Formation Arduino
Centre Culturel Algérien
Future & Energies

Sid-Ali Akil

(Février 2016)

Quand et comment Arduino est-il né - sa petite histoire

Historique

J'ai trouvé une vidéo (en anglais) qui récapitule l'histoire de la conception de la plateforme arduino, ici.
URL: https://vimeo.com/18539129

Après ce bref historique, voyons maintenant le monde des arduino's. Sur le marché il existe une panoplie d'arduino's, simplement parce que les utilisations sont variées et qu'il n'est pas forcément nécessaire d'avoir le meilleur ou le grand modèle dans tous les projets.

Il y a des versions qui sont plus spécialisées, comme par exemple le LilyPad qui est conçu pour être intégré à des vêtements, le Robot qui permet de configurer un robot sur roues, et le Esplora qui peut servir de manette/télécommande.

Nous allons dans ce document nous concentrer sur l'architecture « de base » et ses fonctionnalités, c'est-à-dire la carte avec microcontrôleur, ses entrées/sorties (E/S) et sa programmation.

Comment Arduino est-il structuré - son architecture

Plusieurs architectures arduino se présentent sur le marché. À l'heure ou nous publions cet ouvrage, le plus populaire est le modèle UNO. C'est sur ce modèle que nous allons faire nos expérimentations. La plupart de celles-ci sont directement opérationnelles sur les autres modèles.

Arduino Uno

Figure 1 - Arduino UNO

Caractéristiques principales du modèle UNO

Microcontrôleur (MCU)	ATmega328
Tension d'utilisation	**5V**
Tension d'entrée (recommandée)	7-12V
Tension d'entrée (min, max)	6-20V
E/S digitales	**14 (dont 6 pour les sorties PWM)**
Entrées analogiques	**6**
Courant continu par pin (E/S)	**40 mA**
Courant total sur les 14 pins	**200 mA**
Courant DC de la pin 3.3 volts	50 mA
Mémoire Flash	32 KB (ATmega328) dont 0.5 KB utilisée par le bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Fréquence d'horloge	**16 MHz**
Longueur	68.6 mm
Largeur	53.4 mm
Poids	25 g

Arduino est composé de plusieurs éléments tel que c'est illustré dans l'image suivante.

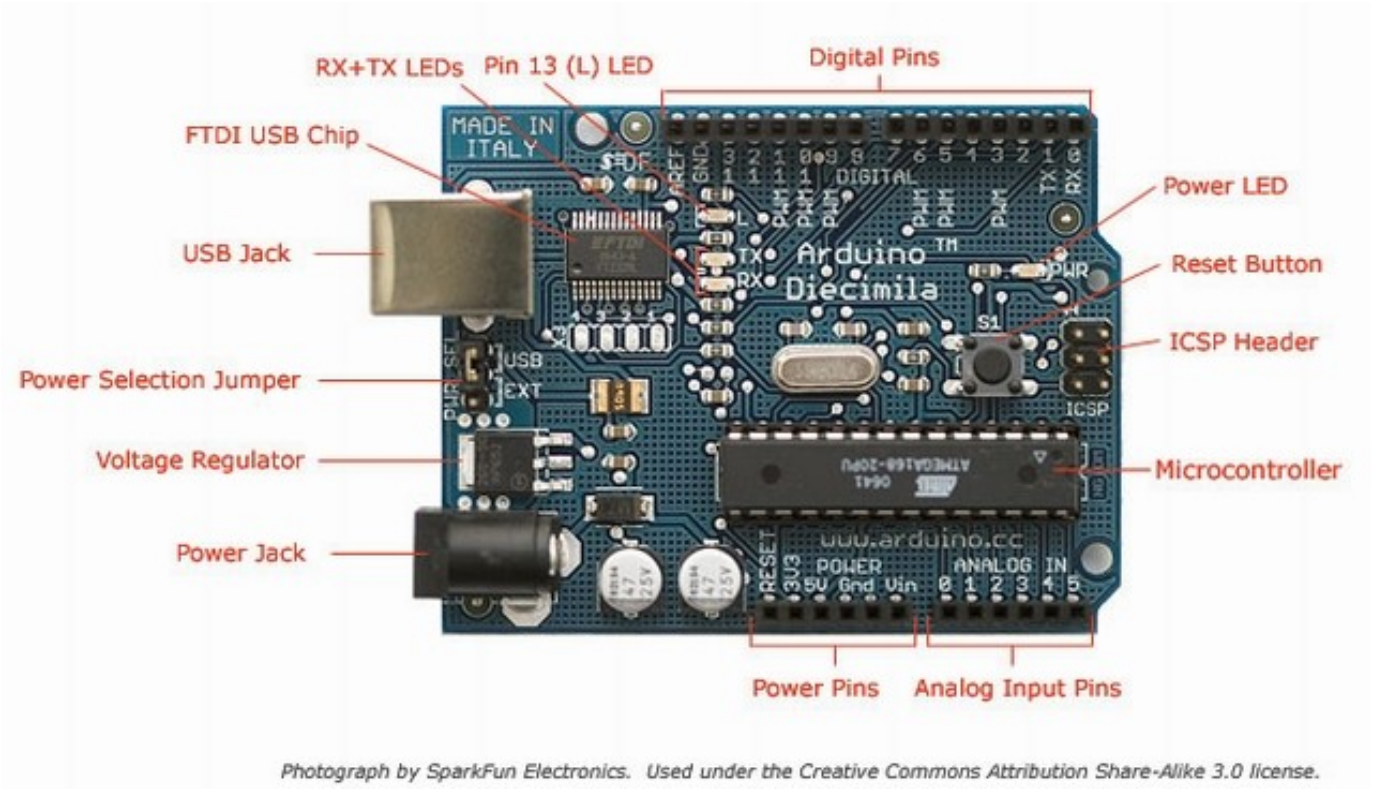


Figure 2 - Architecture de la carte arduino

Le port USB sert à brancher la carte arduino avec le PC. Il est utilisé pour alimenter la carte et pour charger le programme dans la puce arduino pour y être exécuté.
Le connecteur d'alimentation sert comme source d'alimentation électrique (en l'absence de connexion par USB), soit une batterie de 9 volts, soit un adaptateur qui fournit du 9 volts en général.

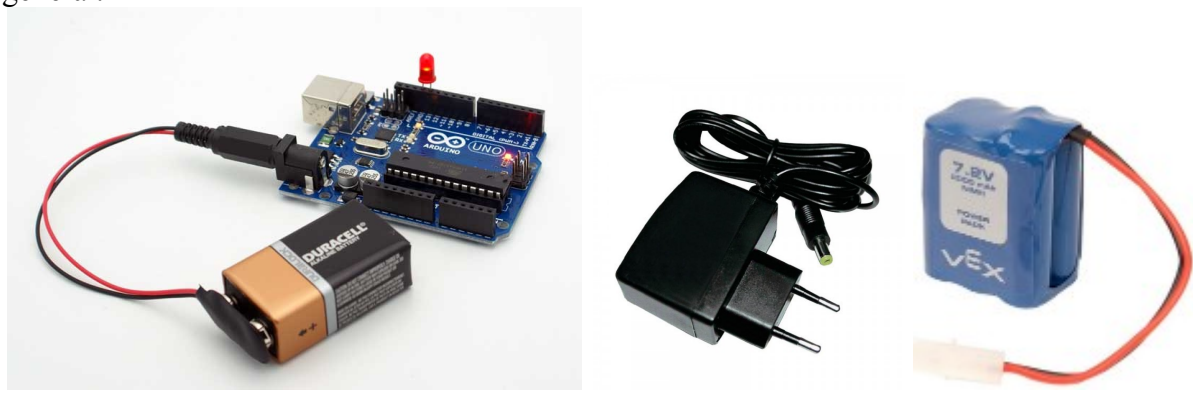


Figure 3 - Arduino alimenté par une batterie 9V ou par Adaptateur électrique (sortie 9V) ou battery pack

Circuit électrique de l'arduino

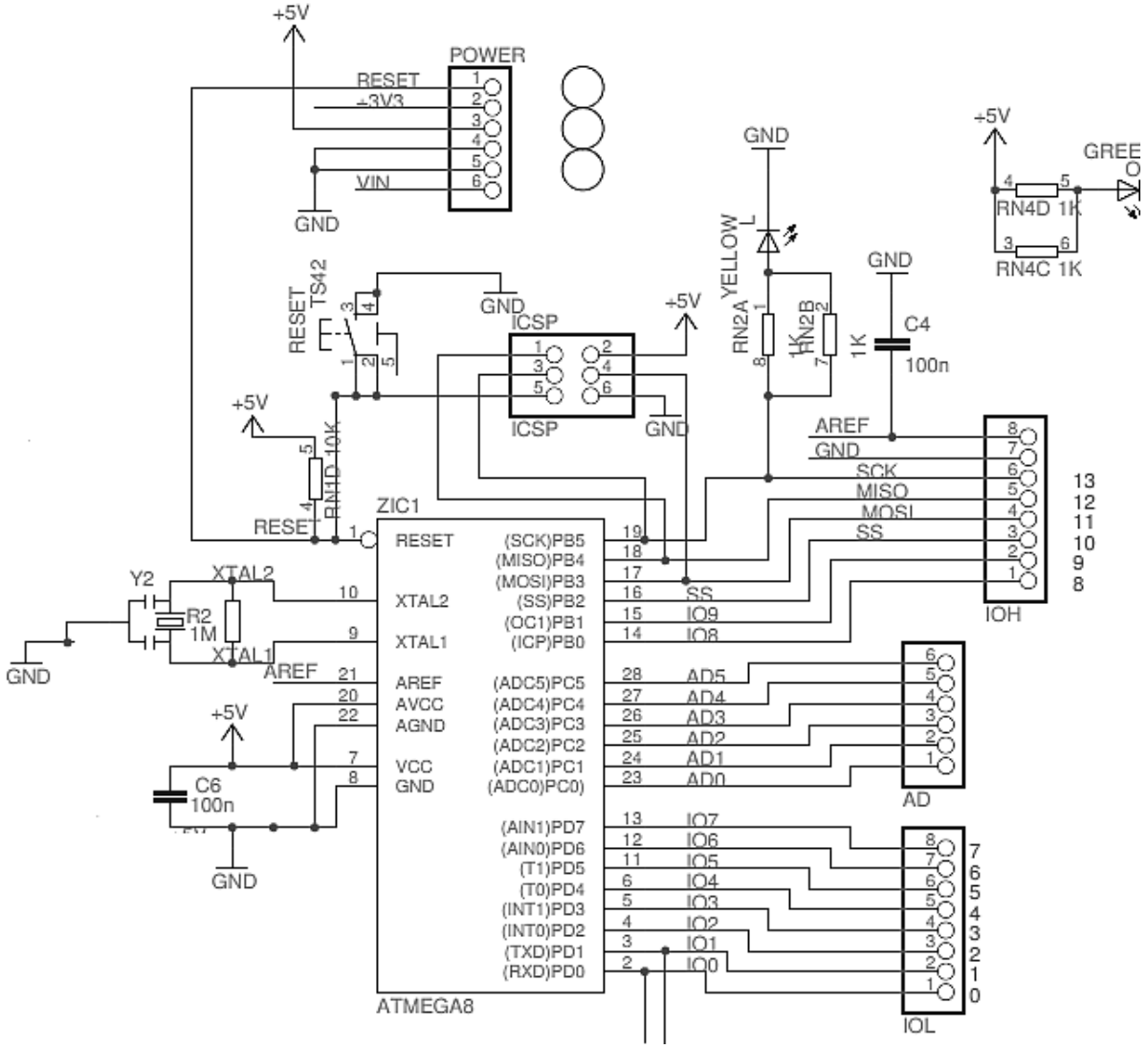
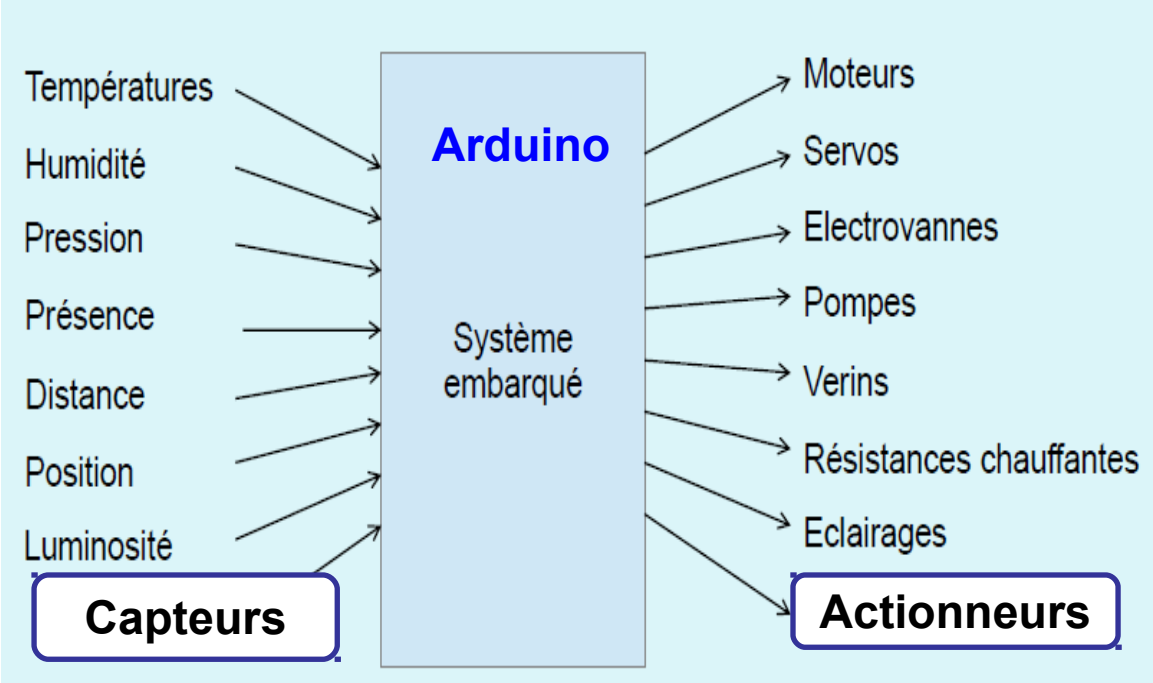


Figure 4 - Circuit électrique de l'arduino

Arduino et différents capteurs et actionneurs



Avec Arduino, nous pouvons commander plusieurs catégories d'actionneurs (relais, moteurs,...) et collecter de l'information à partir des capteurs (pression, température, luminosité, ...).
L'information peut être de type numérique (digitale) ou de type analogique. Les informations sont traitées au sein de l'arduino par des instructions (fonctions) qui permettent de faire les acquisitions ou les commandes sur l'ensemble des éléments physiques qui composent le système au complet.
De plus arduino est très facilement capable de s'intégrer et de performer au sein d'un réseau informatique, au moyen de composants intelligents.

Arduino Duemilanove (Le DUE)

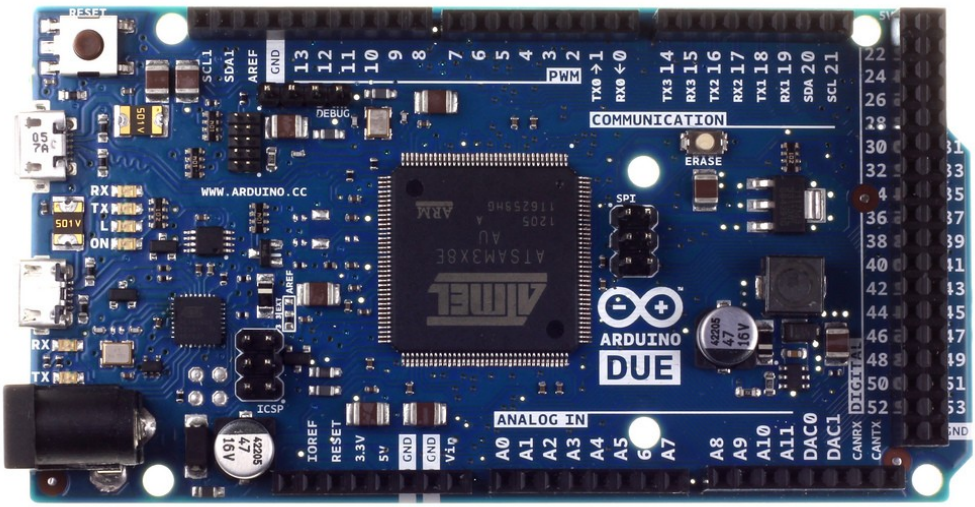


Figure 5 - Arduino Due

L'arduino Due est basé sur le CPU Atmel SAM3X8E ARM Cortex-M3. Il dispose de **54 pins digitales** (dont **12 peuvent servir comme sorties PWM**), de 12 **entrées analogiques**, de **4 UART** (ports série), d'une horloge de **84 MHz**, un port USB OTG, **2 convertisseurs D/A** (digital/analogique), 2 TWI, un power jack , un connecteur SPI, un connecteur JTAG, un bouton reset et un bouton d'effacement.
L'arduino Due fonctionne en 3.3 volts.
Pour plus de détails, il est conseillé de consulter le site suivant:
http://www.arduino.cc/en/Main/ArduinoBoardDue

Arduino Ethernet

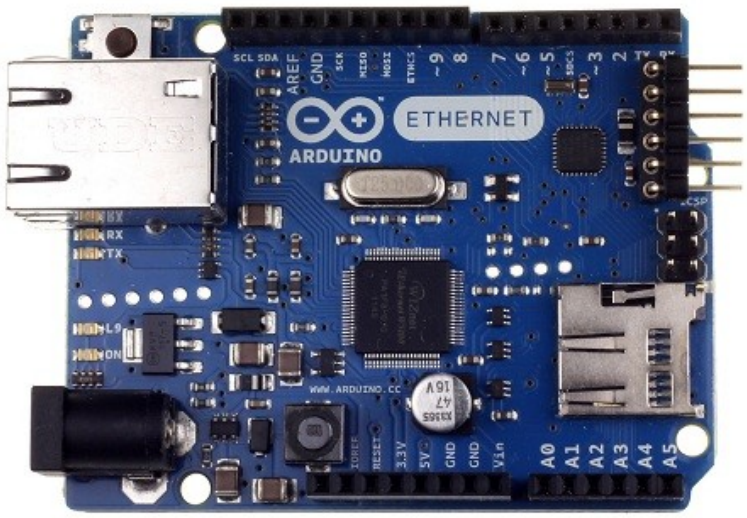


Figure 6 - Arduino ethernet

Arduino Mega

Figure 7 - Arduino Mega

Voici ces principales caractéristiques:

Microcontrôleur	ATmega1280
Operating Voltage	**5V**
Input Voltage (recommended)	7-12V

Input Voltage (limits)	6-20V
Digital I/O Pins	**54 (of which 15 provide PWM output)**
Analog Input Pins	**16**
DC Current per I/O Pin	**40 mA**
DC Current for 3.3V Pin	50 mA
Flash Memory	128 KB of which 4 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	**16 MHz**

Autres Arduinos:

LILYPAD ARDUINO

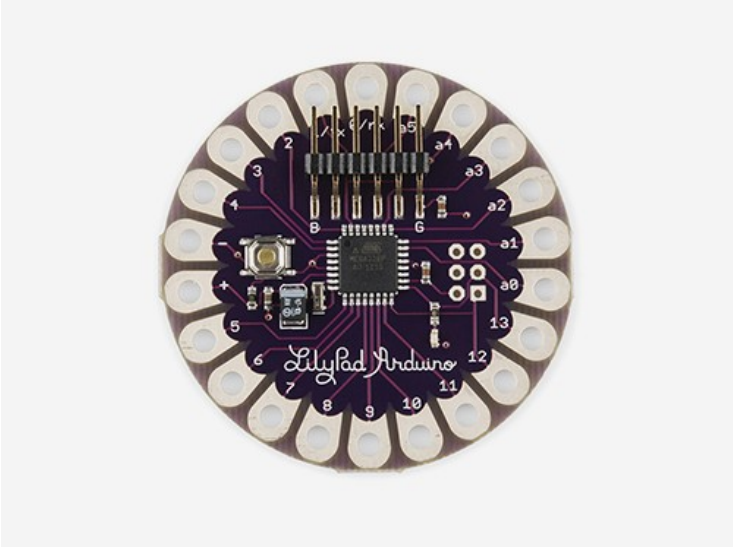


Figure 8 - Lylipad arduino

Le LilyPad Arduino est conçu pour les e-textiles et projets vestimentaires. Il peut être cousu sur le tissu avec du fil conducteur.

ARDUINO NANO

L'Arduino Nano 3.x utilise le CPU ATmega328, tandis que l'Arduino Nano 2.x est bâti autour de l'ATmega168. Il dispose d'un port Mini-B USB. Il est alimenté en 5 volts. Il ne dispose pas de connecteur d'alimentation.

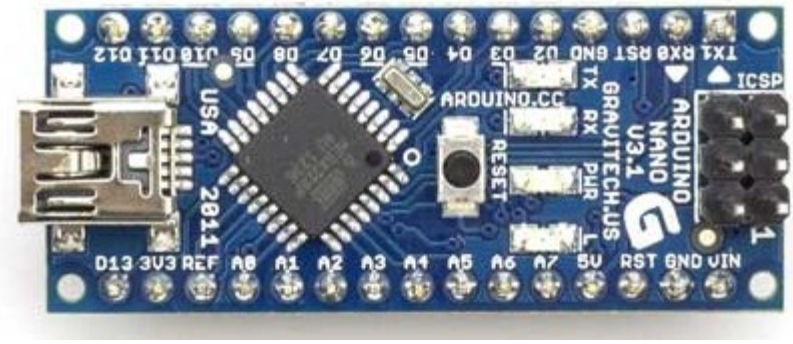


Figure 9 - Arduino Nano

Figure 10 - Câble Mini USB pour arduino nano

Caractéristiques générales

Microcontrôleur	Atmel ATmega168 or ATmega328
Tension de fonctionnement	**5 V**
Tension d'alimentation (recommandé)	7-12 V
Tensions limites	6-20 V
Broches Entrées/Sorties digitales	**14 (dont 6 pour les sorties PWM)**
Broches analogiques	**8**
Courant DC par pin	**40 mA**
Memory Flash	16 KB (ATmega168) ou 32 KB (ATmega328) dont 2 KB utilisés par le bootloader
SRAM	1 KB (ATmega168) ou 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) ou 1 KB (ATmega328)
Vitesse d'horloge	**16 MHz**
Dimensions	0.73" x 1.70"
Longueur	45 mm
Largeur	18 mm
Poids	5 g

Les clones

- Seeeduino
- Seeeduino Film
- Boarduino
- Sippino

Préparation de l'environnement de travail

Installation du logiciel Arduino (IDE et ses bibliothèques)

Il faut installer l'environnement de travail à partir du site consacré à arduino:
http://arduino.cc/en/Main/Software.

Le logiciel est en open source (gratuit) et est disponible pour les trois principales plateformes d'ordinateurs: Windows, Mac OS, Linux. Utiliser celui qui correspond au système d'exploitation de votre ordinateur.
La dernière version 1.6.5 installe aussi le driver qui permet de lier le PC à l'arduino via le câble USB.
Si des problèmes surgissent lors de l'installation, voir
http://arduino.cc/en/Guide/Troubleshooting, pour trouver la solution au problème.

Des guides en ligne sont disponibles pour démarrer correctement avec Arduino:
http://arduino.cc/en/Guide/Windows pour Windows,
http://arduino.cc/en/Guide/MacOSX pour Mac OS X, and
http://www.arduino.cc/playground/Learning/Linux pour Linux.

Une fois l'installation proprement effectuée, il est possible maintenant de lancer l'application arduino, en double-cliquant sur le raccourci de l'application.

Figure 11 - Raccourci arduino

Elle va ouvrir une fenêtre présentant le modèle de programme avec les différentes fonctions de base.

Description de la barre d'outils de l'IDE

Voyons la barre d'outils:

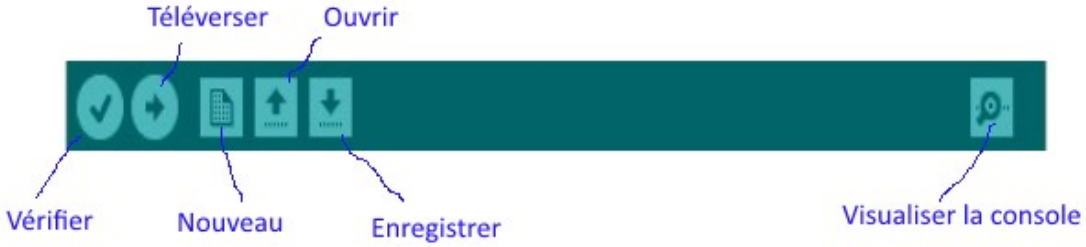


Figure 12 - Barre d'outils de l'IDE arduino

De gauche à droite, voici résumée la fonctionnalité de chaque bouton:
Vérifier :
Permet de vérifier (ou compiler) le programme avant de l'envoyer sur la carte.
Téléverser :
Pour stocker le programme binaire sur la carte et l'exécuter.
Nouveau :

Pour créer un nouveau programme.
Ouvrir :
Pour ouvrir un programme existant
Enregistrer :
Pour sauvegarder le programme
Moniteur série (Visualiser la console):
Ouvrir une fenêtre qui affiche la communication série entre la carte et le PC. Sert surtout pour vérifier l'exécution du programme en affichant des messages. Nous verrons cela en détails plus loin.

Description d'un programme

Un programme arduino est composé en général de 3 sections:

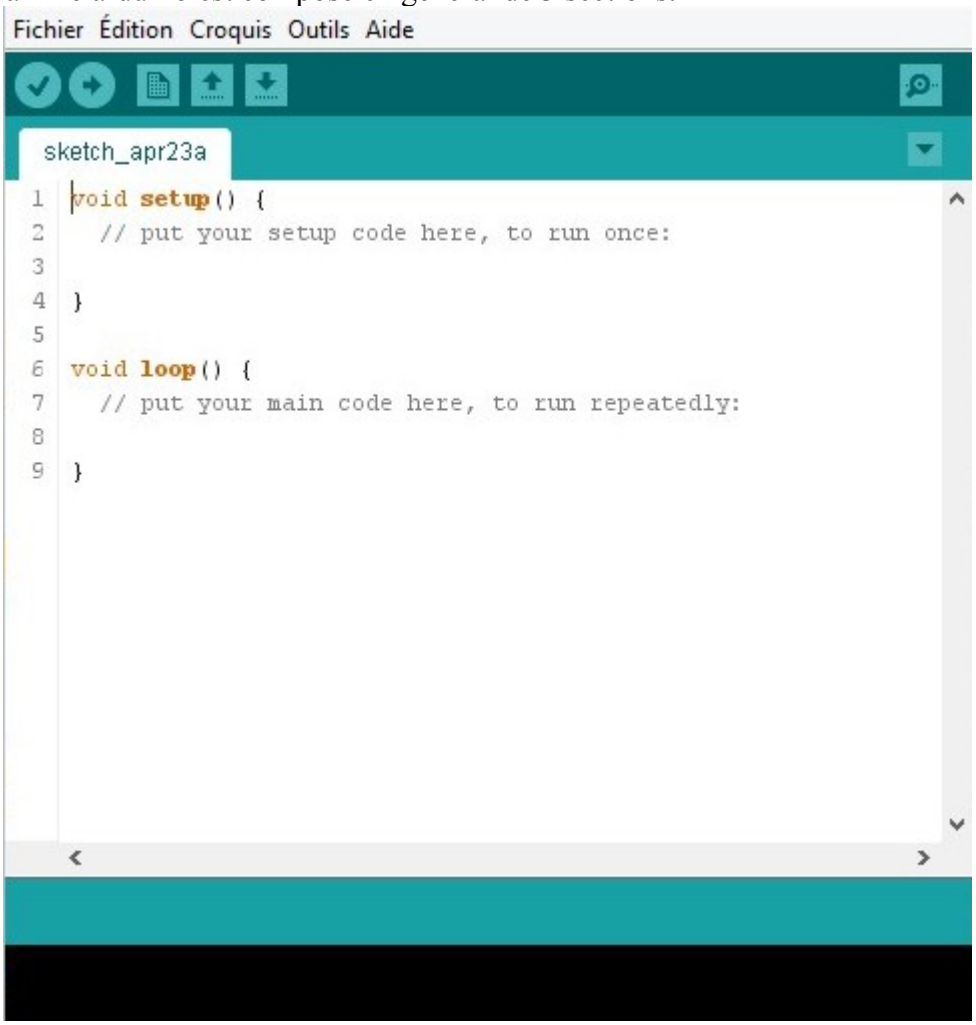


Figure 13 - Exemple de programme (sous IDE)

1- La section générale
Contient en général des déclarations de variables ou de constantes.

