance 220ohm

\*1 carte de prototypage

\*1 tie une bande

Voici le circuit de connexion



référence source :

```
int potpin=0;//Define the analog interface 0 connect the photosensitive resistance  
int ledpin=11;//Define digital interface 11 output PWM adjust LED brightness  
  
int val=0;//Defined variable val  
  
void setup()  
{  
    pinMode(ledpin,OUTPUT);//Define digital interface 11 for output  
    Serial.begin(9600);//Set the baud rate to 9600  
}  
  
void loop()  
{  
    val=analogRead(potpin);//Read the sensor's analog values and assign to val  
    Serial.println(val);//Display Val variable value  
    analogWrite(ledpin,val);// Turn on the LED and set the brightness (the maximum  
    // value of the PWM output is 255)  
    delay(10);//Delay 0.01 seconds  
}
```

Ici nous divisons la valeur des capteurs par 4. La raison est que l'input Analogue a une valeur comprise entre 0 et 1023, et la simulation output a une valeur entre 0 et 255. Télécharger le programme et essayer de changer l'intensité de l'environnement ou la résistance photosensible peut voir les lumières. Dans la vie de tous les jours, l'utilisation de la résistance photosensible est très large.

# Leçon 20. Capteur de température.

## introduction

LM35 est un capteur de température produit par National semiconductor. Il a une précision opérationnelle très élevée, et une portée très large. De taille petite, peu coûteux et fiable, LM35 est largement utilisée dans l'ingénierie. Puisqu'il utilise une compensation interne, l'output peut commencer à 0°C. LM35 a de nombreux lots différents. Sous température normale, LM35 ne nécessite pas une calibration supplémentaire pour atteindre la précision de + ou - 1/4°C. Le mode d'alimentation peut être classifié comme source simple ou double source positive/négative. Ses broches sont comme montrés ci-dessous. En mode double alimentation positive/négative, il peut mesurer une température négative. En mode d'alimentation simple et 25°C, le courant normal est de 50µA et il a une large application de tension, entre 4 et 20V, pour économiser l'électricité.



## Composants

- 1\* Carte Kuman Uno
- 1 \* carte de prototypage
- 1 \* Cable de données usb
- 1 \* r Capteur de température LM35

- 1 \* I2C LCD1602
- cables de démarrage

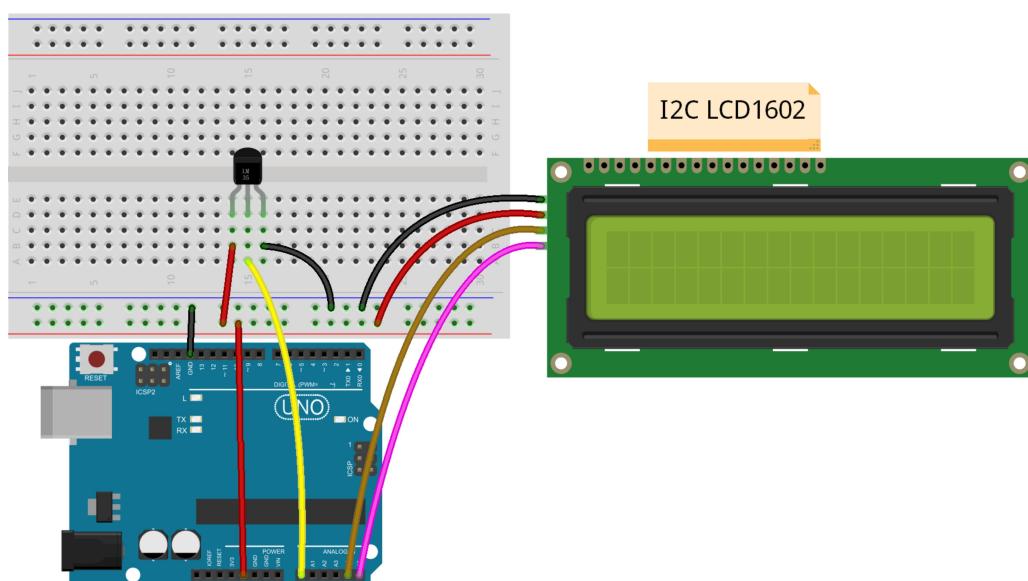
### Principe du test.

La tension output du LM35 est proportionnelle à la température Celsius. Quand c'est placé à 0°C température ambiante, l'output sera de 0. La tension augmentera de 10mV chaque fois que la température augmente de 1°C. La formule de calcul est comme suit :

$$V_{\text{out\_LM35}}(T) = 10 \text{ mV}/^\circ\text{C} \times T^\circ\text{C}$$

