# \\admsrv2\Bureau$\etienne.roy\Bureau\fablab\logo_kreoLab3.png A-00 : Utiliser un « breadboard »

# (panneau de montage)

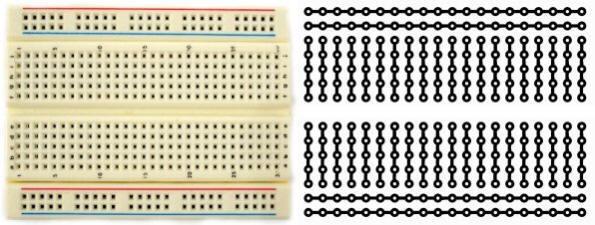
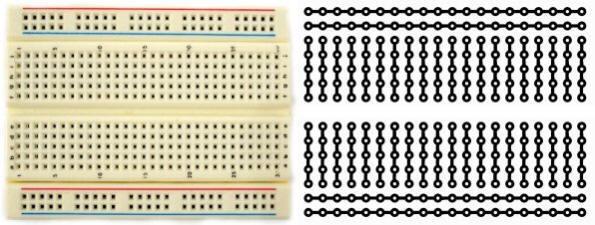
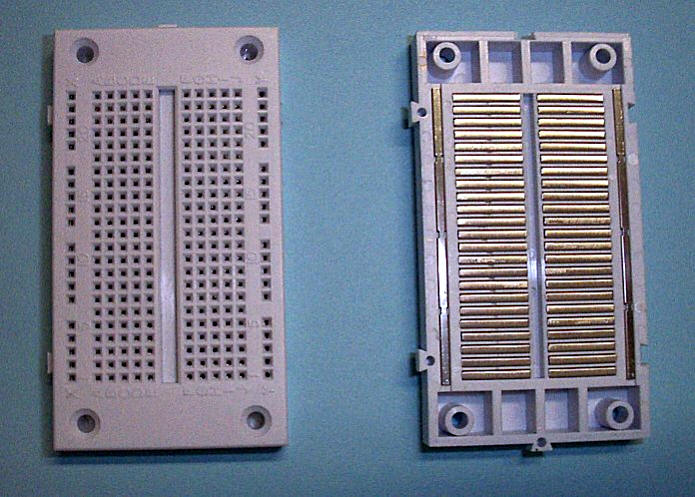
**Difficulté: Débutant**

|  |  |
| --- | --- |
| Apprentissage : **Programmation** : Aucune  **Électricité/Électronique** :   * Fonctionnement d’un panneau de montage. | Matériel  * Aucun |

## Introduction

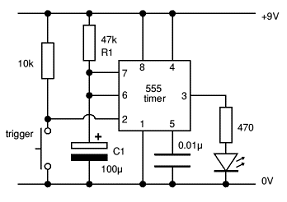
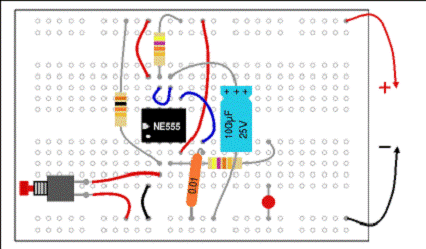
La planche de montage expérimental (appelé en anglais breadboard) est un outil essentiel en électronique. Ce dispositif permet de réaliser le prototype d'un circuit électronique et de le tester. L'avantage de ce système est d'être totalement réutilisable, car il ne nécessite pas de soudure. De plus, il permet de brancher une grande variété de composants afin de réaliser des circuits électroniques, du plus simple circuit au plus complexe.

La planche est recouverte de trous (espacés de 0.1") dans lesquels on insère directement les broches des composants électroniques. Sous ces trous, des languettes métalliques jouent un double rôle. Elles agissent d'abord comme des pinces qui retiennent fermement les broches des composants. De plus, ces languettes permettent de connecter électriquement les broches des composants. La disposition des languettes peut varier suivant le type de breadboard. Par contre, la plupart des modèles de breadboard ont une section plastifiée en leur milieu sur le sens de la longueur. Ces trous sont liés en colonnes et en rangées comme il est illustré dans la partie droite de l'image suivante:



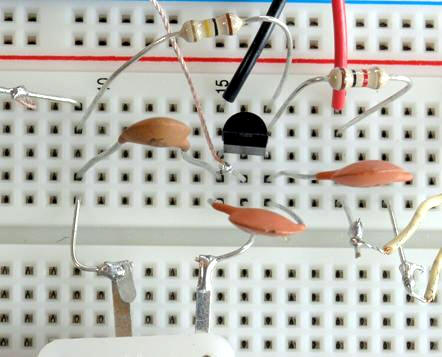
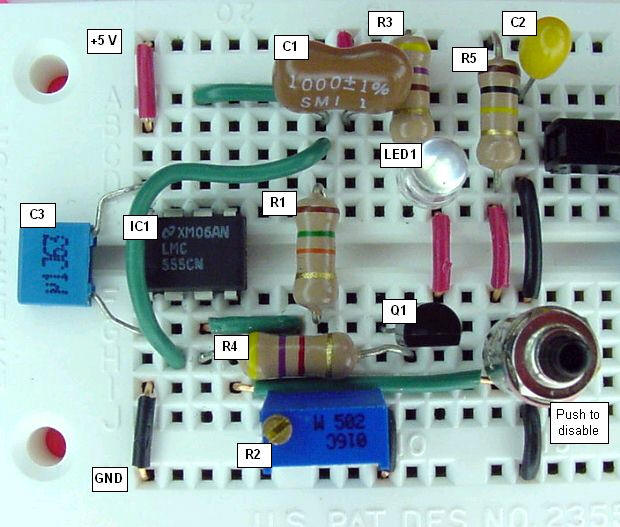
**Comment ça marche?**