2- La section setup
Elle s'écrit comme ceci:

```
void setup() {
        instructions...
}
```

Cette fonction est appelée et exécutée une seule fois lorsque l'arduino est mis sous tension ou bien après un redémarrage à chaud (reset). Elle permet d'initialiser des variables, des bibliothèques.

3- La section loop

Elle s'écrit comme ceci:

```
void loop() {
        instructions...
}
```

Cette fonction exécute les instructions qu'elle contient de façon répétitive comme dans une boucle comme son l'indique.

Exemple:

```
const int buttonPin = 10;
void setup() {
   Serial.begin(9600);
   pinMode(buttonPin, INPUT);
}

void loop() {
   if (digitalRead(buttonPin) == HIGH)
      Serial.write('H');
   else
      Serial.write('L');

  delay(200);
}
```

Si le programme contient des fonctions utilisateur alors il y aura d'autres sections pour les définir, comme dans le programme suivant:

```
/* Declarations (section globale)
...
*/
void setup() {
...
}
void loop() {
        ...
        fonction1();
}
void fct1() {
        ...
}
```

fct1 est une fonction qui est appelée dans la fonction loop().

Description des menus de l'IDE

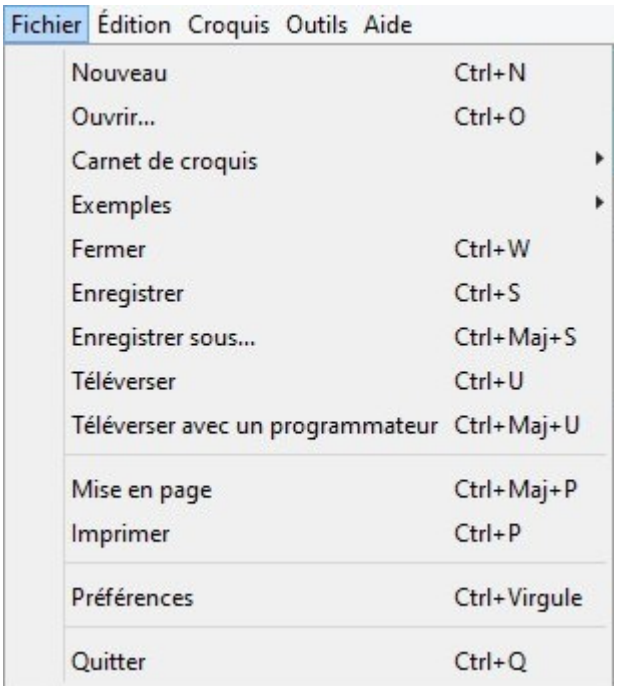


Figure 14 - Menus de l'IDE

Dans ce menu il y a les classiques Nouveau, Ouvrir etc.…

Je dirais que les deux seules choses à retenir sont:
Exemples : Vous permet de charger des programmes d'exemples fournis avec l'IDE et les bibliothèques que vous aurez installées. C'est simple, c'est ici que vous saurez comment fonctionnent vos bibliothèques et que vous trouverez quelques bouts de programmes très intéressants pour vos projets.
Préférences : Simplement pour configurer l'IDE à votre guise.

Il n'y a vraiment pas besoin de s'attarder sur le menu Edition, c'est le classique des classiques sans ajout dont nous aurons besoin, on passe donc au menu Croquis:

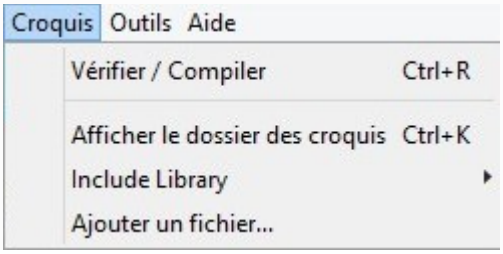


Figure 15 - Le menu Croquis

Ce menu est utile pour 2 choses (vérifier/compiler étant en bouton sur la page, il n'est pas utile de passer par le menu):

Include Library : C'est ici que vous pourrez inclure les bibliothèques que vous aurez installées dans vos programmes.

Ajouter un fichier… : Si vous avez créé une fonction ou si vous incluez un sous-programme à un programme, vous pouvez simplement l'insérer ici.

Et enfin un des menus les plus importants: Outils

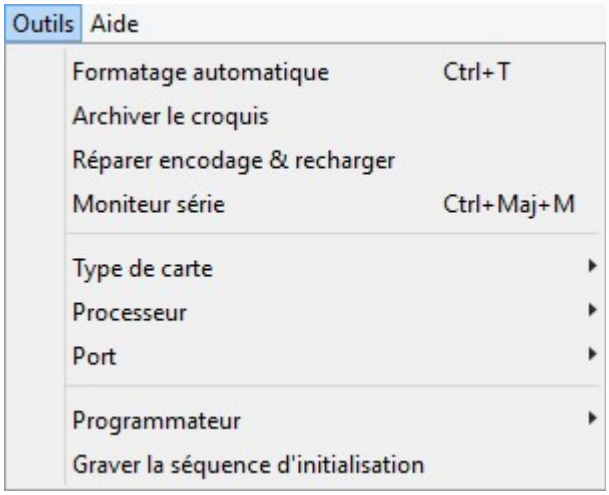


Figure 16 - Le menu Outils

Formatage automatique:
Permet de rajouter les espaces en début de ligne pour rendre votre programme plus lisible.
Archiver le croquis: ?…
Réparer encodage & recharger : ?…
Moniteur série :
Ouvre le moniteur série.
Type de carte :
Permet de choisir le modèle d'arduino à expérimenter.
Processeur :
Certains modèles existent avec des processeurs différents. Il faut choisir le plus adéquat.
Port :
Indique le port série (COM) qui va servir pour le téléversement du programme. Téléversement est le terme utilisé pour signifier l'enregistrement du programme compilé dans la mémoire de l'arduino.
Programmateur et Graver la séquence d'initialisation : Utiliser pour programmer le bootloader. Non recommandé pour les débutants.

Le Moniteur Série

La fenêtre du moniteur (ou console arduino) se présente comme ceci:

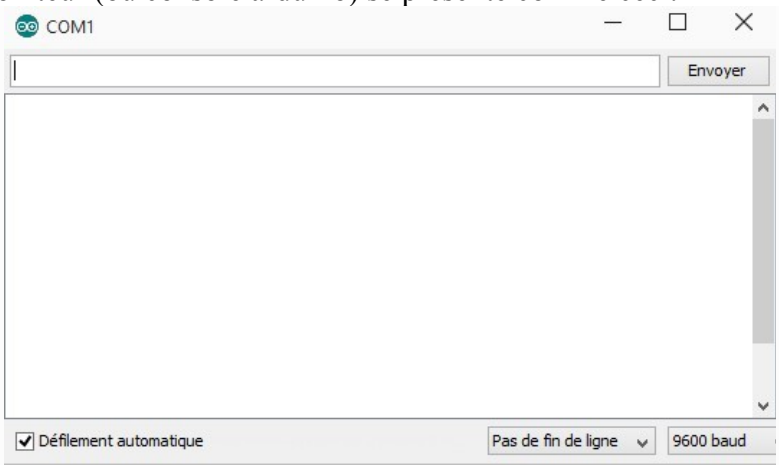


Figure 17 - Le moniteur série

Tout en haut, le port COM est indiqué.

En dessous, une petite zone de texte et le bouton envoyer, pour envoyer des messages ou des nombres à la carte. La donnée rentrée est lue par l'arduino.

En dessous se trouve la fenêtre où s'afficheront toutes les données envoyées par la carte sur le port série. La case défilement automatique permet d'activer ou non le défilement automatique du texte.

9600 bauds : la vitesse de transmission des données sur le port. Cette valeur doit correspondre à celle qui est utilisée dans le programme (voir l'instruction Serial.begin de l'arduino).

Les ports digitaux et les ports analogiques

Dans arduino, il y a les ports digitaux D13 à D2; D1 et D0 étant les ports Tx et Rx pour la transmission sur port série.
Un port digital peut être configuré soit comme entrée pour lire une information qui provient de l'extérieur, soit comme sortie pour commander un circuit électrique externe.
Arduino utilise la fonction pinMode() pour configurer le mode d'entrée ou sortie d'un port.
Exemples:
- pour configurer la pin D10 comme entrée nous écrivons:

```
pinMode(10, INPUT);
```

- pour configurer la pin D9 comme sortie, nous écrivons:

```
pinMode(9, OUTPUT);
```

- on peut aussi utiliser le paramètre INPUT_PULLUP pour utiliser la résistance interne située sur le port digital

L'arduino utilise la fonction digitalWrite() pour placer une tension sur la pin indiquée dand la fonction.
Exemple:
- si nous voulons commander la pin 9 pour envoyer un signal de 5volts (HIGH) ou 0 volt (LOW), nous écrivons:

```
digitalWrite(9, HIGH);
digitalWrite(9, LOW);
```

- si nous voulons connaître la valeur digitale sur la pin 10, nous écrivons:

```
valeur = digitalRead(10);
```

La variable "valeur" doit avoir été déclarée dans le programme.

Un port digital fournit une tension de 5 volts si nous utilisons le paramètre HIGH (ou 1) dans la fonction digitalWrite() ou bien 0 volt si nous utilisons LOW (ou 0).

Pour ce qui est des ports analogiques, le tableau suivant nous renseigne sur les différentes possibilités d'utilisation:

	Analog input	Analog output
Resolution	10 bit (0 through 1023)	8 bit (0 through 255)
Voltage	range 0 through 5 volts	0 through 5 volts
Arduino	pins A0 through A5	Digital pins 3, 5, 6, 9, 10, 11
Arduino Mega	pins A0 through A15	Digital pins 2 through 13

Un port analogique peut sortir une tension comprise entre 0 et 5 volts. Ce sont les pins D3, D5, D6, D9, D10, D11 qui sont les sorties analogiques. Voir la section sur les signaux PWM.
Pour cela il existe la fonction analogWrite(Dx, valeur). *Dx* est le numéro de pin analogique et *valeur* est une valeur comprise entre 0 et 255.
Les entrées analogiques peuvent être utilisées sur les ports A0, A1, A2, A3, A4, A5. La valeur lue est comprise entre 0 et 1023 car la résolution du convertisseur Analogique/Numérique est de 10 bits. Nous utilisons la fonction analogRead(Ax) pour lire un port analogique.

TECHNIQUES DE BASE

Prise de contact: le programme qui fait clignoter une LED

Comment allumer/éteindre une LED

1- Description

La LED est toute indiquée pour commencer à programmer avec l'arduino.
Elle s'allume lorsqu'elle est traversée par un courant d'environ 20 mA. Sinon elle reste éteinte. Le courant qui traverse la LED ne doit pas dépasser la limite indiquée par le constructeur. Les bornes de la LED sont l'anode et la cathode. La tension sur l'anode doit être supérieure à la tension sur la cathode (représentée par une barre sur le symbole de la LED). Pour mieux connaître le fonctionnement des LEDs, aller à l'annexe xxx.

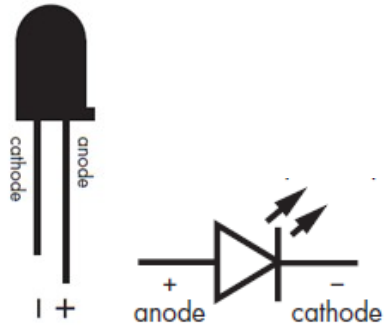


Figure 18 - La LED et son symbole

La pin courte (cathode) est toujours reliée à la tension la plus basse (généralement Gnd) par rapport à l'autre pin qui est plus longue (anode) et qui est reliée au Vcc en général.

2- Connexion

Donc il n'est pas permis de connecter une LED directement sur les bornes d'une source de tension, comme la batterie par exemple. Pour limiter le courant du circuit électrique alimentée par la source de tension, on place une résistance entre 180Ω et 330 Ω en série avec la LED.

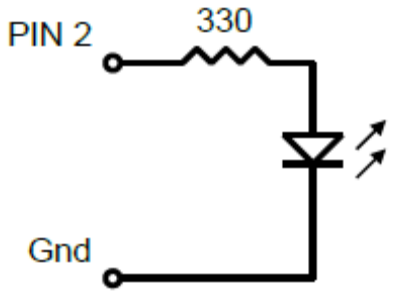


Figure 19 - Connexion d'une LED

On utilise la formule suivante R = (Vs - Vf) / I pour calculer la résistance de limitation de courant d'une LED. Vs est la tension appliquée (5V dans le cas de l'arduino); Vf est la

tension aux bornes de la LED (1.7V dans le cas d'une Led rouge); I vaut 10mA (peut aller entre 5 et 30 mA).
Donc R = 3.3 / 0.010 = 330 Ω.
(3.3 V étant la tension entre les bornes de la résistance).

3- Programme

```
1 void setup() {
2         pinMode(2, OUTPUT);
3 }
4 void loop() {
5         digitalWrite(2, HIGH);
6         delay(1000);
7         digitalWrite(2, LOW);
8         delay(1000);
9 }
```

Description du programme

1	void setup() {	Déclaration de la fonction setup().
2	pinMode(2, OUTPUT);	Utilisation de la fonction pinMode(2, OUTPUT). Ceci sert à positionner la pin 2 de l'arduino comme une pin de sortie (paramètre OUTPUT). On utilise le paramètre INPUT si on veut que la pin soit positionnée comme une pin d'entrée.
3	}	Fin de la fonction setup.
4	void loop() {	Déclaration de la fonction loop().
5	digitalWrite(2, HIGH);	La pin 2 de l'arduino est levée à la valeur 1 logique (HIGH) - correspondant à une tension de 5 volts.
6	delay(1000);	Temporiser pendant 1000 ms (1 sec).
7	digitalWrite(2, LOW);	La pin 2 de l'arduino est baissée à la valeur 0 logique (LOW) - correspondant à une tension de 0 volt.
8	delay(1000);	Temporiser pendant 1000 ms (1 sec).
9	}	Fin de la fonction loop().

Noter l'utilisation de la fonction delay(t) à la ligne 6 et 8. Le paramètre t est exprimé en 1/1000 de secondes (ms). Cette fonction arrête le programme pendant un laps de temps de t millisecondes.
Exemple: delay(1000) arrête le programme pendant 1 sec. On parle de temporisation.

Calcul du courant traversant une LED
Voici résumé dans le tableau suivant comment calculer le courant qui traverse une LED et la résistance de limitation de courant en fonction de la source d'alimentation et du type de LED utilisée:

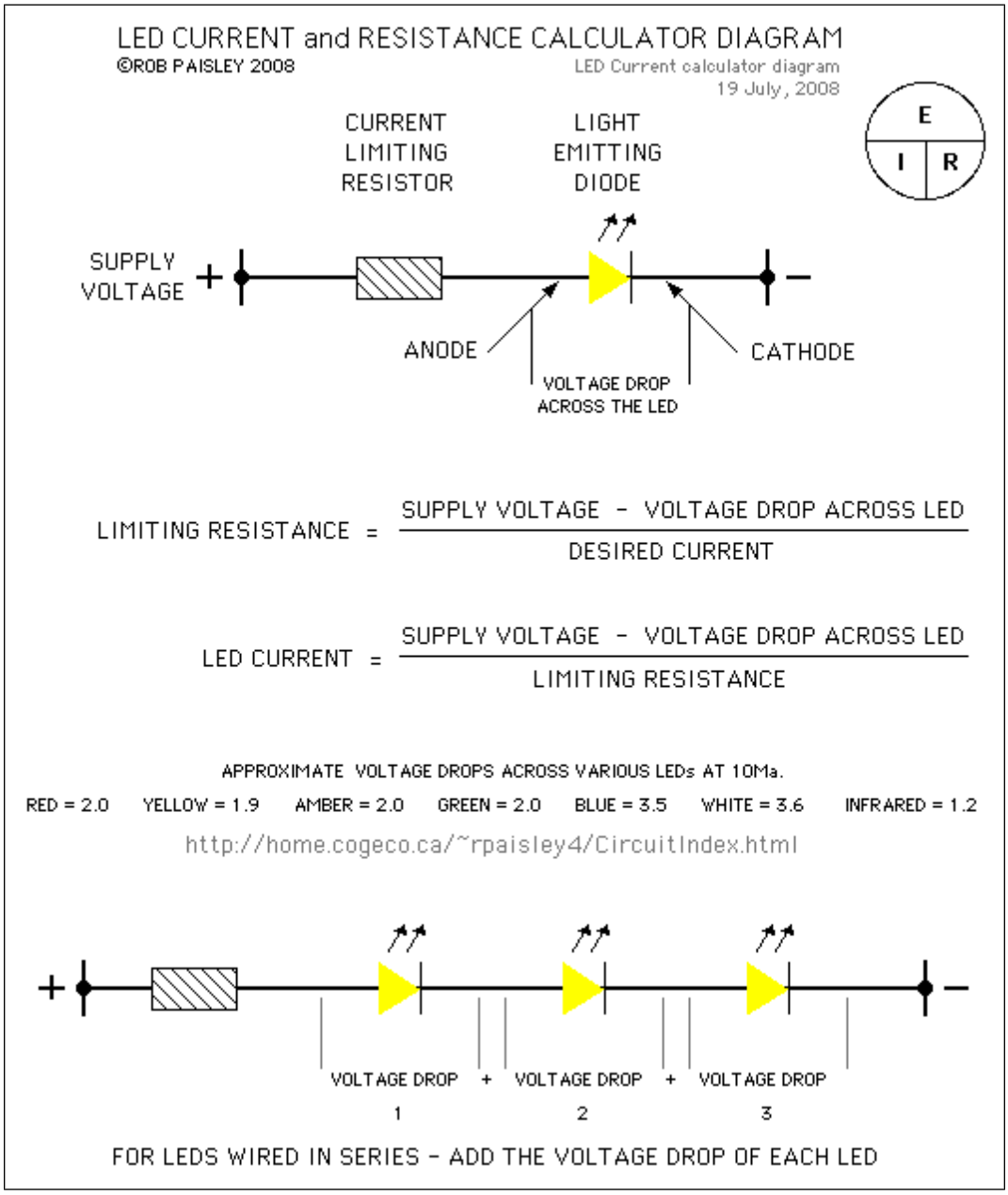


Figure 20 - Diagramme de calcul de courant d'une LED

En général, une LED alimentée en +5volts nécessite une résistance de protection comprise entre 220 ohms et 1K ohms.

Programmation: Description des fonctions Serial.begin et Serial.println

Serial.begin(speed) sert à définir la vitesse de transmission entre l'arduino et le PC. Noter que les données vont transiter entre les deux appareils par le câble USB. La vitesse utilisée par défaut est 9600. Elle se mesure en bauds. Un baud est équivalent à 1 bit par seconde (bps).
Serial.println(paramètre) sert à afficher la valeur du paramètre sur l'écran. Un retour chariot (CRLF) est rajouté pour passer le curseur à la ligne suivante.
Serial.print(paramètre) sert de la même façon mais sans faire passer le curseur à la ligne suivante. Les messages seront placés à la queue-leu-leu.
Exemples:

```
Serial.print("Bienvenue ");
Serial.println("tout le monde");
Serial.println('1');
Serial.println(75);
```

Sur le console on obtient:
Bienvenue tout le monde
1
75

Si on reprend l'exemple sur les LEDs, on aura le programme suivant:

```
void setup() {
        pinMode(2, OUTPUT);
        Serial.begin(9600);
}
void loop() {
        digitalWrite(2, HIGH);
        Serial.println("ON");
        delay(1000);
        digitalWrite(2, LOW);
        Serial.println("OFF");
        delay(1000);
}
```

Note: N'oubliez pas d'ouvrir la console.

Exercices

1- Refaire le programme de clignotement de la LED en utilisant des valeurs différentes de délai. Observez le résultat.

Programmation: Utilisation des variables

Plusieurs types de variables peuvent être utilisées dans arduino:

a) Variable de type int (entier allant de -32768 à 32767): représentée par une case mémoire de 16 bits

exemples:

```
int valeur;          // définit une variable int n'ayant pas de valeur.
int valeur = 150;    // définit une variable int possédant la valeur 150.
```

a) Variable de type unsigned int (entier allant de 0 à 65535): représentée par une case mémoire de 16 bits

b) Variable de type char: représentée par une case mémoire de 8 bits (entre -128 et 127)

exemples:

```
char c;              // définit une variable char n'ayant pas de valeur.
char c = '5';        // définit une variable char possédant la valeur '5'.
char message = "Bienvenue";
// définit une variable char possédant la valeur "Bienvenue".
```

c) Variable de type unsigned char: représentée par une case mémoire de 8 bits (entre 0 et 127)

d) Variable de type byte: représentée par une case mémoire de 8 bits (entre 0 et 255)

exemples:

```
byte b;              // définit une variable byte n'ayant pas de valeur.
byte a = 100;        // définit une variable byte la valeur 100.
```

e) Variable de type float (32 bits): (entre -3.4028235E+38 et 3.4028235E+38)

exemples:

```
float pi = 3.14;     // définit une variable float ayant la valeur mathématique pi.
float F;             // définit une variable float ne possédant pas de valeur.
```

f) Variable de type double(32 bits): (3.4028235E+38 to 3.4028235E+38**)**

exemples:

```
double e = 349.045;  // définit une variable double ayant la valeur mathématique .....
double D;            // définit une variable double ne possédant pas de valeur.
```

Variable type Size Range

g) Variable de type boolean: représentée par une case mémoire de 8 bits (0 ou 1)

h) Variable de type word: représentée par une case mémoire de 16 bits (0 à 65535**)**

i) Variable de type long (de -2,147,483,648 à 2,147,483,647)

j) Variable de type unsigned long (to 4,294,967,295)

k) Variable de type array (tableau)

Un tableau est une collection de données qui sont accessibles au moyen d'un index.

```
int array[] = {valeur0, valeur1, valeur2, …. }
```

Exemple:
int array[] = {180, 250, 1245, …. };

Pour accéder à une valeur dans le tableau, on le fait au moyen de son index. Exemple, pour lire la valeur 180 dans le tableau on fait: array[0], pour 250 on écrit array[1] et ainsi de suite.

Exemple utilisant la collection 'array':

```
byte valeurs[] = {190, 25, 255, 180, 10, 95, 160, 50};
void setup() {
        Serial.begin(9600);
}
void loop() {
        for(int i = 0; i<8; i++) {
                Serial.println(valeurs[i]);
                delay(2000);
        }
}
```

Question: que fait ce programme?

Exemple

Voici un programme arduino qui calcule les 4 opérations arithmétiques entre 2 paramètres a et b. Le résultat est affiché à l'écran.

```
void setup() {
---
}
void loop() {
---
}
```

Pour l'exercice 5 suivant, on devra utiliser les fonctions sinus et cosinus, comme suit:

```
#include  <math.h>          // à inclure dans le programme puis utiliser les fonctions sin et
                            // cos de la manière suivante:
...
sin(double angle)           // cette  fonction retourne le sin de x (x exprimé en radians)
```

Exemple:

```
double val;
val = sin(2*3.14)           // sinus de 2pi; val est
...
cos(double angle)           // cette  fonction retourne le cos of x (x exprimé en radians)
```

Exemple:

```
val = cos(3.14/2)           // cos de pi/2
```

Exercices

Pour tous les exercices, utiliser le moniteur de l'arduino pour vérifier les résultats.

1. Faire clignoter la LED avec une fréquence de 2 Hz.
2. Connecter l'arduino à 2 LEDs (une rouge et une verte). Ne pas oublier les résistances. Faites clignoter les 2 LEDs sachant que lorsque la rouge est allumée alors la verte est éteinte et vice-versa. Choisir une fréquence de 1 Hz.
3. Connecter l'arduino à 5 LEDs. Ne pas oublier les résistances. Faites clignoter les LEDs chacune à son tour. L1 s'allume puis s'éteint, ensuite autour de L2 et ainsi de suite. Choisir une fréquence de 1 Hz.
4. Refaire l'exercice de votre choix en faisant varier la fréquence. À chaque fois, utiliser un délai différent (500, 100, 50). Qu'observe-t-on?
5. Faire le programme qui calcule
    - le sinus et le cos d'un angle pour les angles suivants: 0, 10, 20 et 360 degrés. Rappel: 2*pi = 360 degrés.
    - la tangente d'un angle pour angles suivants: 0, 10, 20 et 360 degrés.

La communication avec l'environnement digital extérieur

L'arduino est capable d'interagir avec son environnement extérieur, composé de composants plus ou moins complexes dont font partie toute une panoplie de senseurs, actuateurs (relais, ...), réseaux avec fils (wired network) et sans fil (wireless network).

Contrôler un circuit avec digitalWrite()

1- Description

Nous avons vu comment un circuit simple contenant une ou plusieurs LEDs peut être commandé en utilisant la fonction digitalWrite(). Voir paragraphe xxx.
La fonction digitalWrite(pin, val) positionne la pin en question à une valeur de tension qui prend soit la valeur 5 volts (si val = HIGH) ou 0 volt (si val = LOW).
Cette fonction agit seulement sur les pins digitales qui sont:
2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 et 13.

2- Connexions

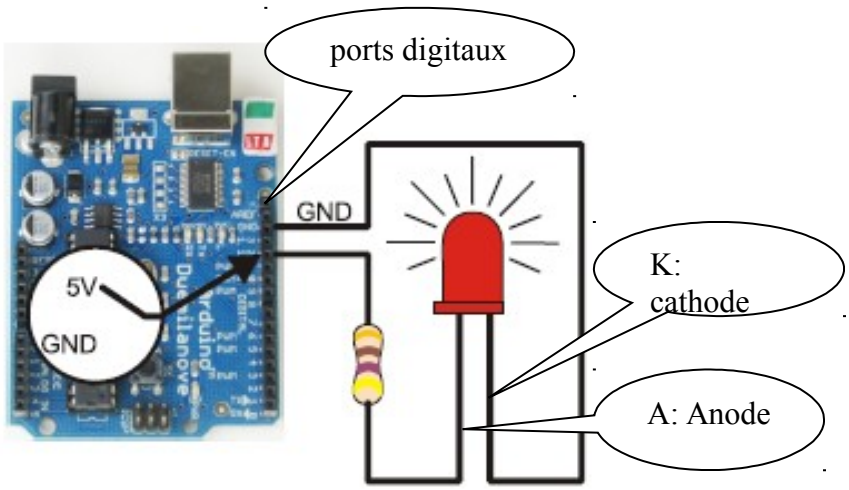


Figure 21 - Mise en 5V d'une pin avec digitalWrite()

L'illumination de la LED se fait en injectant une tension de 5 volts sur la pin de l'arduino qui va alimenter l'anode de la LED, sachant que la cathode de la LED est mise à la masse. Ceci s'effectue en utilisant la fonction digitalWrite, après avoir configurer la pin comme port de sortie, comme ceci:

```
pinMode(pin, OUTPUT);
digitalWrite(pin, HIGH);       // Mise à 1 (led allumée)
```

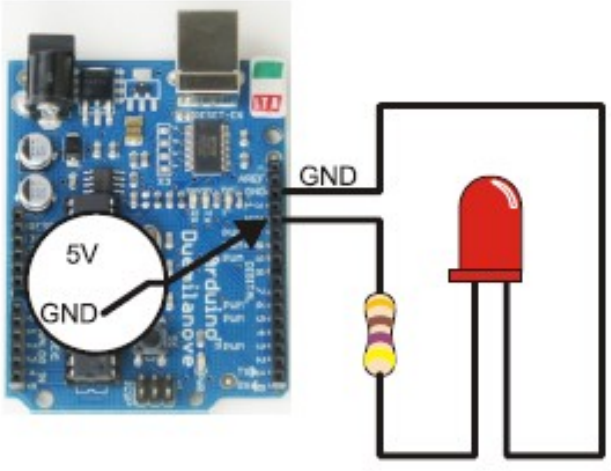


Figure 22 - Mise à la masse d'une pin avec digitalWrite()

L'extinction de la LED se fait en injectant une tension de 0 volt sur la pin de l'arduino qui va alimenter l'anode de la LED, sachant que la cathode de la LED est mise à la masse. Ceci s'effectue en utilisant la fonction digitalWrite, après avoir configurer la pin comme port de sortie, comme ceci:

```
pinMode(pin, OUTPUT);
digitalWrite(pin, LOW);        // Mise à 0 (led éteinte)
```

3- Programme

1- Mise à 1 (5 volts) d'une sortie

```
void setup() {
        pinMode(13, OUTPUT);
}
void loop() {
        digitalWrite(13, HIGH);
}
```

2- Mise à la masse (0 volt) d'une sortie

```
void setup() {
        pinMode(13, OUTPUT);
}
void loop() {
        digitalWrite(13, LOW);
}
```

Capturer de l'information digitale avec digitalRead()

1- Description

Le circuit simple dont nous voulons récupérer l'information est constitué d'un interrupteur (switch). L'interrupteur sert à fermer ou à ouvrir un circuit électrique.

