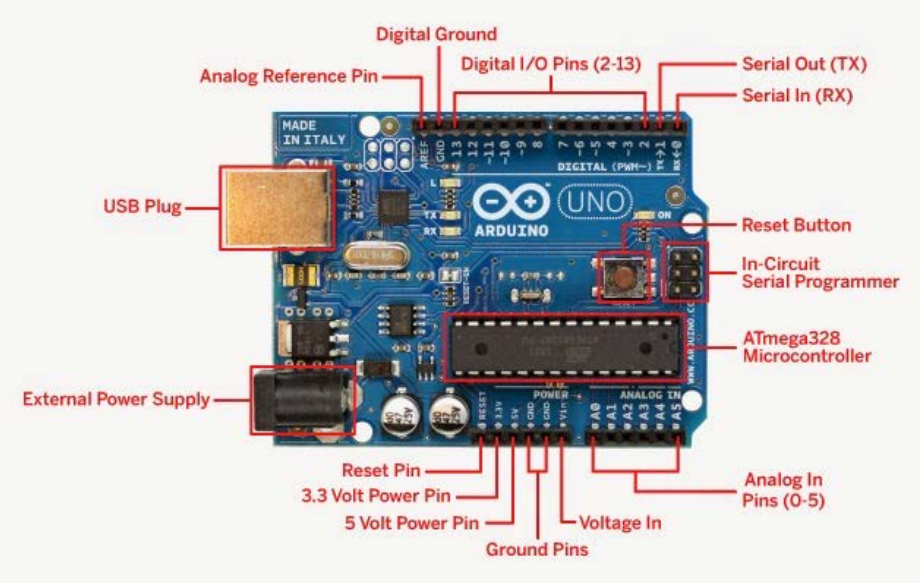
Introduction aux microcontrôleurs

Ce laboratoire vous introduira aux microcontrôleurs qui sont omniprésents et très importants en mécatronique. Un microcontrôleur (en notation abrégée µc, ou uc ou encore MCU en anglais) est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires (mémoire morte et mémoire vive), unités périphériques et interfaces d'entrées-sorties. Les microcontrôleurs se caractérisent par un plus haut degré d'intégration, une plus faible consommation électrique, une vitesse de fonctionnement plus faible (de quelques mégahertz jusqu'à plus d'un gigahertz) et un coût réduit par rapport aux microprocesseurs polyvalents utilisés dans les ordinateurs personnels. Les microcontrôleurs sont fréquemment utilisés dans les systèmes embarqués, comme les contrôleurs des moteurs automobiles, les télécommandes, les appareils de bureau, l'électroménager, les jouets, la téléphonie mobile, etc.[1]

Le Arduino Uno n'est pas un microcontrôleur comme tel. C'est plutôt une plateforme basée sur un microcontrôleur (ATmega328) qui inclut aussi plusieurs interfaces et circuits pour en faciliter l'utilisation.



[2]

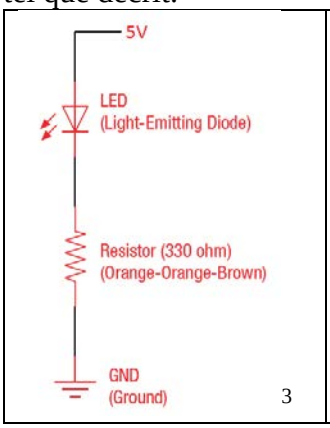
[1] https://fr.wikipedia.org/wiki/Microcontr%C3%B4leur

[2] http://3.bp.blogspot.com/-6spXxiWCc3w/U7N5fY-qmXI/AAAAAAAAAZI/NKdQ-9r_-Ow/s1600/arduino+uno+hardware+components.jpg

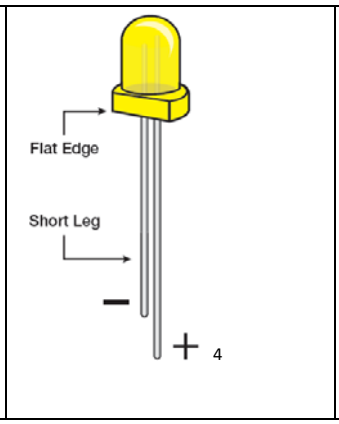
EXERCICE 1 – Allumer une LED

Matériel requis : 1) Arduino; 2) Câble USB; 3) 1 LED Verte; 4) 1 résistance de 330 ohms; 5) des fils; 6) Planche de prototypage breadboard [nous utiliserons celle du Proto-Board, sans alimenter celui-ci – ne branchez pas le Proto-Board].

