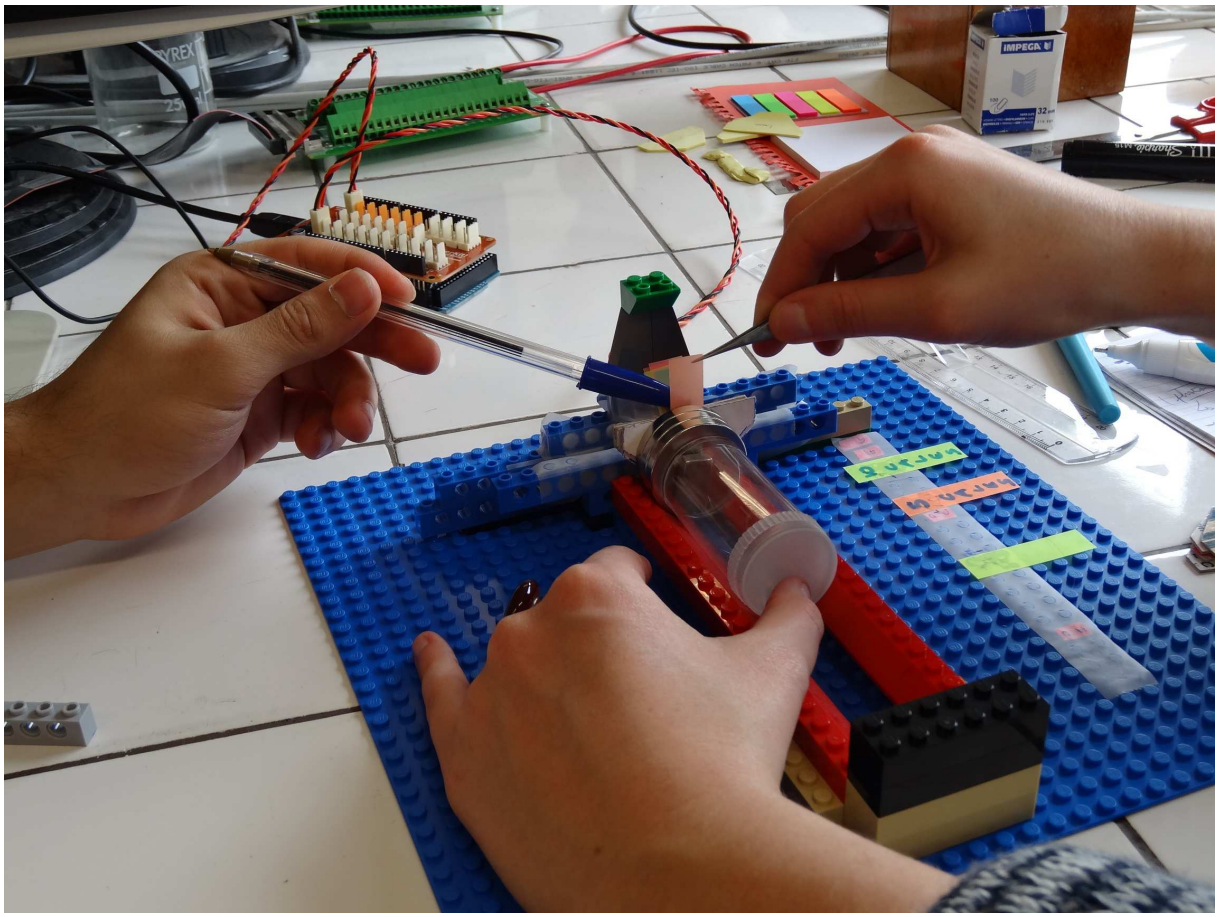
Projets de physique statistique

Magistère de physique fondamentale

Université Paris Sud

MICROCONTROLEUR ARDUINO



Printemps 2015

Frédéric Bouquet et Julien Bobroff

Merci de nous faire suivre vos remarques et corrections :