### Procédure du test

#### Etape 1 : construire le circuit

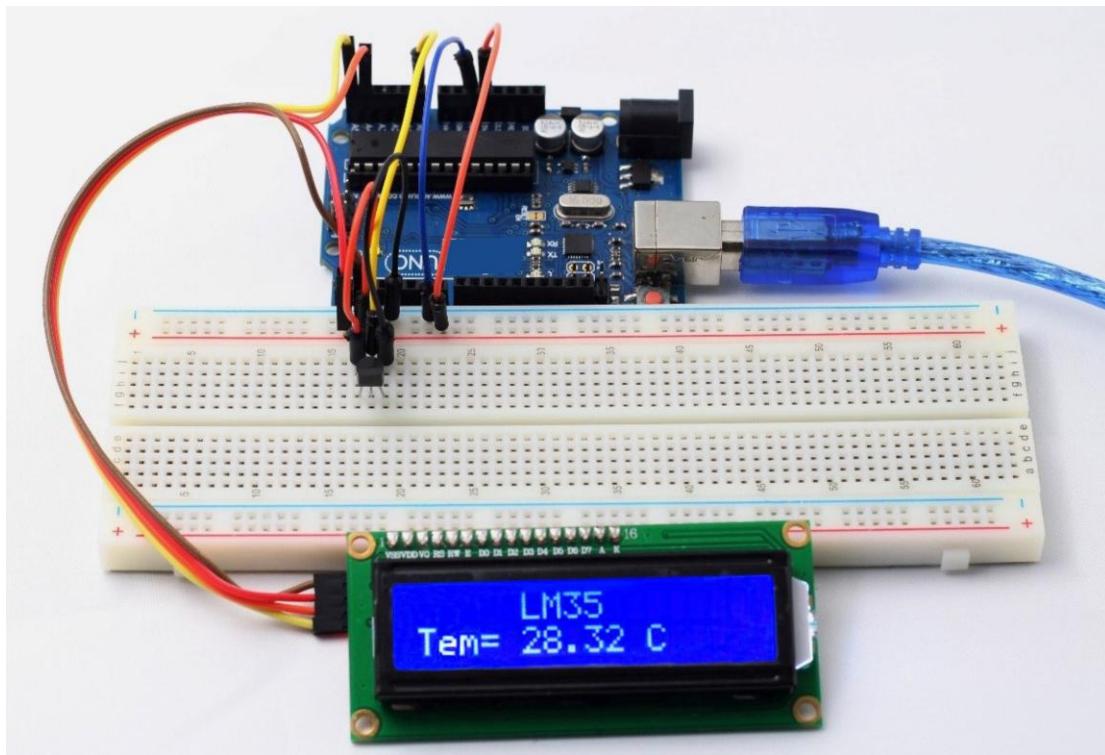


*Note : il faut ajouter une bibliothèque, voir le fichier liquidcrystal*

#### Etape 2 : compiler le code

#### Etape 3 telechargez le sketch sur la carte kuman uno

. Maintenant vous pouvez voir la température affichée sur I2C LCD1602



```
*****
* name:LM35 Temperature Sensor
* function:
* LM35 output voltage has a linear relation with the Celsius temperature, output of 0 v when
0 °C,
* every increase 1 °C, the output voltage increase 10 mv
*****
//lm35 attach to A0
 *****
#define lmPin A0 //LM35 attach to
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2); // set the LCD address to 0x27 for a 16 chars and 2 line display
float tem = 0;
long lmVal = 0;
void setup()
{
    lcd.init(); //initialize the lcd
    lcd.backlight(); //open the backlight
}
void loop()
{
```

```
ImVal = analogRead(ImPin);//read the value of A0
tem = (ImVal * 0.0048828125 * 100); //5/1024=0.0048828125;1000/10=100
lcd.setCursor(5,0); //place the cursor on 5 column,0 row
lcd.print("LM35"); //print "LM35"
lcd.setCursor(0,1); //place the cursor on 0 column,1 row
lcd.print("Tem= "); //print "Tem= "
lcd.setCursor(5,1); //place the cursor on 5 column,1 row
lcd.print(tem); //print tem

lcd.print(char(223)); //print the unit " °C "
lcd.print("C");
delay(200); //delay 200ms}
```

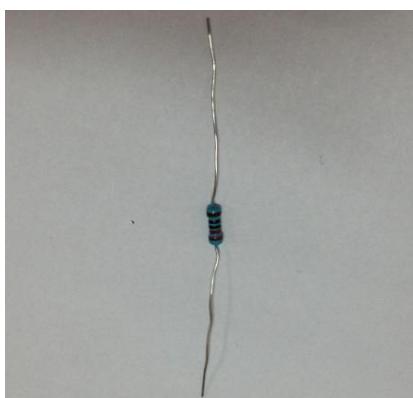
## Leçon 21. Test de scintillement LED

:  
Le test LED est une des bases de test de comparaison, cette fois nous utilisons un autre port I/O et diminuons la lumière LED pour finir le test, nous avons besoin d'équipement comme le contrôleur Arduino et le câble de téléchargement USB