Nous allons d'abord simplement faire allumer une LED. Le circuit que nous souhaitons réaliser est le suivant. Le circuit à connecter est aussi détaillé. Faites attention au branchement de la LED tel que décrit.



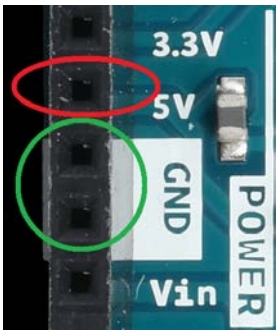
[3]

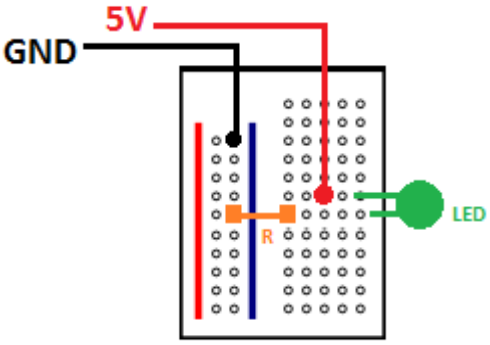


[4]

***** Attention : les LED possèdent une polarité, c'est-à-dire qu'il faut les connecter du bon côté. En regardant la LED, la face plate indique le côté de la patte négative.**

Connecter le 5V du Arduino à la patte + de la LED. Connecter le GND (terre) du Arduino à la plaquette (voir dessin). Placer une résistance entre la patte négative de la LED et la mise à la terre (GND). Il est important de placer une résistance pour éviter que la LED ne reçoive trop de courant, ce qui pourrait la brûler.





Connecter le Arduino à l'ordinateur via le câble USB. C'est l'ordinateur qui alimentera le Arduino et donc le reste du circuit (et non le Proto-Board). Votre LED devrait s'allumer.

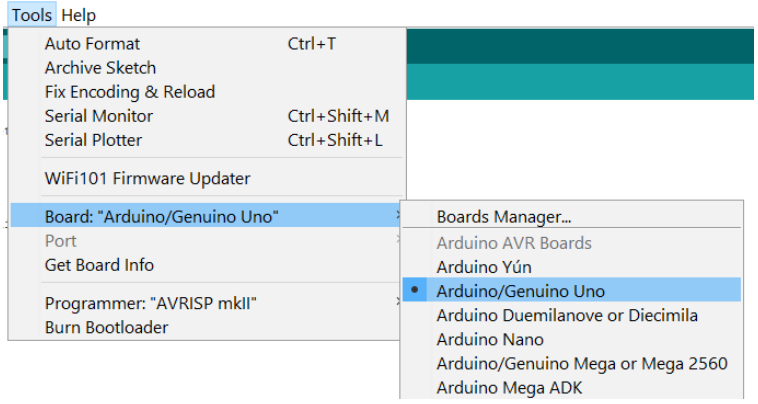
EXERCICE 2 – Clignotement de la LED

Nous allons maintenant utiliser le Arduino afin de faire clignoter la LED. Garder le même circuit qu'à l'exercice 1 mais cette fois remplacer le 5V par un fil connectant la sortie digitale 13 à la LED. À ce moment ci, il est possible que votre LED clignote ou ait un autre comportement. En effet, le microcontrôleur exécute le dernier code qui lui a été programmé. Ne vous inquiétez pas, nous allons remplacer le code par le vôtre bientôt !

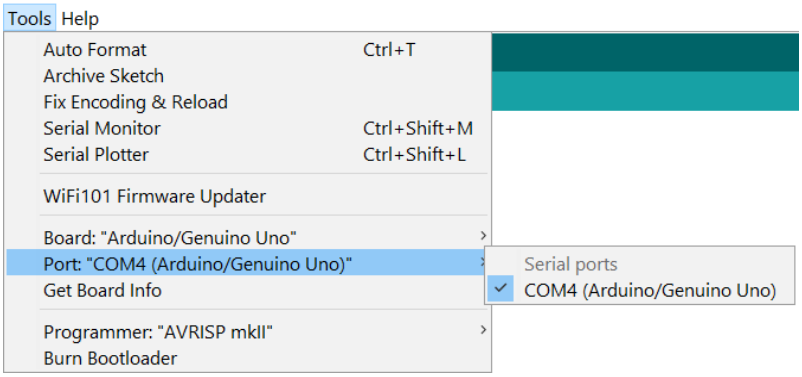
[3] Arduino SIK guide
[4] Arduino SIK guide

Nous allons maintenant nous connecter au Arduino via l'ordinateur.
- Ouvrir le logiciel Arduino
- Créer un nouveau fichier (File/New), vous obtiendrez :

- Nous devons maintenant indiquer au logiciel quelle plateforme nous utilisons. Aller dans Tools/Board et sélectionner Arduino/Genuino Uno.