frederic.bouquet@u-psud.fr, julien.bobroff@u-psud.fr

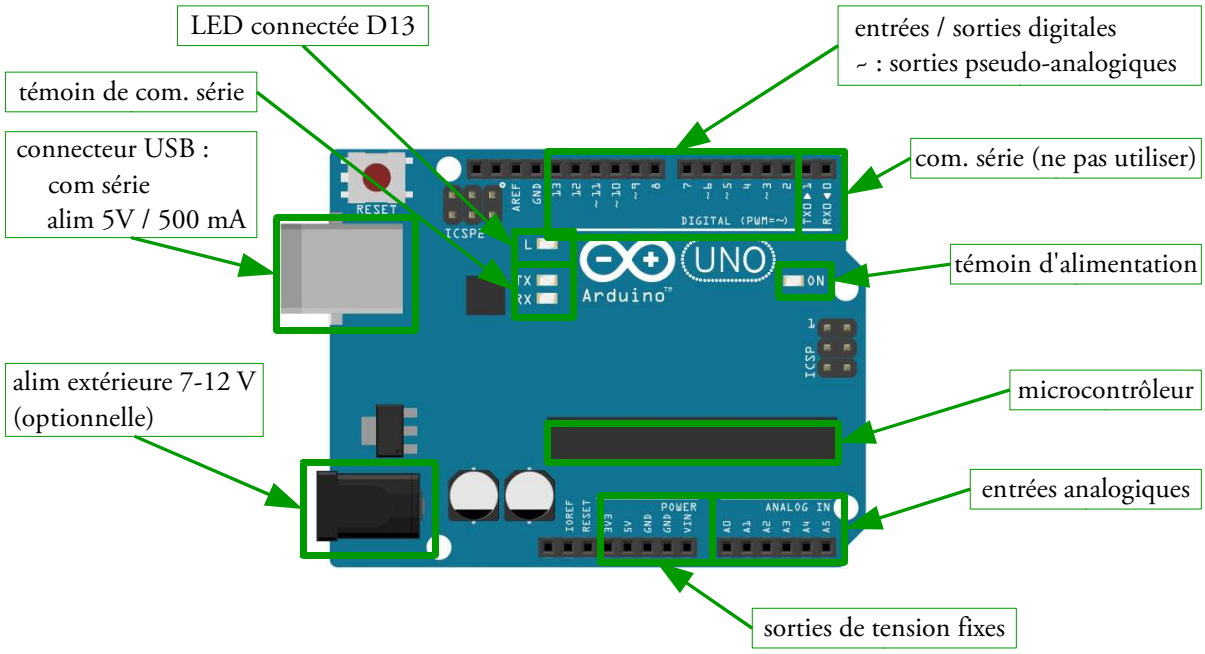
MICROCONTROLEUR ARDUINO

La carte Arduino est un microcontrôleur open source, qui a été adopté par la communauté des Makers. De nombreuses réalisations, conseils, tutoriaux peuvent se trouver facilement sur le net. Arduino permet toute sortes de réalisations diverses, rendant facilement accessible ce qui nécessitait avant de l'électronique compliquée. Le but de ces séances est d'utiliser cet outil pour faire de la physique, de construire votre propre système de mesure.

Les 2 sites webs de référence :
http://www.arduino.cc/
http://playground.arduino.cc/French/Reference

LE MICROCONTROLEUR ARDUINO

Un microcontrôleur est un système qui ressemble à un ordinateur : il a une mémoire, un processeur, des interfaces avec le monde extérieur. Les microcontrôleurs ont des performances réduites, mais sont de faible taille et consomment peu d'énergie, les rendant indispensables dans toute solution d'électronique embarquée (voiture, porte de garage, robots, …). La carte Arduino n'est pas le microcontrôleur le plus puissant, mais son architecture a été publiée en open-source, et toute sa philosophie s'appuie sur le monde du libre, au sens large.

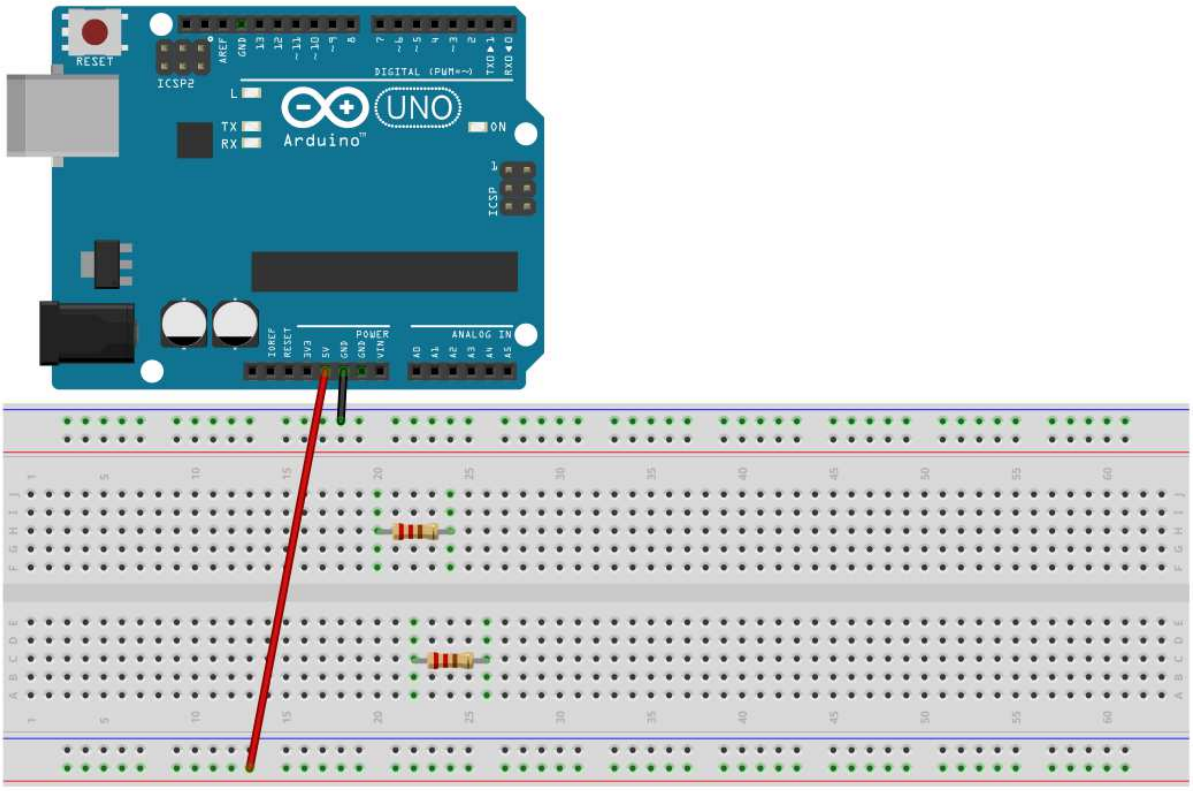


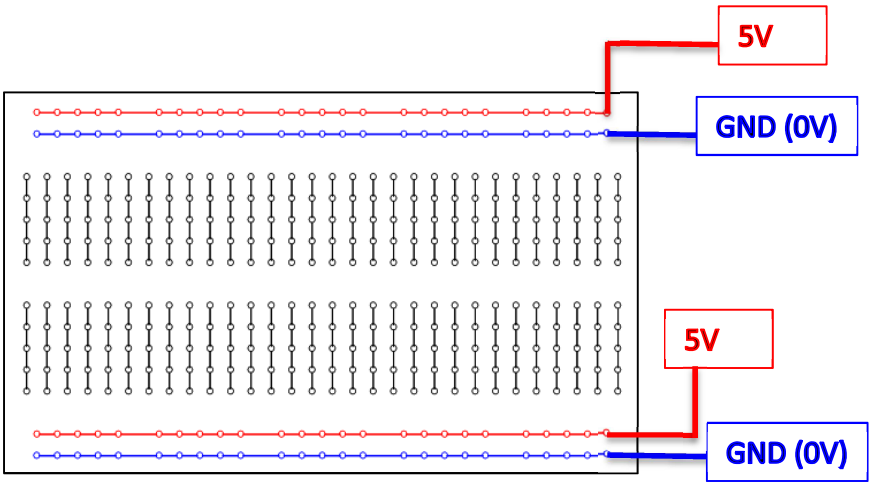
La carte Arduino se relie à un ordinateur par un câble USB. Ce câble permet à la fois l'alimentation de la carte et la communication série avec elle.