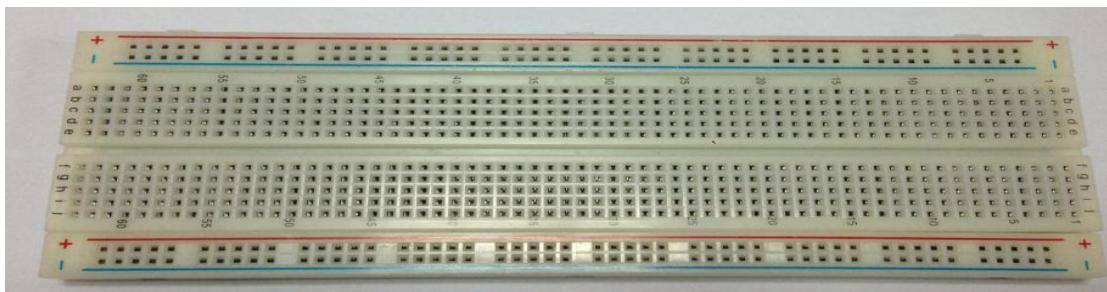
1. Red M5 LED 1 pcs



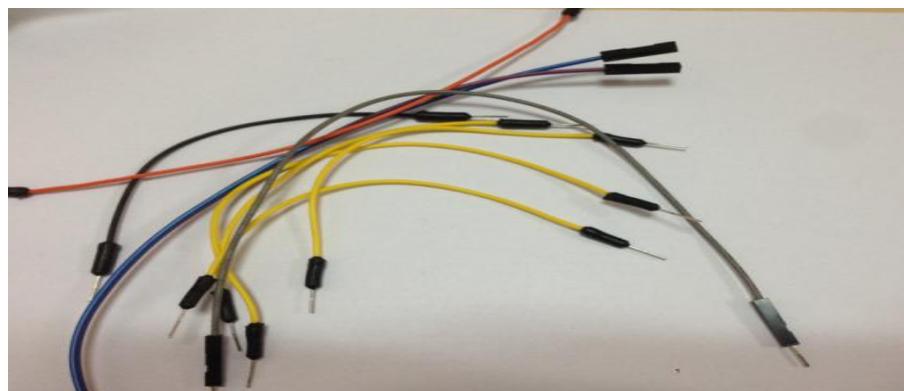
2. 220Ωline resistance\*1



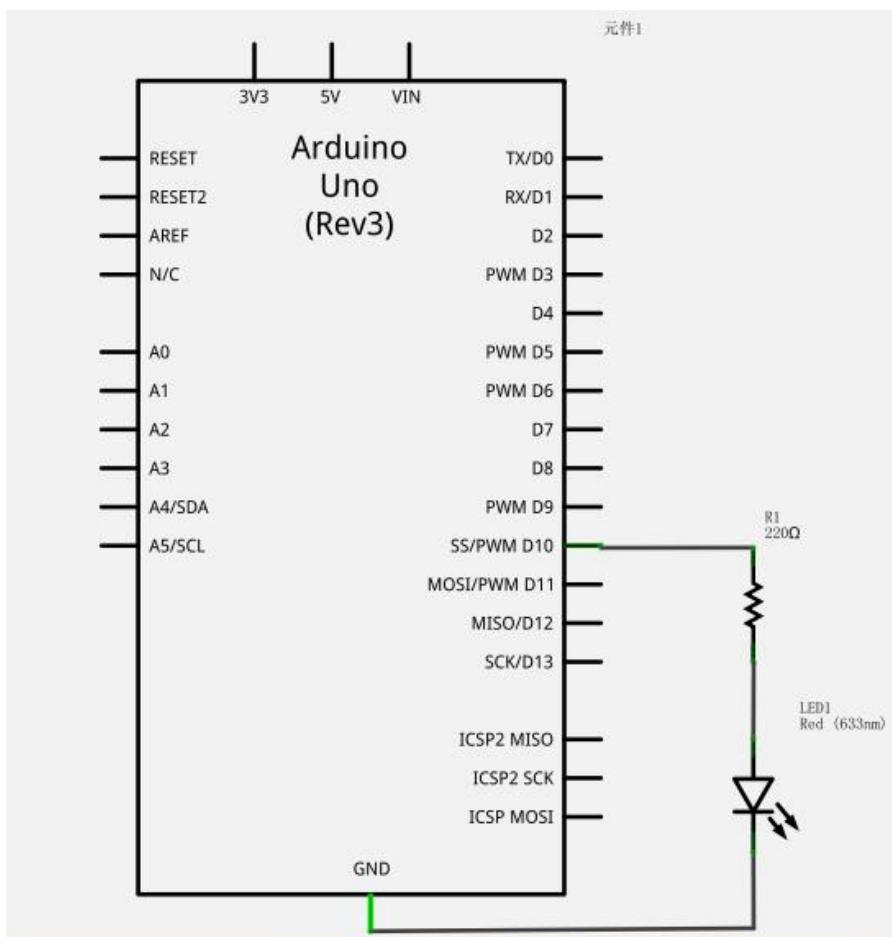
3.bread board\*1



#### 4. Bread board jump line\*1tie

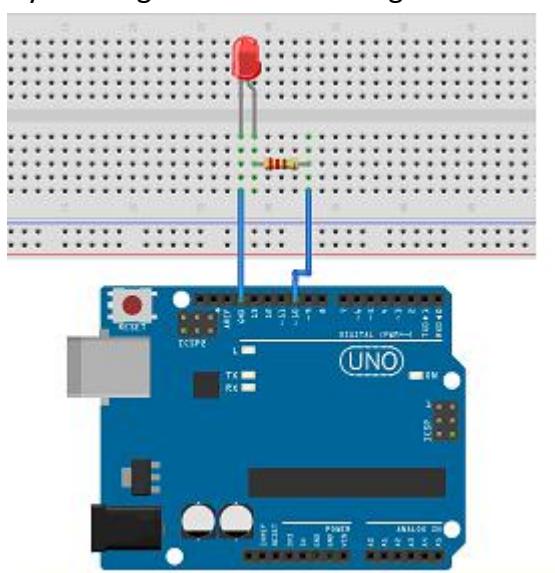


La prochaine étape avec les lumières suivantes, le principe du graphique est comme une carte de physique. Nous utilisons l'interface 10. Avec une diode émettrice de lumière (LED) connectée à un résistor, de 220ohm, ou le courant va brûler la diode.



Lumière carte

Physicaldiagramconnectiondiagram:



En conformité avec le circuit ci-dessus, vous pouvez commencer à graver des programmes, la LED continue à clignoter, 1 seconde sur 2. Cette procédure est très simple et Arduino a une routine dans le clignotement, similaire à l'interface 13 digit pour 10 digit.

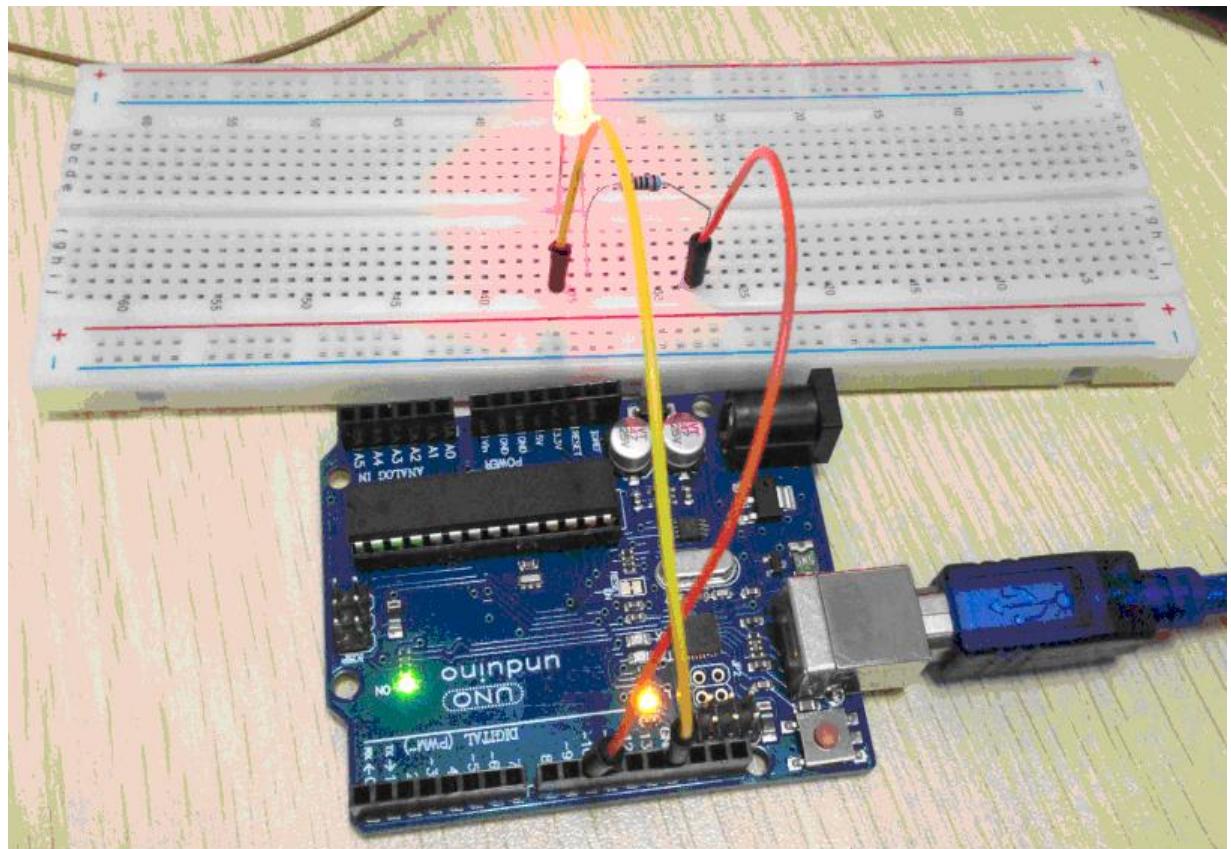
---

Reference procédure ci - dessous

```
int ledPin = 10; //Define number 10 interface  
  
void setup()  
{  
pinMode(ledPin, OUTPUT); //The definition of small interface for output interface  
}  
  
void loop()  
{  
digitalWrite(ledPin, HIGH); //Light the lamp  
delay(1000); //Delay 1 seconds  
digitalWrite(ledPin, LOW); //Lights out  
delay(1000); //Delay 1 seconds  
}
```

---

Télécharger le programme, vous voyez 10 petites lumières clignoter, l'intervalle est de 1 seconde.