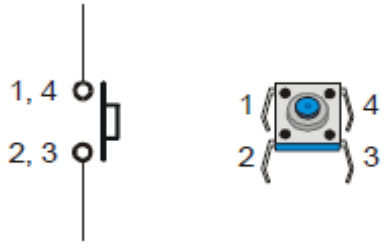


Figure 23 - Exemple d'un switch

Les états classiques d'un bouton-poussoir sont:
- NO (Normalement Ouvert)
- NC (Normalement Fermé)

Figure 24 - Bouton-poussoir Normalement fermé

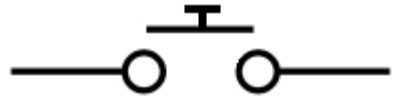


Figure 25 - Bouton-poussoir Normalement Ouvert

2- Connexions

Connexion pour l'allumage d'une LED sans arduino:

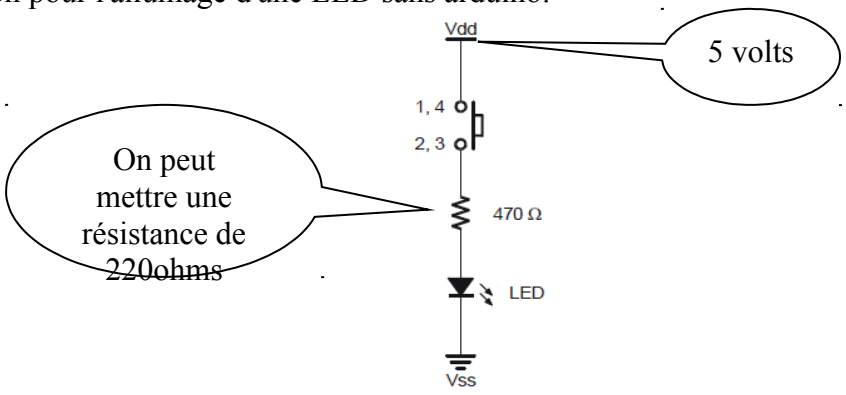


Figure 26 - Connexion d'un bouton-poussoir

Voir le schéma de connexion d'un interrupteur avec arduino:

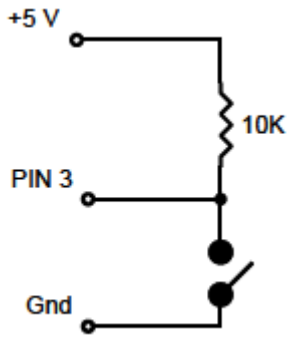


Figure 27 - Connexion avec un interrupteur (switch)

a) Quand l'interrupteur est à l'état ouvert, on dit que le circuit est ouvert. Il n'y a donc pas de lien entre les bornes de l'interrupteur.
b) Quand l'interrupteur est à l'état fermé, on dit que le circuit est fermé. Il y a donc un lien entre les bornes de l'interrupteur.

L'interrupteur est connecté par une borne à la masse du circuit électrique. L'autre borne est connectée à la pin 3 de l'arduino. Cette borne est aussi connectée au pôle positif de la source d'alimentation à travers une résistance. On va pendre une résistance de 10 kΩ.

Pour récupérer l'état d'un interrupteur, l'arduino utilise la fonction digitalRead(pinNumber). Le paramètre 'pinNumber' sert à désigner quelle pin de l'arduino est utilisée.
Exemple: digitalRead(3) permet de lire l'information sur la pin 3. La fonction digitalRead() retourne une valeur qui représente l'état de la pin (1 ou 0, HIGH ou LOW).
Avant d'utiliser la fonction digitalRead(), on informe l'arduino que la pin en question est une pin d'entrée, avec la fonction pinMode(pinNumber, ioMode).
Exemple: pinMode(3, INPUT).

3- Programme

```
void setup()
{
      pinMode(3, INPUT);
      Serial.begin(9600);
}
void loop()
{
      // on lit ce qui se présente sur la pin 3
      // on met le résultat dans la variable 'interrupt'
      int interrupt = digitalRead(3);
      // on affiche la valeur lue
      Serial.println(interrupt);
      // on fait une attente de 250 ms par exemple
      delay(250);
}
```

Connexion du switch sans le rajout de la résistance

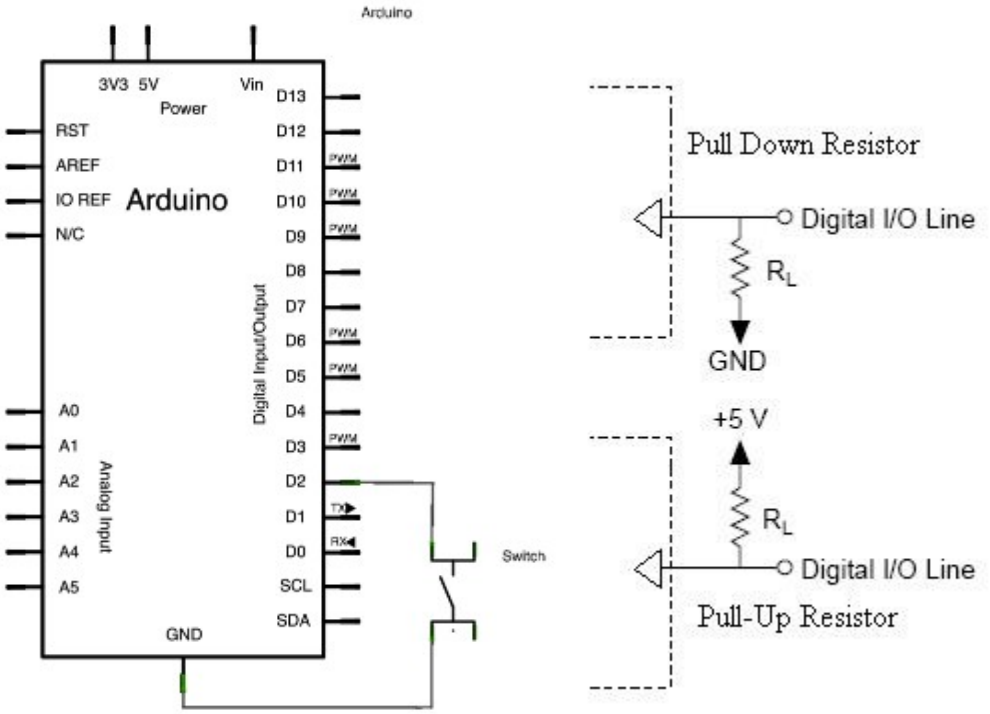


Figure 28 - Connexion d'un switch sans utiliser de résistance

Dans ce cas on va utiliser le paramètre INPUT_PULLUP dans la fonction pinMode().
On aurait le programme suivant:

```
void setup() {
	// start serial connection
	Serial.begin(9600);
	// configure pin2 as an input and enable the internal pull-up resistor
	pinMode(2, INPUT_PULLUP);
	pinMode(13, OUTPUT);
}

void loop() {
	// read the pushbutton value into a variable
	 int sensorVal = digitalRead(2);
	// print out the value of the pushbutton
	Serial.println(sensorVal);

	// Keep in mind the pullup means the pushbutton's logic is inverted. It goes HIGH
	// when it's open, and LOW when it's pressed. Turn on pin 13 when the button's
	// pressed, and off when it's not:
	 if (sensorVal == HIGH) {
		digitalWrite(13, LOW);
	} else  {
		digitalWrite(13, HIGH);
		}
	}
```

Exercices

1- Faire le montage électrique pour allumer une LED lorsqu'on ferme un interrupteur. La LED s'éteint lorsque l'interrupteur est ouvert.
2- Faire le montage électrique pour allumer une LED pendant un certain délai puis l'éteindre lorsqu'on ferme un interrupteur. La LED flashe avec une fréquence de 10 Hz lorsque l'interrupteur est ouvert.
3- Faire le montage à l'aide de 2 interrupteurs et une LED pour réaliser les fonctions logiques OU, ET, OU-X (Ou exclusif). Faire le programme pour chaque fonction. Utiliser le moniteur série de l'arduino pour déboguer les fonctions (utiliser Serial.begin et Serial.println).
4- Faire le montage d'un circuit électrique qui utilise un bouton-poussoir pour allumer une LED pendant 5 secondes (le bouton-poussoir est appuyé puis relâché).
5- Réécrire le programme du switch pour afficher le message "Interrupteur ouvert" ou "Interrupteur fermé" selon le cas.

Programmation: Utilisation de la boucle répétitive

Arduino dispose d'un mécanisme qui consiste à répéter une tâche pendant un certain nombre de fois. Il y a 3 moyens de réaliser la boucle répétitive:

a) 1er mécanisme - for

for (instruction1; condition; instruction2) { ... }

Exemple:

```
for (int i=0; i<10; i++) {
        instructions;
        .....
}
```

Note: N'oublier pas les fonctions void setup() et void loop() dans votre programme.

Cette boucle 'for' permet d'exécuter la série d' instructions entre accolades tant que l'expression 'condition' est vraie, donc 10 fois de suite dans l'exemple. La variable i est utilisée ici comme un compteur. Il va évoluer de 0 à 9. Lorsque la variable i atteint la valeur 10, la boucle est terminée et le programme reprend après l'accolade '}'.

b) 2ème mécanisme - while

while (expression) { ... }

Exemple:

```
int i = 5;
while (i>0) {
        // instructions;
        Serial.println(i);
        i = i - 1;
}
```

Cette boucle 'while' permet d'exécuter la série d' instructions entre accolades tant que la condition est vraie, c'est-à-dire 5 fois de suite dans l'exemple.

c) 3ème mécanisme - do/while

```
do {
  série d'actions...
} while (condition);
```

Exemple:

```
int i = 0;
do {
        actions;
         i = i + 1;
} while (i<10) ;
```

Cette boucle do ... while permet d'exécuter la série d' instructions entre accolades tant que la condition est vraie, c'est-à-dire 10 fois de suite dans l'exemple.

Exemple de programme
.....

Exercices

1- Faire le programme qui affiche la série des nombres pairs de 0 à 20 (inclus). Utiliser les fonctions Serial. Utiliser chacun des trois mécanismes de boucle répétitive (for, while et do/while).
2- Utiliser le montage de l'exercice 4 du paragraphe précédent pour faire flasher la LED pendant 10 secondes, avec une fréquence de flash de 1 Hz.
3- Faire un montage qui permet de générer une valeur de dé (entre 1 et 6) lorsqu'on appuie un bouton-poussoir. Utiliser la fonction random(). Se référer au site https://www.arduino.cc/en/Reference/Random
4- Nous voulons utiliser un buzzer qui vibre pendant quelques secondes (avec la séquence suivante: 2 bips, 1 silence, 2 bips) lorsqu'on appuie sur un bouton-poussoir.
Un bip dure 500 ms.
Avant de réaliser le montage électronique et le programme associé, faisons une description simple d'un buzzer.
5- Faire le montage électrique à base d'arduino pour réaliser le programme suivant:

```
/*
For Loop Iteration
Demonstrates the use of a for() loop.
Lights multiple LEDs in sequence, then in reverse.
The circuit:
* LEDs from pins 2 through 7 to ground created 2006 by David
A. Mellis modified 30 Aug 2011 by Tom Igoe

This example code is in the public domain.
http://www.arduino.cc/en/Tutorial/ForLoop
*/

const int timer = 100;
int Pin;
void setup() {
// use a for loop to initialize each pin as an output:
     for (Pin = 2; Pin < 8; Pin++) {
          pinMode(Pin, OUTPUT);
     }
}

void loop() {
     // loop from the lowest pin to the highest:
     for (Pin = 2; Pin < 8; Pin ++) {
          // turn the pin on:
          digitalWrite(Pin, HIGH);
          delay(timer);
```

```
          // turn the pin off:
          digitalWrite(Pin, LOW);
          delay(timer);
     }

     // loop from the highest pin to the lowest:
     for (Pin = 7; Pin >= 2; Pin --) {
          // turn the pin on:
          digitalWrite(Pin, HIGH);
          delay(timer);
          // turn the pin off:
          digitalWrite(Pin, LOW);
          delay(timer);
     }
}
```

Contrôler un buzzer

1- Description

Le buzzer est utilisé pour créer un son.
Voici quelques unes de ses caractéristiques:

- Fréquence de résonnance: 2300 (+/- 300 Hz)
- Tension: 5V (entre 4 et 8V)
- Courant: 30 ma
- Son minimum perceptible à 10 cm: : 85 dB

Figure 29 - Différentes formes de buzzer

2- Connexions

Voici un exemple de connexion d'un buzzer avec l'arduino:

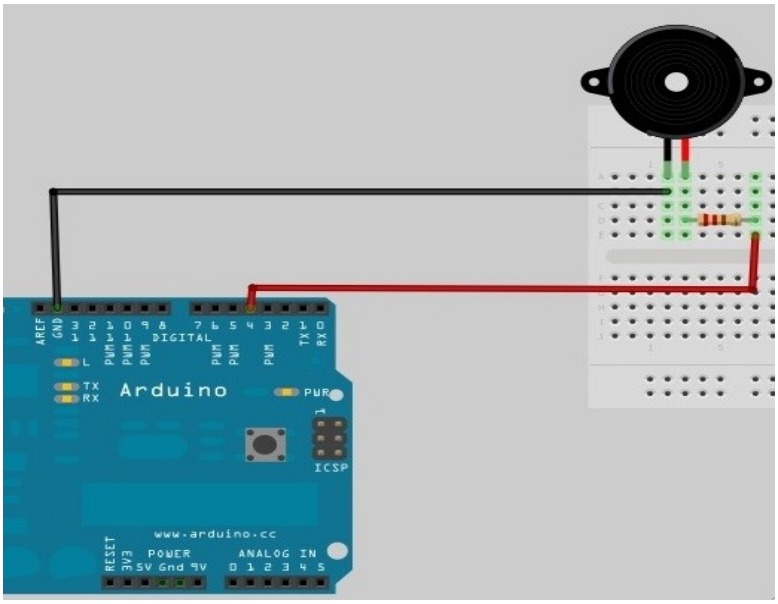


Figure 30 - Connexion d'un buzzer

3- Programme d'essai

```
const int pinBuzzer = 4;        // Connecter le buzzer à la pin 4 de l'arduino
void setup() {
        pinMode(pinBuzzer, OUTPUT);
}
```

```
void loop() {
	unsigned int i;
	while(1) {
		for(i=0; i<80; i++) {
			digitalWrite(pinBuzzer, HIGH);
			delay(1);
			digitalWrite(pinBuzzer, LOW);
			delay(1); //ms
		}
		for(i=0; i<500; i++) {
			digitalWrite(pinBuzzer, HIGH);
			delay(2);
			digitalWrite(pinBuzzer, LOW);
			delay(2);
		}
	}
}
```

Contrôler un joystick

1- Description

Le joystick à 2 axes X-Y est utilisé pour contrôler la position ou le mouvement d'un objet. Il dispose d'un potentiomètre pour chacun des 2 axes. Lorsqu'il est relâché, il retourne toujours à sa position centrale qui est sa position de repos.

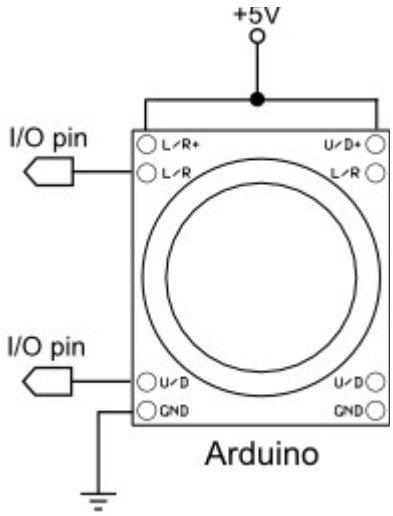


Figure 31 - Une représentation de joystick

2- Connexions

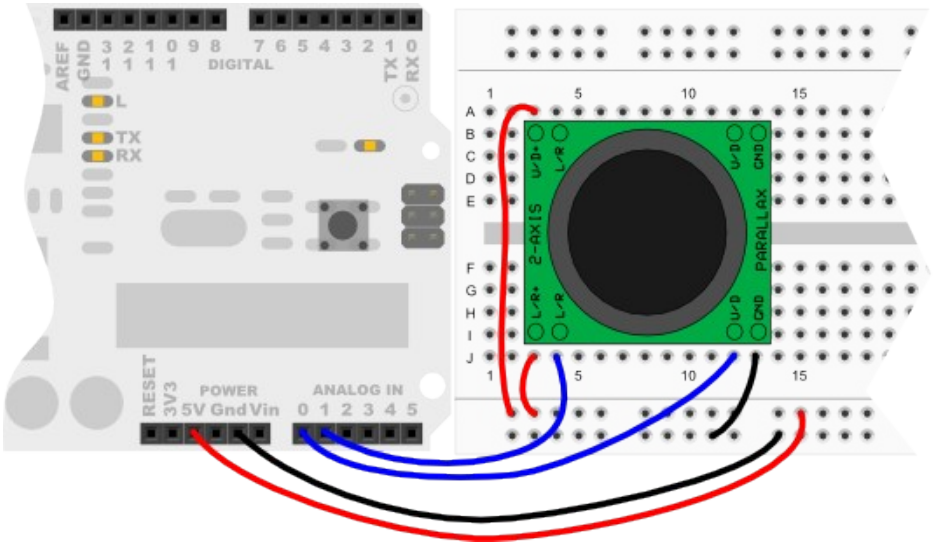


Figure 32 - Connexion avec un joystick

3- Programme

```
int Up_Down = 0;            // UP/DOWN binary value
int Left_Riht = 0;          // LEFT/RIGHT binary value

void setup() {
  Serial.begin(9600);
}
```

```
void loop() {
  Up_Down = analogRead(A0);
  Left_Riht = analogRead(A1);

  Serial.print("UD = ");
  Serial.print(Up_Down, DEC);
  Serial.print(", LR = ");
  Serial.println(Left_Riht, DEC);
  delay(200);
}
```

Contrôler un PIR - détecteur de mouvement

1- Description

Un capteur PIR (Passive Infrared sensor de type HC-SR501) mesure les radiations de lumière émises par les objets dans le champ d'activité du capteur. Ils sont utilisés pour détecter un mouvement.

Permet de détecter le passage d'une personne par exemple, la sensibilité du capteur se fait directement sur le module avec un petit potentiomètre.

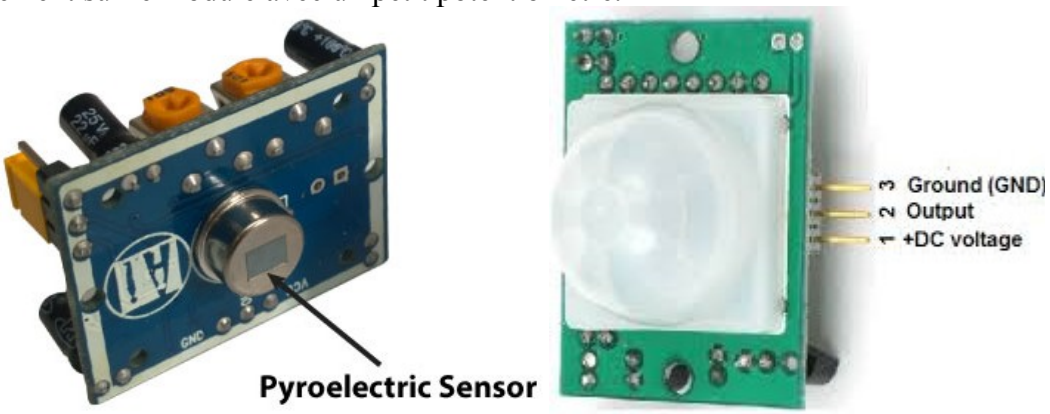


Figure 33 - Un PIR

Remarque: il faut toujours vérifier le pinout du circuit.

Il est aussi possible de régler un retard et de tenir compte aussi de la luminosité ou d'autres critères (ré-enclenchement par exemple).

Il est possible aussi de brancher directement ce capteur sur un relais sans nécessité d'utiliser une carte Arduino ou un Raspberry Pi (fonctionne comme un capteur de mouvement avec lumière, il faut l'alimenter en 5V.)

2- Connexions

Deux broches du PIR sont utilisées pour alimenter le capteur.
La 3ème broche est la broche de donnée qui fournit le signal de présence ou d'absence de mouvement.
Ici nous récupérons le signal 'data' par la pin 2 de l'arduino.

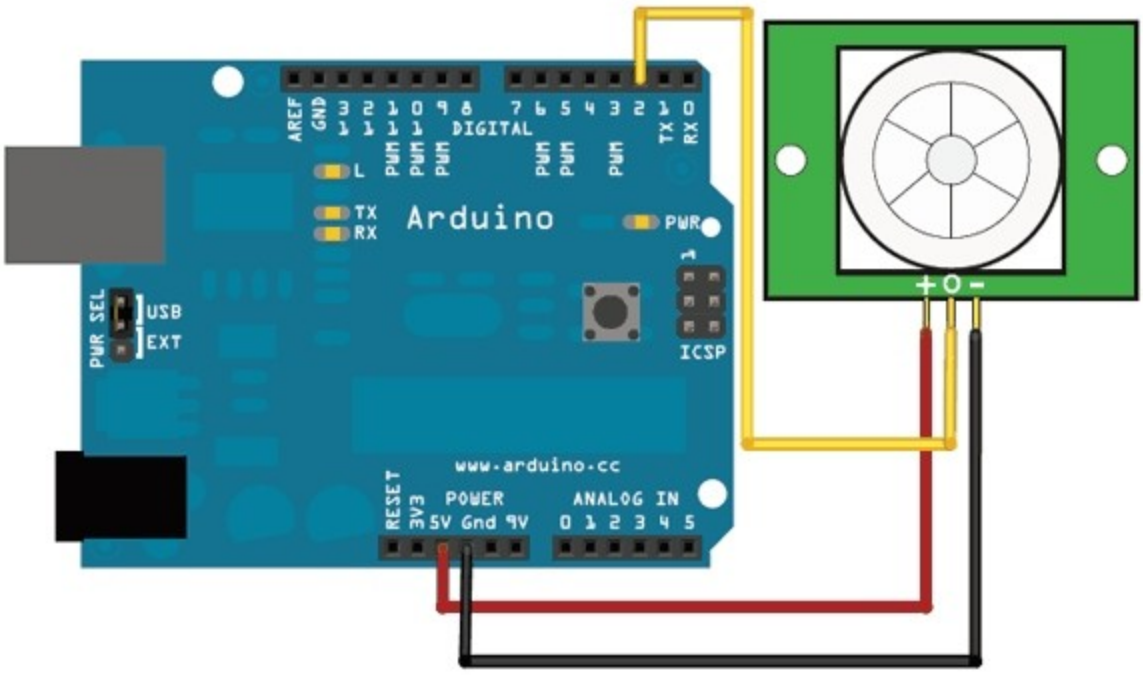


Figure 34 - Connexion à un PIR

3- Programme

```
void setup() {
  pinMode(2, INPUT);
  Serial.begin(9600);
  Serial.println("Alerte ...");
  delay(20000);
}

void loop() {
  int presence = 0;
  Serial.print("PIN 2 = ");
  presence = digitalRead(2);                    // pin 2 : PIR data
  Serial.println(presence, DEC);
  delay(200);
}
```

Exercices

1- Écrire ce code. Vérifier-le et modifier-le pour fonctionner correctement.

```
int inputPin = 3;         // choose the input pin (for PIR sensor)
int pirState = LOW;   // we start, assuming no motion detected
int val = 0;                // variable for reading the pin status

void setup() {
          pinMode(inputPin, INPUT);     // declare sensor as input
          Serial.begin(9600);
}
```

```
void loop() {
        val = digitalRead(inputPin);  // read input value
        if (val == LOW) {           // check if the input is LOW
                if (pirState == LOW) {
                        // we have just turned on
                        Serial.println("Motion detected!");
                        // We only want to print on the output change, not state
                        pirState = HIGH;
                }
        } else  {
                if (pirState == HIGH) {
                        // we have just turned of
                        Serial.println("Motion ended!");
                        // We only want to print on the output change, not state
                        pirState = LOW;
                }
        }
}
```

2- Lire et commenter le programme suivant. Tester-le.

```
int motion_1 = 2;
int light_1 = 13;
void setup() {
  pinMode (motion_1,INPUT);
  pinMode (light_1, OUTPUT);
}
void loop () {
  digitalWrite (light_1,LOW);
  delay(1000); // laisser le capteur se stabiliser avant de prendre une mesure
  int sensor_1 = digitalRead(motion_1);
  if (sensor_1 == LOW) {
    digitalWrite(light_1,HIGH);
    delay(500);
    digitalWrite(light_1,LOW);
    delay(500);
  }
}
```

Capturer l'intensité lumineuse

1- Description

...

2- Connexions

3- Programme

```
void setup() {
---
---
}
void loop() {
---
---
}
```

Projets

1- Faire un montage pour réaliser un générateur de morse en utilisant arduino et quelques composants tels que LEDs et haut-parleur.
Pour vérifier taper un texte sur la console de l'arduino. Le montage doit générer les bips sonores pour chaque caractère du texte en question. Pour faire la correspondance entre un caractère et le code Morse associé, lire la section XXX dans l'annexe YYY.
2- Faire un montage à base d'arduino et d'un convertisseur N/A pour construire un générateur de signaux à fréquence variable (signal carré, triangulaire, sinusoïdal). Utiliser le circuit AD5932 ou AD9850. Voir datasheets.

Créer des sons (avec un crystal piézo-électrique)

1- Description

Le crystal piézo-électrique produit du son (via des vibrations) lorsqu'il est parcouru par un courant alternatif ou par un signal de forme carrée (entre 20Hz à 20kHz). Le piézo-électrique va vibrer et créer ainsi des sons.