Attention : il y a quelques précautions à suivre pour ne pas endommager le matériel. Ne pas respecter ces consignes peut entraîner la perte de la carte, et potentiellement celle du port USB de l'ordinateur. Ces consignes sont données à la fin de ce document, respectez-les !

LA PLATINE D'EXPERIMENTATION (BREADBOARD)

Pour faire des montages électriques rapidement, on utilise une platine d'expérimentation appelée breadboard dans laquelle on peut planter des fils ou des composants sans avoir besoin de soudure. Ces platines contiennent deux bandes latérales de chaque côté, qu'on réserve en général aux tensions d'alimentation (O V et 5 V). La partie centrale est séparée en deux bandes distinctes (voir les connections cachées qui relient les différentes entrées du breadboard ci-dessus).





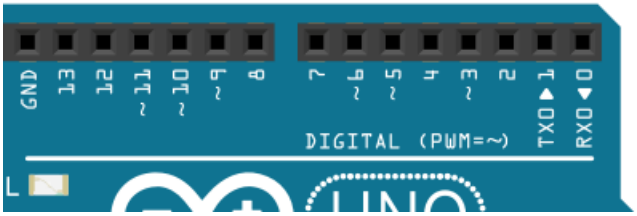
LES ENTREES / SORTIES

Les entrées / sorties (I/O – input/output) représentent le moyen qu'a la carte arduino d'interagir avec l'extérieur. Les sorties sont contrôlées par la carte, cela permet au programme du microcontrôleur de déclencher des actions (allumer ou d'éteindre une LED, un ventilateur, un moteur). Les entrées sont lues par le microcontrôleur, ce qui lui permet de connaitre l'état du système auquel il est relié.

Il y a deux sortes d'I/O : les I/O numériques, et les I/O analogiques.

LES ENTREES / SORTIES NUMERIQUES

Les entrées / sorties numériques ne peuvent prendre que deux valeurs, la valeur LOW (~ GND, 0 V), et la valeur HIGH (~ 5 V). La valeur d'un port numérique peut donc être codée sur un bit, 0 ou 1, true ou false.



La carte arduino comporte 14 I/O numériques (appelées DIGITAL sur la carte), numérotées de 0 à 13 (voir le schéma ci-dessus), et appelées D0, D1, D2, … D13. Chacun de ces ports peut-être déclaré comme étant une entrée ou comme une sortie dans le programme du microcontrôleur. Les deux premiers ports (D0 et D1) sont réservés à la communication série, il ne faut pas les utiliser. Le dernier port, D13, possède un indicateur lumineux, une LED qui s'allume quand le port est HIGH, et qui s'éteint quand le port est LOW.

Le port GND est la masse de la carte (0 V).

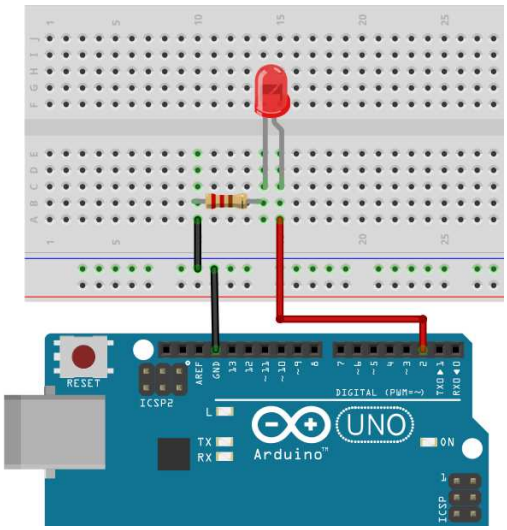
LES SORTIES NUMERIQUES

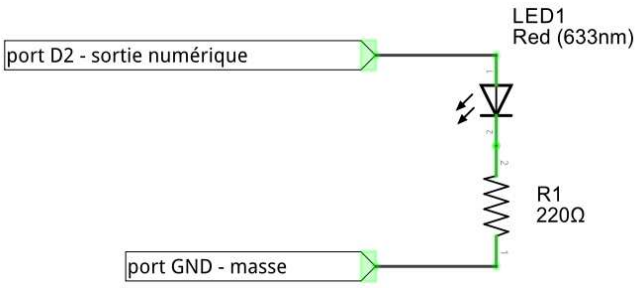
Chacun des 14 ports numériques de la carte peuvent être utilisés en sortie. Si un port est déclaré comme une sortie, le microcontrôleur contrôle la valeur de ce port.

Attention, le courant que peut délivrer un port digital en sortie est limité à 40 mA : en demander plus peut endommager la carte ! Ce genre de situation peut arriver si un port, déclaré comme une sortie, est directement relié à la masse (port GND) avec une résistance très faible (un fil), et que le programme bascule la sortie en HIGH (5 V). L'inverse est également dangereux (une sortie numérique reliée au port 5 V et basculée sur la valeur LOW).

Les sorties numériques ne peuvent pas fournir une grande puissance électrique (40 mA max sur 5 V). On les utilise pour échanger des informations (par exemple les ports D0 et D1 servent à la communication série avec l'ordinateur), ou pour déclencher des actions : par exemple allumer une LED.