# Leçon 22. Récepteur infrarouge.

## Introduction

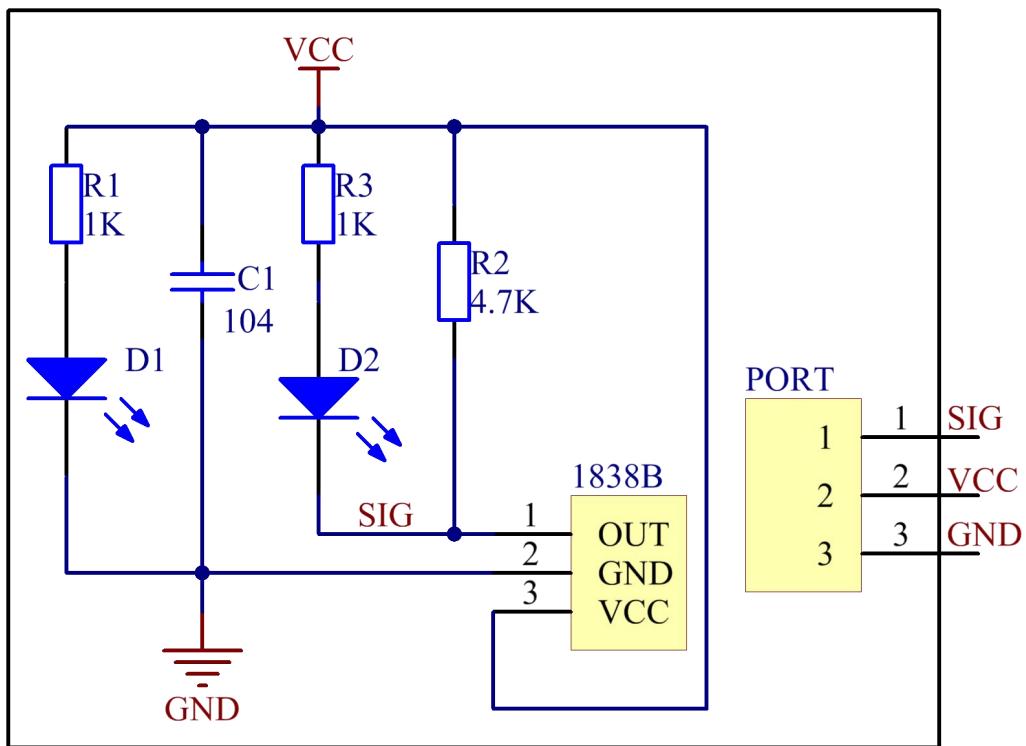
Un récepteur infrarouge est un composant qui reçoit des signaux infrarouges et peut recevoir indépendamment des rayons infrarouges et des signaux compatibles au niveau TTL. C'est similaire à un transistor normal en plastique en taille et approprié pour toutes sortes de transmissions infrarouges et télécommandes.

## composants

- 1 \* KUMAN Uno board
- 1 \* USB data cable
- 1 \* Infrared-receiver module
- 1 \* Remote controller
- 1 \* 3-Pin anti-reverse cable

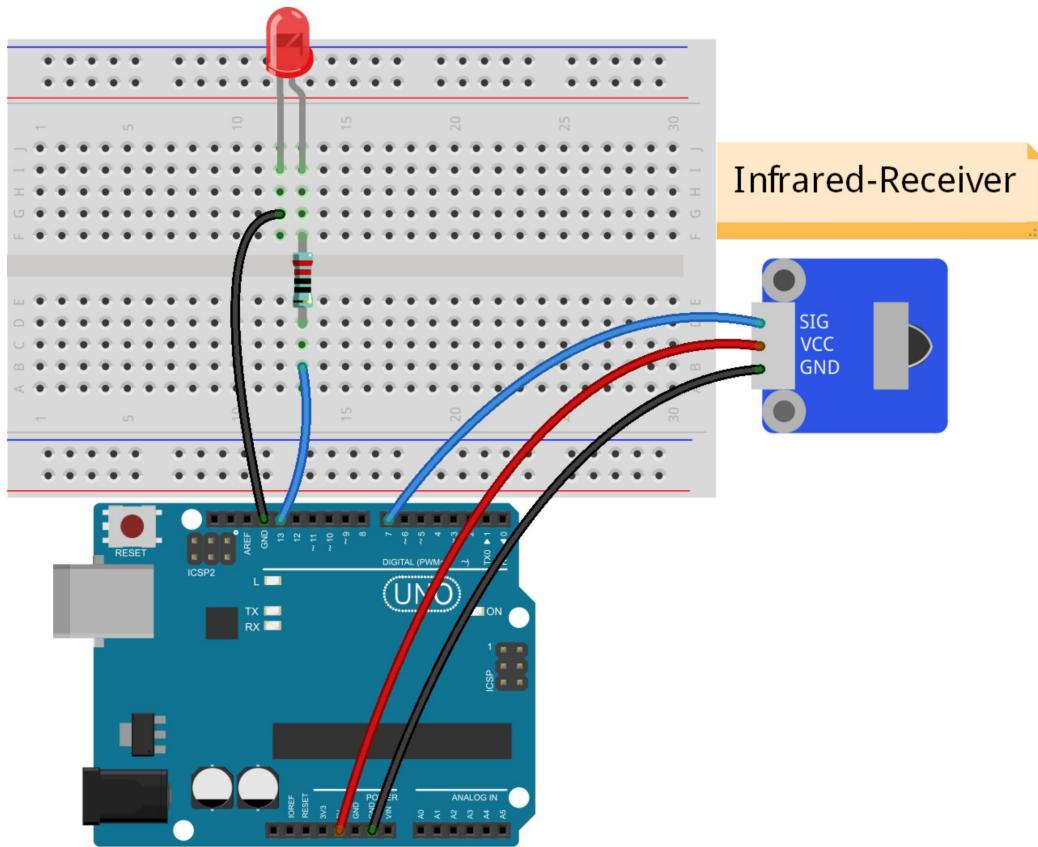
## Principe du test

Contrôle une certaine touche (par exemple celle power) via la télécommande en programmation. Quand vous appuyez sur la touche, il y aura une émission de rayons infrarouges de la télécommande et reçus par le récepteur, la LED sur la carte Kuman Uno va s'allumer. Connecter une LED à la broche 13 sur la carte Kuman Uno pour voir si la touche Power est enfoncée.