Figure 35 - module piézo-électrique

2- Connexions

3- Programme d'essai

1) un premier programme:

```
// piezo_beep.ino - beep on a given frequency with a piezo speaker
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

int speakerPin = 10;

void wave(int pin, float frequency, int duration)
{
  float period=1/frequency*1000*1000; // microseconds (us)
  long int startTime = millis();
  while(millis()- startTime < duration) {
    digitalWrite(pin, HIGH);
    delayMicroseconds(period/2);
    digitalWrite(pin, LOW);
    delayMicroseconds(period/2);
  }
}

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

void loop()
{
  wave(speakerPin, 440, 500);
  delay(500);
}
```

2) Un deuxième programme:

```
/*
 Piezo
 This example shows how to run a Piezo Buzzer on pin 9
 using the analogWrite() function.
 It beeps 3 times fast at startup, waits a second then beeps continuously
 at a slower pace
 */

void setup()  {
  // declare pin 9 to be an output:
  pinMode(9, OUTPUT);
  beep(50);
  beep(50);
  beep(50);
  delay(1000);
}

void loop()  {
  beep(200);
}

void beep(unsigned char delayms){
  analogWrite(9, 20);      // Almost any value can be used except 0 and 255
                           // experiment to get the best tone
  delay(delayms);          // wait for a delay ms
  analogWrite(9, 0);       // 0 turns it off
  delay(delayms);          // wait for a delay ms
}
```

3) troisième essai

```
// piezo_alarmtone.ino
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

int speakerPin = 10;

void wave(int pin, float frequency, int duration)     // <1>
{
  float period=1 / frequency * 1000 * 1000; // microseconds (us)
  long int startTime=millis();
  while(millis()- startTime < duration) {
    digitalWrite(pin, HIGH);
    delayMicroseconds(period/2);
    digitalWrite(pin, LOW);
    delayMicroseconds(period/2);
  }
}

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

void loop()
{
  wave(speakerPin, 440, 40);   // <2>
  delay(25);
  wave(speakerPin, 300, 20);
  wave(speakerPin, 540, 40);
```

```
  delay(25);
  wave(speakerPin, 440, 20);
  wave(speakerPin, 640, 40);
  delay(25);
  wave(speakerPin, 540, 20);
}
```

4) Déclencher une alarme

Le but du programme est de déclencher une alarme.

```
// piezo_alarmtone.ino
// (c) BotBook.com - Karvinen, Valtokari

int speakerPin = 10;

void wave(int pin, float frequency, int duration)
{
  float period = 1 / frequency * 1000 * 1000; // microseconds (us)
  long int startTime = millis();
  while(millis()- startTime < duration) {
    digitalWrite(pin, HIGH);
    delayMicroseconds(period/2);
    digitalWrite(pin, LOW);
    delayMicroseconds(period/2);
  }
}

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

void loop() {
  wave(speakerPin, 440, 40);
  delay(25);
  wave(speakerPin, 300, 20);
  wave(speakerPin, 540, 40);
  delay(25);
  wave(speakerPin, 440, 20);
  wave(speakerPin, 640, 40);
  delay(25);
  wave(speakerPin, 540, 20);
}
```

5) Combiner l'utilisation du piézo et d'un capteur IR

Le but est de déclencher une alarme lorsqu'un switch IR détecte une mauvaise posture (devant un écran par exemple).

```
// posture_alarm.ino - sound an alarm when IR switch detects bad posture
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

int speakerPin = 10;
const int sensorPin = 2;
int switchState = 0;

void wave(int pin, float frequency, int duration) {
  float period=1/frequency*1000*1000; // microseconds (us)
  long int startTime=millis();
  while(millis()-startTime < duration) {
```

```
    digitalWrite(pin, HIGH);
    delayMicroseconds(period/2);
    digitalWrite(pin, LOW);
    delayMicroseconds(period/2);
  }
}

void alarm() {
  wave(speakerPin, 440, 40);
  delay(25);
  wave(speakerPin, 300, 20);
  wave(speakerPin, 540, 40);
  delay(25);
  wave(speakerPin, 440, 20);
  wave(speakerPin, 640, 40);
  delay(25);
  wave(speakerPin, 540, 20);
}

void setup() {
  pinMode(speakerPin, OUTPUT);
  Serial.begin(115200);
  pinMode(sensorPin, INPUT);
}

void loop() {
  switchState = digitalRead(sensorPin);
  Serial.println(switchState,BIN);
  if (switchState==0) {
    alarm();
  }
  delay(10);
}
```

Utiliser un suiveur de ligne

1- Description

2- Connexion

3- Programme d'essai

Afficher un message si on est aligné sur la ligne.

```
// line_sensor.ino - afficher un message si l'objet est sur la ligne
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int sensorPin = 2;
const int ledPin = 13;
int lineFound = -1;

void setup() {
  Serial.begin(115200);
  pinMode(sensorPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  lineFound = digitalRead(sensorPin);
  if(lineFound == HIGH) {
    Serial.println("Sensor is on the line");
    digitalWrite(ledPin, HIGH);
  } else {
    Serial.println("Sensor is off the line");
    digitalWrite(ledPin, LOW);
  }
  delay(10);
}
```

Détecter un obstacle par lumière infrarouge

1- Description

Les cellules infrarouges sont utilisées pour créer une barrière lumineuse pour détecter son franchissement. Le dispositif comprend une Led émettrice et un phototransistor récepteur. La lumière utilisée est de l'infrarouge, donc invisible. C'est ce qui est utilisé dans les télécommandes à infrarouge pour contrôler les appareils, comme par exemple le téléviseur.

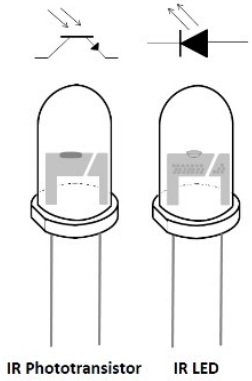


Figure 36 - Phototransistor IR et LED infrarouge

Le signal IR est émis par la LED émettrice puis reçu par le phototransistor

2- Connexions

La LED IR se connecte comme une LED ordinaire sur la tension positive (+5V) à travers une résistance de 330 Ω par exemple.
Le phototransistor IR agit comme un transistor ordinaire. Sa base est activée par la lumière reçue.

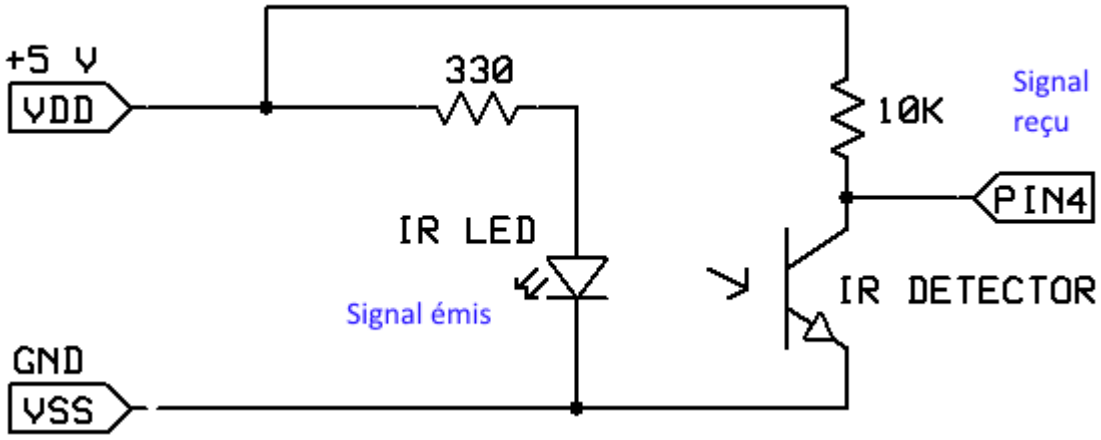


Figure 37 - Connexion avec une barrière IR

L'arduino est branché au circuit par la pin D4.

3- Programme

```
void setup() {
  pinMode(4,INPUT);
  Serial.begin(9600);
```

```
}
void loop() {
  int IR_detect = digitalRead(4);
  Serial.println(IR_detect);
  delay(100);
}
```

La communication avec l'environnement analogique extérieur

Lire une information analogique à partir d'un potentiomètre

1- Description

Le potentiomètre peut se présenter sous différentes formes. Il est utilisé pour fournir une valeur de résistance variable, en général, pour effectuer des réglages dans un circuit donné. Voici les différentes représentations d'un potentiomètre:

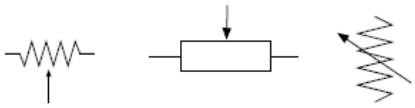


Figure 38 - Représentations symboliques d'un potentiomètre

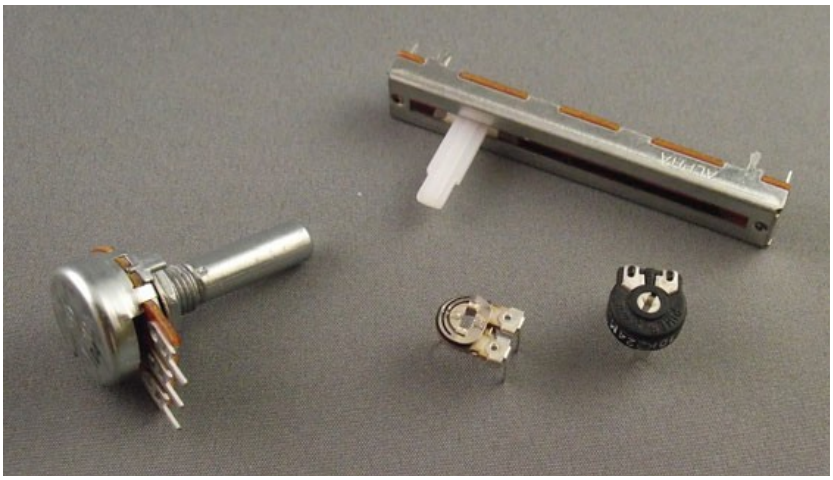


Figure 39 - Différents types de potentiomètres

La valeur d'un potentiomètre fournit une valeur qui peut varier entre 0 et Max ohms, Max étant la valeur maximale du potentiomètre.

2- Connexions

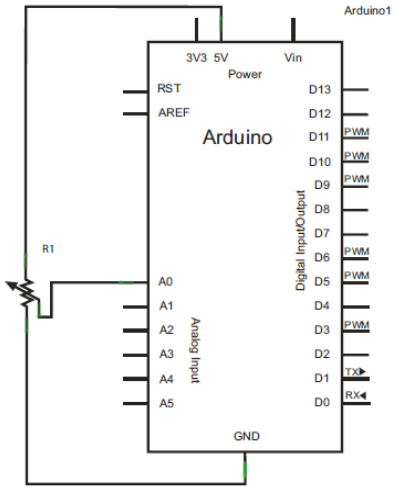


Figure 40 - Schéma de connexion avec un potentiomètre

Remarque: rajouter une capacité de découplage de 0.1 microfarad entre la masse et le +5volts.

3- Programme

```
int sensorVal = 0;                      Déclaration de la variable sensorVal
                                        pour accueillir La valeur analogique.
void setup(){                           -Début de la fonction setup()
   Serial.begin(9600);                  Initialiser la vitesse d'échange avec
                                        le moniteur à 9600 bauds
}                                       -Fin de la fonction setup()
void loop() {                           +Début de la fonction loop()
 sensorVal = analogRead(A0);            La fonction analogRead() lit la valeur
                                        de la tension affichée sur la pin A0,
                                        la convertit et la stocke dans la
                                        variable sensorVal.
 Serial.print("Sensor = ");            Affichage de "Sensor = "
 Serial.println(sensorVal, DEC);        suivi de la valeur en décimal de la
                                        variable sensorVal.
 delay(10);                             Temporiser 10 ms, et retour au début
                                        de la boucle 'loop'. 10 ms est
                                        considéré suffisant pour permettre à
                                        la conversion A/D de se faire
                                        correctement.
}                                       +Fin de la fonction loop().
```

Remarques:
1- La valeur fournie par la fonction analogRead() fournit une valeur entre 0 et 1023 (la résolution du convertisseur analogique/numérique étant de 10 bits, 0 correspond à une tension de 0 volt et 1023 correspond à 5 volts qui est la source d'alimentation de l'arduino).
2- Dans l'instruction Serial.println(sensorVal, DEC), le paramètre DEC sert à préciser que la valeur de sensorVal est affichée en décimal. Il existe d'autres options à la place de DEC, qui sont: HEX (pour afficher la valeur en hexadécimale), BIN (en binaire), OCT (en octal).

Programmation

Le test des expressions

Avant de faire l'exercice suivant, nous allons décrire comment arduino effectue des tests en faisant des comparaisons. Pour tester une variable par rapport à une autre, le programme utilise l'instruction suivante:

```
if (variable1 op_relation variable2) {
        instructions...
}
else {
        autres instructions ...
}
```

op_relation est l'opérateur qui peut avoir l'une des relations suivantes:

==	Égal
!=	Différent
<	Inférieur
>	Supérieur
<=	inférieur ou égal
>=	supérieur ou égal

Exemple:

```
if (sensVal > 200)
{
        allumer la LED rouge
}
else
{
         allumer la LED verte
}
```

Note: la partie else {} est optionnelle.

Exercices

1- Faire le programme qui affiche les nombres entiers de 0 à 255, en décimal, hexadécimal, binaire et octal. Utiliser le moniteur pour voir les résultats.
2- Faire le montage avec un potentiomètre de 10 KΩ et 2 LEDs (verte et rouge) et le programme pour allumer la LED verte si la valeur de tension fournie par le potentiomètre (provenant du curseur) est inférieure à 2.5 volts sinon allumer la LED rouge.
On considère que le potentiomètre est alimenté en 5 volts. Utiliser le moniteur pour vérifier la logique du programme.

Lire une information analogique à partir d'un capteur piézo-électrique

1- Description

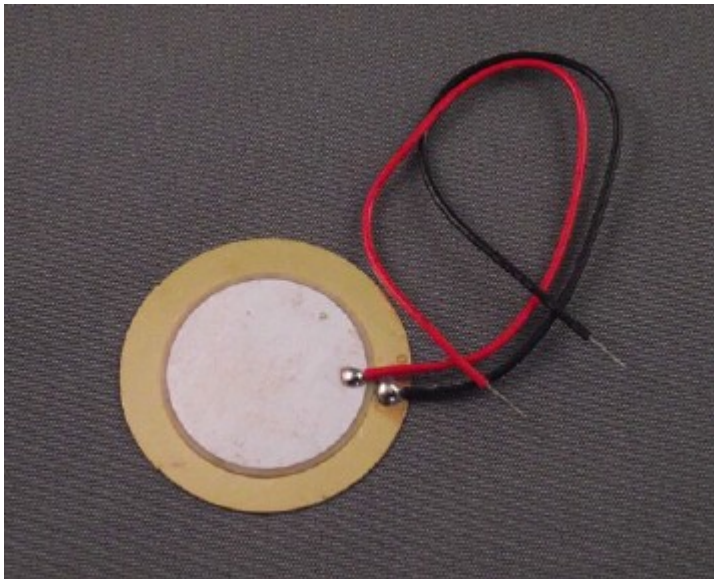


Figure 41 - Capteur piézo-électrique

2- Connexions

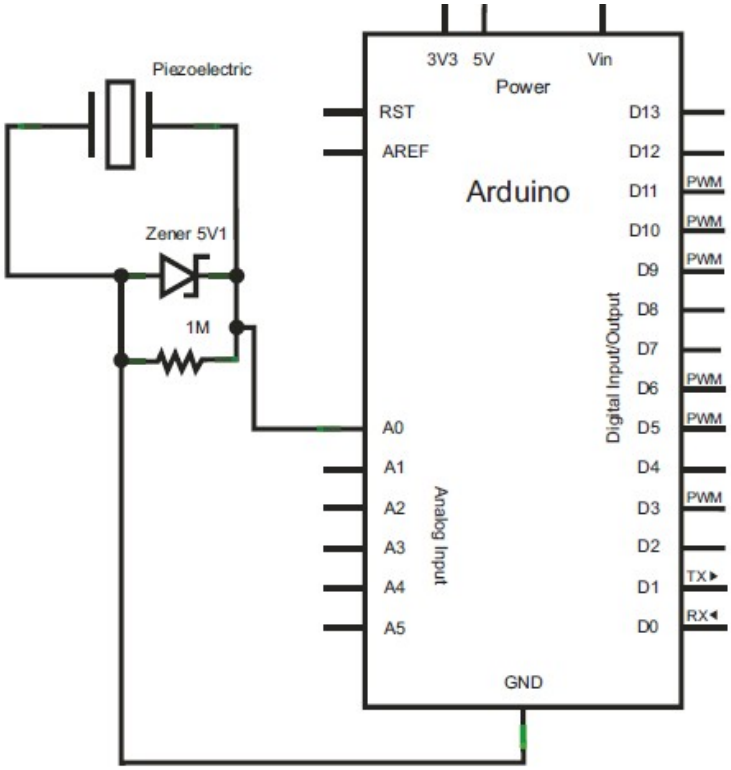


Figure 42 - Schéma de connexion arduino - capteur piézo-électrique

3- Programme

Ce programme fait la lecture de la valeur analogique fournie par un capteur piézo-électrique:

```
int analogPin = A0;
int sensVal = 0;
const int seuil = 500;
void setup(){
      Serial.begin(9600);
}
void loop(){
      sensVal = analogRead(analogPin);     // read the voltage on pin A0
      if (sensVal > seuil) {               // compare with threshold
            Serial.print("Valeur lue = ");
            Serial.println(sensVal, DEC);
      }
      delay(10);  // délai pour la conversion A/D
}
```

Exercices

1- Décrire le programme de lecture du capteur piézo-électrique
2- Faire le montage contenant le capteur piézo-électrique et un potentiomètre et écrire le programme pour régler le seuil à l'aide du potentiomètre et allumer la led (pin D13) si la valeur du capteur dépasse ce seuil. Vérifier avec le moniteur série.

Lecture de la valeur d'une photorésistance (capteur de lumière)

1- Description

La photorésistance (LDR: Light Dependant Resistance) agit comme un capteur de lumière. Elle agit comme une résistance variable, selon l'intensité de lumière au lieu où elle est exposée. Elle fournit une grande valeur de résistance lorsqu'il fait sombre et une faible résistance en présence de lumière.
Elle peut être utilisée par exemple pour allumer une lampe lorsque la nuit tombe.
Les valeurs de résistance obtenues dépendent du modèle de photorésistance. Il est nécessaire de vérifier en faisant des tests.

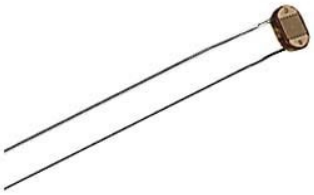


Figure 43 - Une cellule photoélectrique (photorésistance)

2- Connexions

La tension sur la pin analogique A1 (exemple) peut être convertie par arduino à travers son A/D convertisseur. La valeur numérique qui en résulte va se situer entre 200 si la lumière est faible et 800 environ si la lumière est brillante. Reste à expérimenter.

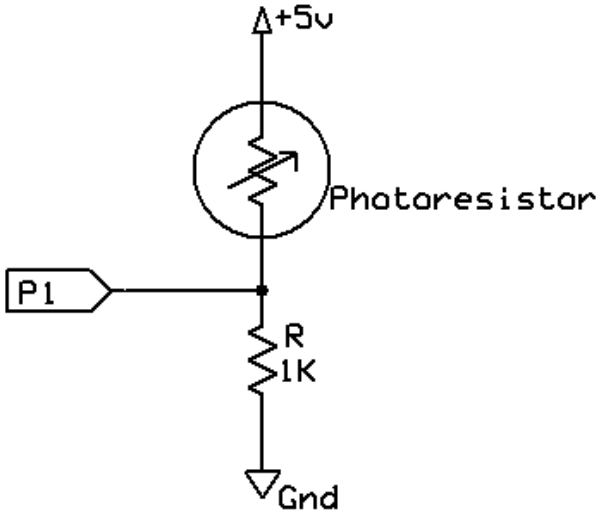
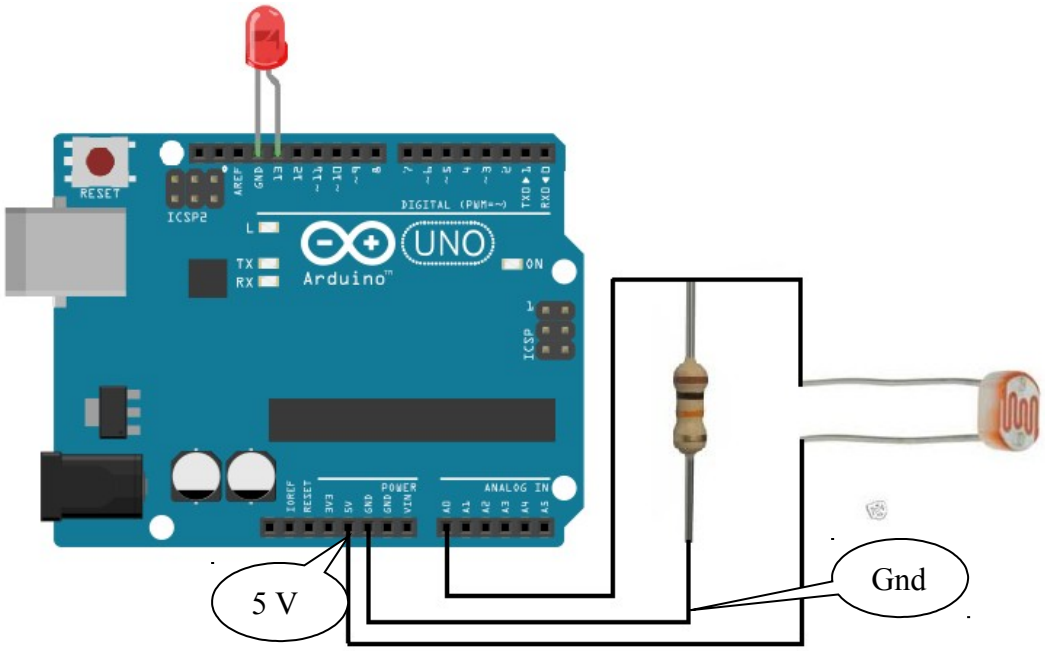


Figure 44 - Connexion d'un capteur de lumière



3- Programme

Remarque: Faire l'essai avec une résistance de 10K puis avec une résistance de 1K.

```
int val;
void setup() {
  Serial.begin(9600);              //  setup serial
}
void loop() {
  val = analogRead(A0);          // lire la pin A0
  Serial.println(val);      // afficher à l'écran
  delay(100);
}
```

Autre essai:

```
// Pin de la Photo-Résistance
int lightPin = 0; // analog pin de la photo-résistance
// Pin de la LED
int ledPin = 9;   // la pin de la LED:  nous utilisons le mode
PWM (pulse width modulation)
void setup()
{
        pinMode(ledPin, OUTPUT);
}
void loop() {
        int lightLevel = analogRead(lightPin);      // lire le niveau de lumière
        // vérifier si la valeur est dans le domaine  750 à 900
         lightLevel = constrain(lightLevel, 750, 900);
        // ajuster la valeur 750 à 900 dans le domaine 255 à 0
```

```
        lightLevel = map(lightLevel, 750, 900, 255, 0);
        analogWrite(ledPin, lightLevel);
}
```

Autre code d'essai

```
const int ledPin=13;
const int sensorPin = 0;

int level; // la variable qui va contenir le niveau de lumière mesuré

const int threshold=800;        // Seuil préconfiguré. Si < 800, la LED doit être mis à ON

void setup() {
        pinMode (ledPin, OUTPUT);
        pinMode (sensorPin, INPUT);
        Serial.begin(9600);

}

void loop() {
        level = analogRead(sensorPin);          // lire la mesure (pin A0)
        if (level > threshold) {
                digitalWrite(ledPin, HIGH);  // si le niveau < seuil, allumer la LED
        }
        else {
                digitalWrite(ledPin, LOW);   // sinon éteindre la LED
        }
        delay(100);
}
```

Contrôler un circuit par modulation de largeur d'impulsion (PWM, Pulse Width Modulation)

1- Description

Arduino peut contrôler un circuit analogique. Nous pouvons utiliser la fonction analogWrite(pin, valeur), ou pin est un numéro de pin parmi 3, 5, 6, 9, 10, 11 sur Arduino Uno ou parmi 2 à 13 sur Arduino Mega.

Modèle d'Arduino	Broches PWM
Uno, Pro Mini, Nano	3, 5, 6, 9, 10 et 11
Mega	2 à 13, 44 à 46
Leonardo	3, 5, 6, 9, 10, 11 et 13
Due	2 à 13
Zero	2 à 13

Le paramètre valeur n'est autre que le rapport de cycle égal entre 0 et 255 correspondant à un rapport compris entre 0 et 100%.
En somme cette fonction produit un signal PWM (Pulse Width Modulation) modulation de largeur d'impulsion.
Par exemple, la valeur à appliquer pour un rapport cyclique de 75% sera égal à 0,75 x 255 = 191.
V = Rapport cyclique x 255

Voici quelques cas de figure montrant un signal PWM:

- 0 correspondant à un rapport de cycle de 0%
- 64 correspondant à un rapport de cycle de 25%
- 127 correspondant à un rapport de cycle de 50%
- 191 correspondant à un rapport de cycle de 75%
- 255 correspondant à un rapport de cycle de 100%

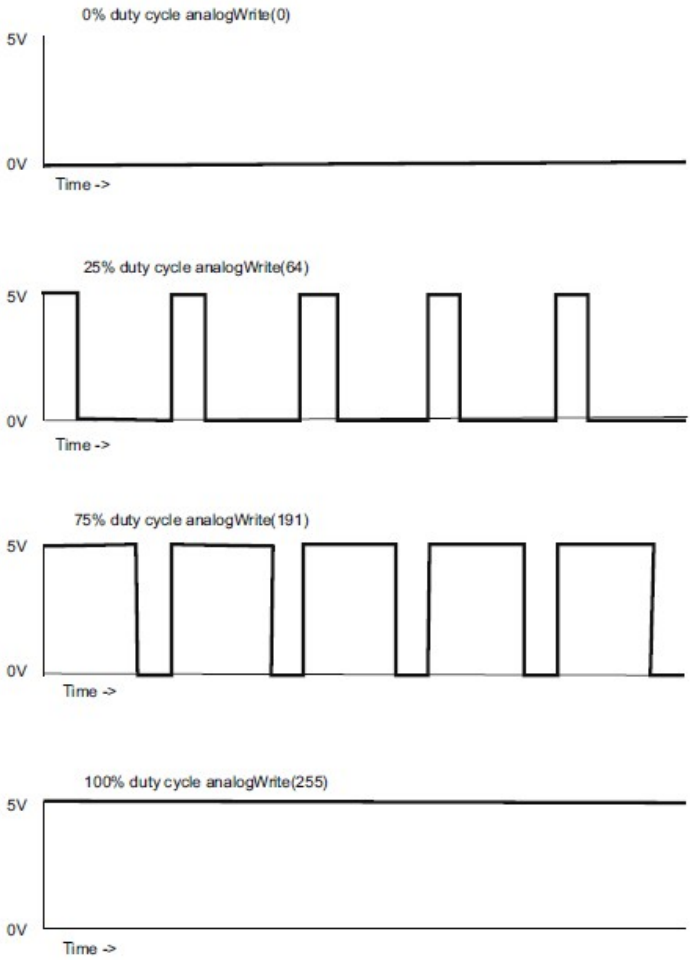


Figure 45 - Quelques formes de signaux PWM

La fréquence du signal PWM est prédéterminée sur l'Arduino. La fréquence n'est pas la même selon les broches. Sur le Uno, la fréquence est de **490Hz** sur les broches **3, 9, 10 et 11** et de **980Hz** sur les broches **5 et 6**.
Nous verrons cela en vérifiant avec un analyseur logique.

2- Connexions

...

3- Programme

Voici un programme qui montre la forme d'un signal sur une sortie PWM de l'arduino. On va prendre la pin 5 dans cet exemple.

```
const int analogPin = 5;
const int delay = 1000;
void setup() {
	Serial.begin(9600);
}
void loop() {
	for (int i = 0; i<=255; i++) {
		analogWrite(analogPin, i);	// tester cette pin en branchant un voltmètre
		Serial.println(i);
		delay(delay);
```

```
        }
        analogWrite(analogPin, 0);
        delay(2000);
}
```

Autre séquence à tester:

```
const byte pinDEL = 3;
byte luminosite = 0; // peut varier de 0 à 255

void setup()
{
}
void loop()
{
  analogWrite(pinDEL, luminosite);
  luminosite++;
  delay(4);
}
```

Exercices

1- Calculer le pourcentage (%) correspondant aux valeurs suivantes:
12,33, 66. Extraire la formule générale.

2- Faire le montage en branchant une LED sur la pin A5 par exemple. Écrire le programme pour moduler l'intensité lumineuse de la LED. Identique au programme précédent. Ne pas oublier la résistance de limitation de courant.

3- Tester le programme suivant:

```
const int pinLED = 9;
int i = 0;
void setup() {

}
void loop() {
        for (i = 0; i < 255; i++) {
                analogWrite(pinLED, i);
                delay(10);
        }
        for (i = 255; i > 0; i--) {
                analogWrite(pinLED, i);
                delay(10);
        }
}
```

Commander un haut-parleur

1- Description

...