Voici un montage simple pour contrôler l'état de la LED :





Le programme téléversé sur le microcontrôleur définit le port D2 comme une sortie. Quand le programme bascule la valeur de ce port à LOW, la LED est éteinte. Quand le programme bascule la valeur de D2 à HIGH, la LED s'allume. La résistance R1 sert à limiter le courant, ce qui protège à la fois le microcontrôleur et la LED (qui a également un courant critique au-delà duquel elle fume). Si la LED est montée en sens inverse, elle ne s'allumera jamais (c'est une diode).

Instructions de programmation utiles :

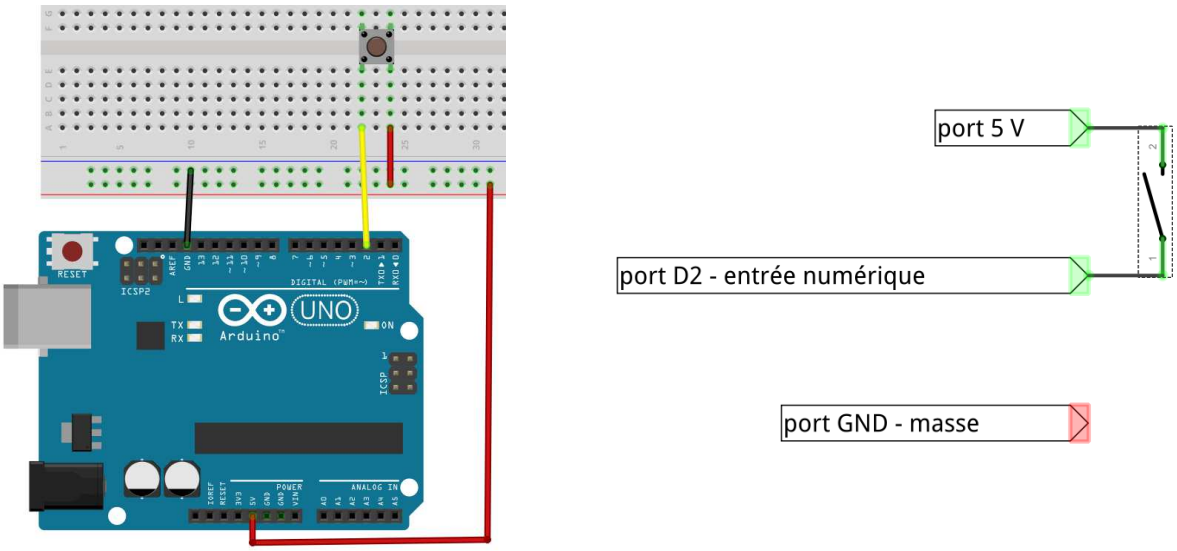
- `pinMode(2, OUTPUT);` // définit le port D2 comme une sortie ;
- `digitalWrite(2, HIGH);` // bascule l'état du port D2 à HIGH ;
- `digitalWrite(2, LOW);` // bascule l'état du port D2 à LOW.
-

LES ENTREES NUMERIQUES

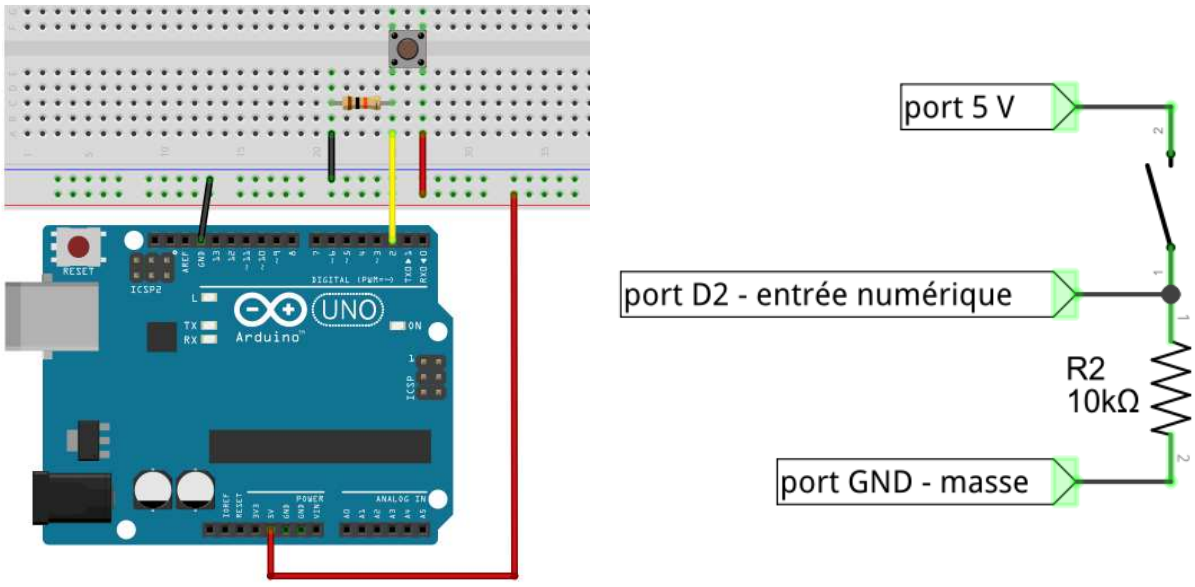
Chacun des 14 ports numériques de la carte peuvent être utilisés en entrée. Si un port est déclaré comme entrée, l'état du port sera lu par arduino (HIGH ou LOW), et cette valeur pourra être utilisée dans le programme pour déclencher telle ou telle action.

Déclaré comme une entrée, un port numérique sera considéré comme HIGH ou LOW selon la valeur de la tension mesurée par la carte. En gros, les tensions inférieures à 1 V seront lues comme LOW, les tensions supérieures à 4 V seront lues comme HIGH. Il faut éviter les tensions intermédiaires, qui risquent de donner un résultat indéterminé. Attention, une tension supérieure à 5.5 V peut détruire la carte arduino.