## Experimental Procedures

etape 1 : construire le circuit



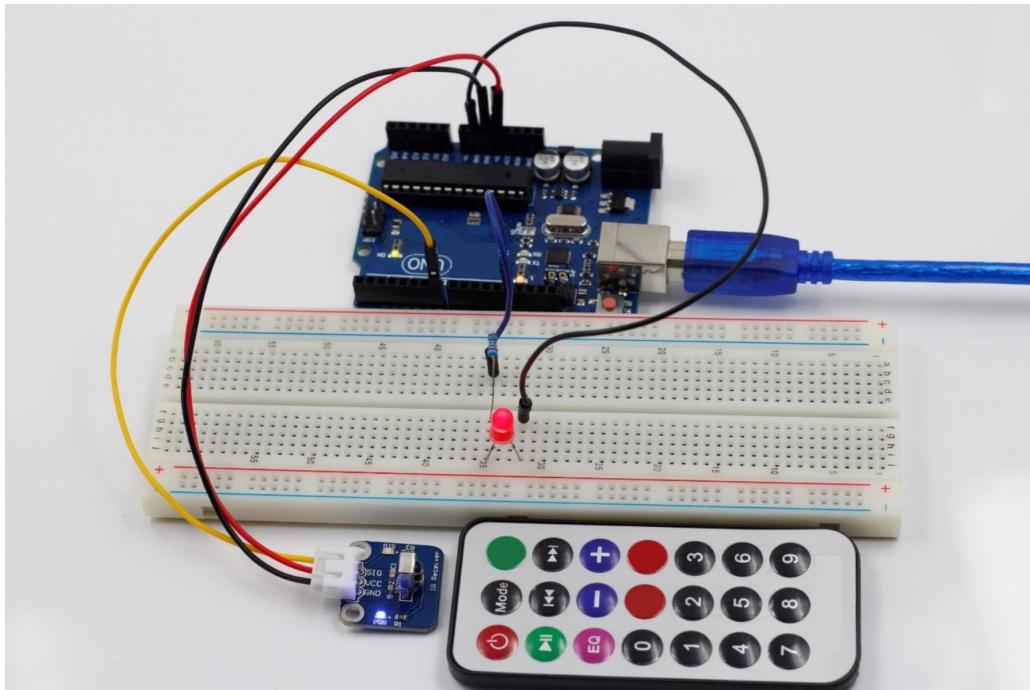
**Note:**

*Il faut ajouter une bibliothèque. Voir la description dans Bibliothèque.*

etape 2 : compiler le code.

etape 3 : télécharger le sketch sur la carte Kuman uno

Maintenant, appuyer sur la touche power de la télécommande, les 2 LED attachées et connectées au pin 13 de la carte Kuman vont s'allumer. Appuyez ensuite sur n'importe quelle touche, les LED vont s'éteindre.



## 1、 Infrared receiving head introduction

Qu'est ce que la tête de réception infrarouge.

Il y a une séries d'impulsions binaires qui proviennent de la télécommande infrarouge. Afin

de le faire à partir d'autre signaux en transmission sans fil, il faut en général d

Abord faire la modulation sur une fréquence spécifique, et ensuite par les diodes qui

émettent l'infrarouge, ils recoivent le signal de fréquence spécifique et le restaure en

impulsion binaire, comme démodulation

## **Principe de fonctionnement**

Le récepteur intégré va transmettre le signal infrarouge en signal faible, qui ira via l'amplificateur interne IC, et ensuite via le contrôle automatique, filtre band-pass, démodulateur, forme de vagues après l'encodage original pour restaurer l'output du signal d'origine par la broche du récepteur sur les circuits de reconnaissance des appareils électriques.

## **Broche et connexion de la tête récepteur infrarouge**



La tête récepteur infrarouge a 3 broches. Lors de l'utilisation du port analogue VOUT, la carte expérimentale GND est reçue, ainsi que le VCC, +5v

## **test de télécommande infrarouge**

### **Appareil expérimental**

#### **1 télécommande infrarouge**

#### **tête recepteur infrarouge**

#### **6LED**

résistance 220 : 6

## 2、 Connexion experimentale

Tout d'abord, la carte est connectée, la tête recepteur infrarouge comme la méthode ci-dessus, le VOUT connecté à la broche digitale 11, la lampe LED via la résistance connecté à la broche 2,3,4,5,6,7.

## 3、 principe de l'expérience

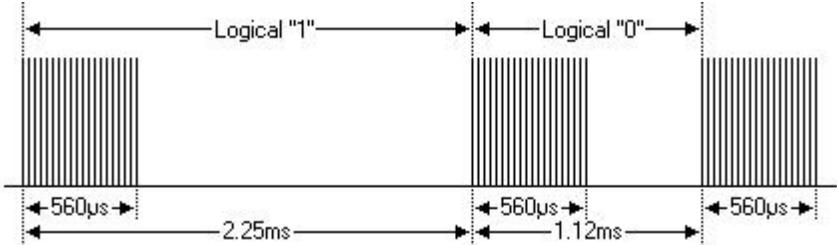
Il faut connaitre l'encodage de la télécommande. Ce produit est utilisé pour controler le code : NEC.  
Suivant le protocole NEC, description du protocole NEC, caractéristiques, bits 8

(3) pulse position modulation

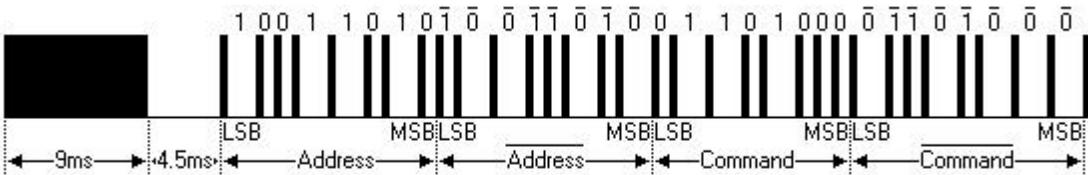
(4) carrier frequency 38kHz

(5) each time pour les 1.125ms ring 2.25ms

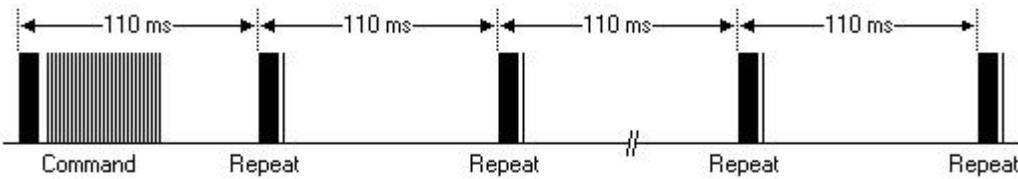
·Logic 0 et 1 sont défini comme ci -dessous .:



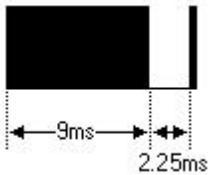
·Press the launch pulse at once:



L'image ci-dessus montre une séquence d'impulsions typique de NEC, cela lance d'abord le protocole LSB. La commande de transmission est 0x59 , 0x16. Un message du haut niveau 9ms, et de bas niveau 4.5ms, et le code de commande. Tous les bits sont recuperés ensuite, pouvant etre utilisés pour la confirmation du message.