2- Connexions

Attacher la pin D8 de l'arduino avec une borne de la résistance de 1ko; l'autre borne de cette résistance est attachée à une borne du haut-parleur et l'autre borne du HP mise à la masse (GND), tel que c'est dessiné sur le schéma suivant:

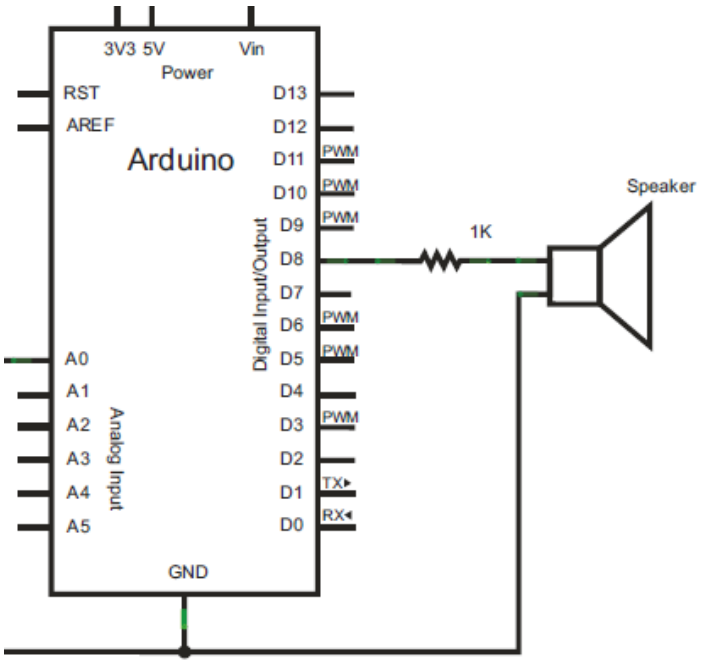


Figure 46 - Schéma de connexion arduino avec HP (40 ohms et plus)

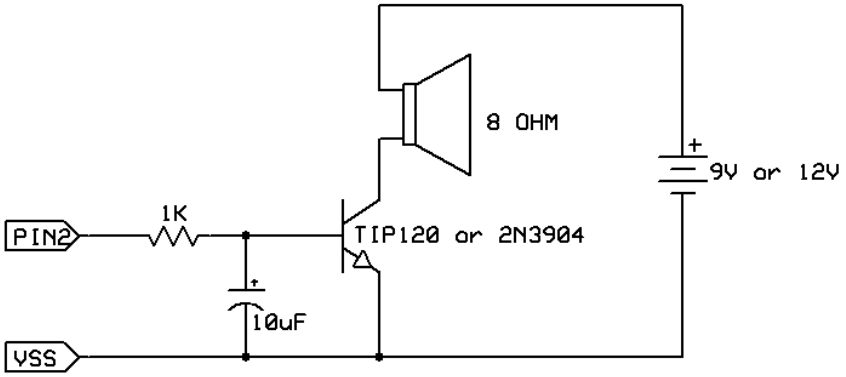


Figure 47 - Connexion avec un HP de 8 ohms

Nous allons commander le haut-parleur pour lui faire jouer les notes suivantes:

- C = 262 Hz
- D = 294 Hz
- E = 330 Hz

- G = 392 Hz
- A = 440 Hz

3- Programme

```
const int toneDuration = 10;
const int hpPin = 8;    // ou pin D2 selon le cas
int tones[] = {262, 294, 330, 392, 440}; // tableau de fréquences

void setup() {
...
}
void loop() {
        for (int sensVal = 0; sensVal < 5; sensVal ++) {
                tone(hpPin, tones[sensVal], toneDuration);
        }
}
```

Commander une LED RGB

1- Description

La LED RGB dispose à l'intérieur de 3 LEDs rouge (R), vert (V), bleu (B).
Leur combinaison des intensités lumineuses va offrir une multitude de couleurs.

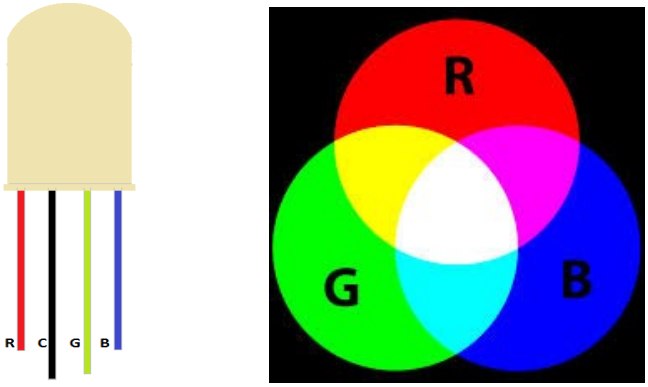


Figure 48 - Pins d'une LED RGB et couleurs RGB

2- Connexions

Remarque:
Si la LED RGB est à anode commune, connecter la pin C au 5volts; si elle est à cathode commune connecter-la à la masse (GND).
Les autres pins de la LED seront connectées à 3 pins PWM de l'arduino (ne pas oublier de brancher la résistance pour chaque diode).
Exemple: les pins de connexion sont

- Pin 3 de l'arduino connectée à la pin R (Red) de la LED
- Pin 5 connectée à la pin G (Green) de la LED
- Pin 6 connectée à la pin B (Blue) de la LED

GND connectée à la pin la C (Common) de la LED si elle est de type cathode sinon connecter le +5volts à la pin C.

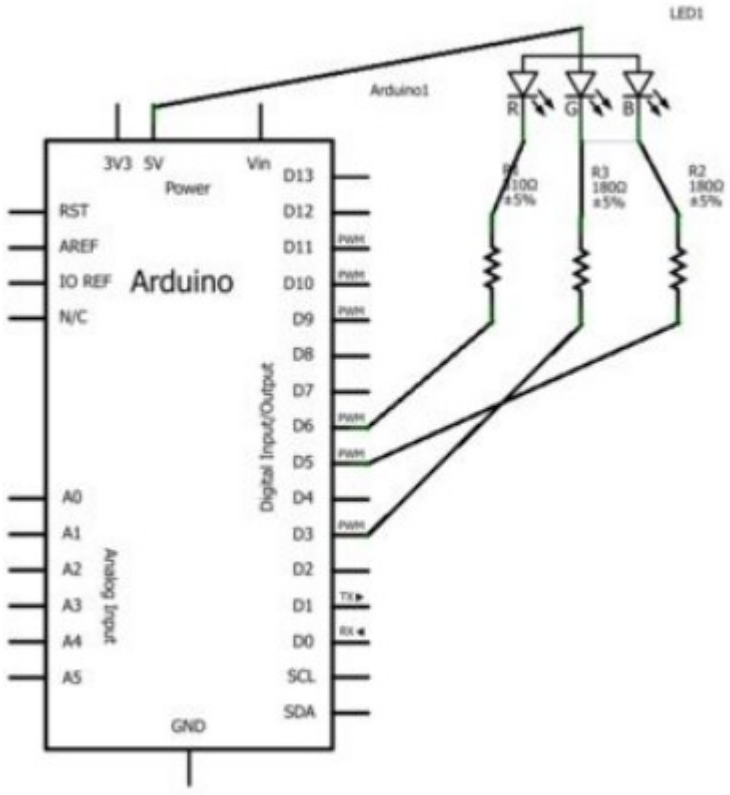


Figure 49 - Connexion avec une LED RGB

3- Programme

```
void setup() {
  // déclarer le port série avec une vitesse de 9600 bauds
  Serial.begin(9600);
  // déclarer les pins de commande des LEDs en OUTPUT
  pinMode(6, OUTPUT); // Rouge
  pinMode(5, OUTPUT); // Bleu
  pinMode(3, OUTPUT); // Vert
}

void loop()
{
  // Vérifier la valeur retournée par la fonction ReadFromConsole()
  switch(ReadFromConsole() ) {
     case 'R':
          analogWrite(6, 255);
          break;
     case 'B':
          analogWrite(5, 255);
          break;
     case 'V':
          analogWrite(3, 255);
          break;
  }
}

// Lire un caractère sur la console (taper R ou B ou V)
```

```
int ReadFromConsole() {
      while ( Serial.available()<=0 ) {
      }
      return Serial.read();
}
```

Cas du fade-in fade-out

Exercice: faites varier l'intensité lumineuse de chaque LED (en allant de 0 à 255 puis de 255 à 0) avec un délai de 20 ms entre chaque variation. Indication: programmer 3 boucles for pour le fade-in et 3 boucles for pour le fade-out.

Commander un afficheur digital à 7 segments

1- Description

L'afficheur dispose de 7 LEDs (segments lumineux); il peut être de type décimal (0 à 9) ou hexadécimal (0 à F), selon sa référence.
L'afficheur peut se présenter soit avec une cathode commune, soit avec une anode commune.

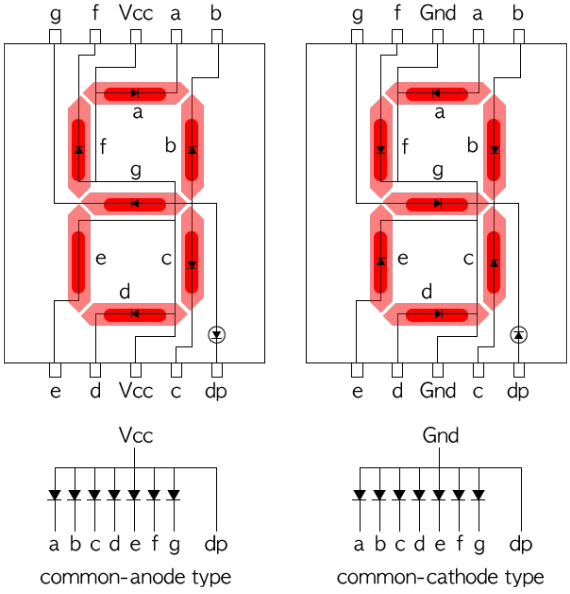


Figure 50 - L'afficheur à 7 segments

2- Connexions

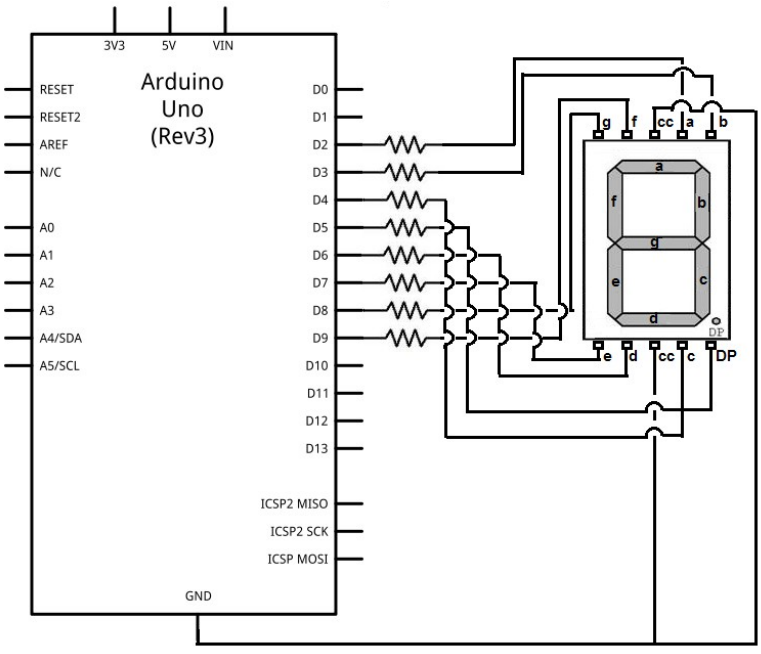


Figure 51 - Connexion à un afficheur 7 segments (cathode commune)

Remarque: utiliser une résistance de 220 ohms pour le courant des LEDs.

3- Programme

Le tableau suivant représente les différentes valeurs (en hexadécimal) qui correspondent à l'affichage des nombres hexadécimaux (0 à F).

Digit	ABCDEFG	A	B	C	D	E	F	G
0	0×7E	1	1	1	1	1	1	0
1	0×30	0	1	1	0	0	0	0
2	0×6D	1	1	0	1	1	0	1
3	0×79	1	1	1	1	0	0	1
4	0×33	0	1	1	0	0	1	1
5	0×5B	1	0	1	1	0	1	1
6	0×5F	1	0	1	1	1	1	1
7	0×70	1	1	1	0	0	0	0
8	0×7F	1	1	1	1	1	1	1
9	0×7B	1	1	1	1	0	1	1
A	0x77	1	1	1	0	1	1	1
B	0x1F	0	0	1	1	1	1	1
C	0x4E	1	0	0	1	1	1	0
D	0x3D	0	1	1	1	1	0	1
E	0x4F	1	0	0	1	1	1	1
F	0x47	1	0	0	0	1	1	1

```
// Bits représentant les chiffres 0-9 (base décimale)
const byte numeral[11] = {
        B11111100,    // 0
        B01100000,    // 1
        B11011010,    // 2
        B11110010,    // 3
        B01100110,    // 4
        B10110110,    // 5
        B00111110,    // 6
        B11100000,    // 7
        B11111110,    // 8
        B11100110,    // 9
        B00000000,    // pas d'affichage
};

// Les pins (a-g) de l'afficheur 7 segments connectées sur  les pins correspondantes de
// l'arduino :
const int segmentPins[8] = { 5, 8, 9, 7, 6, 4, 3, 2 };

void setup() {
        for (int i = 0; i < 8; i++) {
                pinMode(segmentPins[i], OUTPUT);
        }
}
void loop() {
        for (int i = 0; i <= 10; i++) {
                showDigit(i);
```

```
                delay(1000); // laisser un délai
        }
        delay(2000); // un autre délai
}

void showDigit (int number) {
boolean isBitSet;
for (int segment = 1; segment < 8; segment++) {
        isBitSet = bitRead(numeral[number], segment); // bitRead lit le bit 'segment' du
                                                      // nombre
      digitalWrite(segmentPins[segment], isBitSet);
   }
}
```

Commander une barrette de 4 afficheurs à 7 segments

1- Description

Nous allons décrire le module de 4 afficheurs à 7 segments et sa mise en œuvre avec l'arduino. Il peut être utilisé comme support d'information numérique.
Il est représenté par la figure ci-dessous (Figure 52).

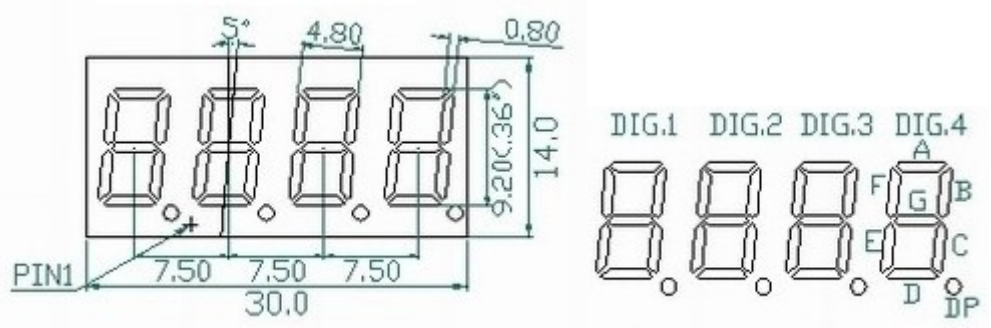


Figure 52 - Pin 1 et segments du module HS420361K-32

2- Connexion

Pour mettre en œuvre le module, nous devons avoir le diagramme des pins (pinout) ainsi que la disposition des segments.

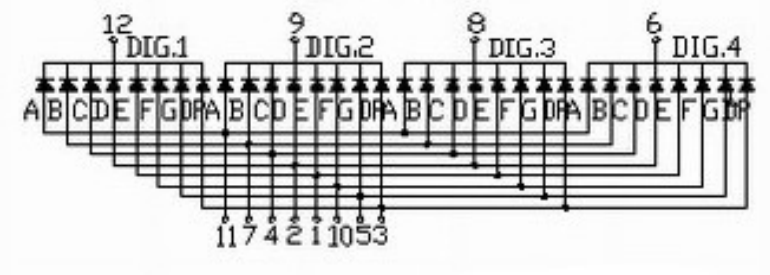
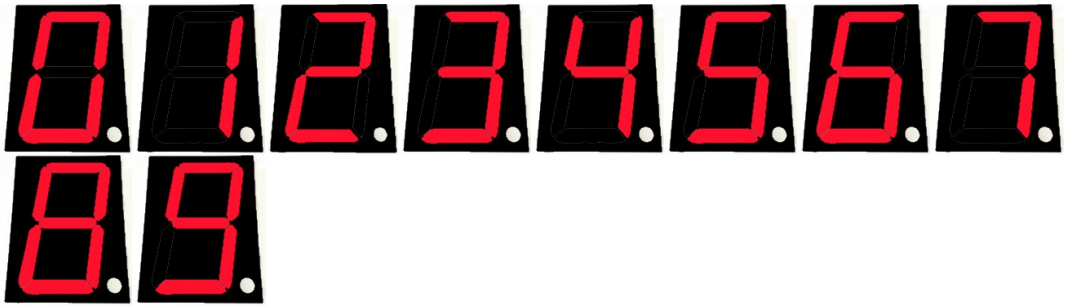


Figure 53 - Description des pins du module HS420361K-32

D'après le schéma, le module est conçu avec des diodes à cathode commune. Ce sont les pins 12 (DIGIT 1), 9 (DIGIT 2), (DIGIT 3) et 6 (DIGIT 4); donc elles doivent être mises à la masse. Les pins 11 (A), 7 (B), 4 (C), 2 (D), 1 (E), 10 (F), 5 (G) et 3 (DP) seront utilisés pour allumer/éteindre les différents segments. La pin DP (decimal point) est le point décimal.



3- Programme

Essayer le programme précédent (pour un seul afficheur). Modifier le code pour l'adapter au module. Allumer le module afficheur par afficheur.

Capturer l'intensité lumineuse à l'aide d'un capteur de lumière

1- Description

Nous allons utiliser une photorésistance. Son rôle est de capter la lumière et de fournir son intensité sous forme de tension analogique entre 0 et 5 volts.

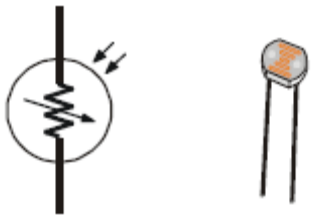


Figure 54 - Photorésistance et son symbole électrique

2- Connexion

On utilise un pont diviseur constitué d'une résistance de 10 ko et d'une photorésistance. Une extrémité de la résistance de 10 ko est reliée au +5 volts du montage; l'autre extrémité est reliée à la pin A0 de l'arduino (entrée analogique). La photorésistance relie une borne à la masse (GND) et l'autre borne à la pin A0. Ceci constitue le pont diviseur.
L'autre partie du montage comprend une LED ordinaire reliée à une résistance de limitation de courant de 220 ohms (non obligatoire). On utilise la pin D9 pour contrôler la LED.
Pour vérifier, faites laisser passer la lumière ou l'obstruer pour voir le résultat.
Le dessin est représenté par le schéma suivant:

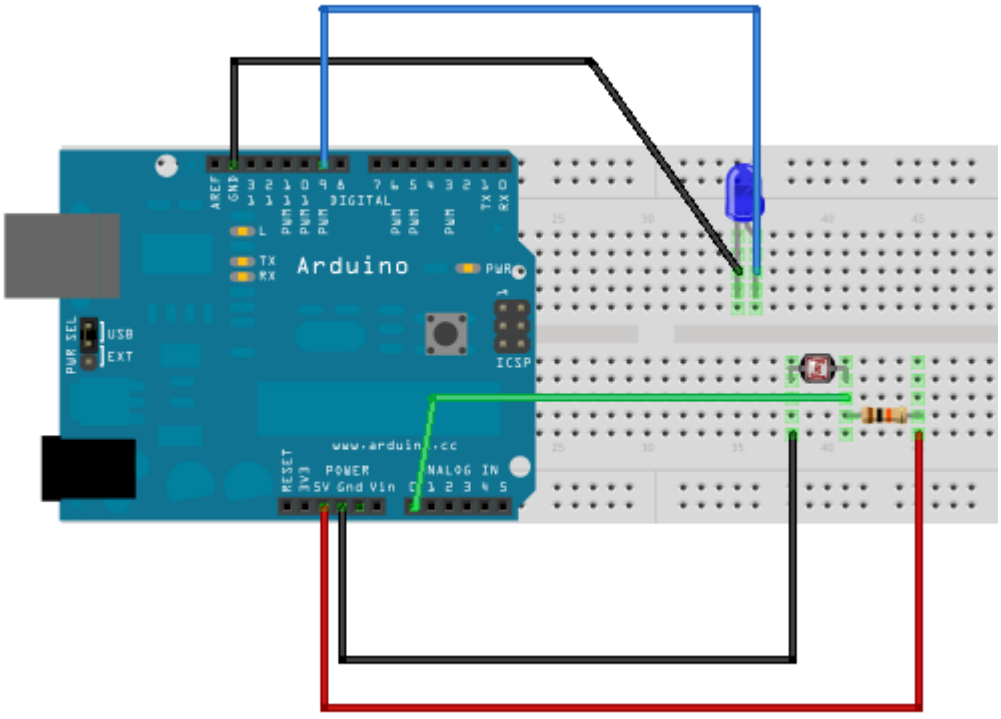


Figure 55 - Connexion avec une cellule photoélectrique

3- Programme de test

```
const byte photoPin = A0;
const byte ledPin = 9;
int intensite;
void setup() {
        pinMode(ledPin, OUTPUT);
}
void loop() {
  intensite = analogRead(photoPin); // On lit la valeur sur la pin A0 (entre 0 et 1023)
  intensite = map(intensite, 0, 900, 0, 255);  // On transforme la valeur - domaine 0..255
  intensite = constrain(intensite, 0, 255);      // On garantit que la valeur obtenue est valide
  analogWrite(ledPin, intensite);                // On contrôle la LED
}
```

Exercices:

1- Vérifier en utilisant le moniteur de l'arduino. Utiliser la bibliothèque Serial.

Contrôler le niveau d'humidité et la température ambiante (capteur DHT11/DHT22)

1- Description

Les capteurs DHT sont utilisés pour mesurer la température ambiante et le taux d'humidité. Ils contiennent à l'intérieur un convertisseur analogique/digital, et fournissent en parallèle les valeurs de température et d'humidité.

Il y a deux versions de ce capteur. Voici les différences:

	DHT11	**DHT22**
Range d'humidité	20%-80% (±5%)	0%-100% (±2%)
Température	0-50°C (±2°C)	-40-80°C (±0.5°C)

Comme on peut le constater, le DHT22 a une meilleure précision mais coûte un peu plus cher. Très utilisé dans les déshumidificateurs et autres appareils de précision.

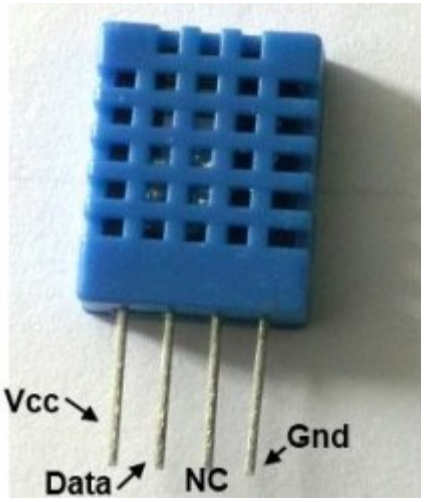


Figure 56 - Capteur DHT11 de température et d'humidité

2- Connexion

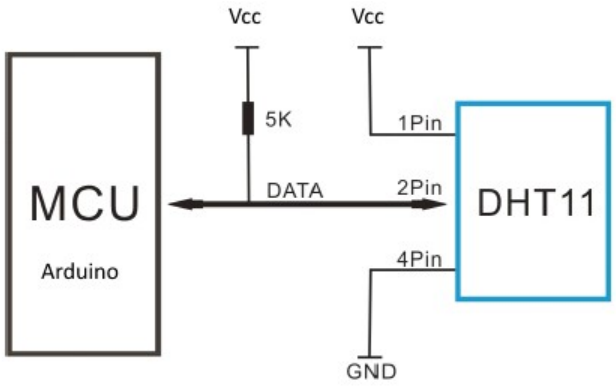


Figure 57 - Connexion à un capteur d'humidité

3- Programme

Le programme suivant utilise le capteur DHT11:

```
#include "DHT.h"                        // Utiliser la librairie DHT
#define         DhtPin          2       // La pin 'data' du DHT22 (pin digitale)
// Utiliser le bon DHT(11/22)
//#define       DHTTYPE     DHT11           // Dans l'exemple, on utilise le DHT 22
#define         DHTTYPE     DHT22           // DHT 22       (chip AM2302)
//#define       DHTTYPE     DHT21           // DHT 21       (chip AM2301)

// Initialiser le DHT22
DHT dht(DhtPin, DHTTYPE);

void setup() {
        Serial.begin(9600);
        Serial.println("DHTxx test!");
        dht.begin();
}

void loop() {
        // Attendre un instant entre 2 mesures:  Le capteur est un relativement lent
        delay(2000);
        float h = dht.readHumidity();
         float tc = dht.readTemperature();      // Lire la température en Celsius
         float tf = dht.readTemperature(true);          // Lire la température en Fahrenheit
        // Vérifier les données lues
        if (isnan(h) || isnan(tc) || isnan(tf)) {
                Serial.println("Echec de lecture!");
                return;
        }
        // Calculer l'index de chaleur
        float hi = dht.computeHeatIndex(tf, h);
        Serial.print("Humidite:  ");  Serial.print(h);  Serial.print(" %\t");
        Serial.print("Temperature:  ");  Serial.print(tc);  Serial.print(" *C ");
        Serial.print(tf);  Serial.print(" *F\t");
        Serial.print("Index de chaleur:  ");  Serial.print(hi);  Serial.println(" *F");
}
```

Contrôler un relais à travers un transistor

1- Description

L'arduino n'est pas capable de contrôler des charges qui demandent de fortes intensités de courant, comme un relais, une bobine, un moteur, etc..
Nous devons alors utiliser un transistor car il est fait entre pour amplifier le courant en lui fournissant un courant très faible.
L'arduino est capable de fournir sur sa pin un courant allant jusqu'à 40 mA. Le relais utilisé est de type DPDT.
Schéma du relais:

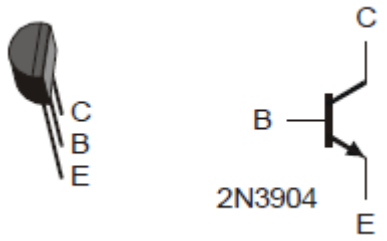


Figure 58 - Schéma d'un transistor et son symbole électrique

2- Connexions

Nous utilisons un transistor bien connu sur le marché, le 2N2222.
Son pôle émetteur est relié à la masse (GND), sa base est relié à la pin D2 de l'arduino à travers une résistance de 2.2 kOhms. Son pôle collecteur est une borne de la bobine du relais. L'autre borne de la bobine est reliée au +5 volts du montage; une diode de protection 1N4001 est placée entre les bornes de la bobine.
Les 2 sorties du relais sont utilisées pour commander la charge. Placer 2 LED connectées à 2 résistances pour vérifier l'action du relais.

3- Programme

```
const byte ledPin = 2;          // la pin qui commande le transistor
void setup()  {
        pinMode(ledPin, OUTPUT);
}
void loop() {
        digitalWrite(ledPin, HIGH);
        delay(500);
        digitalWrite(ledPin, LOW);
        delay(500);
}
```

Contrôler un afficheur à 7 segments

1- Description

Un afficheur 7 segments dispose de LEDs sous forme de segments appelés a, b, c, d, e, f, g, dp.

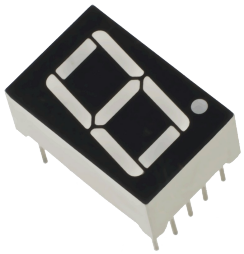


Figure 59 - Afficheur en circuit intégré

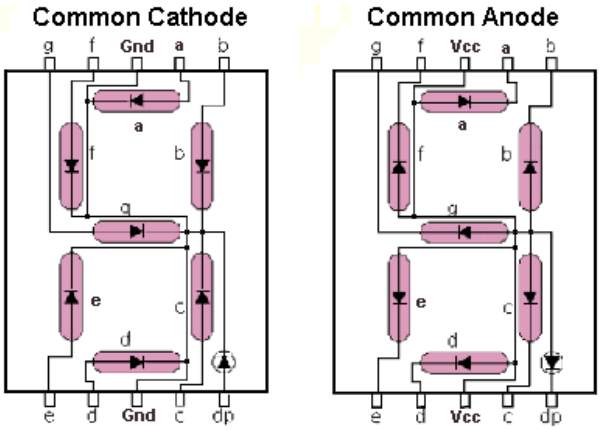


Figure 60 - Représentation d'un afficheur digital

Il peut se présenter sous 2 formes: l'afficheur à anode commune ou à cathode commune.