Attention : une entrée numérique non connectée sera flottante (sans tension affectée), son état sera indéterminé. La lecture de ce port peut donner n'importe quel résultat. Il faut éviter cette situation. Par exemple, si on veut déterminer si un interrupteur est appuyé ou non, le montage suivant n'est pas bon car lorsque l'interrupteur est ouvert, le port D2 n'est alors plus connecté à rien, il n'y a aucune tension définie. Si le microcontrôleur lit l'état de cette entrée, le résultat sera aléatoire. C'est le cas du montage suivant :



Le montage suivant corrige ce problème : cette fois, la masse GND est reliée au via une résistance R2 très élevée (10kOhms) montée en parallèle , appelée résistance de pull-down.



Maintenant, quand l'interrupteur est ouvert, D2 et la masse sont reliés directement sans courant entre eux, et la carte mesure bien 0 V (D2 est au même potentiel que la masse GND). Quand l'interrupteur est fermé, le potentiel du port D2 monte à 5 V (HIGH), et un courant de 0.5 mA circule entre le port 5 V et le port GND, ce qui n'est pas un problème.

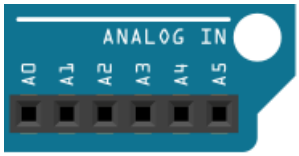
Attention : si le port D2 est déclaré par erreur comme une sortie au lieu d'une entrée dans le programme, il y a un risque de court-circuit si l'interrupteur est refermé.

Instructions de programmation utiles :

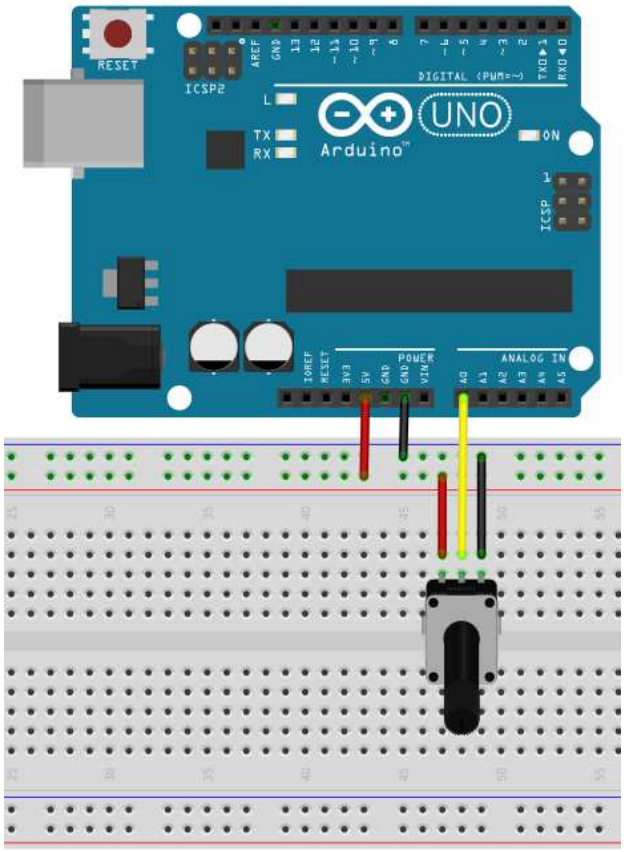
- `pinMode(2, INPUT); //` le port D2 est définit comme une entrée ;
- `buttonState = digitalRead(2); //` retourne la valeur du port D2 (true ou false, 0 ou 1, HIGH ou LOW).

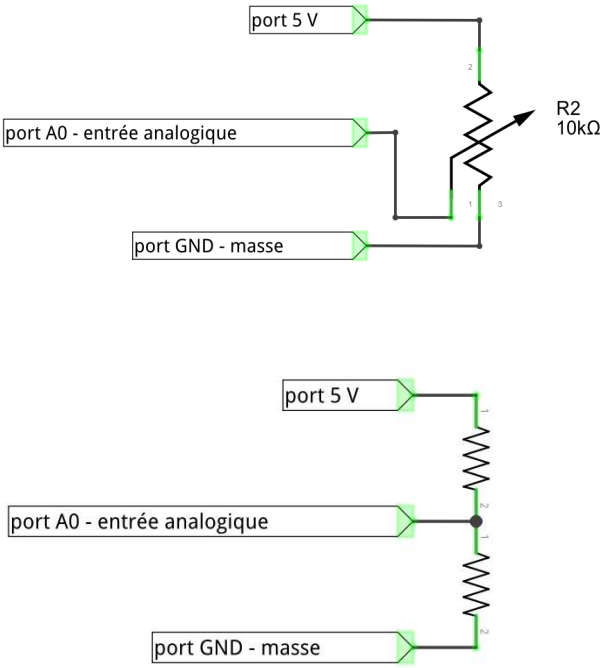
LES ENTREES ANALOGIQUES.

Une entrée analogique est une sorte de voltmètre : la carte lit la tension qui est appliquée sur le port. Cependant le microcontrôleur ne travaille qu'avec des chiffres : il faut donc transformer la tension appliquée en sa valeur numérique. C'est le travail du convertisseur analogue/digital, dit « CAD ».



Le CAD de la carte arduino travaille sur 10 bits : il accepte en entrée une tension comprise entre 0 V et V_{ref} une tension de référence, et fournit au microcontrôleur un chiffre entier compris entre 0 et 1023 (= 2^{10} - 1). Une tension inférieure à 0 V est lue comme 0, une tension supérieure à V_{ref} est lue comme 1023, une tension intermédiaire est lue comme un entier entre 0 et 1023, avec une relation linéaire. La tension V_{ref} est 5 V par défaut, mais cette valeur peut être changée dans le programme.