Une commande envoyée une fois, le bouton de la télécommande est toujours enfoncé. Quand le bouton a été pressé, les premières 110ms de l'impulsion et l'image ci-dessus, après la répétition du code chaque 110ms.



Note :

Line connection diagram:

Program code:

```
#include <IRremote.h>
int RECV_PIN = 11;
int LED1 = 2;
int LED2 = 3;
int LED3 = 4;
int LED4 = 5;
int LED5 = 6;
int LED6 = 7;
long on1 = 0x00FFA25D;
long off1 = 0x00FFE01F;
long on2 = 0x00FF629D;
long off2 = 0x00FFA857;
long on3 = 0x00FFE21D;
long off3 = 0x00FF906F;
long on4 = 0x00FF22DD;
long off4 = 0x00FF6897;
long on5 = 0x00FF02FD;
long off5 = 0x00FF9867;
long on6 = 0x00FFC23D;
long off6 = 0x00FFB047;
IRrecv irrecv(RECV_PIN);
decode_results results;
// Dumps out the decode_results structure.
// Call this after IRrecv::decode()
```

```

// void * to work around compiler issue
//void dump(void *v) {
//  decode_results *results = (decode_results *)v
void dump(decode_results *results) {
    int count = results->rawlen;
    if (results->decode_type == UNKNOWN)
    {
        Serial.println("Could not decode message");
    }
    else
    {
        if (results->decode_type == NEC)
        {
            Serial.print("Decoded NEC: ");
        }
        else if (results->decode_type == SONY)
        {
            Serial.print("Decoded SONY: ");
        }
        else if (results->decode_type == RC5)
        {
            Serial.print("Decoded RC5: ");
        }
        else if (results->decode_type == RC6)
        {
            Serial.print("Decoded RC6: ");
        }
        Serial.print(results->value, HEX);
        Serial.print(" (");
        Serial.print(results->bits, DEC);
        Serial.println(" bits)");
    }
    Serial.print("Raw (");
    Serial.print(count, DEC);
    Serial.print("): ");
}

for (int i = 0; i < count; i++)
{
    if ((i % 2) == 1) {
        Serial.print(results->rawbuf[i]*USECPERTICK, DEC);
    }
    else
    {
        Serial.print(-(int)results->rawbuf[i]*USECPERTICK, DEC);
    }
}

```

```

        }
        Serial.print(" ");
    }
    Serial.println("");
}

void setup()
{
    pinMode(RECV_PIN, INPUT);
    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
    pinMode(LED3, OUTPUT);
    pinMode(LED4, OUTPUT);
    pinMode(LED5, OUTPUT);
    pinMode(LED6, OUTPUT);
    pinMode(13, OUTPUT);
    Serial.begin(9600);

    irrecv.enableIRIn(); // Start the receiver
}
int on = 0;
unsigned long last = millis();

void loop()
{
    if (irrecv.decode(&results))
    {
        // If it's been at least 1/4 second since the last
        // IR received, toggle the relay
        if (millis() - last > 250)
        {
            on = !on;
            // digitalWrite(8, on ? HIGH : LOW);
            digitalWrite(13, on ? HIGH : LOW);
            dump(&results);
        }
        if (results.value == on1 )
            digitalWrite(LED1, HIGH);
        if (results.value == off1 )
            digitalWrite(LED1, LOW);
        if (results.value == on2 )
            digitalWrite(LED2, HIGH);
        if (results.value == off2 )
            digitalWrite(LED2, LOW);
    }
}

```

```
if (results.value == on3 )
    digitalWrite(LED3, HIGH);
if (results.value == off3 )
    digitalWrite(LED3, LOW);
if (results.value == on4 )
    digitalWrite(LED4, HIGH);
if (results.value == off4 )
    digitalWrite(LED4, LOW);
if (results.value == on5 )
    digitalWrite(LED5, HIGH);
if (results.value == off5 )
    digitalWrite(LED5, LOW);
if (results.value == on6 )
    digitalWrite(LED6, HIGH);
if (results.value == off6 )
    digitalWrite(LED6, LOW);
last = millis();
irrecv.resume(); // Receive the next value
}
}
```

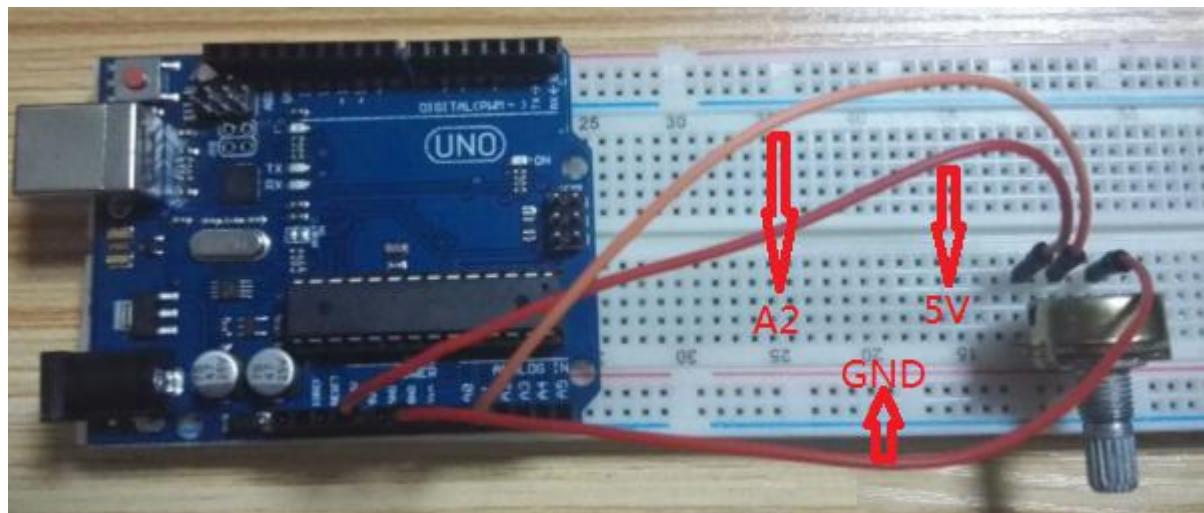
## Leçon 23. Potentiomètre.

Un potentiomètre est un simple bouton qui fournit une résistance variable, que nous pouvons lire sur la carte Arduino en valeur analogue. Dans cet exemple, cette valeur contrôle la rapidité du clignotement de la LED.

Nous connectons 3 câbles à la carte Arduino. Un va à la terre via une des broches externes du potentiomètre. Le second va via l'autre broche de 5volts du potentiomètre. Le 3ème va sur l'input 2 analogue sur la broche du milieu du potentiomètre.