Pour calculer la valeur des résistances à insérer dans les sorties a – b – c – d – e – f – g – dp, on peut utiliser cette simple formule:
$\Omega = (V - 1,5) / 0,016$
Exemple:
Par conséquent, si on veut allumer un afficheur avec une tension de 4,5 volts, on doit utiliser 8 résistances de:
$(4,5 - 1,5) / 0,016 = 187,5\ \Omega \approx 200\ \Omega$ ou $180\ \Omega$.

2- Connexions

3- Programme

```
int i;
void setup() {
```

```
        pinMode(8, LOW);
        for (i=0; i ;i++)
                pinMode(i, HIGH);
}
void loop() {
        unsigned int LedPins[] =

        for(i=0; i 16; i++)  {
                PORTD = LedPins[i];
        }
}
```

Contrôler plusieurs afficheurs à 7 segments à la fois

1- Description

2- Connexions

3- Programme

"A refaire

```
/*

Easy way to control 7 segment displays by Arduino

Stefan Hermann
www.fritzing.org


 _________  _________  _________  _________
/____a____\ /____a____\ /____a____\ /____a____\
|  | |  | | |  | |  | | |  | |  | | |  | |  |
|f| |b| |f| |b| |f| |b| |f| |b|
|  | |  | | |  | |  | | |  | |  | | |  | |  |
\_/_______\_/ \_/_______\_/ \_/_______\_/ \_/_______\_/
_|___g___|_ _|___g___|_ _|___g___|_ _|___g___|_
/ \ / \ / \ / \ / \ / \ / \ / \ / \
|  | |  | | |  | |  | | |  | |  | | |  | |  |
|e| |c| |e| |c| |e| |c| |e| |c|
\_/_______\_/ __ \_/_______\_/ __ \_/_______\_/ __
\_/_______\_/ __
\____d____/ |DP| \____d____/ |DP| \____d____/ |DP| \____d____/
|DP|
Digit 4 Digit 3 Digit 2 Digit 1
*/

const int numberOfSegments = 8; // do not change
int segmentPins[numberOfSegments] = {2,3,4,5,6,7,8,9}; //
change to your pins a,b,c,d,e,f,g,DP
const int numberOfDigits = 4; // change to the number of
digits
int commonPins[numberOfDigits] = {13,12,11,10}; // change to
the pins of digits starting with the left digit

int displayType=1; // change to the type of your display:
common anode = 1; common cathode = 2;

int myNumber=0;
```

```
int theFigures[11][7]={
{1,1,1,1,1,1,0},
{0,1,1,0,0,0,0},
{1,1,0,1,1,0,1},
{1,1,1,1,0,0,1},
{0,1,1,0,0,1,1},
{1,0,1,1,0,1,1},
{1,0,1,1,1,1,1},
{1,1,1,0,0,0,0},
{1,1,1,1,1,1,1},
{1,1,1,0,0,1,1},
{0,0,0,0,0,0,1}};

void setup() {
     // config for the display
     for (int i=0; i<numberOfSegments; i++){
          pinMode(segmentPins[i],OUTPUT);
     }
     for (int i=0; i<numberOfDigits;i++){
          pinMode(commonPins[i],OUTPUT);
     }
     // type your setup code here
}

void loop(){
     // type in your loop code here
     // this is just for the example:
     showNumber(myNumber); // display the number
     if ( myNumber++;
          delay(1);
}

// do not change the methods below
void showNumber(int theNumber) {
     for (int stelle=numberOfDigits; stelle>=0; stelle--) {
          char szZahl[numberOfDigits];
          sprintf(szZahl, "%d", theNumber);
          int stellen = strlen(szZahl);
          if (stelle > stellen) {
          } else {
               int ergebnis = szZahl[stelle] - '0';
               if (ergebnis>-1) {
                    digit(((numberOfDigits-1)-(stellen-
(stelle+1))),ergebnis);
               }
               else {
```

```
                    digit(((numberOfDigits-1)-(stellen-
(stelle+1))),0);
                }
            }
            delay(1);
            clearDisplay();
}
}
```

```
void digit(int theDigit, int theNumber) {
      if (displayType==1) {
            digitalWrite(commonPins[theDigit], HIGH);
            for (int i=0; i<7; i++) {
                  if (theFigures[theNumber][i] == 0) {
                        digitalWrite(segmentPins[i],HIGH);
                  } else {
                        digitalWrite(segmentPins[i],LOW);
                  }
            }
      } else {
            digitalWrite(commonPins[theDigit], LOW);
            for (int i=0; i<7; i++) {
                  digitalWrite(segmentPins[i],theFigures[theNumber][i]);
            }

      }
}
```

```
void clearDisplay() {
      for (int theDigit=0; theDigit<numberOfDigits; theDigit++) {
            if (displayType == 1) {
                  digitalWrite(commonPins[theDigit], LOW);
                  for (int i=0; i<7; i++) {
                        digitalWrite(segmentPins[i],HIGH);
                  }
            } else {
                  digitalWrite(commonPins[theDigit], HIGH);
                  for (int i=0; i<7; i++) {
                        digitalWrite(segmentPins[i],LOW);
                  }
            }
      }
                                    }
```

Programmation

Utilisation du système d'interruption de l'arduino

Le système d'interruption de l'arduino permet de laisser le programme effectuer la tâche principale normalement puis lorsqu'un évènement bien défini survient alors la tâche principale est interrompue pour effectuer la tâche d'interruption.

1- Description

Le montage suivant va nous permettre d'expliquer comment nous mettons en œuvre le système d'interruption de l'arduino. Les pins D2 et D3 sont aussi utilisées pour la mise en œuvre des interruptions.
Le fonctionnement: l'arduino va allumer les LEDs une à une puis les éteindre une à une, de manière continue (loop). Ceci est la tâche principale. Lorsque ce bouton est actionné, la séquence d'interruption s'exécute et va éteindre les LED pendant 5 sec puis la tâche principale reprend son exécution.

2- Connexions

Nous disposons d'un bouton-poussoir S1 relié à la pin D2. L'autre borne du bouton est raccordée au +5 volts du montage. La pin D2 est aussi rattachée à une résistance dont l'autre borne est raccordée la masse. Nous plaçons ausi 5 Leds (ou moins, selon la disponibilité des éléments) comme il est montré dans la figure ci-dessus.

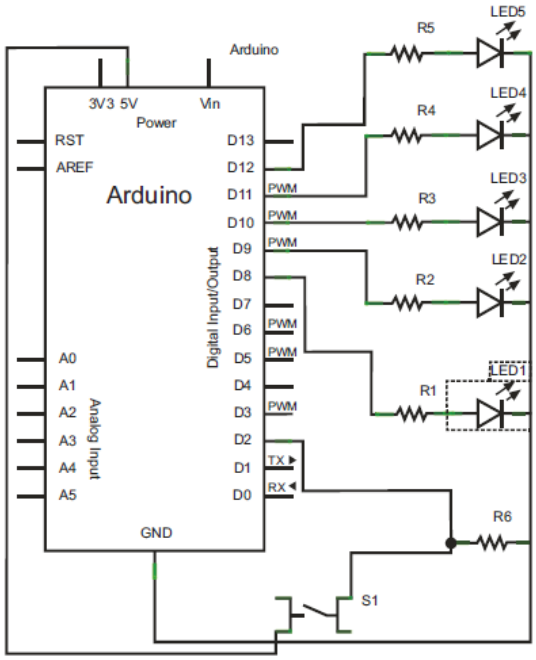


Figure 61 - Mise en œuvre du système d'interruption

3- Programme

```
byte ledTab[] = {8, 9, 10, 11, 12)};   // utilisation d'un tableau d'éléments
byte ct;
const byte timer = 100;
```

```
const int silence = 1000;
void setup() {
	Serial.begin(9600);
	for(ct =0; ct <5; ct ++) {
		pinMode(ledTab[ct], OUTPUT);
	}
	attachInterrupt(0, action_bp, RISING);
}
void loop() {
	for(ct=0; ct<5; ct++) {
		digitalWrite(ledTab[ct], HIGH);
		delay(timer);
	}
	delay(silence);
}
void action_bp() {                              // séquence d'interruption
	Serial.println("BP appuye...");
	for(ct=0; ct<5; ct++) {
		digitalWrite(ledTab[ct], LOW);
		delay(timer*5);
	}
}
```

Afficher des données sur écran LCD

1- Description

L'arduino peut afficher du texte et des valeurs numériques sur un écran à cristaux liquides (LCD: Liquid Crystal Display).
Nous allons utiliser notre circuit qui est le LCD 1602. Il peut afficher 2 lignes de 16 caractères. D'autres écrans LCD peuvent afficher 4 lignes de 16 caractères. D'autres peuvent afficher du graphique.

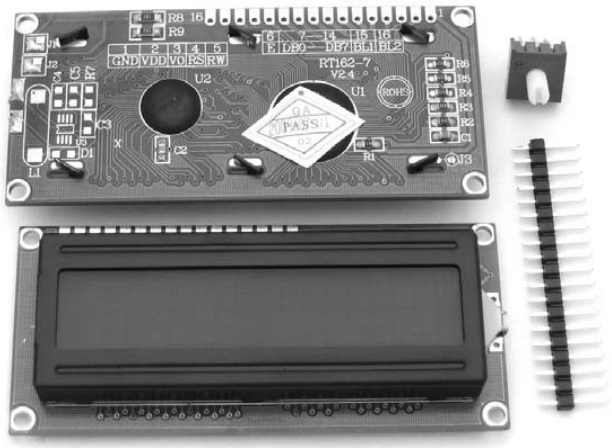


Figure 62 - Écran LCD 1602

Tout afficheur LCD HD44780- or KS0066-compatible ayant la compatibilité avec HD44780 ou KS0066 peut être utilisé avec l'arduino.
Nous allons montrer comment l'arduino peut communiquer avec le LCD1602 pour faire de l'affichage numérique.

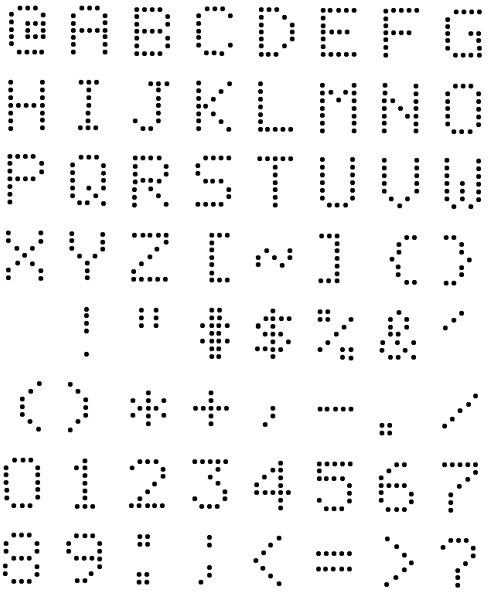


Figure 63 - Représentation des caractères par matrice de points

Chaque caractère affiché va illuminer les différents points de la matrice. La matrice est en général représentée par 7 lignes et 5 colonnes, chaque intersection d'un ligne avec une colonne est en fait une LED qui représente le point lumineux.

Voyons la matrice du caractère A:

	0	0	0	
0				0
0				0
0	0	0	0	0
0				0
0				0
0				0

Figure 64 - Représentation du caractère A

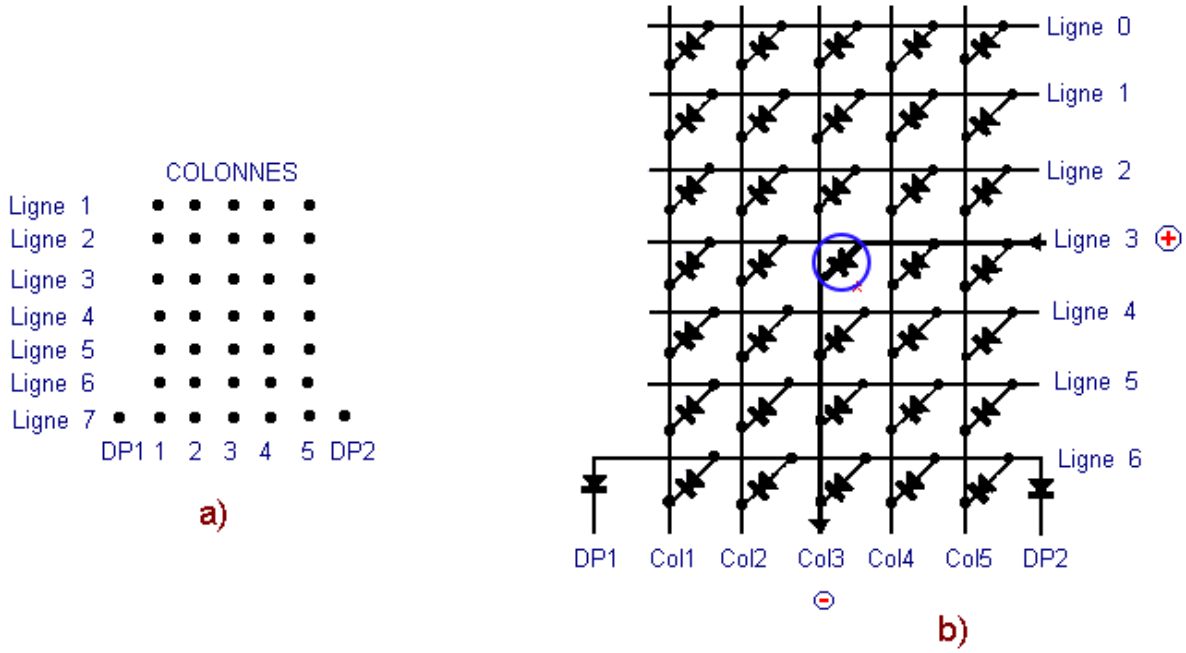


Figure 65 - Représentation d'une matrice à LED

En (a) nous voyons les différents points de luminosité. En (b) la matrice est représentée par un ensemble de diodes luminescentes.
Exemple: Si la Led située à l'intersection de la ligne 3 et de la colonne 3 est parcourue par un courant (de l'anode vers la cathode), cette Led va s'illuminer.

2- Connexions

Voici le pin out (brochage) du module LCD1602.

PIN ASSIGNMENT		
Pin no.	Symbol	Function
1	Vss	Power supply(GND)
2	Vdd	Power supply(+)
3	Vo	Contrast Adjust
4	RS	Register select signal
5	R/$\overline{W}$	Data read / write
6	E	Enable signal
7	DB0	Data bus line
8	DB1	Data bus line
9	DB2	Data bus line
10	DB3	Data bus line
11	DB4	Data bus line
12	DB5	Data bus line
13	DB6	Data bus line
14	DB7	Data bus line
15	A	Power supply for LED B/L (+)
16	K	Power supply for LED B/L (–)

Figure 66 - Pins du LCD1602

La pin V0 du LCD va nous permettre de régler le contraste de l'afficheur, à l'aide d'un potentiomètre de 10 ko.

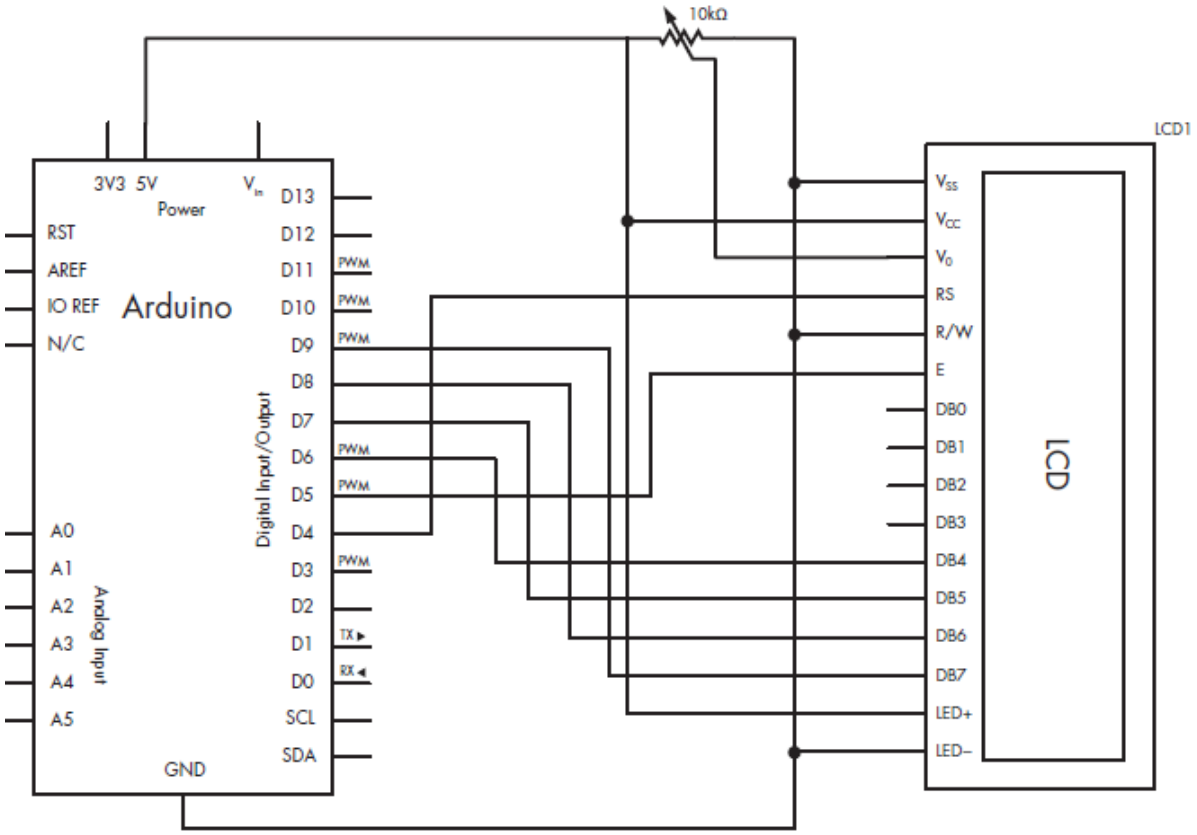
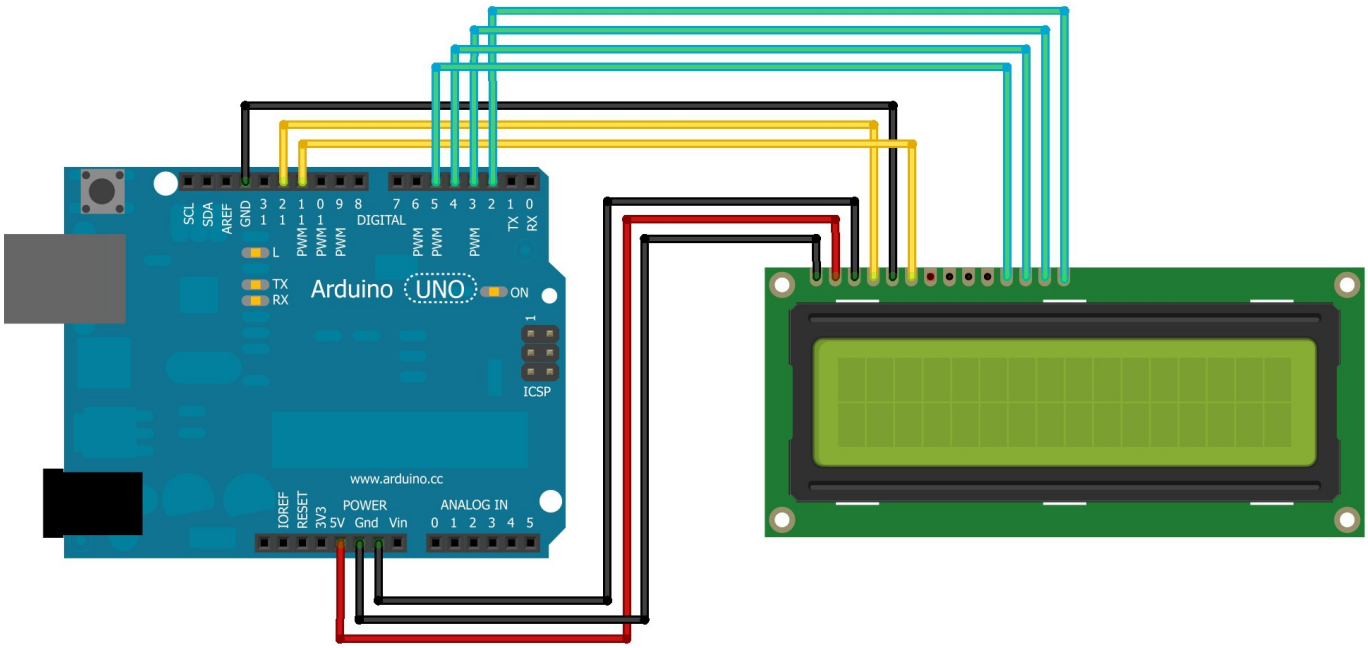


Figure 67 - Connexion avec un écran LCD



3- Programme de test

```
#include <LiquidCrystal.h>          // utiliser la librairie LiquidCrystal de l'arduino
LiquidCrystal lcd(4, 5, 6, 7, 8, 9);   // pins for RS, E, DB4, DB5, DB6, DB7
void setup() {
```

```
        lcd.begin(16, 2);               // LCD de 2 lignes de 16 caractères
        lcd.clear();                    // effacer l'écran
}
void loop() {
        lcd.setCursor(0, 5);    // positionner le curseur de l'écran à la première ligne (0) et
                                // à la 6ème colonne (5)
        lcd.print("Bonjour "); // afficher le texte à la position du curseur (0, 5)
        lcd.setCursor(1, 3);    // positionner le curseur de l'écran à la seconde ligne (1) et
                                // à la 4ème colonne (3)
        lcd.print("tout le monde!");
        delay(1000);
}
```

Note: D'autres fonctions LCD peuvent être utilisées. comme:
- lcd.blink() pour faire clignoter l'écran
- lcd.noblink() arrêter le clignotement
- lcd.clear() effacer l'écran
- lcd.noDisplay()
- lcd.display()
- lcd.scrollDisplayRight()
- lcd.scrollDisplayLeft()

On peut aussi créer ses propres caractères en utilisant la fonction lcd.createChar. Exemple, pour créer le symbole slash (/) on procède comme ceci:
1) on crée le tableau binaire pour représenter la matrice du caractère (5x8) comme ceci:

```
byte slash[8] = {
        B00001,
        B00010,
        B00100,
        B01000,
        B10000,
        B00000,
        B00000,
        B00000
}
```

la fonction s'écrit lcd.createChar(n, slash); n étant le code du caractère à créer (compris entre 0 et 7); exemple n=7.
puis on utilisera lcd.write(7) pour afficher le slash.

Connexion du LCD par interface I²C

1- Description

Le module d'affichage LCD peut aussi être contrôlé par une interface de type I²C qui est le sigle d'Inter-Integrated Circuit.

2- Connexion

L'arduino communique avec les modules I2C par les broches SDA et SCL. Pour le module LCD 1602 ou 1604, il y a un module électronique représenté ici (Figure 68):

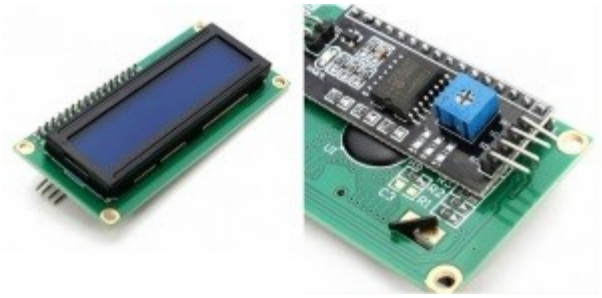


Figure 68 - Écran LCD1602 et son module I²C

3- Programme de test

a) Scanner I²C: Pour utiliser un module I²C, on a besoin de connaître son adresse. Pour cela il faut exécuter un programme de balayage d'adresse (I²C scanner).

```
#include <Wire.h>
void setup() {
    Wire.begin();
    Serial.begin(9600);
    while (!Serial);
        Serial.println("\nI2C Scanner");
  }

void loop() {
    byte error, address;
    int nDevices = 0;
    Serial.println("Scanning...");

    for (address = 1; address < 127; address++ )
    {
        // The i2c_scanner uses the return value of the Write.endTransmisstion to see if
        // a device did acknowledge to the address.
        Wire.beginTransmission(address);
        error = Wire.endTransmission();

         if (error == 0)
        {
                Serial.print("Module I2C non trouvé à l'addresse 0x");
```

```
            if (address<16)
                Serial.print("0");
            Serial.print(address, HEX);
            Serial.println("  !");

            nDevices++;
        }
        else if (error == 4)
        {
            Serial.print("Erreur inconnue à l'addresse 0x");
            if (address<16)
                Serial.print("0");
            Serial.println(address, HEX);
        }
    }
    if (nDevices == 0)
      Serial.println("Pas de module I2C trouvé\n");
    else
      Serial.println("Ok\n");

    delay(5000);          // pause de 5 secondes avant le prochain scan
  }
```

b) Programme de test du module LCD avec I2C

Note: Utiliser la librairie LiquidCrystal-I2C (à télécharger à partir du GitHub).

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

// Positionner l'adresse du LCD ici // faites un scan de l'adresse pour votre LCD
// en utilisant le programme I2C_scanner (ex. 0x27)

LiquidCrystal_I2C lcd(0x27, 16, 2); // pour un écran de 2 lignes de 16 caractères

void setup() {
        // initialiser le LCD
        lcd.begin();

        // Allumer le blacklight.
        lcd.backlight();
        // positionner le curseur en colonne 0 et ligne 0 (exemple)
        lcd.setCursor(0,0);
        lcd.print("Bonjour !");          // afficher un message
        // positionner le curseur en colonne 1 et ligne 1 (exemple)
         lcd.setCursor(1,1); // C1, L1
         lcd.print("`Je suis ici ....."); // afficher un message
}
void loop() {  }
```

Programmation

Affichage d'un nombre sur LCD1602

Il existe un paramètre dans la fonction d'affichage sur LCD, selon le type d'information à afficher c'est-à-dire décimal, binaire, hexadécimal.

a- Afficher un nombre réel, on utilise la fonction 'print' de la librairie LiquidCrystal et on peut indiquer aussi le nombre de décimales souhaitées.

Exemple:

```
float N = 1.61803398875;
lcd.print("Nombre d'or:");
lcd,println(N, 4);              // 4 décimales
```

b- Afficher un nombre décimal en indiquant la base (binaire BIN, décimale DEC, hexadécimale HEX). Exemples:

```
int nbre = 170;
lcd.setCursor(0, 0);            // placer le curseur
lcd.print("Binaire: ");
lcd.println(nbre, BIN);         // afficher en binaire
lcd.println(nbre, DEC);         // afficher en décimal
lcd.setCursor(0, 1);
lcd.print("Hexadecimal: ");
lcd.println(nbre, HEX);         // afficher en hexadécimal
```

Exercices

1- Modifier le programme pour afficher votre nom et prénom sur l'écran LCD. Vos nom et prénom sont rentrés à la console. Pour cela, utiliser la fonction Serial.

2- Faire le montage et le programme qui affiche la valeur de la tension d'un potentiomètre. Revoir le paragraphe qui explique l'utilisation d'un potentiomètre à la section xxxx.

3- Écrire le programme qui affiche le sinus et le cosinus d'un nombre tapé à la console de l'arduino.

Connecter un clavier à 16 touches

1- Description

Le clavier à 16 touches se présente sous forme de matrice composée de lignes et de colonnes; à l'intersection de chaque ligne et de chaque colonne il y a une touche qui laisse ou pas passer le courant. Donc lorsqu'un un courant est envoyé sur une ligne, on va vérifier s'il se propage sur l'une des colonnes de la matrice. S'il n'y a pas de courant cela signifie que la touche correspondante n'a pas été appuyée. Si par contre on récupère le courant, donc la touche a été appuyée à cet endroit.

2- Connexions

Les pins 1 à 4 représentent les lignes; les pins 5 à 8 représentent les colonnes de la matrice.

Pins du clavier	Ligne (L) Colonne (C)	Connexions avec l'arduino
1	L1	D9
2	L2	D8
3	L3	D7
4	L4	D6
5	C1	D5
6	C2	D4
7	C3	D3
8	C4	D2

Tableau 1 - Exemple de connexion d'un clavier

3- Programme

Le programme d'essai utilise la librairie keypad (à downloader à partir de GitHub: http://playground.arduino.cc/code/keypad).