- Il faut maintenant indiquer à quel port de communication de l'ordinateur votre Arduino est attaché. Assurez-vous que votre Arduino est connecté à l'ordinateur. Aller dans Tools/Port. S'il n'y a qu'un seul port, sélectionnez-le. *S'il y a plusieurs ports d'indiqués, cela veut dire que d'autres équipements sont aussi connectés à l'ordinateur. Afin de savoir quel est le port connecté au Arduino, déconnecter le Arduino, réouvrir le menu des ports et regarder le port qui a disparu. Celui-ci devrait être votre port connecté au Arduino. Reconnecter le Arduino et sélectionner ce port.*



Nous allons maintenant écrire le code permettant de faire clignoter la LED. L'objectif est d'envoyer sur la PIN 13 une tension étant parfois 5V et le reste du temps à 0V.
- Plusieurs PINs du Arduino peuvent servir d'entrée ou de sortie (Input/Output → I/O). Ceci laisse plus de flexibilité à l'utilisateur dépendamment de ses besoins. Ici, nous allons utiliser la PIN 13 comme sortie. Nous devons donc indiquer que nous allons l'utiliser comme sortie. Pour ce faire, dans le bloc « setup », entrer le code suivant : « pinMode(13, OUTPUT) » :

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

Il est ici très important de faire une parenthèse. Votre code contient deux fonctions : la fonction *setup* et la fonction *loop*. Lorsque le Arduino est lancé (lorsqu'il est ouvert ou lorsqu'un nouveau code est envoyé au Arduino), il exécutera une fois le code qui se trouve dans *setup* et il exécutera ensuite en boucle à l'infini le code qui se trouve dans *loop*. Ici nous désirons indiquer une seule fois que la PIN13 sera une sortie et non à chaque boucle.

Nous allons ensuite faire clignoter la LED avec le code suivant. Le code *digitalWrite(13, HIGH)* indique de mettre la PIN 13 à une tension élevée (5V). Cette tension se maintiendrait ensuite à l'infini. Nous allons ensuite appeler la fonction *delay(1000)* qui mettra le programme en pause pendant 1000ms (1 seconde). Finalement, nous allons remettre la tension à *Low* et reprendre une pause de 1000ms.

En résumé, une fois le programme lancé, la PIN13 sera configurée en sortie. Ensuite en boucle nous aurons : mettre la PIN13 à 5V (allumer la LED), attendre 1 seconde, mettre la PIN13 à 0V, attendre 1 seconde et ainsi de suite, ce qui aura pour effet de faire clignoter la LED.