Un exemple avec le petit schéma d’un NE555 (timer). Le mieux avec un breadboard est de positionner les composants comme sur votre schéma:



On utilise surtout le breadboard de côté. Ainsi, chaque ligne horizontale du haut et du bas représente le même point pour le circuit électrique. Les lignes horizontales sont souvent utilisées pour brancher la source. Les lignes verticales fonctionnent par groupe de cinq trous. Chaque groupe vertical représente un point de circuit. Pour insérer des composants, il faut rentrer la tige de métal bien au fond du trou pour que le contact soit efficace.

Voici d’autres exemples :

Évitez quand même d’en arriver là:



**Références :**

<http://www.zpag.net/Electroniques/Breadboard/Breadboard.htm>

**COMPLÈTE LES EXERCICES DANS TON DOCUMENT A-00**

# \\admsrv2\Bureau$\etienne.roy\Bureau\fablab\logo_kreoLab3.png A-01 : Construire des circuits alimentés par la carte Arduino

**Difficulté: Débutant**

|  |  |
| --- | --- |
| Apprentissage : **Programmation** : Aucune  **Électricité/Électronique** :   * Fonctionnement d’un panneau de montage; * Alimenter la carte Arduino via le port USB; * Alimenter un circuit en utilisant la carte Arduino comme source 5V; * Comprendre et construire un circuit simple comprenant DEL, résistor et interrupteur. | Matériel  * 1 – carte Arduino Uno; * Un résistor de 220 Ω ; * Une diode électroluminescente (DEL); * Quelques fils; * Un interrupteur bouton-poussoir normalement ouvert; * 1 ordinateur et 1 câble USB. |

## Introduction

Une carte « Arduino » est un microcontrôleur, c’est-à-dire un petit « ordinateur » servant à recevoir des signaux électriques (intrants), à les traiter en suivant les instructions d'un programme (système) et à en émettre de nouveaux (Extrants).

Elle a une fonction de commande étendue c'est-à-dire qu’elle traite l'information reçue des capteurs et envoie aux effecteurs. Les capteurs prennent l'information de l’environnement comme la quantité de lumière et les effecteurs effectuent une action sur l'environnement comme par exemple faire tourner un moteur ou faire allumer une lumière.

Arduino peut aussi servir d’alimentation électrique de faible puissance fournissant une tension de 3V ou 5V.

Le schéma suivant décrit la fonction de principaux connecteurs :

**Entrées et sorties numériques**

Les entrées numériques: ce sont des broches numérotées de 1 à 13 qui nous permettent de recevoir des signaux électriques qui sont de 0 V (Off) ou de 5 V (On).

Les sorties numériques sont aussi les broches numérotées de 1 à 13 qui nous permettent d'émettre des signaux électriques qui sont de 0 V (Off) ou de 5 V (On).

**Entrées analogiques**

Les entrées analogiques: ce sont des broches numérotées de A0 à A5 qui nous permettent de recevoir des signaux électriques qui sont de 0 V (signal faible) à 5 V (signal fort).

Par exemple, dans un programme, on pourrait dire:

si

la valeur lue sur la broche A0 (photorésistor) est plus petite que 409

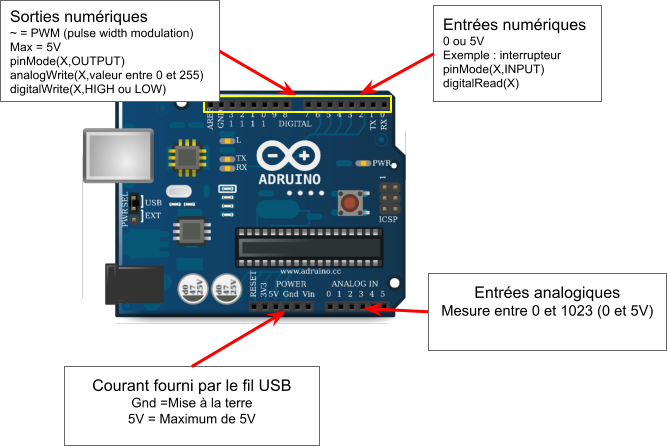
alors,

place la broche 2 (ampoule) à 5 V

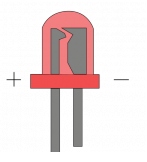
Arduino transforme la valeur de tension en nombre de 0 à 1023. Par exemple, un photorésistor a une tension de 2V sur une possibilité de 5V lorsqu'il reçoit de la lumière dans une pièce. Arduino va lire 409 (nombre compréhensible pour lui qui vient de 2V/5V de 1023 = 409).

**Broche 5 V (+) et GND (-)**

La carte Arduino peut servir de source d'alimentation 5 V à la condition qu'elle soit branchée à une source de tension électrique. La broche 5 V sert de borne positive et la broche GND (Ground ou la mise à la terre) sert de borne négative. La broche 5 V joue le rôle de fonction d'alimentation et non de la fonction commande, car elle est toujours sous une tension de 5V.



