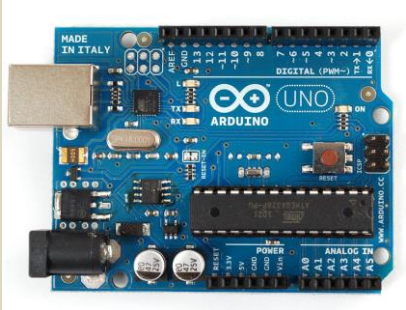
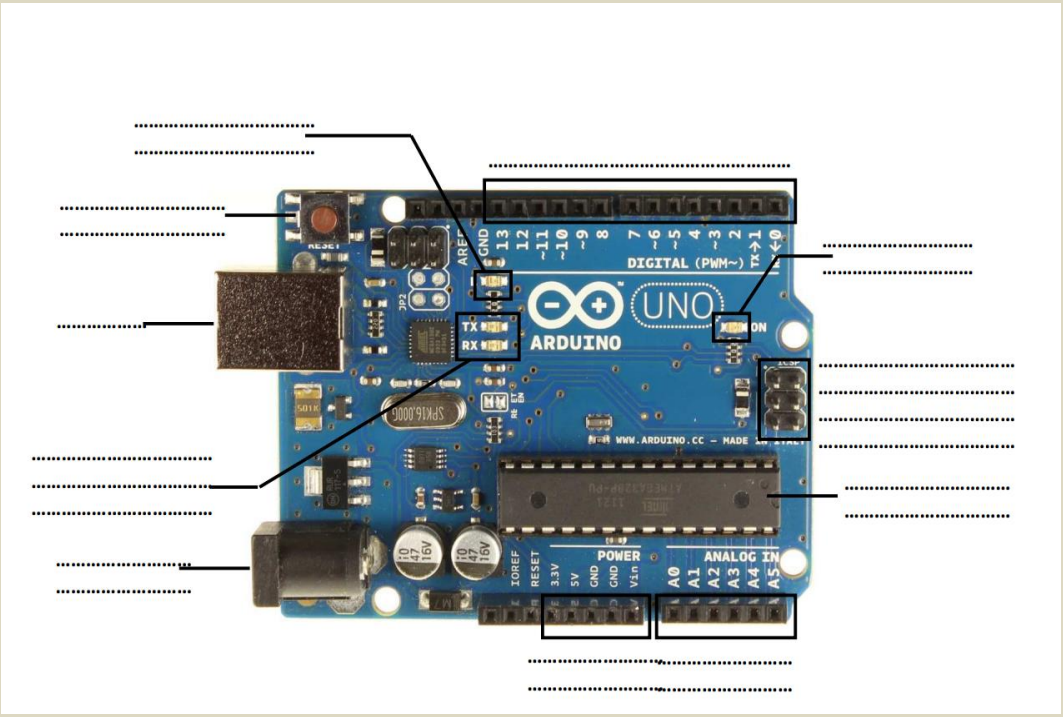
Je veux m'initier à Arduino

1. Présentation du matériel.



1. Préciser ce qu'est « Arduino ».
2. Citer les principaux avantages des systèmes « Arduino »
3. Indiquer quel type de microcontrôleur équipe les cartes « Arduino Uno ».
4. Lister le type et la taille des mémoires disponibles.
5. Donner la gamme des tensions d'alimentation normalisées de la carte « Arduino Uno ».
6. Préciser le nombre d'entrées/sorties numériques disponibles sur la carte « Arduino Uno ».
7. Donner le nombre d'entrées/sorties analogiques disponibles sur la carte « Arduino Uno ».
8. Lister les liaisons séries disponibles par défaut sur la carte « Arduino Uno ».
9. Compléter la figure ci-dessous.



Clignotement d'une LED

Objectifs :

Réaliser une fonction astable permettant d'allumer et d'éteindre une LED pendant une durée définie et ceci de façons perpétuelle. Donc, de faire clignoter une LED.

Travail à réaliser :

Notez dans votre compte-rendu individuel ce que vous avez mis en œuvre autant matériel que logiciel.

A quoi sert la fonction setup()

A quoi sert la fonction loop()

Par quoi commence et fini une fonction ?

Dans cet exemple, pourquoi n'y a-t-il pas de résistance en série avec la LED ?

Pour chaque instruction, notez sur le listing source du programme un commentaire expliquant son rôle.

1 - Le programme minimum

Description

Cet exemple contient le code minimum qui est nécessaire et indispensable pour un programme Arduino compilable. Le code minimum doit comporter :

- la fonction setup() qui est exécutée une fois au début du programme.
- et la fonction loop() qui est exécutée en boucle tant que la carte est sous tension.