En tournant la tige du potentiomètre, nous changeons la résistance des cotés du wiper qui est connecté à la broche du centre du potentiomètre. Cela change la proximité relative de la broche de 5 volts et terre, donnant un input analogue différent. Quand la tige est tournée dans une direction, il y a 0 volt allant dans la broche, et nous lisons 0. Quand la tige est tournée dans l'autre direction, il y a 5 volts allant dans la broche et nous lisons 1023. Analogread() retourne à un nombre entre 0 et 1023 qui est proportionnel au total de volt appliqués à la broche.

### Diagramme du cablage:



### Composants

- 1 \* kuman Uno board
- 1 \* USB cable
- une resistance ajustable
- able de demarrage
- une carte de prototypage

## Code :

```
/* Analog Read to LED

* -----
*
* turns on and off a light emitting diode(LED) connected to digital
* pin 13. The amount of time the LED will be on and off depends on
* the value obtained by analogRead(). In the easiest case we connect
* a potentiometer to analog pin 2.

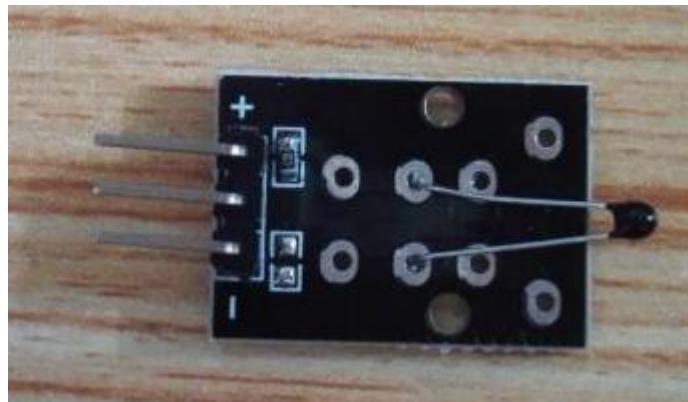
*/
int potPin = 2;      // select the input pin for the potentiometer
int ledPin = 13;     // select the pin for the LED
int val = 0;          // variable to store the value coming from the sensor

void setup() {
    pinMode(ledPin, OUTPUT); // declare the ledPin as an OUTPUT
}

void loop() {
    val = analogRead(potPin); // read the value from the sensor
    digitalWrite(ledPin, HIGH); // turn the ledPin on
    delay(val);             // stop the program for some time
}
```

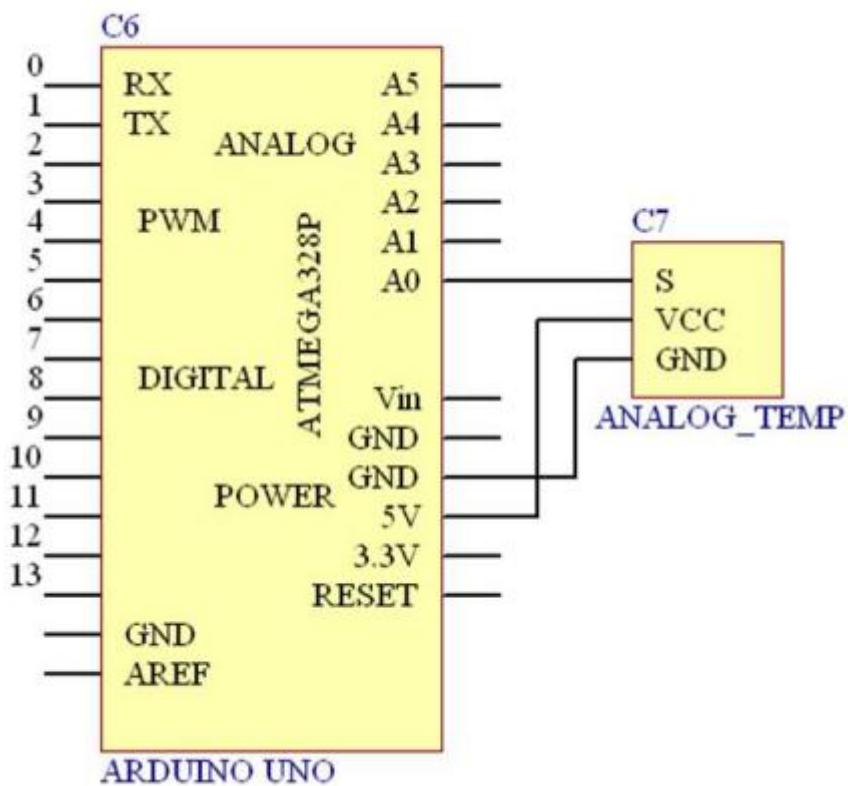
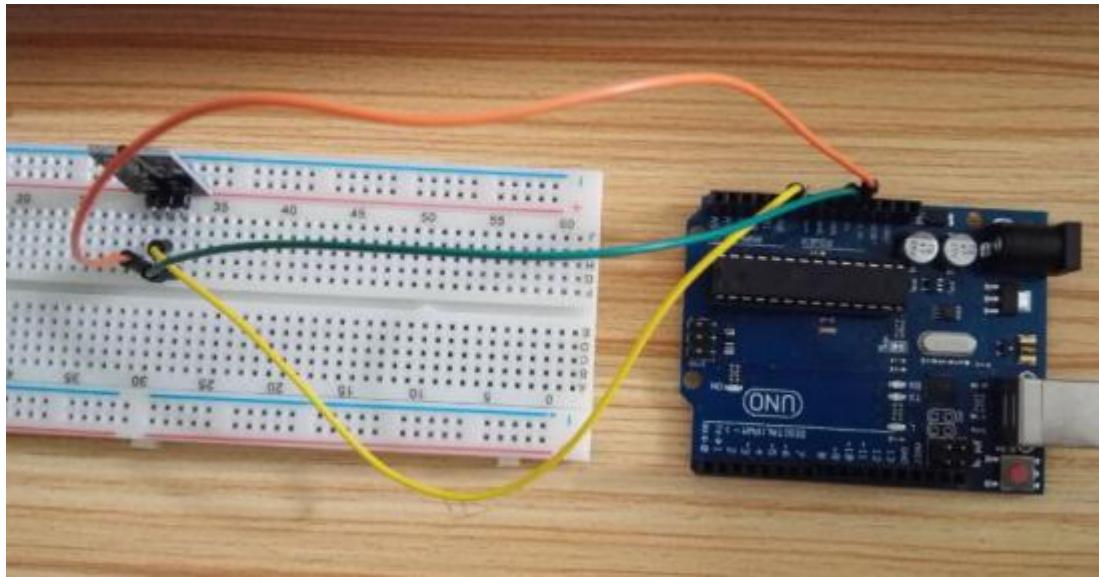
```
digitalWrite(ledPin, LOW); // turn the ledPin off  
  
delay(val); // stop the program for some time  
  
}
```