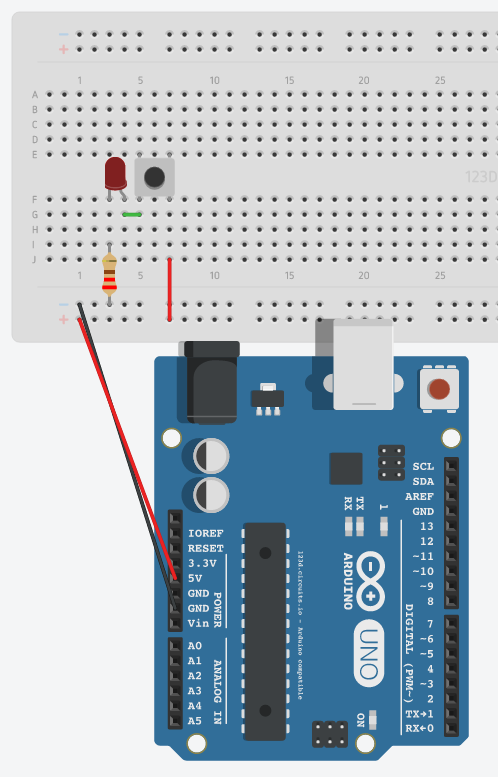
## Tâche 1 : **Allumer une DEL avec bouton poussoir (interrupteur)**

Comme première tâche avec la carte Arduino, vous devez réaliser un circuit en série alimenté d’une tension de 5 V par Arduino et comprenant une DEL, un résistor, un interrupteur bouton-poussoir.

ATTENTION : la patte la plus longue de la DEL est l'anode, donc elle doit être connectée sur la borne positive de notre circuit.(Voir image ci-contre)

## Montage

Réalisez le montage suivant :



On utilise la tension de 5V fournie par le port USB (on doit donc connecter le fil USB de l’ordinateur à la carte Arduino) de l’ordinateur pour alimenter notre circuit.



Débranchez toujours le câble USB lorsque vous réalisez ou modifiez un circuit.

Une fois le montage réalisé et la carte Arduino sous tension, appuyez sur le bouton poussoir.

** A-02 : Une première automatisation avec la carte Arduino**

**Difficulté: Débutant**

|  |  |
| --- | --- |
| Apprentissage :  **Programmation** :   * void setup() * void loop() * pinMode() * digitalWrite() * delay()   **Électricité/Électronique** :   * Fonctionnement d’un panneau de montage; * Alimenter la carte Arduino via le port USB; * Alimenter un circuit en utilisant une broche en sortie numérique de la carte Arduino; * Comprendre et construire des circuits simples comprenant une DEL, résistors et interrupteurs. | Matériel   * 1 – carte Arduino Uno; * Quelques résistors de 220 Ω * Quelques diodes électroluminescentes (DEL); * Quelques fils; * 1 ordinateur et 1 câble USB. |

Introduction

La réussite d'un projet Arduino compte deux types de savoir-faire : le montage de circuits électriques ou électroniques et la programmation. Ces deux types de tâches ont leur lot de difficultés.

La réalisation de circuits électroniques demande la manipulation de composants très petits. Le risque d'effectuer une erreur de branchement est grand. Lorsque le projet ne fonctionne pas, la première étape est de vérifier le circuit électrique.

L'écriture d'un programme Arduino demande une syntaxe particulière. Le choix de majuscules ou de minuscules est important. Le langage Arduino a son vocabulaire particulier.

Les explications qui suivent décriront le vocabulaire et la syntaxe nécessaires au présent projet.

Programmation - références minimales du langage Arduino

***void*** Le mot-clé **void** est utilisé uniquement pour les déclarations de fonctions. Il indique à Arduino qu'on lui donne une commande.

***void setup()*** La fonction **setup()** est appelée au démarrage du programme. Cette fonction est utilisée pour initialiser les variables et le sens des broches. La fonction setup n'est exécutée qu'une seule fois, après chaque mise sous tension ou reset (réinitialisation) de la carte Arduino. La fonction **setup()**, même vide, est obligatoire dans tout programme Arduino.

## Syntaxe

void **setup**()  
{   
  
}

***pinMode()*** Configure la broche spécifiée pour qu'elle se comporte soit en entrée, soit en sortie.

## Syntaxe

void **setup**()  
{   
pinMode(broche, mode)

}

Paramètres :

broche: le numéro de la broche de la carte Arduino dont le mode de fonctionnement (entrée ou sortie) doit être défini.

mode: soit INPUT (entrée en anglais) (=0) ou OUTPUT (sortie en anglais) (=1)

***void loop()*** Après avoir créé une fonction setup(), qui initialise et fixe les valeurs de démarrage du programme, la fonction **loop ()** (boucle en anglais) fait exactement ce que son nom suggère et s'exécute en boucle sans fin, permettant à votre programme de s'exécuter et de répondre. Utiliser cette fonction pour contrôler activement la carte Arduino. La fonction loop() est obligatoire, même vide, dans tout programme.

## Syntaxe

void loop()  
{  
  
}

## Syntaxe

void loop()  
{  
digitalWrite(broche, valeur)  
}

***digitalWrite()*** Met un niveau logique HIGH (HAUT en anglais) ou LOW (BAS en anglais) sur une broche numérique. Si la broche a été configurée en SORTIE avec l'instruction pinMode(), sa tension est mise à la valeur correspondante : 5V (ou 3.3V sur les cartes Arduino 3.3V) pour le niveau HAUT, 0V (masse) pour le niveau BAS.