Une règle essentielle : les deux fonctions setup() et loop() (même vides) sont OBLIGATOIRES dans tout programme Arduino .

Explications du programme

La fonction setup() est la première fonction qui est appelée quand un programme démarre. Il faut utiliser cette fonction pour notamment :

- initialiser les variables,
- initialiser le mode de fonctionnement (entrée/sortie) des broches,
- démarrer les librairies utilisée, etc....

La fonction setup() sera exécutée seulement une fois, après la mise sous tension ou une réinitialisation (par appui sur le bouton 'reset') de la carte Arduino.

Après la fonction setup, la fonction loop va faire exactement ce que son nom suggère (loop = boucle en anglais) : elle va s'exécuter en boucle, sans fin, permettant au programme d'exécuter des instructions et de réagir durant son exécution. C'est le code placé dans la fonction loop qui est activement utilisé pour contrôler la carte Arduino : on peut dire que c'est le "cœur" de votre programme.

Le code que vous allez tester ne fera absolument rien... mais sa structure vous sera très utile pour réaliser un copier/coller avant de démarrer un programme.

Ce programme minimal vous montre également comment mettre des commentaires dans votre code. Toute ligne qui commence par deux slashs // ne sera pas lue par le compilateur : vous pouvez donc écrire à ce niveau ce que vous voulez. Commenter votre code de cette façon sera particulièrement utile pour expliquer à vous-mêmes ou aux autres votre programme pas à pas.

Toute fonction utilisée dans un programme doit être déclarée en définissant :

- son type, défini par un mot clé mis avant le nom de la fonction,
- les paramètres qu'elle reçoit, mis dans les () après le nom de la fonction.

Le code d'une fonction est encadré par une accolade { de début et une } de fin : ce sont les limites du code de la fonction. Pour plus de détails, voir la page Accolades. Bon à savoir également : chaque ligne de code active comportant une instruction (sauf quelques cas particuliers) doit se terminer par un ; .

Le type de la fonction correspond au type de valeur (byte, int, float, etc...) qu'elle renvoie après son exécution. Dans le cas le plus simple, lorsqu'une fonction ne renvoie aucune valeur, elle est de type void : on utilisera donc ce mot clé pour déclarer une fonction qui ne renvoie rien.

Si une fonction utilise des paramètres, ils doivent être définis entre les (). Les paramètres reçus seront utilisés par la fonction pendant son exécution. Si une fonction n'utilise aucun paramètre, on laissera les ( ) vides tout simplement.

Les fonctions setup() et loop() obligatoires dans tout programme Arduino sont des fonctions de la forme la plus simple : elles ne renvoient aucune valeur et elles n'utilisent aucun paramètre. On utilisera donc le mot clé void pour les déclarer et on laissera les parenthèses vides.

Le code du programme

```
void setup()
{
```

```
 // début de la fonction setup () : est exécutée une fois
  // mettre ici les instructions de votre programme à exécuter une seule fois au démarrage
} // fin de la fonction setup() : le programme exécute ensuite la fonction loop

void loop()
{
// début de la fonction loop() : est exécutée en boucle sans fin
  // mettre ici les instructions de votre programme qui seront exécutées en boucle
} // fin de la fonction loop : le programme reprend au début de loop
```

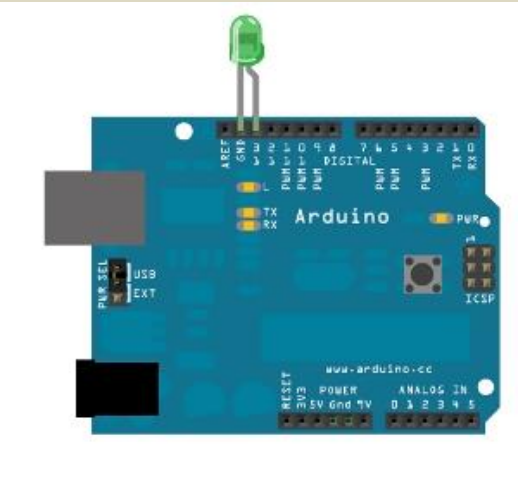
2 – Faire clignoter une LED

Matériel requis

- une carte Arduino Uno
- une LED standard

Le circuit à réaliser

- prendre une LED et connecter sa patte longue (patte positive appelée anode) à la broche 13 de la carte Arduino,
- connecter la patte courte (patte négative, appelée cathode) à la broche de masse de la carte Arduino (notée GND pour GROUND en anglais, cette broche correspond au 0V).