## Leçon 24. Capteur de température analogue



### Description

Le capteur de température est une thermistance NTC  
Mesure de température multi-points. 55 °C / +125 °C  
récision + / - 0.5 °C  
Matériau : mixte  
taille : 3 x 1.5 x 0.6cm  
Poids : 2g



### Composants

- 1 \* kuman Uno board
- 1 \* USB cable
- module de température analogue
- câble de démarrage
- carte de prototypage

Test 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;           // Convert Kelvin to Celcius
return Temp;
}
void setup() {
Serial.begin(9600);

}

void loop() {
Serial.print(Thermister(analogRead(0))); // display Fahrenheit
Serial.println("c");
delay(500);
}
```

Les thermistances peuvent aussi prendre la température, car elles combinent le Steinhart - Hart Thermistor équation, les fonctions du code test dans le double Thermister (int RawADC) est juste l'incarnation de l'équation.

Jetez un coup d'oeil aux résultats du test.



The test completed successfully.

## Leçon 25. Module tactile

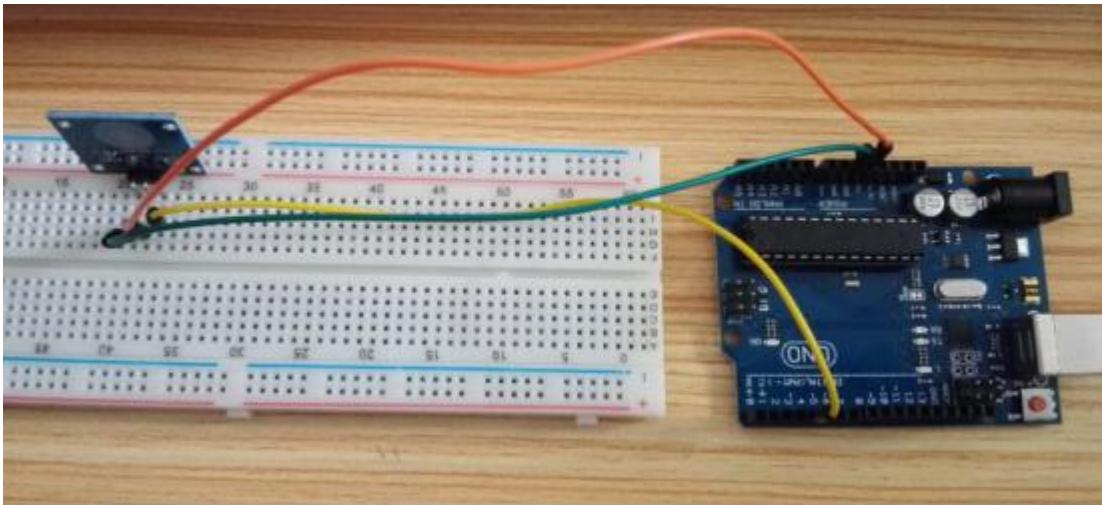


C'est un module de contrôle à interrupteur direct, quand un doigt touché l'écran ou il y a les cercles blancs, la broche SIG va envoyer un signal de haut niveau, quand le doigt se retire, SIG sera à bas niveau.

Ci-dessous est le mode de connexion

Touch module	Arduino
GND	GND
VCC	5V
SIG	D3

Ci-dessous est le diagramme de cablage



### Composants

- 1 \* carte kuman uno
- 1 \*cable usb
- 1 \* module tactile
- cables de demarrage
- 1 \* carte de prototypage

### Le test montre que :

Nous utilisons la broche D3 avec détection haute/basse, quand le signal de la broche D3 est haut, la broche D13 est haute, quand le signal du D3 est bas, le D13 est bas. Nous choisissons la broche D13 car elle est avec la LED sur la carte principale

### Test code:

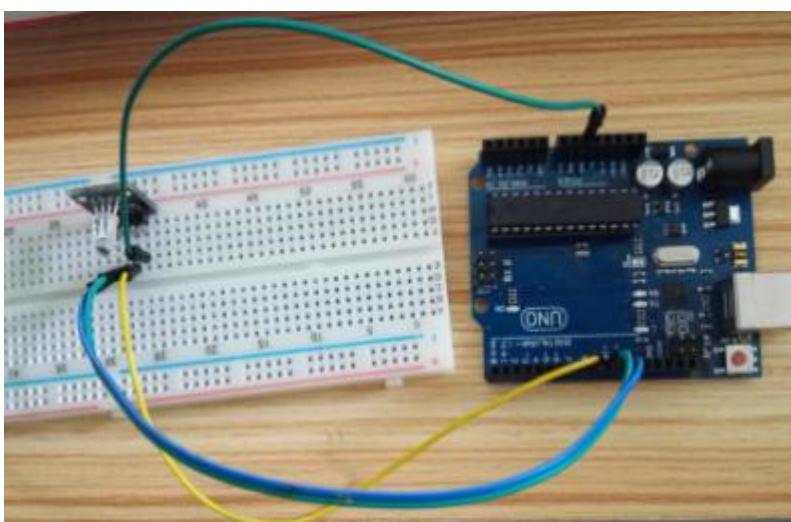
```

int Led=13;// Define LED interface
int touch=3;//Define the touch module interface
int val;// Digital variables "val"
void setup()
{
pinMode(Led,OUTPUT);// Define the LED for the output interface
pinMode(touch,INPUT);/*Define the touch module for the input interface*/
}
void loop()
{
val=digitalRead(touch);/* Will the value of the digital interface 3 read assigned to val*/
if(val==HIGH)/* When the key switch sensor detection signal, LED light*/
{
digitalWrite(Led,HIGH);
}

```

```
    }
} else
{
digitalWrite(Led,LOW);
}
}
```

## Leçon 26. Module LED full color- 3 couleurs



Compatible with Arduino DIY project

Voici le module LED 3 couleurs RGB pour Arduino. Rouge, vert, bleu

MAtiere : PCB

RGB signal output

3 couleurs : rouge, vert, bleu

Compatible avec les projets Arduino DIY

composants

- 1 \*carte kuman uno
- 1 \* USB cable
- 1 \* RGB module
- cables de demarrage
- 1 \*carte prototypage

Arduino test code:

```
int redpin = 11; //select the pin for the red LED
int bluepin =10; // select the pin for the blue LED
int greenpin =9;// select the pin for the green LED

int val;

void setup() {
  pinMode(redpin, OUTPUT);
  pinMode(bluepin, OUTPUT);
  pinMode(greenpin, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
for(val=255; val>0; val--)
{
  analogWrite(11, val);
  analogWrite(10, 255-val);
  analogWrite(9, 128-val);
  delay(1);
}
for(val=0; val<255; val++)
{
  analogWrite(11, val);
  analogWrite(10, 255-val);
  analogWrite(9, 128-val);
  delay(1);
}
Serial.println(val, DEC);
```