Paramètres :

broche: le numéro de la broche de la carte Arduino.

valeur: HIGH ou LOW (ou bien 1 ou 0)

***delay()*** Réalise une pause dans l'exécution du programme pour la durée (en millisecondes) indiquée en paramètre. (En passant, il y a 1000 millisecondes dans une seconde...)

## Syntaxe

void loop()  
{  
delay (ms);  
}

Paramètres :

ms: le nombre de millisecondes que dure la pause.

## CONSEIL

Quand vous essayez ou mettez au point votre code, mettre en commentaire des parties ou des lignes de votre programme est une façon pratique de retirer du code les lignes qui peuvent être la source d'une erreur. Ces lignes restent dans le code (pas besoin de les réécrire...), mais en les passant en commentaire, le compilateur les ignore. Cette technique est très pratique et utile lorsque vous essayez de localiser un problème, ou quand un programme refuse de compiler et que le message d'erreur de compilation est énigmatique ou peu informatif.

***// et /\* \*/*** Les commentaires sont des lignes de texte incluses dans le programme, un peu comme un aide-mémoire, et qui ont pour but de vous informer et d'informer les autres de la façon dont le programme fonctionne. Ces lignes ajoutées sont ignorées par Arduino. Les commentaires ont pour seul but de vous aider à comprendre (ou à vous rappeler) comment votre programme fonctionne ou d'en informer les autres. Il y a deux façons de créer des lignes de commentaires : // et /\*\*/

### Syntaxe de base

## CONSEIL

**Toujours vérifier la présence du point-virgule à la fin d'une instruction.**

Oublier le point-virgule en fin de ligne donnera une erreur de compilation. Le texte d'erreur pourra être évident, et se référer à un point-virgule oublié, mais parfois ça sera moins évident. Si une erreur de compilation incompréhensible et apparemment illogique survient, **la première chose à vérifier est l'oubli d'un point-virgule**, dans le voisinage immédiat, précédent la ligne à laquelle Arduino signale le problème.

***Point-virgule***Utilisé (et obligatoire) à la fin d'une instruction. Pour Arduino les sauts de lignes « enter » n'ont pas de signification : c'est le point-virgule qui marque la fin de la ligne.

### Syntaxe de base

À noter que l'on peut séparer plusieurs instructions sur une même ligne à l'aide de virgule, le point-virgule devant être alors mis à la fin.

**À vérifier s’il y a un message d’erreur!**

***Accolades {}*** Elles servent à marquer le début et la fin d’une fonction, d’une boucle ou d’une condition. Elles sont importantes.

## CONSEIL

**Toujours vérifier la présence d’une accolade d’ouverture et de fermeture**

On dit souvent qu'il faut que les accolades soient équilibrées (le nombre d'accolade d'ouverture doit correspondre au nombre d'accolade de fermeture donc autant de"{" que de "}". Arduino inclut une fonctionnalité pratique pour vérifier la correspondance des accolades entre elles. Il suffit de sélectionner une accolade, ou même de cliquer juste après l'accolade, et son compagnon logique sera mis en surbrillance.

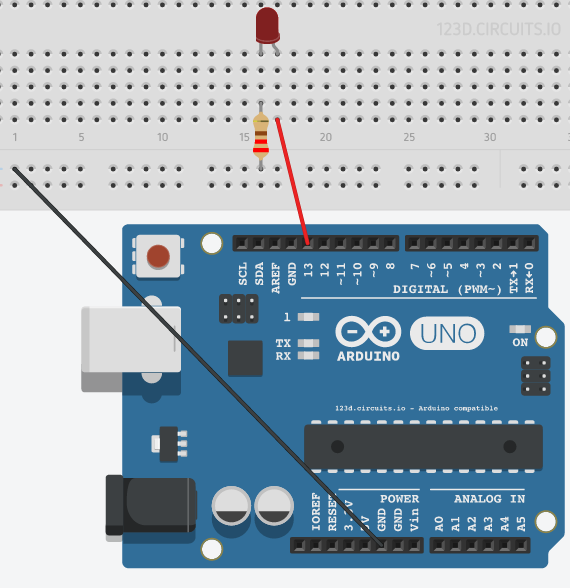
Des accolades déséquilibrées : chaque "{" n'étant pas accompagnée de son pendant "}" peuvent donner lieu à des messages d'erreur de compilations énigmatiques et incompréhensibles, difficiles à résoudre dans de long programme. À cause de leurs utilisations très variées, les accolades sont incroyablement importantes dans la syntaxe d'un programme et déplacer une accolade d'une ou deux lignes peuvent souvent perturber profondément la signification d'un programme.

### Syntaxe de base

**À vérifier s’il y a un message d’erreur!**

Tâche 1 : **Faire clignoter une DEL**

Vous devez faire clignoter une DEL de façon à ce qu’elle s’allume 1 seconde et s’éteigne 1 seconde.