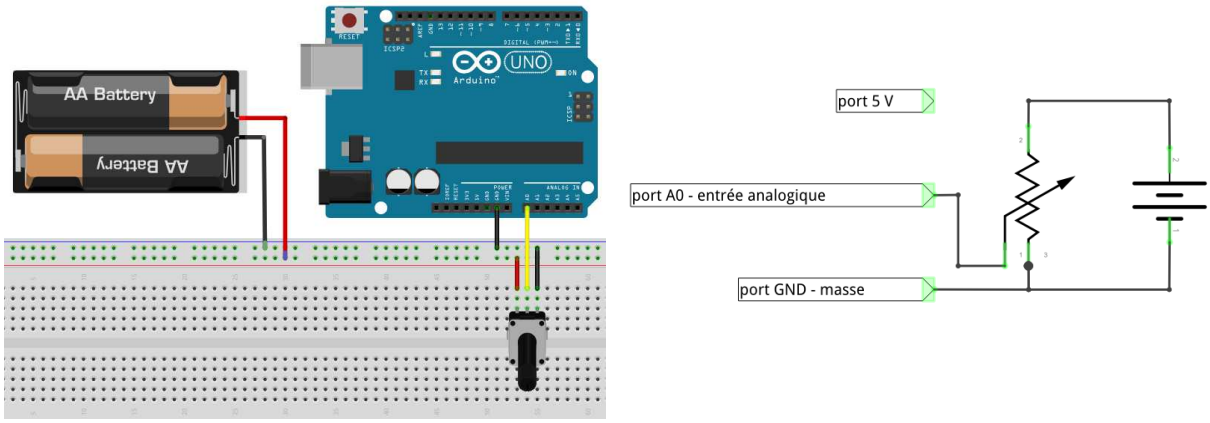




(schéma équivalent)

Le montage ci-dessus permet de mesurer une tension modifiée par un potentiomètre. Ce montage est équivalent à un pont diviseur de tension (voir le schéma équivalent). À la place du port 5 V de la carte arduino, on peut aussi utiliser une source externe pour alimenter le

potentiomètre. Il faut alors faire attention à bien définir une masse unique dans le circuit. C'est le cas du montage ci-après :

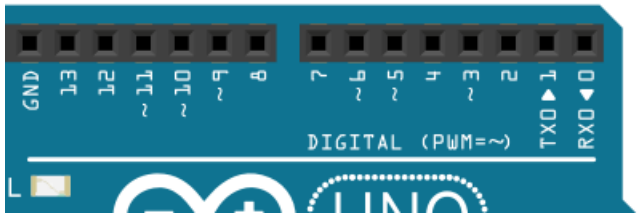


Instructions de programmation utiles :

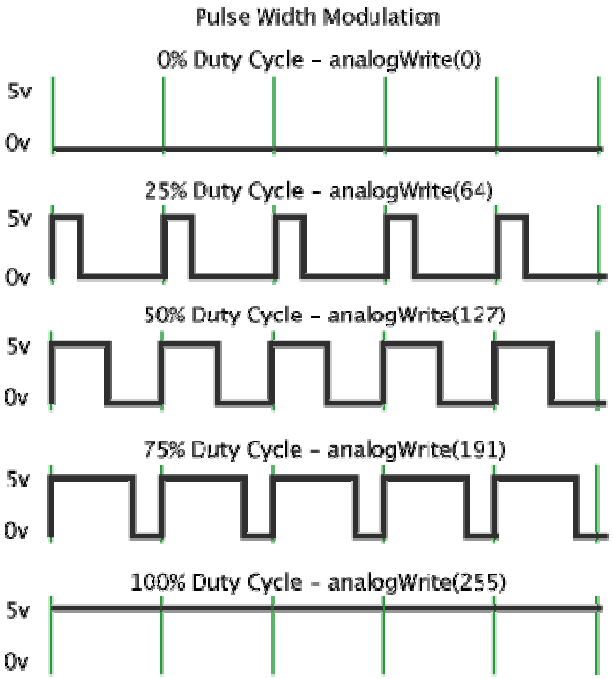
- `sensorValue = analogRead(A0); //` fonction qui retourne un entier compris entre 0 et 1023, selon la tension appliquée sur le port A0.

LES SORTIES ANALOGIQUES

La carte arduino ne possède pas de vraie sortie analogique, capable de produire une tension d'une valeur arbitraire choisie par l'utilisateur. Certains ports numériques peuvent cependant servir de sortie analogique en utilisant la technique de PWM (Pulse Width Modulation) : il s'agit des ports 3, 5, 6, 9, 10 et 11 (signalés par un ~ sur la carte). Ces ports peuvent simuler une tension entre 0 et 5 V en basculant rapidement entre leur état LOW (0 V) et HIGH (5 V). La valeur moyenne de la tension est alors 2.5 V si le port passe autant de temps dans un état que dans l'autre, mais en changeant ce rapport, la valeur moyenne de la tension peut être contrôlée de 0 à 5 V.



La carte arduino est capable de faire varier la valeur moyenne de ces ports avec une sensibilité de 8 bits : on fournit un chiffre entier compris entre 0 et 255 (= 2^8 - 1), et le port délivre une tension moyenne entre 0 et 5 V (0 = 0 V, 255 = 5 V).