ATTENTION : Avant de tester il est impératif de faire vérifier son montage par le professeur

Ensuite, connecter la carte Arduino par un câble USB à votre ordinateur sur lequel est installé le logiciel Arduino

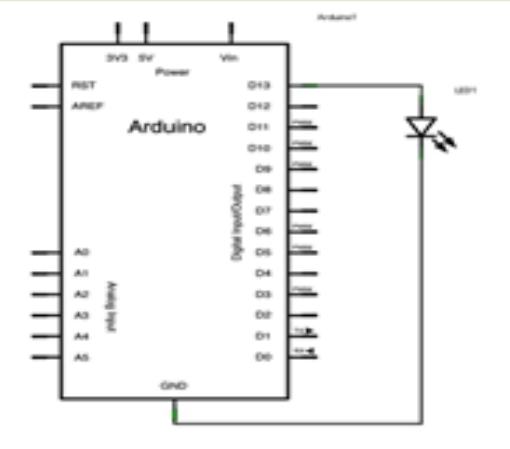
Mise en oeuvre du programme

- Lancer le logiciel Arduino,
- copier/coller le code ci-dessous
- programmer votre carte en suivant la procédure indiquée ici.

La LED connectée sur la broche 13 doit se mettre à clignoter.

Schéma théorique du montage

Voici le schéma théorique du montage :



Vous noterez que la LED n'a pas de résistance en série. Ceci car l'intensité disponible sur une broche en sortie est assez basse pour ne pas endommager la LED. Ceci simplifie le montage. Cependant, d'une manière générale, il est bien sûr préférable d'ajouter une résistance en série avec la LED afin de limiter l'intensité utilisée sur la broche en sortie.

- Les broches de la carte Arduino peuvent fournir jusqu'à 40mA en entrée comme en sortie. Cependant, il ne faut pas oublier que l'intensité maximale disponible pour l'ensemble des broches en sortie est de 200mA pour une carte UNO.
- Ainsi en pratique, il est judicieux de limiter l'intensité utilisée de 10 à 15 mA par broche en moyenne. Pour mémoire, la chute de tension aux bornes d'une LED est de 1,5V environ. On pourra donc, avec une LED, utiliser une résistance en série de 300 Ohms environ (R = U/I = (5V - 1,5V) / 0,012 = 300 Ohms).

Explications du programme

Au niveau de la fonction setup ()

Dans le programme qui suit, 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 de 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 LED 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'oeil 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);
```

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 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.

Le code du programme

```
/*
  Clignotement de LED
  Allume une LED pendant 1 seconde, puis l'éteint pendant 1 seconde puis le programme se répète
indéfiniment

*/

void setup() {

  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

- ✓ Une fois ce petit programme testé, tu peux essayer de diminuer le délai afin de voir

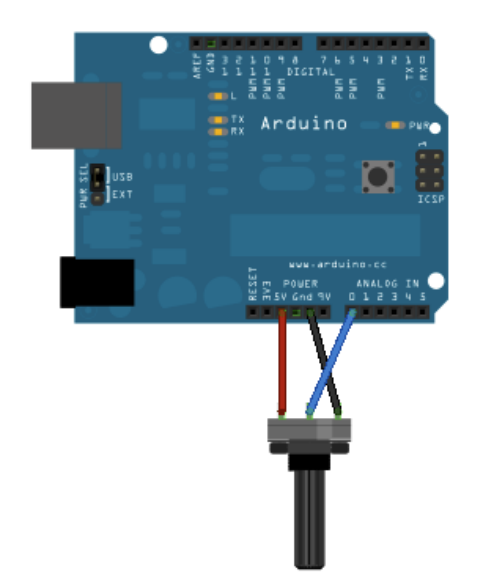
quelle est la valeur à partir de laquelle l'œil ne voit plus la LED clignoter (persistance
rétinienne).
Donner la valeur.

- ✓ Ensuite tu vas écrire le logigramme pour faire un SOS lumineux en langage Morse avec la LED (aide-toi d'Internet pour trouver les informations). Utilise la boucle for (….,….,….) pour réaliser ton programme.

3- Acquérir une grandeur analogique et afficher sa valeur numérique au PC

Cet exercice vous montre comment lire la tension venant du monde physique présente sur une entrée analogique de la carte Arduino, en utilisant un potentiomètre. Un potentiomètre est un composant mécanique simple qui fait varier la résistance entre ses broches lorsque l'on tourne son axe. En connectant un potentiomètre à une tension et vers une entrée analogique de la carte Arduino; il est possible de mesurer la résistance du potentiomètre en tant que valeur analogique.