```
/*  Matrice 4x4 connectée à Arduino
Le code affiche la valeur de la touche sur la console */

#include <Keypad.h>           // utiliser la librairie 'keypad'
const byte numRows = 4;       // nombre de lignes du clavier
const byte numCols = 4;       // nombre de colonnes du clavier

// on définit la matrice des caractères
char keymap[numRows][numCols]   =
{
        { '1',  '2', '3', 'A' },
        { '4',  '5', '6', 'B' },
        { '7',  '8', '9', 'C' },
        { '*',  '0',  '#', 'D' }
};

// on définit les pins correspondant aux lignes et aux colonnes qui vont être raccordés à

// l'arduino
byte rowPins[numRows] = {9,8,7,6};          // Lignes 0 to 3
byte colPins[numCols]    = {5,4,3,2};          // Colonnes 0 to 3

// on initialise l'instance de keypad
Keypad myKeypad= Keypad(makeKeymap(keymap), rowPins, colPins, numRows,
numCols);

void setup() {
        Serial.begin(9600);
}

// Si une touche est appuyée elle sera affichée sur la console
void loop() {
        char keypressed = myKeypad.getKey();
        if (keypressed != NO_KEY)   {
                Serial.print(keypressed);
        }
}
```

Les protocoles d'échange I2C et ISP

I- Le protocole I2C

À la fin des années 1970, la compagnie Philips (maintenant devenue NXP) avait vu la nécessité de simplifier et standardiser les échanges de données entre les différents circuits intégrés dans leurs produits. Le bus I^2C fut inventé, et réduisait le nombre de lignes nécessaires à seulement deux lignes, SDA - Serial DAta, et SCL - Serial CLock.

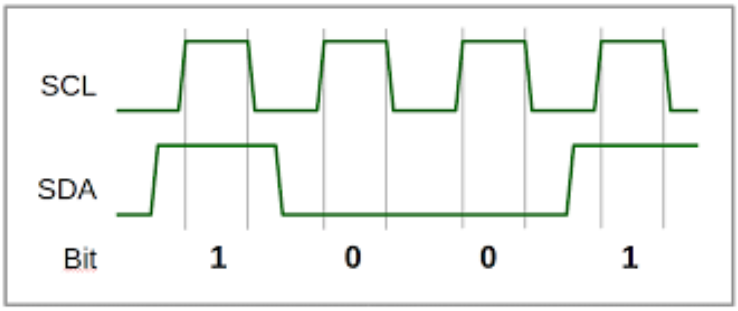
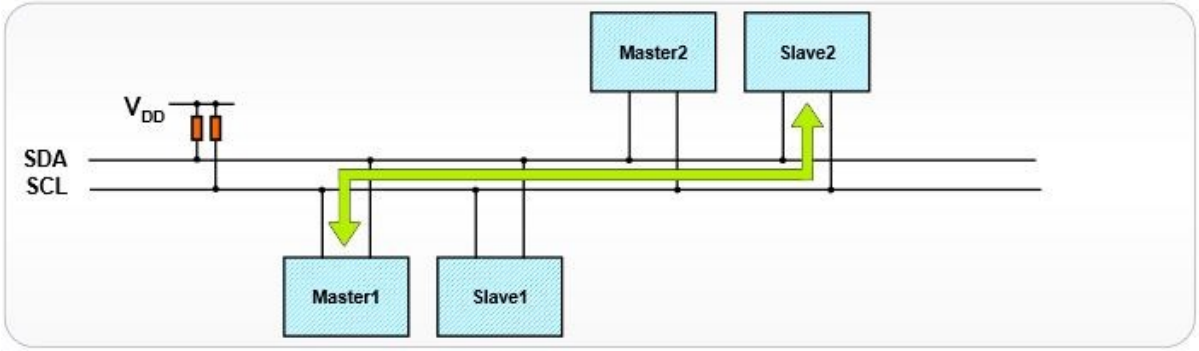


Figure 69 - Diagramme des signaux SDA et SCL de l'interface I2C

De nouvelles spécifications sont nées pour améliorer la vitesse de communication. D'abord un Fast Mode à 400 kbit/s, puis un Fast Mode plus (FM+) à 1 Mbit/s. Depuis 1998, il y a une version High Speed à 3,4 Mbit/s. Le débit maximal possible via un bus I^2C est spécifié dans l'Ultra Fast mode à 5 Mbit/s, mais avec un fonctionnement un peu particulier.

Caractéristiques principales du bus I^2C :



Note: Les résistances pull-up de quelques k ohms sont nécessaires si plusieurs modules I2C sont connectés sur le bus.

- seulement deux lignes (bidirectionnelles) sont nécessaires, une pour transporter les données - SDA -, une autre pour l'horloge de synchronisation - SCK - (1 bit échangé à chaque top d'horloge) ;
- la transmission est synchrone. Pas besoin de spécifier une vitesse de transfert comme pour la liaison RS232. Ici, le périphérique maître (master) génère le signal d'horloge qui synchronise et cadence les échanges ;
- la relation entre les périphériques du bus est de type maître-esclave (master/slave). Le maître est à l'initiative de la transmission et s'adresse à un esclave (ou tous les esclaves) ;

- chaque périphérique sur le bus I^2C est adressable, avec une adresse unique pour chaque périphérique du bus ;
- le protocole I^2C gère le fonctionnement multi-maître (multi-master), plusieurs périphériques maîtres peuvent prendre simultanément le contrôle du bus (système d'arbitrage des maîtres et gestion des collisions).

La distance maximale maître-esclave pour un bus I^2C est d'environ un mètre et dépend de plusieurs facteurs comme la capacité électrique du bus ou le débit de transmission. Cette distance peut être sensiblement augmentée en passant par des interfaces spécifiques (un i2c-bus extender amplifie le courant débité par les broches SDA et SCL, ce qui augmente la portée du signal).

Projets à réaliser

P1:

Réaliser un circuit ainsi que le programme qui répondent aux spécifications suivantes.
1/ Nous aurons besoin du matériel suivant:

- Un clavier de 16 touches
- Un écran LCD 1602
- Un capteur de lumière
- Un capteur de température et d'humidité
- Un module RTC DS1307
- Un contrôleur arduino

2/ Le programme va fonctionner comme suit:

- L'écran LCD affichera la température lorsque la touche A est appuyée
- L'écran LCD affichera l'humidité lorsque la touche B est appuyée
- L'écran LCD affichera l'intensité de lumière lorsque la touche C est appuyée
- L'écran LCD affichera la date et l'heure lorsque la touche D est appuyée

Notions d'électricité

Un montage électrique a toujours besoin d'une source d'alimentation qui fournit une tension et l'intensité du courant.
Exemple: une batterie 9 volts fournit une tension de 9 volts et 200mAh?.
Pour faire fonctionner un circuit électrique il faut l'alimenter.
Dans le cas de l'arduino, il est possible de l'alimenter soit en utilisant:
- une batterie de 9 volts
- un adaptateur électrique
- PC à travers un bus USB

La plaquette arduino va convertir la source de tension en:
- +5 volts
- +3.3 volts

L'arduino UNO fonctionne en 5 volts. D'autres plateformes fonctionnent en 3.3 volts. La plaquette arduino produit du 3.3 volts pour pouvoir s'interfacer avec des cartes alimentées en 3.3 volts. Exemple: module Bluetooth HM10, module wifi ESP8266, etc.

1) La résistance

La résistance est un composant qui permet de limiter le courant qui circule dans le circuit électrique. Exemple: la Led nécessite seulement environ 20mA. Si on la branche directement entre les bornes de la batterie de 9 volts, elle va se détruire. Pour éviter de l'endommager, nous plaçons une résistance pour ne laisser passer que 20mA.
Nous devons calculer la valeur de la résistance R. Nous utilisons la loi d'Ohm dont la formule est la suivante: V = R x I. La tension V (en volts) est proportionnelle au produit de la résistance R (en ohms) et de l'intensité du courant I (en ampères) qui traverse le circuit.

a- Schéma et symboles

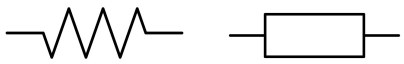


Figure 70 - Représentation classique de la résistance

Figure 71 - Une autre forme de résistance sous forme de chip

Ce circuit (Figure 71) représente une résistance de $10^3\Omega$.

b- Comment connaître la valeur d'une résistance

La valeur d'une résistance peut être connue en déchiffrant les couleurs sur la résistance:

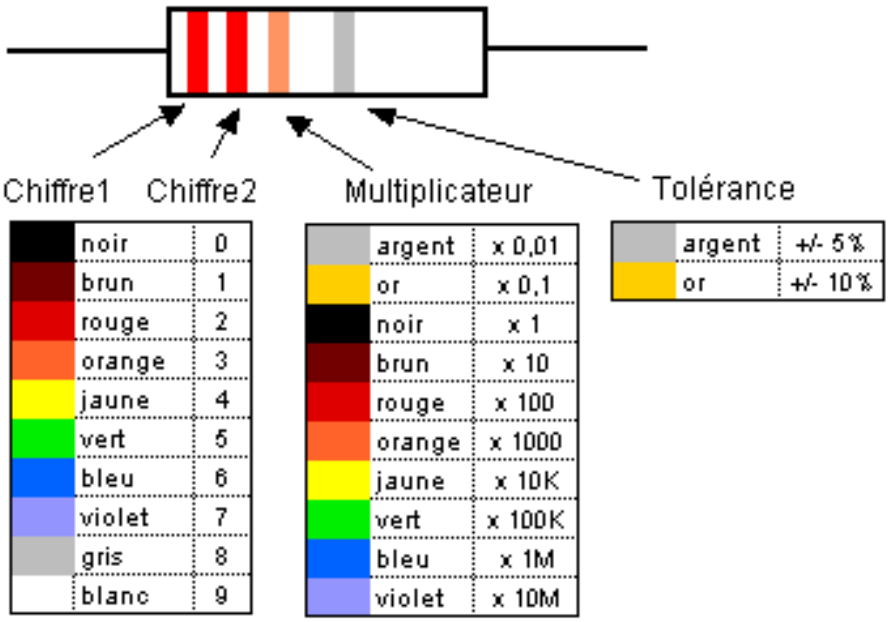


Figure 72 - Le code couleur des résistances

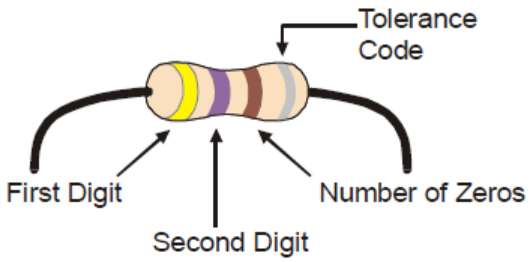
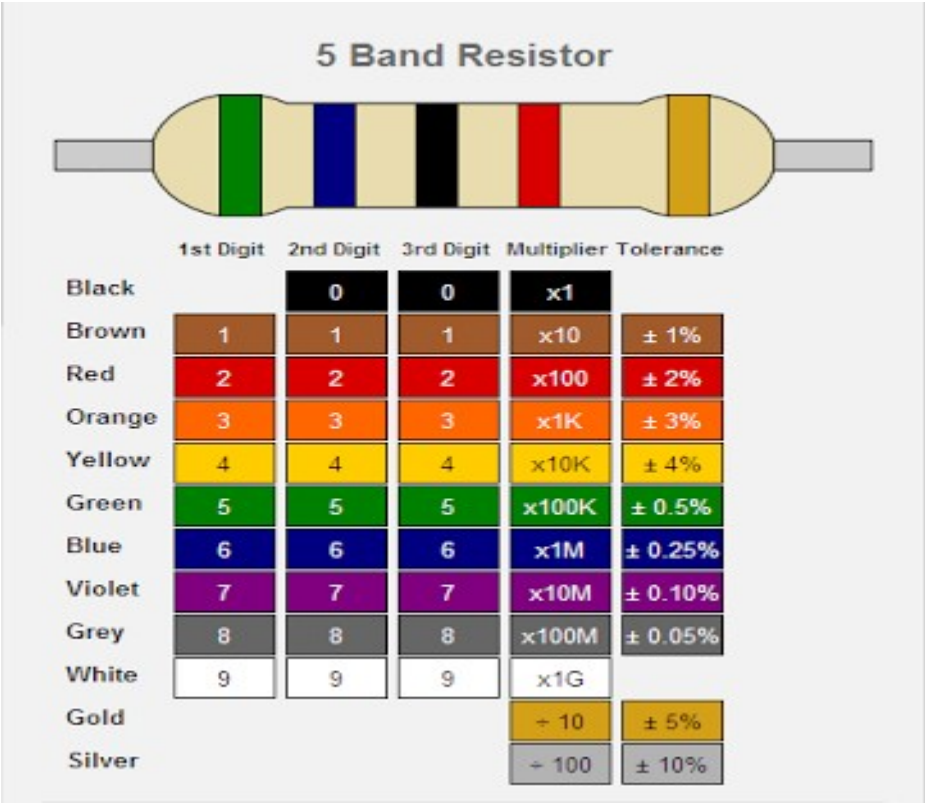


Figure 73 - Résistance de 470 ohms

Le circuit (Figure 71) représente une résistance de $10x10^3\Omega = 10k\Omega$. Pour plus de détails sur ce type de résistance (appelées résistances SMD), il est préférable de se référer à la norme EIA-96.

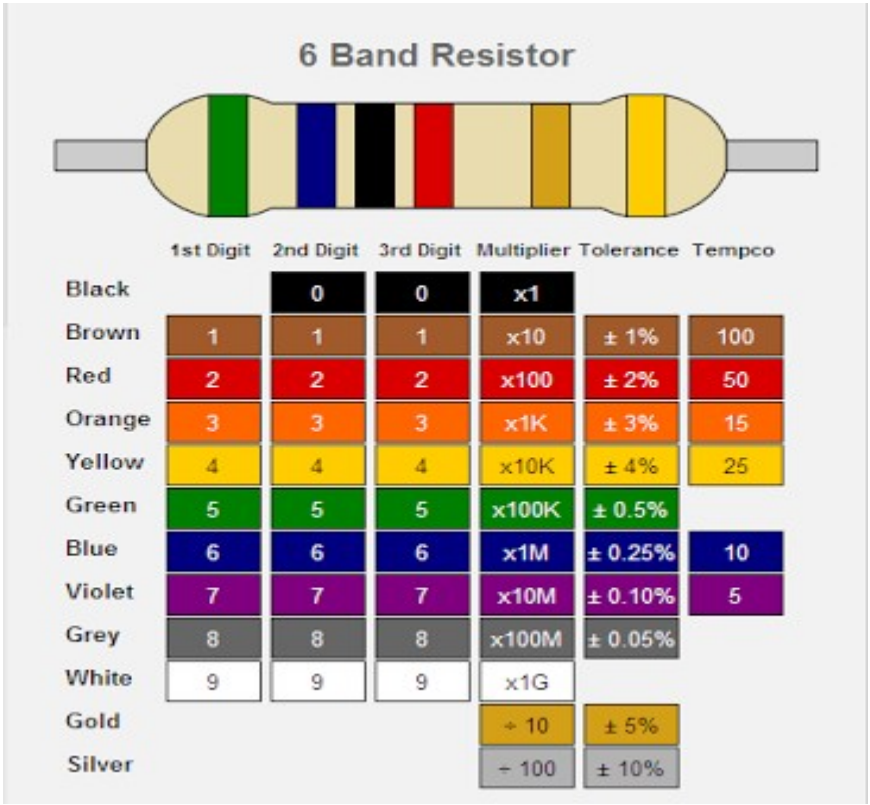
c- calcul de résistance avec 5 bandes

On utilise le tableau suivant:



d- Résistance 6 bandes

On utilise le tableau suivant:



e- Calcul de la résistance d'un circuit

La loi d'Ohm s'applique: V = R / I.
Exemple: En utilisant une batterie de 9 volts, la valeur de R est:
R = V/I = 9V/ 0,020 = 450 ohms. Nous utilisons une résistance de 470 ohms qui est une valeur standard.
Si le même circuit est alimenté en 5 volts, alors R = 5/0,02 = 250 ohms. Nous pouvons utiliser une résistance de 220 ohms, vu que 250 n'est pas une valeur standard.

1) Disposition des résistances en série:
Nous pouvons combiner en mettant plusieurs résistances en série, pour avoir une résistance équivalente dont la valeur est: $R = R_1 + R_2 + R_3 + ... + R_n$.
Exemple de 2 résistances: $R = R_1 + R_2$.

2) Disposition des résistances en parallèle:
Si on dispose plusieurs résistances en parallèle, la valeur de la résistance équivalente est calculée comme suit: $1/R = 1/R_1 + 1/R_2 + 1/R_3 + ... + 1/R_n$.

f- Calcul du courant I d'un circuit

La loi d'Ohm s'applique: V = R / I.
Le calcul utilise donc la fonction I = V/R, I exprimée en ampères (A), V en volts (V), R en ohms.

Exercices

1- Mettre 2 résistances en série et calculer la résistance équivalente. R1=220 ohms; R2=330 ohms. Mesurer avec un contrôleur et vérifier.
2- Mettre 2 résistances en parallèle et calculer la résistance équivalente. R1=220; R2=330. Mesurer avec un contrôleur et vérifier.

2) Le condensateur

Le condensateur est un composant électrique capable d'emmagasiner une charge électrique.
Il y en a 2 types: le condensateur céramique (non polarisés) et le condensateur électrolytique (polarisés).
Les valeurs sont de 1000 µF à 1 µF pour les condensateurs polarisés et de 1µF à 1 pF pour les condensateurs non polarisés .

a- Schéma et symboles

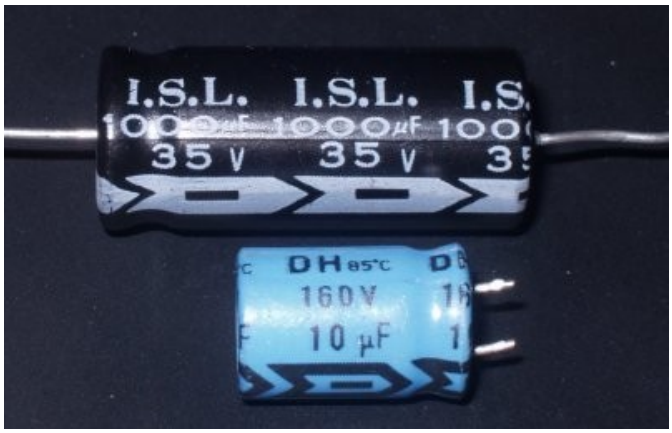


Figure 74 - Condensateurs électrolytiques (polarisés)

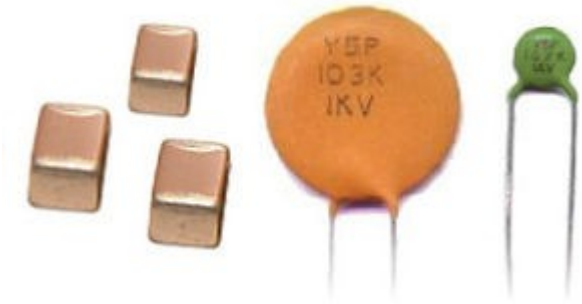


Figure 75 - Condensateurs céramique (non polarisés)

b- Comment lire la valeur d'une capacité

La valeur d'une capacité se mesure en Farads (F). Mais la plupart des capacités sont évaluées en microfarads (µF).
D'autres fabricants indiquent la valeur en picofarads (pF) avec un code de 3 digits. 1 Farad = 1000000 µFarads. Le 3ème chiffre donne le nombre de 0 zéros. Les chiffres 6 et 7 ne sont pas utilisés. 8 indique de multiplier le nombre formé des 2 premiers chiffres par 0.01. 9 indique de multiplier le nombre formé des 2 premiers chiffres par 0.1.
Exemple: 104 donne 10 x 10^4 pF = 100000 pF = 100 nF = 0.1 µF.
339 = 33 x 0.1 pF = 3.3 pF.

c- Calcul de la capacité équivalente

1) Disposition des capacités en série:
Nous pouvons combiner en mettant plusieurs résistances en série, pour avoir une résistance équivalente dont la valeur est: $1/C = 1/ C_1 + 1/C_2 + 1/C_3 + ... + 1/C_n$.
2) Disposition des capacités en parallèle:
Si on dispose plusieurs résistances en parallèle, la valeur de la résistance équivalente est calculée comme suit: $C = C_1 + C_2 + C_3 + ... + C_n$.

3) Le transistor

Réf: http://www.pixelab.org/theoprat/elektronik/C4.htm#Transistors

3-1) Le transistor 2N2222

Le transistor est utilisé lorsque le circuit électrique nécessite plus de courant que celui qui est fourni par l'arduino. Comme la pin de l'arduino peut fournir 40 mA, il est nécessaire d'amplifier le courant lorsque les composants commandés demandent plus de courant que 40 mA. Le 2N2222 peut fournir jusqu'à 800mA jusqu'à 40 volts.

4) La bobine

5) Le relais

Un relais électronique est un interrupteur qui se commande avec une tension continue de faible puissance. La partie interrupteur sert à piloter des charges secteur de forte puissance (jusqu'à 10A couramment).

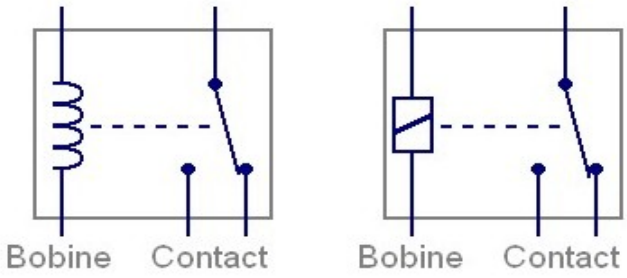


Figure 76 - Symboles du relais

Le symbole du relais doit inclure ce qui représente la bobine et ce qui représente le ou les interrupteurs (contacts).

Lorsqu'on choisit ou qu'on récupère un relais, il peut être intéressant de savoir de quel type de connexion il s'agit. Il existe 4 grandes familles de relais, en fonction des contacts.

- Relais SPST : Single Pole Single Throw. Le relais SPST possède 2 broches de contacts. Dans ce cas, le relais possède 4 broches au total : 2 pour les contacts, 2 pour la bobine.

- Relais SPDT : Single Pole Double Throw. Le relais SPDT possède un seul contact mais avec une borne commune, un contact normalement ouvert (quand il n'y a pas de tension sur la bobine) et un contact normalement fermé (quand il n'y a pas de tension sur la bobine). Quand on applique une tension sur la bobine, on entend "clic" : la borne commune va se connecter sur le contact normalement ouvert (NO = normally open) et le contact normalement fermé (NC = normally closed) s'ouvre. Dès qu'on coupe la tension aux bornes de la bobine, on entend "clic" et le relais revient à son état de repos. On peut ainsi basculer d'un circuit à l'autre (allumer soit l'ampoule rouge soit l'ampoule verte par exemple).

Le relais SPDT possède 5 broches au total : 3 pour les contacts, 2 pour la bobine. Si on n'utilise pas le contact "normalement fermé" le relais SPDT est équivalent à un relais SPST.

Relais DPST : Double Pole Single Throw. Le relais DPST est équivalent à 2 relais SPST qui fonctionnent ensemble, pilotés par la même bobine. On peut par exemple commuter 2

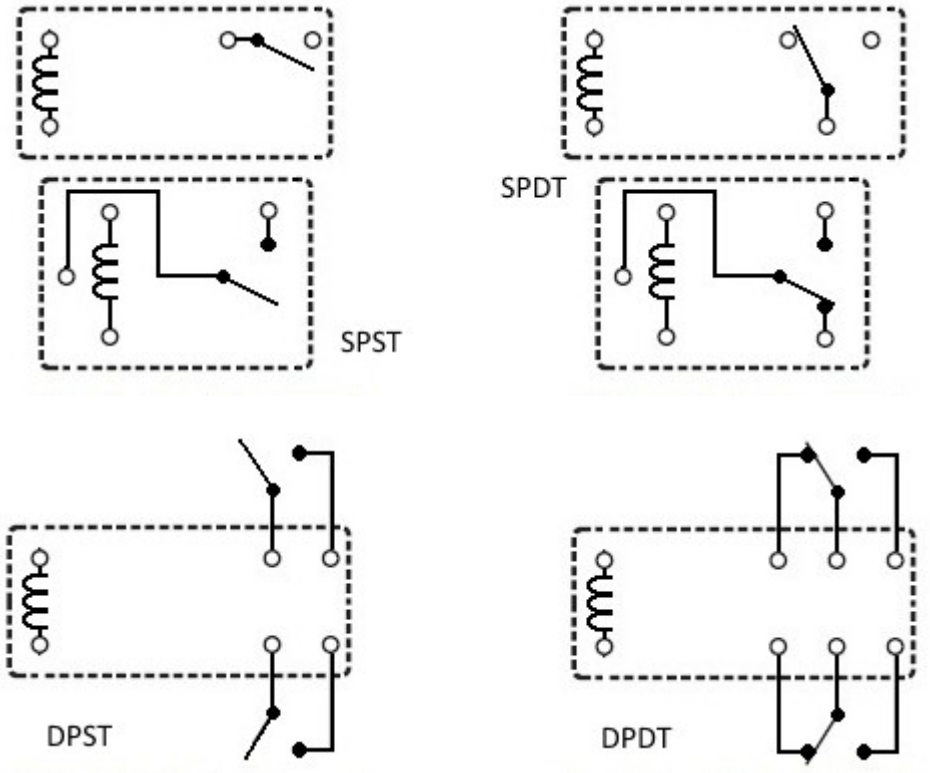
circuits indépendants en même temps par la même action (mettre du 12V sur la bobine par exemple), ou commuter à la fois phase et neutre pour brancher un circuit électrique au secteur.

Le relais DPST possède 6 broches au total : 2x2 pour les contacts, 2 pour la bobine.

Relais DPDT : Double Pole Double Throw. Le relais DPDT est équivalent à 2 relais SPDT qui fonctionnent ensemble, pilotés par la même bobine. Le relais DPDT possède 8 broches au total : 2x3 pour les contacts, 2 pour la bobine.

La lettre "S" ou "D" peuvent être remplacées par le nombre de contacts pilotés ensemble. Par exemple : 4PDT signifie 4 ensembles DT (12 bornes de contacts + 2 bornes de bobine). A la place de "DT", on rencontre parfois "CO" (change-over)

Pour récapituler ces 4 types de relais :



Tension de bobine du relais

Pour chaque relais, il existe une tension de bobine. Elle peut être continue (5V, 12V, etc.) ou alternative (110V, 230V). C'est la tension qu'il faut appliquer sur la bobine pour faire commuter le relais (vérifier le son du "clic"). Le courant consommé dépend du type de relais et est inversement proportionnel à la tension de bobine. Dans une gamme de relais (5V, 12V, 24V, etc.), la puissance de la bobine est constante.

Par exemple : 12V 40mA, 24V 20mA, 48V 10mA, etc.

Pour déterminer le courant consommé par la bobine, il suffit de mesurer la résistance à l'ohmmètre et déduire le courant. C'est utile pour dimensionner l'alimentation de la commande (montage qui pilote la bobine du relais).

Si on mesure 1410 Ohms sur une bobine de relais 24V, le courant consommé par la bobine sera de 24V/1410 Ohms = 17mA. Ce sera proportionnel à la tension appliquée, loi d'Ohm oblige.

Sur la plupart des relais, il existe une large tolérance sur la tension de bobine. La tension va souvent de 70% à 150% (ou même 200%) de la tension nominale. Une tension trop faible ne permet pas de garantir un bon contact. Une tension trop élevée va faire brûler la bobine.

Exemples :

Relais alimenté en 12V : 8.4V - 18V
Relais alimenté en 5V : 3.5V - 7.5V

On n'a pas besoin d'une alimentation régulée de précision pour faire commuter un relais.

Le pilotage de la bobine par un transistor bipolaire NPN (le plus classique) nécessite l'ajout d'une diode de roue libre en parallèle avec la bobine du relais : http://www.astuces-pratiques.fr/electronique/la-diode-de-roue-libre

Polarité de la tension de bobine

Sur la plupart des relais, la tension continue peut être appliquée dans n'importe quel sens sur la bobine. Il existe de très rares exceptions où la bobine du relais est polarisée. Si on applique la tension à l'envers, le relais ne commute pas.

Charges pilotables par les relais

Les relais peuvent souvent commuter un courant élevé sur de l'alternatif (250V AC), en revanche, ils ne commutent pas de tension supérieures à 20 ou 30V continus environ. Bien sûr, cela dépend des modèles, mais il faut retenir que commuter une charge en tension et courant continu est une opération délicate (coupure de courant issu de panneaux solaires ou de batteries). L'arc engendré à l'ouverture du contact ne peut pas s'éteindre spontanément comme dans le cas d'un courant alternatif qui passe par zéro deux fois par période...

Relais statiques

Il existe des relais statiques : ce sont des composants qui ont la même fonction qu'un relais à bobine et contacts mais sans élément mécanique. Le contact est assuré par la conduction d'un triac piloté par opto-coupleur pour garantir l'isolement électrique entre commande et puissance :

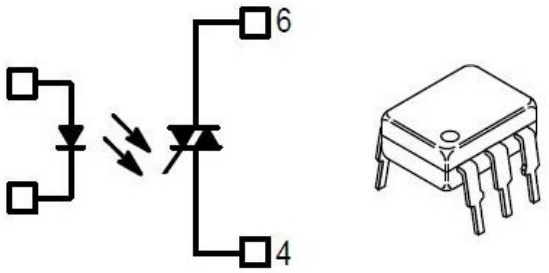


Figure 77 - Relais statique

Le courant maximum du côté des contacts est plus faible : un triac possède une chute de tension de 1V environ contrairement à des contacts d'un relais mécanique à bobine. Il y a aussi un très léger courant de fuite lorsque la LED n'est pas pilotée (LED éteinte). Ce courant est de 1nA à 100nA couramment.

Exemple de relais statique : MOC3041 et MOC3042 (Indiquer la sortie de courant max et typique).

Défaillance des relais

Lorsqu'un relais ne fonctionne plus, il se peut qu'on entende toujours le "clic". On peut penser que le relais est encore bon et chercher la panne ailleurs. Pourtant, malgré le "clic", il ne se produit plus de contact. C'est parfois un piège !

On peut aussi vérifier la continuité de la bobine en mesurant à l'ohmmètre la résistance entre les 2 broches de la bobine (valeurs courantes entre 100 Ohms et quelques kOhms).

Application des relais

Les applications sont nombreuses. Les relais peuvent piloter de nombreuses charges sur le secteur, grâce à une tension isolée (non dangereuse) qu'on applique sur la bobine : chauffage électrique, luminaires…

Et toutes les applications où des moteurs sont commutés : porte de garage, lit motorisé, électroménager, etc..

Les relais peuvent aussi piloter commuter les haut parleurs d'un ampli pour éviter le ploc lorsqu'on allume l'ampli.

6) La led

Caractéristiques électriques d'une LED

Une LED fonctionne en courant et tension continus. Comme une diode classique, la LED ne conduit que dans un sens. Dans l'autre sens, elle est bloquée mais ne supporte pas des tensions inverses élevées (souvent 5V max). Lorsqu'une LED est passante, il s'établit à ses bornes une tension assez indépendante du courant : on peut appeler cette tension la tension de seuil. Elle dépend du matériau utilisé dans la LED, et donc de sa couleur.

LED rouge, orange, jaune (ambre) : 1.8V à 2V
LED verte standard (vert clair) : 1.8V à 2.2V

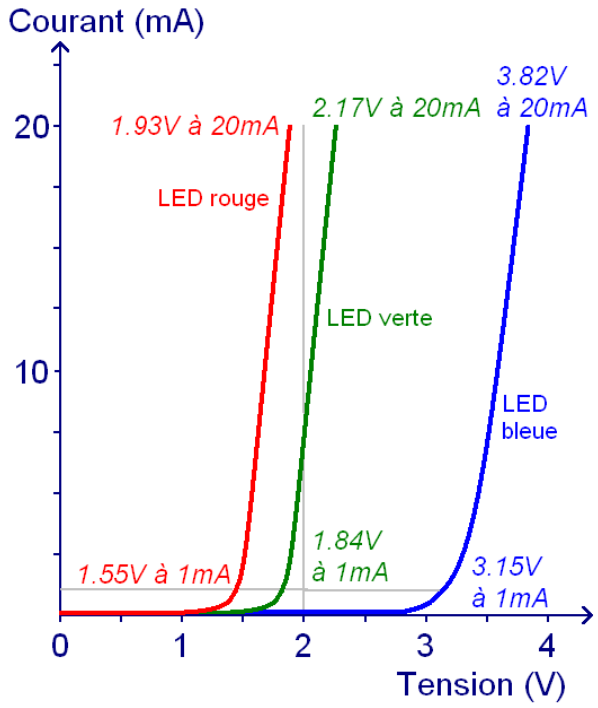


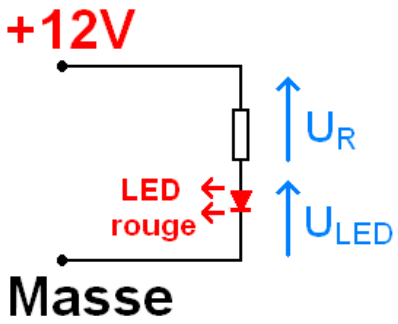
Figure 78 - Caractéristique typique pour LED rouge, LED verte et LED bleue

La tension aux bornes de la LED rouge ne varie que de 1.55V à 1.93V quand on fait varier le courant de 1mA à 20mA. Si on dispose d'une alimentation stabilisée réglable fixée à 1.93V, on pourrait y brancher directement une LED rouge. Il faut éviter de faire ainsi parce **qu'une petite variation de tension entraîne une grande variation de courant.**
En passant de 1.93V à 2.20V, le courant dans la LED est multiplié par 2 !

Calcul de la résistance série pour la LED

Prenons un exemple : à partir d'une **alimentation 12V**, on souhaite allumer une **LED rouge qui supporte 20mA**. Sa tension de seuil est de **1.8V**.

L'additivité des tensions s'écrit : ULED + UR = 12V



ULED est égale à la tension de seuil (supposée indépendante du courant).
La tension UR vaut donc : UR = 12V - ULED = 12V - 1.8V = 10.2V
Or, le courant dans la résistance est égal à celui de la LED, c'est-à-dire 20mA.
La valeur de la résistance se déduit à partir de ces *deux* informations :
R = 10.2V / 20mA = 10.2V / 0.02A = 510 Ohms

Cette valeur correspond à une valeur standard. On retiendra donc R = 510 Ohms. Si la valeur trouvée n'est pas standard, choisir la valeur standard supérieure la plus proche (exemple : pour 637 Ohms, on choisit 680 Ohms).

La formule générale pour calculer la résistance s'écrit donc :

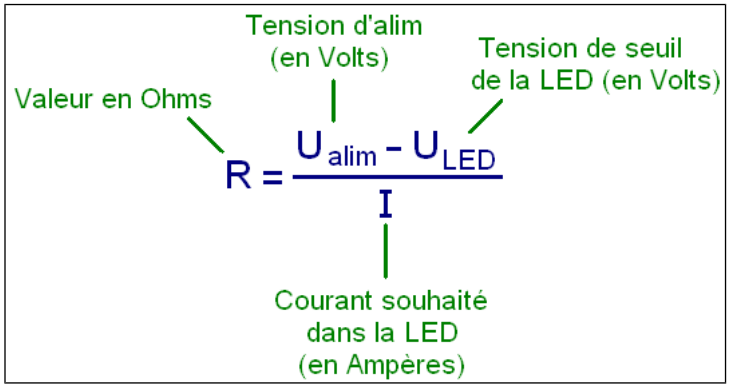


Figure 79 - Formule de calcul de la résistance d'une LED

Dissipation dans la résistance

La dissipation de chaleur est à considérer pour des tensions d'alimentations élevées comme 24V. La puissance dissipée vaut :

$$P = \frac{(U_{alim} - U_{LED})^2}{R}$$

Liste de capteurs et actuateurs du monde des microcontrôleurs

1. Sonar à ultrasons (HC-SR04)

Le capteur sonar à ultrason est utilisé pour déterminer la distance qui le sépare d'un obstacle. Il offre une bonne précision, de quelques cm (2 cm) jusqu'à 3-4 m.

Figure 80 - Capteur sonar à ultrasons

2. Capteur IR détecteur d'obstacle

C'est un détecteur d'obstacle à infrarouges. La distance par rapport à l'obstacle est entre 2 et 30 cm, réglable manuellement à l'aide du potentiomètre. L'angle de détection est de 35°. La sortie est activée lorsque la distance réglée est atteinte.

Figure 81 - Capteur IR détecteur d'obstacle (à réglage manuel)

3. Capteur d'humidité du sol

Ce module sert à détecter l'humidité du sol. Il fournit un signal haut lorsque le sol n'est pas humide (selon un seuil à définir et régler). Peut facilement servir à irriguer les plantes d'un jardin de façon automatique.

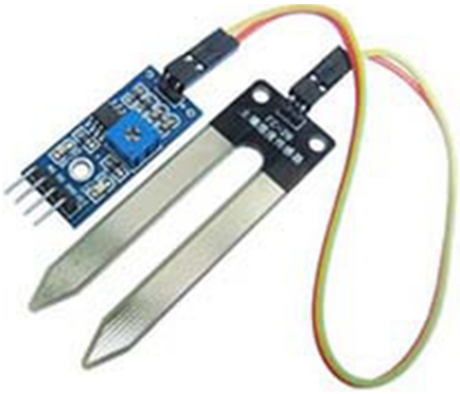


Figure 82 - capteur et sonde d'humidité du sol

Exemple de programme:

```
const int VAL_PROBE = 0; // pin Analog 0
const int MOISTURE_LEVEL = 250; // seuil à définir

void setup() {
   Serial.begin(9600);
}
void loop() {
   int moisture = analogRead(VAL_PROBE);
   Serial.println(moisture);
   if(moisture > MOISTURE_LEVEL) {
      digitalWrite(13, HIGH);
   } else   {
      digitalWrite(13, LOW);
   }
   delay(100);
}
```

4. Capteur de son

Ce capteur de son fournit deux sortie: AO (sortie analogique) donnant la tension du signal capté par le microphone; DO (sortie digitale) passe à 1 lorsque l'intensité du son atteint un certain seuil de son qui, lui, est réglé manuellement via le potentiomètre.

Figure 83 - Capteur de son

Exemple de programme:

```
//SOUND Sensor

const int sensorPin = 4;    //Microphone Sensor Pin on analog 4
const int YellowLED = 1;
const int RedLED = 4;
int sensorValue = 0;

void setup() {
        pinMode(YellowLED, OUTPUT);
        pinMode(RedLED, OUTPUT);
        Serial.begin(9600);
}

void loop() {
        // Lire la valeur du capteur
        sensorValue = analogRead(sensorPin);
        Serial.println(sensorValue);

/* If you feel that the sensor values on the serial monitor
are going by too quickly for you to read, you can make it
so that the values only show up on the serial monitor
if you make a noise.  You can replace
Serial.println(sensorValue);
  with:
if (sensorValue < 509){
   Serial.println(sensorValue);
}
*/

        digitalWrite(YellowLED, HIGH);  //Yellow is always on.
        if (sensorValue>25) {  //The 'silence' sensor value is 509-511
                digitalWrite(RedLED,HIGH);
                delay(15 );          // The red LEDs stay on for 2 seconds
        } else {
                digitalWrite(RedLED,LOW);
```

```
        }
}
```

5. Capteur digital de pression barométrique

You can use this module to measure the absolute pressure of the environment using a digital barometer such as this has some interesting applications. By converting the pressure measures into altitude, you have a reliable sensor for determining the height of your robot or projectile for example.

6. Photoresistor Sensor Module Light Detection Light

You can use this module for light detection with an Arduino.

7. Digital Thermal Sensor Module Temperature Sensor Module

The thermal sensor module is very sensitive to the ambient temperature, generally used to detect the ambient temperature. Through the adjustment of the

potentiometer, you can change the temperature detection threshold.

8. Rotary Encoder Module Brick Sensor Development Board

When you rotate the rotary encoder it counts in the positive direction and the reverse direction. Rotation counts are not limited. With the buttons on the rotary encoder, you can reset to its initial state and start counting from 0.

9. MQ-2 MQ2 Gas Sensor Module Smoke Methane Butane Detection

This sensor module utilizes an MQ-2 as the sensitive component and has a protection resistor and an adjustable resistor on board. The MQ-2 gas sensor is sensitive to LPG, i-butane, propane, methane, alcohol, Hydrogen and smoke. It could be used in gas leakage detecting equipments in family and industry. The resistance of the sensitive component changes as the concentration of the target gas changes.

10. SW-420 Motion Sensor Module Vibration Switch Alarm

This module can be used to trigger the effect of various vibration, theft alarm, intelligent car, earthquake alarm, motorcycle alarm, etc.

11. Humidity and Rain Detection Sensor Module

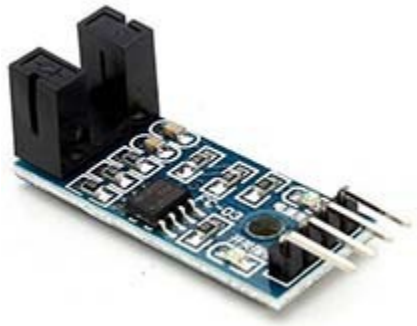
This is a rain sensor. It’s used for all kinds of weather monitoring.

12. Passive Buzzer Module

A simple sound making module. You set high or low to drive this module by changing the frequency you’ll hear a different sound.

13. Speed Sensor Module

Tachometer is a simple module that allow you to test the speed of the motor. Widely used for motor speed detection, Pulse count, position limit, and so on.



14. IR Infrared Flame Detection Sensor Module

Flame Detection Sensor Module is sensitive to the flame, but also can detect ordinary light. Usually used as a flame alarm. Detects a flame or a light source of a wavelength in the range of 760nm-1100 nm. Detection point of about 60 degrees, particularly sensitive to the flame spectrum.

Figure 84 - Capteur de flamme (ou lumière)

This module is sensitive to the flame- but also can detect ordinary light. Usually used as a flame alarm. On board- digital output interface can be directly connected with the microcontroller IO.

Feature:

Detects a flame or a light source of a wavelength in the range of 760nm-1100 nm

Detection angle about 60 degrees- it is sensitive to the flame spectrum.
Accuracy adjustable
Operating voltage 3.3V ~ 5V
Output
 -Analog voltage output
 -Digital switch outputs (0 and 1)
With a mounting screw hole
Power indicator (red) and digital switch output indicator (green)
Comparator chip LM393- it is stable
Flame detection distance- lighter flame test can be triggered within 0.8m- if the intensity of flame is high- the detection distance will be increased.

Interface(4-wire):

VCC: 3.3V ~ 5V
GND: GND
DO: board digital output interface (0 and 1)
AO: board analog output interface
Size: 3.2 x 1.9cm

Exemple de programme:

```
// lowest and highest sensor readings:
const int sensorMin = 0;     // sensor minimum
const int sensorMax = 1024;  // sensor maximum

void setup() {
        // initialize serial communication @ 9600 baud:
        Serial.begin(9600);
}
void loop() {
        // read the sensor on analog A0:
        int sensorReading = analogRead(A0);
        // map the sensor range (four options):
        // ex: 'long int map(long int, long int, long int, long int, long int)'
        int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

        // range value:
        switch (range) {
                case 0:    // A fire closer than 1.5 feet away.
                        Serial.println("** Feu proche **");
                        break;
                 case 1:    // A fire between 1-3 feet away.
                        Serial.println("** Feu distant **");
                        break;
                case 2:    // No fire detected.
                        Serial.println("Pas de feu");
                        break;
        }
        delay(10);  // delay between reads
```

}

15. 5V 2-Channel Relay Module

This is a 2 Channel 5V Relay Module, it allows you to control various appliances, and other equipment with large current. It can be controlled directly by any microcontroller.

16. Breadboard Power Supply Module 3.3V 5V

A simple module to power your breadboard. It supplies both 3.3V and 5V.

17. Module HC-SR501 (Pyroelectric Infra-Red) détecteur de mouvement

Un capteur PIR est un capteur détecteur de mouvement produit par un humain ou un animal.

Figure 85 - Module PIR de détection de mouvement

18. Module de calcul d'accélération

Ce module sert à mesurer l'accélération d'un objet en mouvement.

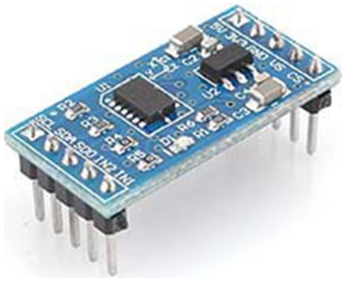


Figure 86 -Module d'accélération

19. DHT11 Temperature and Humidity Sensor

Sert à mesurer les température et humidité à l'intérieur ou à l'extérieur d'un lieu.

Figure 87 - Module DHT11

20 or 21. RF 433MHz Transmitter/Receiver

Ces modules servent à échanger des données avec ondes radio de fréquence 433 MHz.

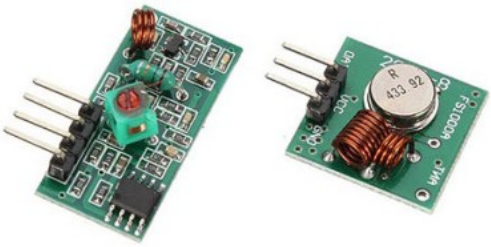
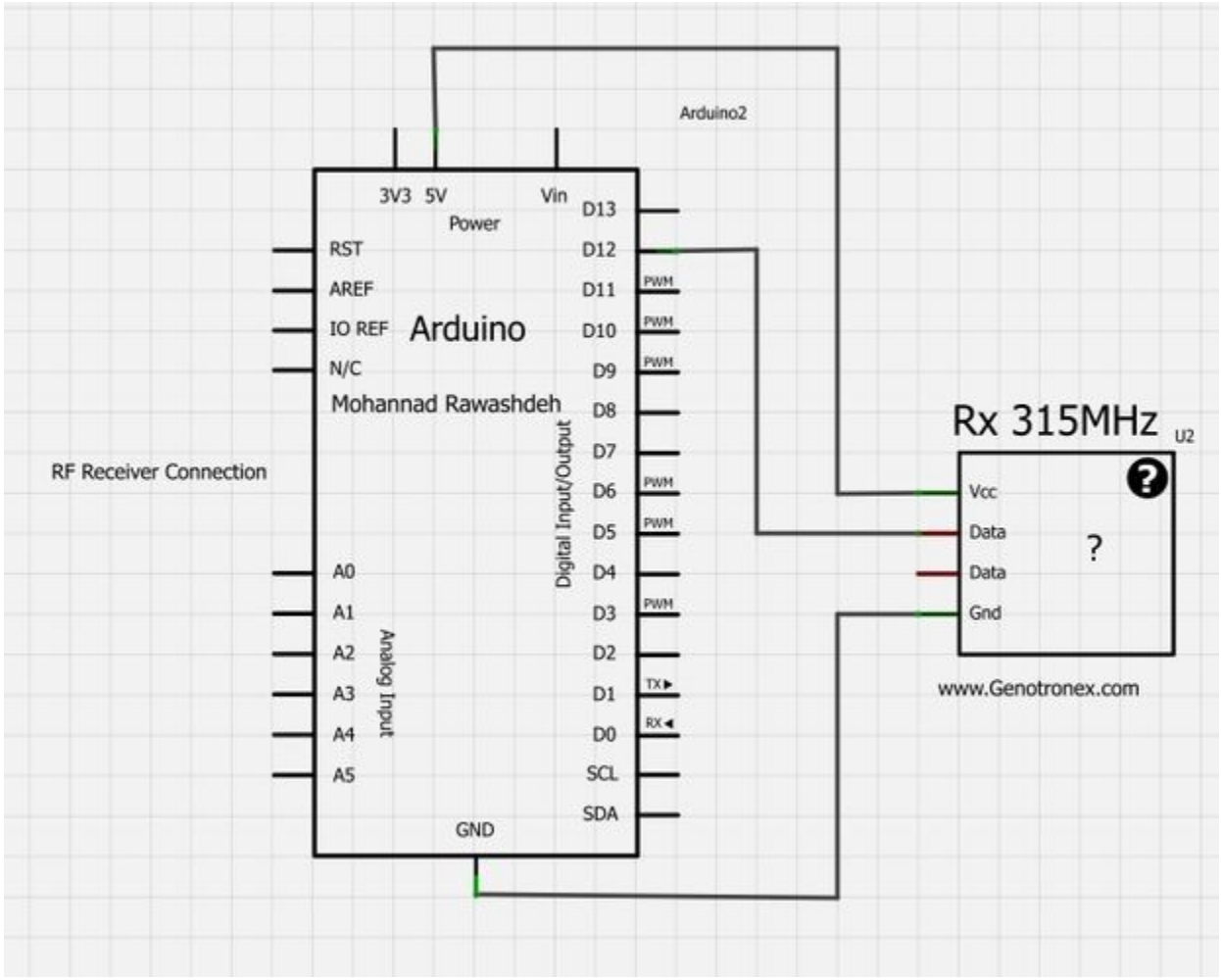
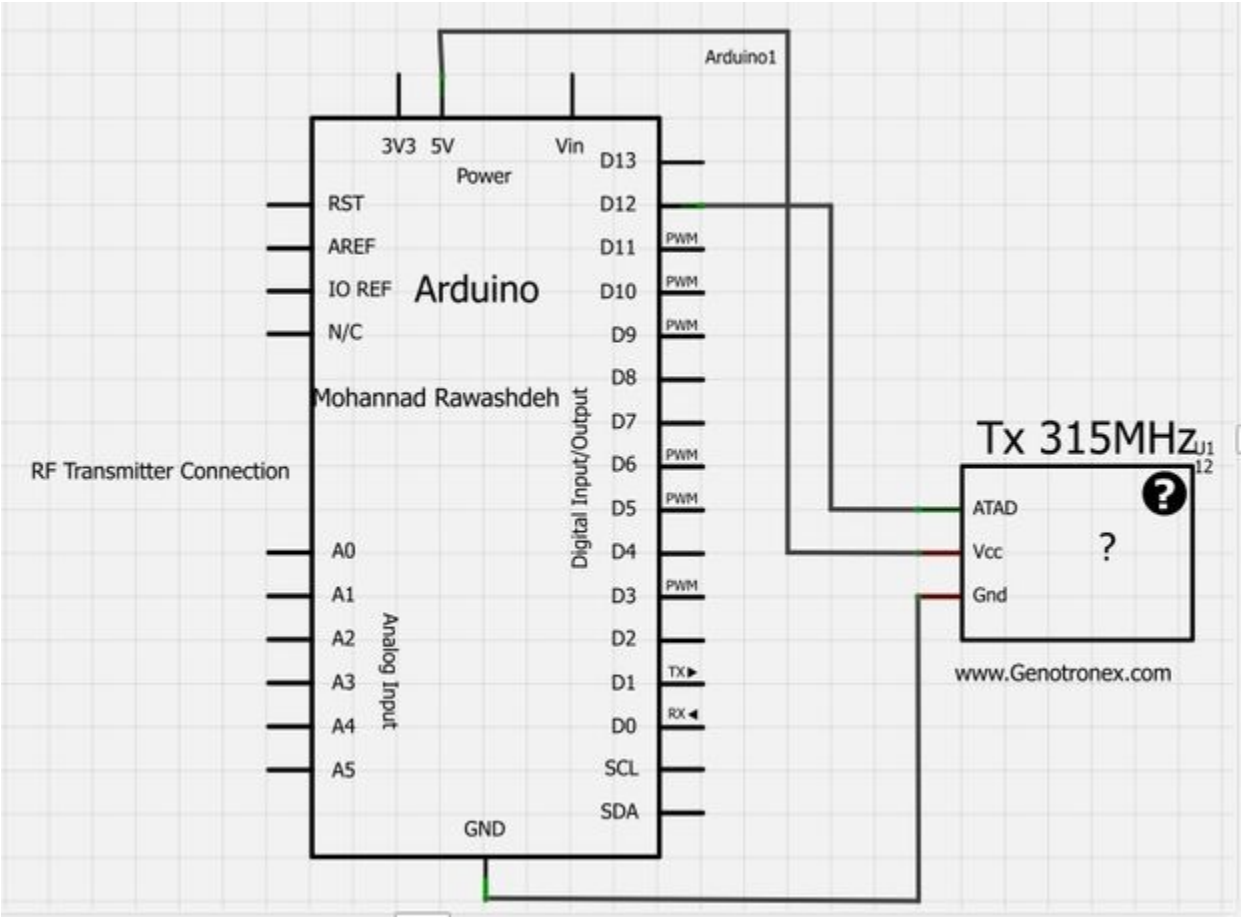


Figure 88 - 433 Mhz radio modules

Connexions:


Arduino2
3V3
5V
Vin
Power
RST
AREF
IO REF
N/C
Arduino
Mohannad Rawashdeh
RF Receiver Connection
A0
A1
A2
A3
A4
A5
Analog Input
GND
D13
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
D0
SCL
SDA
Digital Input/Output
PWM
TX
RX
Rx 315MHz
U2
Vcc
Data
Data
Gnd
?
www.Genotronex.com



Connexions de l'émetteur :

Côté émetteur RF	Côté arduino
Vcc	5V
ATAD	pin D12
Gnd	Gnd

Connexions du récepteur :

Côté récepteur RF	Côté arduino
Vcc	5V
Data	pin D12
Gnd	Gnd

Utiliser la librairie: Virtual Wire
VirtualWire is an Arduino library that provides features to send short messages, without addressing, retransmit or acknowledgment, a bit like UDP over wireless, using ASK (amplitude shift keying). Supports a number of inexpensive radio transmitters and receivers.
This library allow You to send and receive data"byte" and string easily

Programme d'essai
1/ Transmission
this is a simple code , it will send character '1' and after 2 sec will send character '0' and so on

```
//simple Tx on pin D12
//Written By : Mohannad Rawashdeh
// 3:00pm , 13/6/2013
//http://www.genotronex.com/
//.....................................
#include <VirtualWire.h>
char *controller;
void setup() {
        pinMode(13, OUTPUT);
        vw_set_ptt_inverted(true);     //
        vw_set_tx_pin(12);
        vw_setup(4000);                // speed of data transfer Kbps
}

void loop() {
        controller = "1";
        vw_send((uint8_t *)controller, strlen(controller));
        vw_wait_tx();          // Wait until the whole message is gone
        digitalWrite(13, 1);
        delay(2000);
        controller = "0";
        vw_send((uint8_t *)controller, strlen(controller));
        vw_wait_tx();          // Wait until the whole message is gone
        digitalWrite(13, 0);
        delay(2000);
}
```

2/ Réception

The D13 LED On the arduino board must be turned ON when received character '1' and Turned Off when received character '0'

La led D13 s'allume lorsqu'on reçoit un '1' et s'éteint lorsqu'on reçoit un '0'.

```
//simple Rx on pin D12
//Written By : Mohannad Rawashdeh
// 3:00pm , 13/6/2013
//http://www.genotronex.com/
//.....................................
#include <VirtualWire.h>
void setup()
{
   vw_set_ptt_inverted(true); // Required for DR3100
   vw_set_rx_pin(12);
   vw_setup(4000);            // Bits per sec
   pinMode(13, OUTPUT);
   vw_rx_start();             // Start the receiver PLL running
}

void loop()
{
```

```
        uint8_t buf[VW_MAX_MESSAGE_LEN];
        uint8_t buflen = VW_MAX_MESSAGE_LEN;

         if (vw_get_message(buf, &buflen)) // Non-blocking
        {
                if (buf[0] == '1') {
                        digitalWrite(13,1);
                }

                if (buf[0] == '0') {
                        digitalWrite(13,0);
                }

        }
}
```

Avec la librairie RCswitch:

22. Module Compas

23. Module Gyroscope

24. [Extra] ESP8266 WiFi Module

This module costs a little bit more than $2, but it's totally worth it. This module can send emails, host a web server, control LEDs, etc.

From www.NeuroGoody.com

Robotics - Obstacle Avoidance with IR sensors

L	F	R	M1	M2	M3	M4	Mvt
0	0	0	1	0	0	1	F
0	0	1	1	0	0	1	F
0	1	0	1	0	1	0	RIGHT

0	1	1	0	1	0	1	LEFT
1	0	0	1	0	0	1	F
1	0	1	1	0	0	1	F
1	1	0	1	0	1	0	RIGHT
1	1	1	0	1	1	0	REV

24. Hall switch sensor

25. nRF24L01