tension moyenne = 0 V

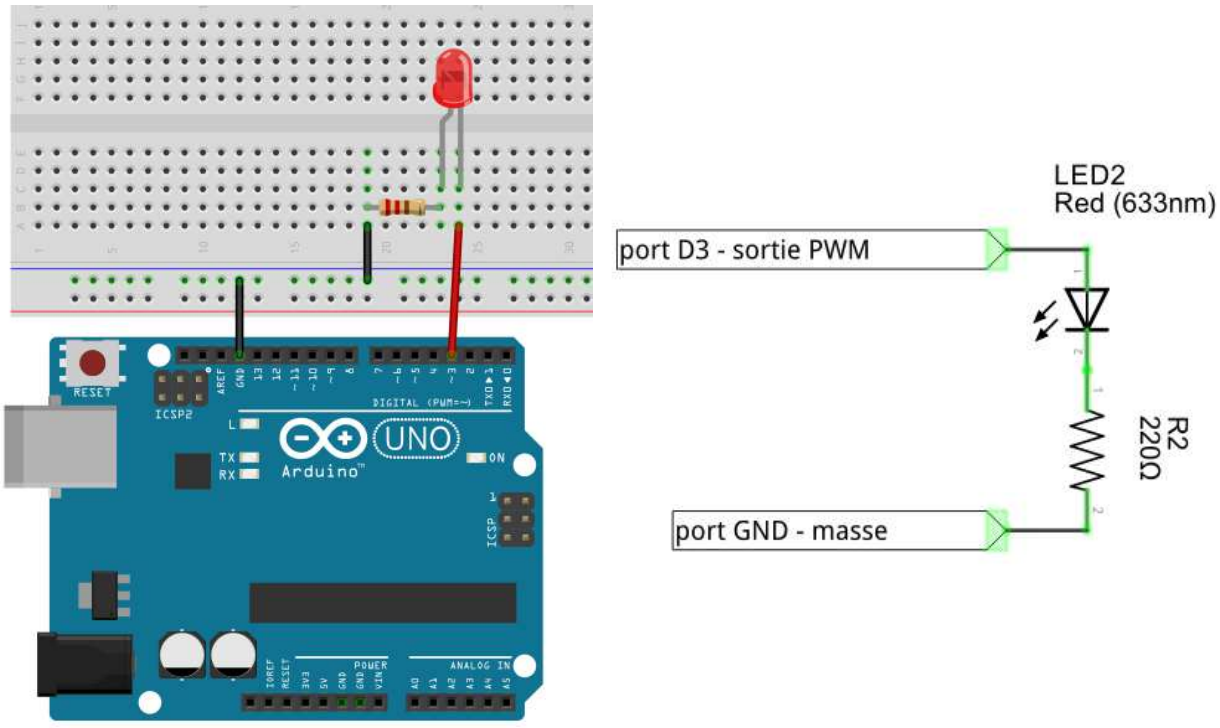
tension moyenne = 1.25 V

tension moyenne = 2.5 V

tension moyenne = 3.75 V

tension moyenne = 5 V

Pour certaines applications, une sortie PMW convient tout à fait. Par exemple, pour alimenter un moteur, ou pour faire varier la puissance d'une LED. Le circuit ci-après permet de faire varier l'intensité de la LED en fonction de la valeur demandée au port D3 défini par le programme comme une sortie analogique.



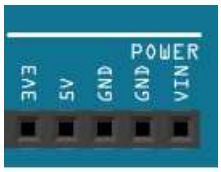
Si on a besoin d'une vraie tension analogique continue, il faut alors mettre un filtre passe-bas qui éliminera les hautes-fréquences et ne gardera que la valeur moyenne.

Instructions de programmations utiles

- `analogWrite(3, 127); //` envoie sur le port D3 une tension moyenne de 2.5 V.

LES TENSIONS DE REFERENCES

La carte arduino fournit des ports permettant d'accéder à certaines tensions de référence.



GND est la référence de la carte arduino par rapport à laquelle toutes les différences de tension sont mesurées. Si la carte est reliée à l'ordinateur par un câble USB, cette tension est celle de la terre.

Les ports 5V et 3V3 donnent accès aux tensions de 5 V et de 3.3 V. Ces tensions sont normalement régulées et précises. Une exception : quand la carte est branchée sur un port USB sans alimentation externe, le port 5 V ne provient plus de la carte arduino mais directement du câble USB, la tension de référence 5 V n'est alors plus aussi bien régulée.

VIN est la tension de l'alimentation externe, quand il y en a une.

Attention : si vous reliez directement le port 5 V au port GND (ou le port 3V3 au port GND, ou le port 5V au port 3V3), vous provoquerez un court-circuit qui endommagera la carte !