Montage

Réalisez le montage suivant :

Programmer le clignotement

|  |  |
| --- | --- |
| Une fois le montage réalisé, il faut brancher la carte Arduino à l’ordinateur à l’aide du câble USB.  Sur l’ordinateur, démarrez le programme Arduino. Il se trouve dans les applications distribuées. | \\admsrv2\MesDoc$\etienne.roy\Mes documents\1_matante\moodlecsbe\GetF.PNG |
| Le logiciel Arduino s’ouvrira. | C:\Users\etienne.roy\AppData\Local\Temp\SNAGHTML13ac94d.PNG |
| Vérifiez que la bonne carte est choisie à partir du menu « Outils », puis « Type de carte ».  Il doit s’agir de « Arduino / Genuino UNO ».  **À vérifier s’il y a un message d’erreur!** |  |
| Vérifiez que le port de communication avec la carte est choisi.  Le port de votre ordinateur peut être différent de celui de l’exemple  **À vérifier s’il y a un message d’erreur!** |  |
| Effacez ce qui se trouve dans la fenêtre d’édition. |  |
| Écrivez le programme suivant dans la fenêtre d’édition.  Le respect des majuscules et de minuscules est important!  **À vérifier s’il y a un message d’erreur!** | void setup()  {  pinMode(13, OUTPUT);  }  void loop()  {  digitalWrite(13, HIGH);  delay(1000);  digitalWrite(13, LOW);  delay(1000);  } |
| * Enregistrez le programme en cliquant sur la flèche blanche pointant vers le bas. |  |
| * La fenêtre suivante s’ouvrira. | C:\Users\etienne.roy\AppData\Local\Temp\SNAGHTML152156a.PNG |
| Envoyez le programme à la carte Arduino, en utilisant le cercle avec la flèche pointant à droite. |  |
| La DEL devrait maintenant clignoter. |  |

**MONTRE TON CIRCUIT FONCTIONNEL À LA TECHNICIENNE OU À TON ENSEIGNANTE POUR APPROBATION**

Explications du programme

**AU NIVEAU DE LA FONCTION SETUP ()**

Dans le programme que tu viens de faire, on commence par initialiser la broche 13 en sortie avec l'instruction pinMode() selon :

***pinMode(13, OUTPUT);***

**AU NIVEAU DE LA FONCTION LOOP()**

Dans la fonction loop, "coeur" du programme, on commence par allumer la LED à l'aide l'instruction digitalWrite()

***digitalWrite(13, HIGH);***

Ceci met la broche 13 au niveau HAUT, soit +5V. Ceci crée une différence de potentiel entre les 2 broches de la DEL et elle s'allume donc. Ensuite, on éteint la LED toujours à l'aide de l'instruction digitalWrite() :

***digitalWrite(13, LOW);***

Ceci met la broche 13 au niveau BAS, soit 0V : la LED s'éteint. Entre l'allumage et l'extinction de la LED, il faut prévoir un délai assez long pour que l'œil humain puisse percevoir le changement. Pour cela, on utilise l'instruction delay() pour dire à la carte Arduino de ne rien faire pendant 1000 millisecondes, c'est à dire une seconde :

***delay(1000);***

Quand vous utilisez l'instruction delay(), rien ne se passe pendant le temps précisé.

Si vous ne mettez pas d'instruction delay() entre la mise au niveau HAUT puis BAS de la broche 13, la LED va clignoter très très rapidement... et elle apparaîtra toujours allumée. Si on connecte un oscilloscope sur la broche, on pourra voir cependant que la broche change d'état HAUT/BAS plusieurs millions de fois par seconde !! Cela vous donne une petite idée de la rapidité de votre carte Arduino, un véritable petit ordinateur

**Références :**

<http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.ExempleLEDBlink>

Tâche 2 : **Faire clignoter deux DEL**

**Consulte ton document de l’élève : dessine le montage, écris le programme et fais vérifier ton travail par la technicienne ou l’enseignante. Bon défi !**

# \\admsrv2\Bureau$\etienne.roy\Bureau\fablab\logo_kreoLab3.png A-03 : Lire l’état d’un capteur

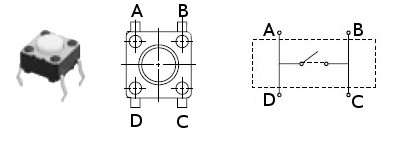
**Difficulté: Débutant**

|  |  |
| --- | --- |
| Apprentissage : **Programmation** :   * Int * Serial.begin() * Serial.print() * Serial.println() * digitalRead() * analogRead() * string   **Électricité/Électronique** :   * Comprendre ce qu’est un capteur; * Comprendre et construire des circuits simples comprenant un photorésistor, résistors et interrupteurs. | Matériel  * 1 – carte Arduino Uno; * 1 résistor de 220 Ω ou 100 Ω et 10 000 Ω; * photorésistor * Un interrupteur bouton-poussoir normalement ouvert; * Quelques fils; * 1 ordinateur; * 1 câble USB; * Logiciel Arduino. |

## Introduction

Un capteur est un dispositif transformant l'état d'une grandeur physique observée en une tension électrique.

**L’interrupteur :**

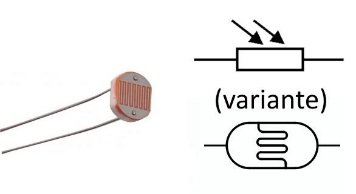


L’interrupteur est un capteur numérique (digital), car il ne peut y avoir que deux positions :

* Sous tension (interrupteur fermé);
* Hors tension (interrupteur ouvert);

C’est l’un de ces deux états que les broches numériques (2 à 13) de la carte Arduino peuvent lire.

**Le photorésistor :**



Le photorésistor est une résistance qui varie en fonction de l’éclairage. Puisque sa résistance varie, la différence de potentiel (tension) à ses bornes varie aussi. Elle peut prendre n’importe quelle valeur entre un minimum et un maximum.