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

void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000); // wait for a second
  digitalWrite(13, LOW); // set the LED off
  delay(1000); // wait for a second
}
```

Note : La PIN 13 est une entrée/sortie digitale, c'est-à-dire qu'elle peut prendre 2 valeurs : 0V ou 5V. Nous pouvons donc choisir une tension de 5V (high) ou de 0V (low). Au contraire, une entrée et/ou sortie qui pourrait recevoir des valeurs continues entre 0 et 5V se dirait analogique (voir PIN A0-A5).

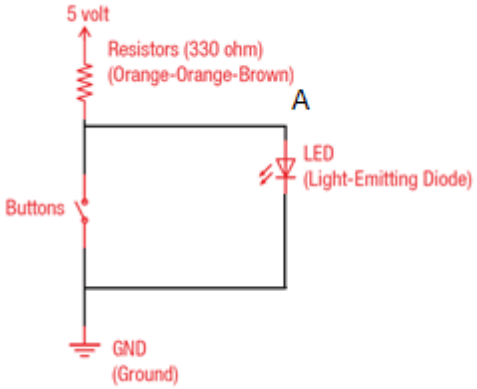
Il nous reste maintenant à programmer ce code dans le Arduino. Au haut du logiciel Arduino, vous avez différentes options qui sont respectivement de vérifier (verify), téléverser (upload), nouveau (new), ouvrir (open), sauver (save).

- Commencer par sauvegarder votre sketch dans votre compte. Ensuite appuyer sur *Verify* ce qui aura pour effet de vérifier la validité de votre code. Vous verrez une barre verte se remplir. S'il y a une erreur dans votre code, la barre deviendra rouge et le logiciel vous indiquera l'erreur dans la fenêtre du bas.
- Appuyer ensuite sur *Upload* afin d'envoyer le code au Arduino. Lors de cette étape, il se peut que votre LED démontre un comportement erratique, c'est normal en phase de programmation. Une fois le transfert terminé, votre LED devrait clignoter !
- Essayer différents temps de délai (ex. [300-300], [500-2000]) afin de vous assurer que votre code fonctionne bien.

→ Félicitations ! Vous avez fait votre premier programme ! Ça mérite un high-five. Mais pas un fort, juste un petit. On a quand même juste fait flasher une LED. Mais ça s'en vient bien !

EXERCICE 3 – Contrôle de la LED avec un interrupteur

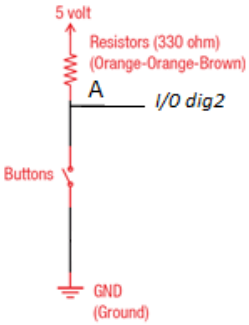
Nous allons maintenant contrôler la LED avec un interrupteur (bouton) seulement de manière électronique. Nous allons utiliser le 5V du Arduino mais sans programmer aucun code. **Déconnecter le Arduino de l'ordinateur avant de faire le montage et faites-le vérifier avant de reconnecter le Arduino.**

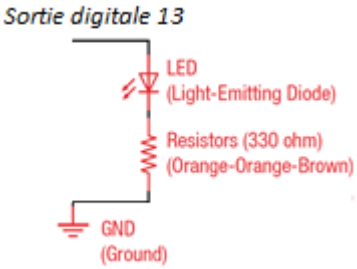


On désire donc faire le montage de droite. Lorsque le bouton n'est pas pressé, il ne laisse pas passer le courant. Le point « A » est alors à 5V. Lorsque le bouton est pressé, il laisse passer le courant et il n'y a donc plus de courant qui passe dans la LED car celle-ci est court-circuité par le lien direct à la terre via le bouton qui a une résistance presque nulle. La LED est donc allumée par défaut et s'éteint lorsqu'on appuie sur le bouton.

EXERCICE 4 – Interruption

Comme nous l'avions vu précédemment, le Arduino exécute en boucle le code contenu dans la fonction *loop*. Il arrive cependant souvent que nous voulons réagir à des évènements qui arrivent en temps réel. Ici, nous allons modifier le comportement de notre LED lorsque le bouton est appuyé lors d'un cycle. Pour ce faire, connecter le bouton (le fil allant au point A) à l'I/O digitale 2 et l'entrée de la LED à la sortie digitale 13.





- Ouvrir un nouveau sketch.
- Entrer le code suivant

```
int time=1000;

void setup()
{
  pinMode(13, OUTPUT);
  pinMode(2, INPUT);
  attachInterrupt(0, pin_ISR, FALLING);
}

void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(time); // wait for a second
  digitalWrite(13, LOW); // set the LED off
  delay(time); // wait for a second
}

void pin_ISR()
{
  if (time==1000)
  {
    time=300;
  }
  else
  {
    time=1000;
  }
}
```

Différents éléments se sont ajoutés ici. Il est important de bien comprendre ce code.
- Premièrement, nous avons déclaré la variable « int time=1000 », c'est-à-dire une variable entière (-2,-1,0,1,2,3,etc.) au haut de tout sans qu'elle soit dans une fonction. Ceci se nomme une variable globale car elle peut être utilisée et modifiée par toutes les fonctions. En effet, si on l'avait déclarée dans *loop* uniquement, elle n'aurait pas pu être utilisée dans *pin_ISR*.
- Nous avons ensuite déclaré la PIN2 comme entrée (qui recevra le signal 0 ou 5V du bouton).
- Nous avons ensuite attaché une fonction d'interruption *pin_ISR* qui sera immédiatement lancée lorsque l'entrée de la PIN2 passe de 0 à 5V. (Sans entrer dans les détails ici le chiffre 0 dans l'appel de *attachInterrupt* représente la PIN2.

Sans se préoccuper de la fonction *pin_ISR* pour l'instant, vous pouvez voir que *delay(1000)* a été remplacé par *delay(time)*. C'est équivalent ici car *time* vaut 1000 mais ceci nous permettra de modifier cette variable *time* avec le code.

Ce que l'on désire maintenant c'est que la fréquence du clignotement change lorsqu'on fait un clic sur le bouton (appuyer-relâcher). On pourrait par exemple rajouter une condition à la fin de la boucle qui vérifie si le bouton est appuyé et qui modifie le temps si c'est le cas. Par contre, cette condition ne serait vérifiée qu'à la toute fin de la boucle. Il faudrait donc maintenir le bouton 2 secondes, le temps que le code soit rendu à cette section. Ici nous voulons simplement appuyer une fois rapidement sur le bouton n'importe quand dans son cycle et que ceci modifie immédiatement le temps de clignotement.

Ceci est fait par la fonction d'interruption *pin_ISR*. Celle-ci est lancée lorsqu'il y a un front descendant sur le bouton (lorsque la sortie du bouton passe de 5V à 0V, donc quand il est pressé de la manière dont nous avons fait le circuit). En regardant ce code, vous verrez que si la variable *time* est de 1000, elle sera modifiée pour 300. Sinon, elle reviendra à 1000. Essayez ce code.

EXERCICE 5 – Interruption 2

Nous allons voir un deuxième exercice sur les interruptions.
La LED va clignoter avec des délais de 1000ms. Si lors d'un cycle le bouton est pressé, nous voulons qu'à la fin du cycle la LED clignote 4 fois rapidement. Un cycle est simplement défini comme LED allumée, LED éteinte. Pour ce faire, nous allons utiliser le code suivant.