Dans cet exercice, vous allez visualiser l'état de votre potentiomètre après avoir établi une communication série entre votre carte Arduino et votre ordinateur.



Comment çà marche ?

En tournant l'axe du potentiomètre, on modifie la résistance entre la broche médiane et les broches externes du potentiomètre. Ceci a pour effet de modifier la tension sur la broche de sortie du potentiomètre (l'intérieur du potentiomètre réagissant en diviseur de tension) :

- quand la résistance entre la broche médiane et la broche externe reliée au 5V est nulle (et donc la résistance entre la broche centrale et la broche externe reliée au 0V vaut 10 kiloOhms), la tension de la broche centrale vaut 5V.
- à l'inverse quand la résistance entre la broche médiane et la broche externe reliée au 0V est nulle (et donc la résistance entre la broche centrale et la broche externe reliée au 5V vaut 10 kiloOhms), la tension de la broche centrale vaut 0V.
- en dehors de ces deux positions extrêmes, la tension sur la broche de sortie prend des valeurs variables entre le 0V et le 5V en fonction de la position de l'axe.

C'est cette tension analogique (= variable) que l'on va lire sur l'entrée analogique de la carte Arduino.

La carte Arduino dispose d'un circuit intégré interne (présent dans le microcontrôleur de la carte), appelé "convertisseur analogique-numérique" qui convertit la tension en volts présente à son entrée en une valeur numérique comprise entre 0 et 1023 (soit une échelle de 10 bits), valeur très précisément proportionnelle à la tension présente à l'entrée :

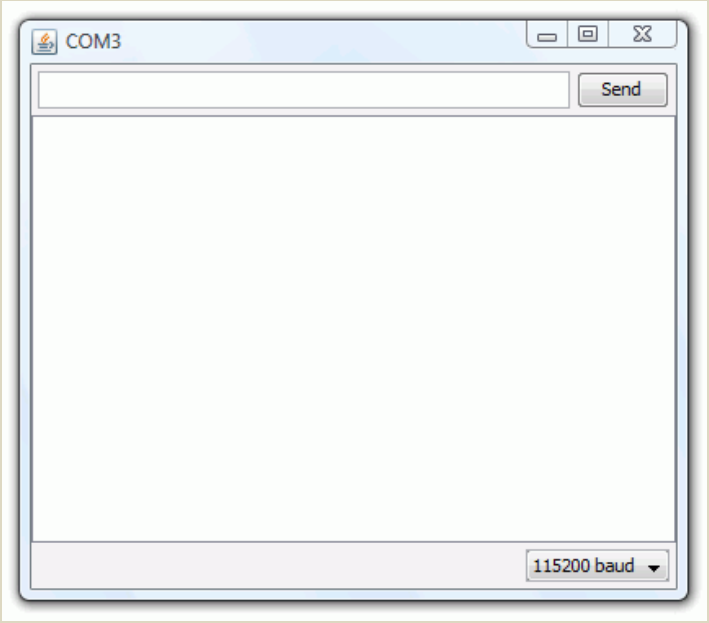
- quand l'axe du potentiomètre est tourné "à fond" dans un sens, la tension de la broche centrale vaut 5V et le convertisseur analogique-numérique renvoie la valeur maximale soit 1023.
- quand l'axe du potentiomètre est tourné "à fond" dans l'autre sens, la tension de la broche centrale vaut 0V et le convertisseur analogique-numérique renvoie la valeur minimale soit 0.
- quand l'axe du potentiomètre est en position intermédiaire, la tension de la broche centrale varie entre 0 et 5V et le convertisseur analogique-numérique renvoie la valeur proportionnelle à la tension comprise entre 0 et 1023.

Note : la précision de la mesure est de l'ordre de 5V/1023 = 0.0048 soit 5millivolts environ, ce qui est très précis et largement suffisant pour l'utilisation des nombreux capteurs analogiques disponibles et utilisables pour un robot, une station météo, mini-oscilloscope, etc...

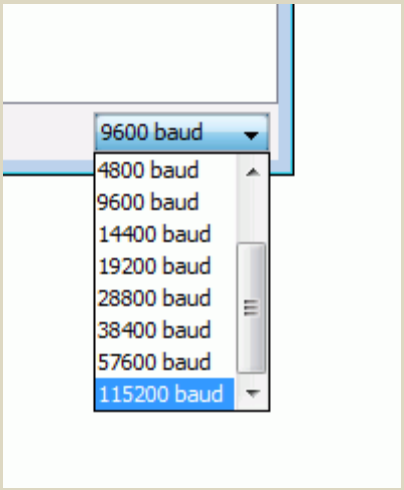
Préparation du Terminal côté PC dans le logiciel Arduino

- Côté PC, il faut ouvrir la fenêtre terminal de l'IDE Arduino : pour ce faire, un simple clic sur le bouton « Sérial Monitor ».