C’est cette valeur que les broches analogiques (A0 à A5) de la carte Arduino peuvent lire. Cette valeur sera transformée en un nombre compris entre 0 et 1083.

### **Programmation - références minimales du langage Arduino**

## Syntaxe

int var = val;

***int*** Déclare une variable de type int (pour *integer*, entier en anglais). Les variables de type int sont votre variable de base pour le stockage de nombres.

Paramètres :

var : le nom de votre variable de type int

val : la valeur d'initialisation de la variable

## Syntaxe

String(valeur);

***String()*** C’est le constructeur qui crée une « phrase » à partir de différents types de données qui sont alors transformées en séquence de caractères

Paramètres :

valeur : une variable à convertir en un objet String – la valeur peut-être une chaîne de caractère, une constante ou une variable

## Syntaxe

void **setup**()  
{   
Serial.begin(debit);  
}

***Serial.begin()*** Fixe le débit de communication entre arduino et l’ordinateur en nombre de caractères par seconde. L'unité est le baud pour la communication série. C’est le nombre de symboles transmissibles par seconde.

Paramètres :

debit: Pour une carte Aduino UNO, on le fixe à 9600.

## Syntaxe

void loop()  
{  
Serial.print(val);  
}

***Serial.print()*** Affiche les données sur le port série.

Paramètres :

Val : la valeur à afficher. N'importe quel type de données.

***Serial.println()*** Affiche les données sur le port série, suivi d'un caractère de "retour de chariot"

## Syntaxe

void loop()  
{  
Serial.println(val);  
}

Paramètres :

Val : la valeur à afficher. N'importe quel type de données.

***digitalRead()*** Lit l"état (= le niveau logique) d'une broche précise en entrée numérique, et renvoie la valeur HIGH (HAUT en anglais) ou LOW (BAS en anglais)

## Syntaxe

void loop()  
{  
digitalRead(broche);  
}

Paramètres :

broche: le numéro de la broche de la carte Arduino.

VALEUR RENVOYÉE

Renvoie la valeur HIGH (HAUT en anglais)(1) ou LOW (BAS en anglais)(0).

***analogRead()*** Lit la valeur de la tension présente sur la broche spécifiée. La carte Arduino comporte 6 voies (8 voies sur la Mini et la Nano) connectées à un convertisseur analogique-numérique 10 bits. Cela signifie qu'il est possible de transformer la tension d'entrée entre 0 et 5V en une valeur numérique entière comprise entre 0 et 1023.

## Syntaxe

void loop()  
{  
analogRead(broche\_analogique);  
}

Paramètres :

broche\_analogique: le numéro de la broche analogique de la carte Arduino.

VALEUR RENVOYÉE

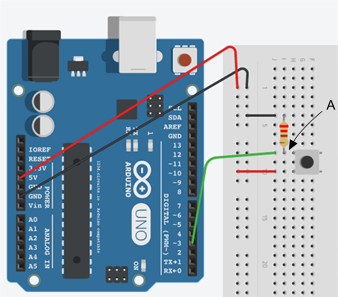
valeur int (0 to 1023) correspondant au résultat de la mesure effectuée

## Tâche 1 : **L’interrupteur**

Vous devez réaliser le montage et le programme permettant de savoir si un interrupteur est à la positon « ouvert (off) » ou « fermé (on) ».

### Montage

Réalisez le montage suivant :



### Programmer

* Branchez la carte Arduino à l’ordinateur à l’aide du câble USB.
* Sur l’ordinateur, démarrez le programme Arduino.
* Vérifiez que la bonne carte est choisie à partir du menu « Outils », puis « type de carte ». Il doit s’agir de « Arduino / Genuino UNO ».
* Vérifiez que le bon port de communication avec la carte est choisi :
* Écrire le programme suivant :

int etat;

void setup() {

Serial.begin(9600);

pinMode(3, INPUT);

}

void loop() {

etat = digitalRead(3);

Serial.print("Interrupteur = ") ;

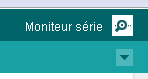
Serial.print(etat) ;

}

* Envoyez le programme à la carte Arduino et vérifier qu’il fonctionne correctement.

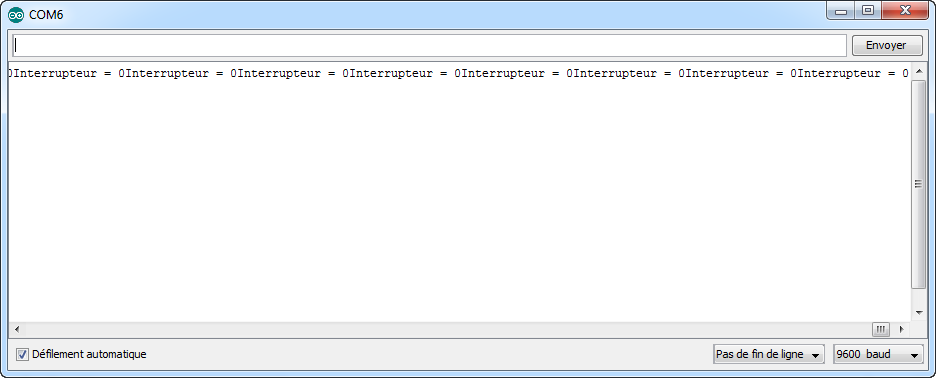
Pour visualiser les données recueillies par la carte Arduino :

* Cliquez sur « Moniteur série » (carré avec une loupe);



La fenêtre suivante apparaîtra.