```
int buttonState=0;

void setup() {
  pinMode(13, OUTPUT);
  pinMode(2, INPUT);
  attachInterrupt(0, pin_ISR, FALLING);
}

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

  if (buttonState==HIGH)
  {
      buttonState=LOW;
      for (int i=0;i<4;i++)
      {
        digitalWrite(13, HIGH);
        delay(300);
        digitalWrite(13, LOW);
        delay(300);
      }
  }
}

void pin_ISR() {
  buttonState = HIGH;
}
```

Vous pouvez analyser ce code pour bien le comprendre. En gros, la variable globale *buttonState* est initialisée à 0.

À la fin d'un cycle, lorsque celle-ci vaut 1, on remet la valeur à 0 (pour éviter que le code qui suit soit ensuite effectué à chaque boucle) et on fait clignoter la LED 4 fois plus rapidement.

Il est important de comprendre que le code contenu dans les interruptions doit normalement être très bref.

Vous ne pouvez pas copier-coller ce code directement, c'est une image. C'est voulu, c'est pour que vous puissiez bien comprendre où il faut une accolade ou une parenthèse ou un point-virgule, désolé !

Juste en passant, vous êtes en train de coder en C++. C'est tout.

EXERCICE 6 – Gestion d'un feu de circulation

C'est le temps des choses sérieuses !
On vous demande ici de faire la gestion d'un feu de circulation.
- Placer 3 LED (vert, jaune, rouge) à gauche et centrées verticalement sur la plaque
- Placer 3 LED (vert, jaune, rouge) au centre horizontalement et au haut de la plaque de prototypage (un peu pour simuler une intersection, le placement n'est pas critique).
- Connecter les LED aux sorties digitales 8 à 13 du Arduino.
- **<u>Ne pas oublier les résistances de 330 ohms et le sens de branchement des LED.</u>**
→ C'est en fait le même branchement que vous aviez fait mais avec six LED plutôt qu'une.
- Le cycle doit commencer avec le feu de gauche avec la lumière verte.
- Pour la première mise en marche, lorsque le circuit est allumé, les lumières rouges doivent être allumées pendant 2 secondes pour les deux feux (juste la première fois, pas tous les cycles)

Voici les requis du directeur des transports :
- Pour le feu de circulation de gauche : la lumière verte doit être active pendant 4 secondes.
- Pour le feu de circulation du haut : la lumière verte doit être active pendant 2 secondes.
- Dans les deux cas, la lumière jaune doit être active 2 secondes.
- Dans les deux cas, vous devez gérer la lumière rouge afin que le résultat soit logique et qu'il n'y ait pas d'accidents !
- Un requis supplémentaire est que lorsqu'une lumière passe de jaune à rouge, il ne faut pas immédiatement que l'autre passe au vert. Il faut un moment de 1 seconde où les deux lumières sont rouges (pour éviter les accidents avec ceux qui passent tardivement sur la jaune).
→ N'hésitez pas à poser des questions

Lorsque vous aurez terminé, ça vaut un gros top-gun high-five, tel qu'expliqué en classe.

EXERCICE 7 – Gestion d'un feu de circulation – Ajout d'un indicateur piéton

Ici on reprend le même montage qu'à l'exercice 6. Par contre, on doit ajouter un bouton lié à une interruption sur la PIN2. Lorsque le bouton est pressé durant un cycle, nous voulons activer une lumière piétonne à la fin du cycle. Pour indiquer ceci, vous allez faire clignoter toutes les lumières avec des délais de 300ms pour environ 2 secondes.