- La fenêtre « Terminal » s'ouvre alors :

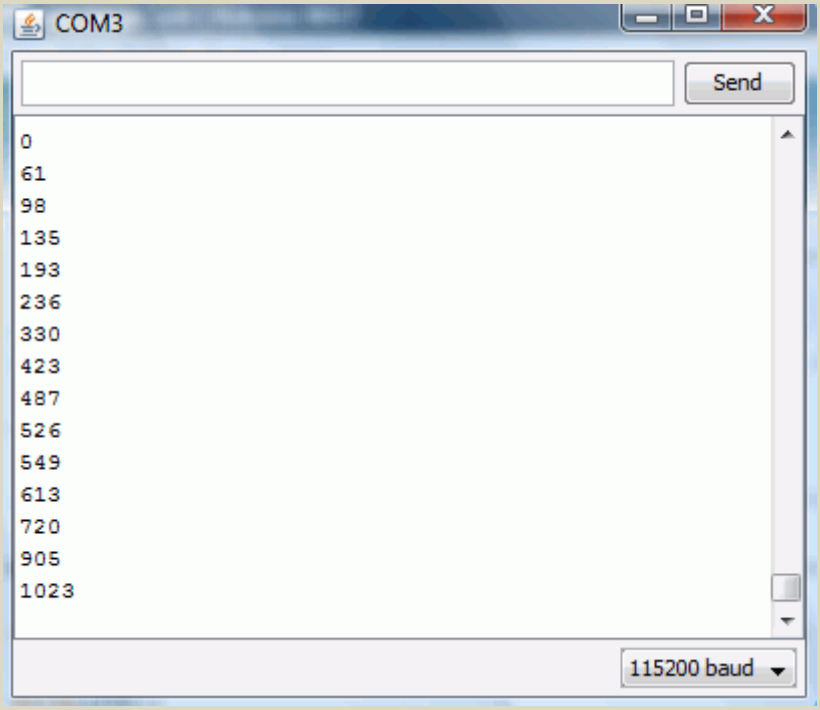


- Il faut alors régler le débit de communication sur la même valeur que celle utilisée par le programme avec lequel nous allons programmer la carte Arduino :



Résultat obtenu

Lorsque vous ouvrez la fenêtre du Terminal Série du logiciel Arduino (voir ci-dessus), vous verrez s'afficher une série de valeurs comprises entre 0 et 1023, directement corrélées à la position de l'axe du potentiomètre. Lorsque vous tournez l'axe, les valeurs affichées varient instantanément. A noter qu'il s'agit d'une valeur brute que l'on pourra aisément convertir en volt ou autre grandeur voulue par simple calcul.



Au niveau de la fonction setup ()

Dans votre programme, la toute première chose que l'on fait est d'initialiser la communication série entre votre ordinateur et la carte Arduino, à 9600 bits de données par seconde à l'aide de l'instruction Serial.begin :

Au niveau de la fonction loop()

Ensuite, dans la fonction loop, vous devez créer une variable pour stocker la valeur de la mesure analogique en provenance du potentiomètre (qui pourra prendre une valeur entre 0 et 1023, parfait pour une variable de type int). Dans cette variable, nous mettrons la valeur obtenue à l'aide de l'instruction analogRead qui renverra la valeur de la tension sur la broche analogique 2 symbolisée par A2 :

Les broches analogiques sont identifiées par le symbole A0, A1...A5.

Enfin, on réalise l'affichage de l'information obtenue dans la fenêtre du Terminal Série en tant que valeur décimal.

Ceci est réalisé à l'aide de la commande Serial.println .

Ecrire le code du programme puis valider son fonctionnement.

4- Lire l'état d'un bouton poussoir et suivant sa valeur allumer ou éteindre une LED ➔ If (condition) Then

Cet exemple vous montre comment allumer ou éteindre une LED suivant l'état d'un bouton poussoir (ou BP).

L'appui sur le bouton poussoir affiche un message dans la fenêtre Terminal Série sur l'ordinateur et allume une LED.

Réaliser le schéma de câblage avec le logiciel Fritzing.

A télécharger ici : http://fritzing.org/home/

```
pinMode(2, INPUT_PULLUP);
```

Expliquer cette ligne de commande en vous aidant de cette page internet.

https://openclassrooms.com/courses/programmez-vos-premiers-montages-avec-arduino/le-bouton-poussoir

Réaliser le programme et valider son fonctionnement.