Remarquez que les informations défilent rapidement sur une même ligne.



Il serait intéressant d’avoir une donnée par ligne. Pour y arriver :

* Remplacez Serial.print par Serial.println.

int etat;

void setup() {

Serial.begin(9600);

pinMode(3, INPUT);

}

void loop() {

etat = digitalRead(3);

Serial.println("Interrupteur = ") ;

Serial.println(etat) ;

}

* Envoyez le programme à la carte Arduino et vérifier qu’il fonctionne correctement.

Il serait intéressant d’avoir le mot et la donnée sur une seule ligne. Pour y arriver :

* Utilisez String().

int etat;

void setup() {

Serial.begin(9600);

pinMode(3, INPUT);

}

void loop() {

etat = digitalRead(3);

Serial.println((String("Interrupteur = ") + String(etat)));

}

* Envoyez le programme à la carte Arduino et vérifier qu’il fonctionne correctement.

### **Explications du programme**

**AU NIVEAU DE LA DÉFINITION DE PARAMÈTRES ()**

On définit la variable « etat » comme étant un entier sans lui attribuer de valeur.

**int etat;**

**AU NIVEAU DE LA FONCTION SETUP ()**

On commence par initialiser la communication série à une vitesse de 9600 baud

**Serial.begin(9600);**

Par la suite, on initialise la broche 3 en entrée avec l'instruction pinMode() selon :

***pinMode(3, INPUT);***

**AU NIVEAU DE LA FONCTION LOOP()**

On attribue à la variable « etat » la valeur lue sur la broche 3 .

***etat = digitalRead(3);***

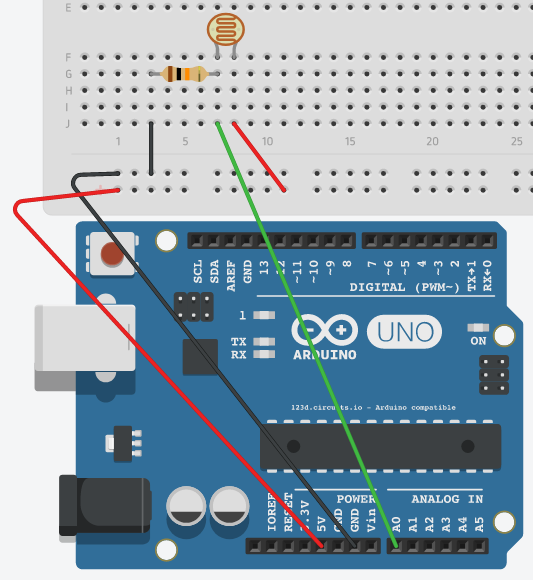
On affiche avec des retours de chariot une phrase composée des caractères « Interrupteur = » et de la valeur de la variable « état ».

***//Serial.println((String("Interrupteur = ") + String(etat)));***

## Tâche 2 : **La photorésistance**

Réalisez le montage et le programme permettant de savoir si un le niveau d’éclairage reçu par une photorésistance.

### Montage



### Programmer

* Modifiez le programme de l’activité précédente pour obtenir le programme suivant :

int etat;

void setup() {

Serial.begin(9600);

pinMode(A0, INPUT);

}

void loop() {

etat = analogRead(A0);

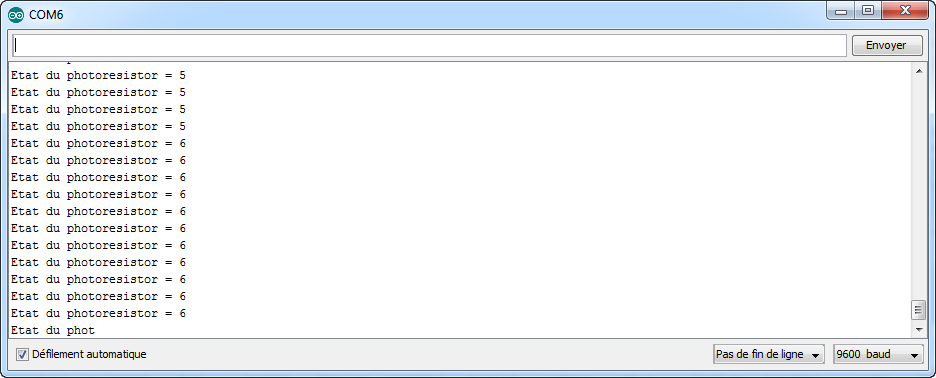
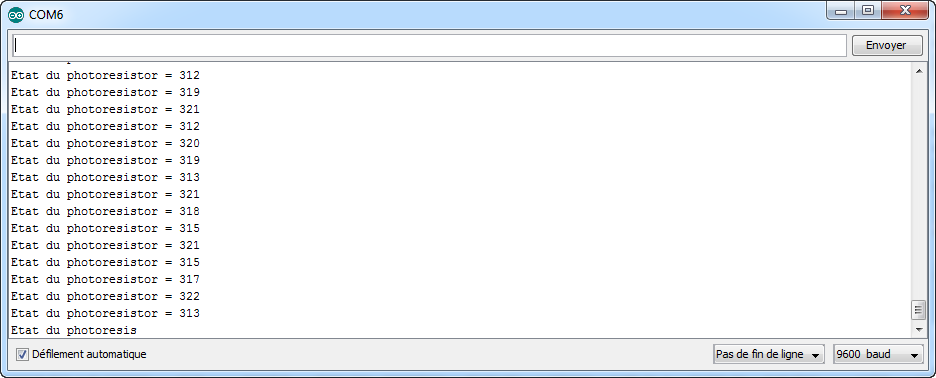
Serial.println((String("Interrupteur = ") + String(etat)));

}

* Envoyez le programme à la carte Arduino et vérifier qu’il fonctionne correctement.

La carte Arduino envoie des nombres compris entre 0 et 1083

**Lumière assez forte Lumière faible**



# \\admsrv2\Bureau$\etienne.roy\Bureau\fablab\logo_kreoLab3.png A-04 : Une première interactivité…

**Difficulté: Débutant**

|  |  |
| --- | --- |
| Apprentissage : **Résumé des activités précédentes :**   * ***A-01 A-02 – Vous avez appris à commander une DEL avec la carte Arduino.*** * ***A-03 – Vous avez appris à lire l’état d’un capteur.***   **Programmation** :   * ***if else***;   **Électricité/Électronique** :   * Comprendre ce qu’est un capteur; * Comprendre et construire des circuits simples comprenant un photorésistor, résistors et interrupteurs. | Matériel  * 1 – carte Arduino Uno; * 1 résistor de 220 Ω ou 100 Ω et 10 000 Ω; * photorésistor * Un interrupteur bouton-poussoir normalement ouvert; * Quelques fils; * 1 ordinateur; * 1 câble USB; * Logiciel Arduino. |

## Introduction

### Programmation - références minimales du langage Arduino

***if / else*** L'instruction if /else, est utilisée avec un opérateur logique de comparaison et permet de tester si une condition est vraie. Si c'est le cas, le programme va réaliser une action particulière. Sinon, le programme va réaliser une autre action particulière.

## Syntaxe

void loop()  
{  
  
if (condition){

{

// action A

}

else

{

// action B

}

}

Paramètres :

Condition : opérateurs logiques

* x == y (x est égal à y)
* x != y (x est différent de y)
* x < y (x est inférieur à y)
* x > y (x est supérieur à y)
* x <= y (x est inférieur ou égal à y)
* x >= y (x est supérieur ou égal à y)

Action : action particulière

***for()*** L'instruction for est utilisée pour répéter l'exécution d'un bloc d'instructions regroupées entre des accolades. Un compteur incrémental est habituellement utilisé pour incrémenter et finir la boucle. L'instruction for est très utile pour toutes les opérations répétitives et est souvent utilisées avec des tableaux de variables pour agir sur des données ou broches.

## Syntaxe

void loop()  
{  
for(initialization; condition; incrementation) {

//instruction(s)à exécuter;

}  
}

Paramètres :

initialization: nom de la variable

condition : condition d’arrêt;

incrementation : variation de la variable

<http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.ReferenceMaxi>

## Tâche 1 : **L’interrupteur**

Combiner les savoirs appris lors des activités A-02, A-03 en réalisant le montage et le programme permettant de faire clignoter 2 DEL en alternance à toutes les secondes, pendant 4 fois lorsqu’un interrupteur est fermé.

### Montage

Dessine ton montage dans le document de l’élève

**Programmer**

Réfléchissons dans un langage plus facile (le français) les actions à programmer.

Si l’interrupteur est fermé :

Alors

Fais 4 fois

Allume la DEL 1

Éteint la DEL 2

Attend 1 seconde

Éteint la DEL 1

Allume la DEL 2

Attend 1 seconde

Autrement

Éteint la DEL 1

Éteint la DEL 2

Commençons la traduction en langage Arduino

« L’interrupteur est fermé » 🡺 État de la broche = HIGH ou 1.

digitalRead(2) == HIGH

« Allume la DEL » 🡺 Place la broche à HIGH ou 1.

digitalWrite(3, HIGH);

« Éteint la DEL » 🡺 Place la broche à LOW ou 0.

digitalWrite(4, LOW);

« Attend 1 seconde » 🡺 Délai de 1000 ms

delay(1000);

Si l’interrupteur est fermé :

if (digitalRead(2) == HIGH) {

Fais 10 fois

for (int count = 0; count < 4; count++) {

}

**Voyons ce que ça donne avec le programme complet…**

* Branchez la carte Arduino à l’ordinateur à l’aide du câble USB.
* Sur l’ordinateur, démarrez le programme Arduino.
* Vérifiez que la bonne carte est choisie à partir du menu « Outils », puis « type de carte ». Il doit s’agir de « Arduino / Genuino UNO ».
* Vérifiez que le bon port de communication avec la carte est choisi :
* Écrire le programme suivant :

void setup() {

pinMode(2, INPUT);

pinMode(3, OUTPUT);

pinMode(4, OUTPUT);

}

void loop() {

if (digitalRead(2) == HIGH) {

for (int count = 0; count < 4; count++) {

digitalWrite(3, HIGH);

digitalWrite(4, LOW);

delay(1000);

digitalWrite(3, LOW);

digitalWrite(4, HIGH);

delay(1000);

}

} else {

digitalWrite(4, LOW);

digitalWrite(3, LOW);

}

}

## Tâche 2 : **La photorésistance**

Créer un circuit et un programme faisant clignoter 2 DEL en alternance lorsqu’il fait sombre dans la pièce. Bonne réflexion…