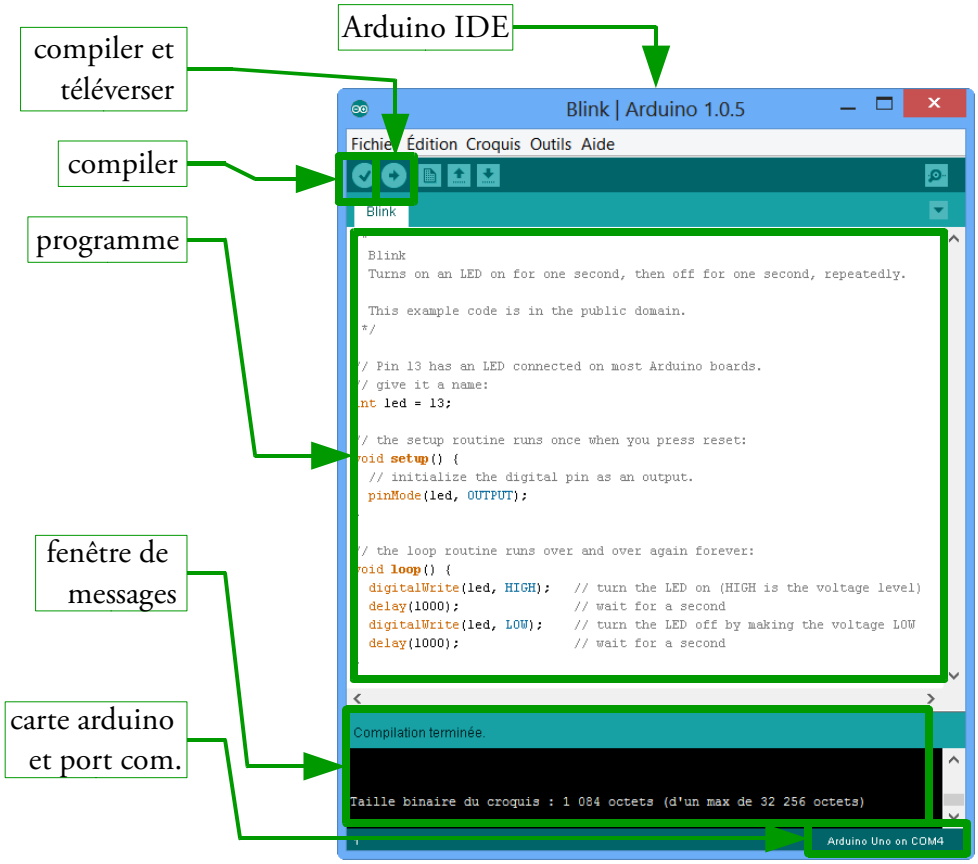
LE PORT USB

Le port USB permet à la fois l'alimentation de la carte Arduino et la communication série entre la carte et l'ordinateur. Une fois connectée, la carte Arduino apparaît dans le gestionnaire de matériel de votre ordinateur, connecté à un port série (COM1, COM4, …). Vous devez vérifier que l'IDE (le programme fourni par Arduino) est bien configuré pour dialoguer sur le bon port COM (et pour le bon type de carte Arduino !). L'IDE permet de scanner le port COM et de récupérer les messages éventuels de la carte. Utiliser toujours cet outil pour debugger votre programme quand il y a des informations qui circulent sur le port série : Outils / Moniteur série.

TRANSFERER UN PROGRAMME A LA CARTE

La façon dont le microcontrôleur gère ses entrées / sorties est fixée par un programme, contenu dans le microcontrôleur. Ce programme doit être écrit par l'utilisateur. En pratique, l'utilisateur écrit le programme en langage C, en utilisant un environnement de développement spécialisé (IDE) installé sur un ordinateur. Ce programme est ensuite compilé et téléversé dans le microcontrôleur par liaison série (USB).

Nous utiliserons l'IDE standard Arduino (arduino.exe). Il suffit de taper le code dans la fenêtre dédiée, de compiler et de téléverser le programme sur la carte arduino. La carte doit être reliée à l'ordinateur par un câble USB. La modèle de la carte arduino (il y a plusieurs type de carte) ainsi que le port série sur lequel elle est branchée doivent être déclarés dans le menu de l'IDE Outils/type de carte et Outils/port série.



Une fois téléversé dans le microcontrôleur, le programme s'exécute. La fonction setup() s'exécute une seule fois, la fonction loop() s'exécute en boucle.

De nombreux exemples de programmes sont disponibles via le menu Fichier/Exemples de l'IDE, classés par thèmes. N'hésitez pas à vous inspirer de ces exemples pour vos propres programmes. N'hésitez pas non plus à regarder sur internet les nombreux sites d'exemples et de conseils liés à arduino, mais essayez de bien comprendre la logique des exemples que vous utiliserez.

LES PRECAUTIONS (OU COMMENT NE PAS DETRUIRE VOTRE CARTE ARDUINO)

Attention : il y a quelques précautions à suivre pour ne pas endommager le matériel. Ne pas respecter ces consignes peut entraîner la perte de la carte, et potentiellement celle du port USB de l'ordinateur.

- ne jamais connecter une tension supérieure à 5 V sur les ports d'entrée de la carte ;
- ne jamais faire débiter à la carte plus de 40 mA par port, et 200 mA au total. Donc ne jamais envoyer une sortie de la carte vers un circuit de trop faible résistance.

Si vous devez relier une sortie à la masse (entre D13 et GND par exemple), il faut donc que votre circuit contienne une résistance d'au moins 125 Ω, ce qui correspond à un courant de 40 mA. Une résistance de 200 Ω est préférable – soit 25 mA – pour garder une marge de sécurité.

Quelques consignes importantes :

- Ne jamais relier directement une sortie digitale à la masse ;
- Ne jamais relier directement deux sorties digitales ;
- Bien vérifier la polarité d'une alimentation externe avant de l'allumer (ou via un multimètre avant de brancher) ;
- Ne pas envoyer de tension dans ports des sorties fixes (ceux intitulés 5V 3V3 GND VIN) ;Ne jamais envoyer plus de 5.5 V dans une entrée, analogique ou digitale ;
- Toujours vérifier les courants demandés par le circuit, et les comparer aux limites de la carte.
- Toujours mettre une résistance en série à une diode branchée à une sortie digitale pour ne pas cramer la diode.

Toutes les différentes tensions sont mesurées par rapport à GND, qui est relié à la masse quand la carte est connectée à un ordinateur par câble USB. « Ne pas dépasser une tension de 5.5 V sur un port » veut dire « ne pas dépasser une différence de tension de 5.5 V entre un port et le GND ».

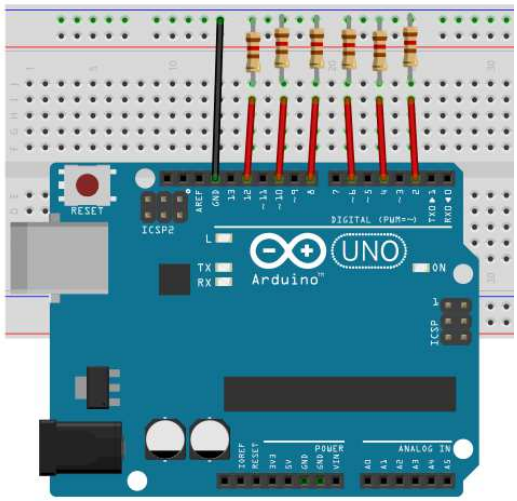
CE QU'IL NE FAUT PAS FAIRE N°1 : UN COURANT TROP FORT

La façon la plus simple de détruire une sortie numérique est de lui faire débiter un courant électrique trop élevé. Une résistance inférieure à 100 Ω entre le port numérique et la masse (port GND) demandera un courant supérieur à 50 mA si le port bascule en HIGH. Un fil, qui a une résistance quasi-nulle, fera un court-circuit et imposera un courant énorme.

NE FAITES JAMAIS CE MONTAGE !

CE QU'IL NE FAUT PAS FAIRE N°2 : UN COURANT TOTAL TROP FORT

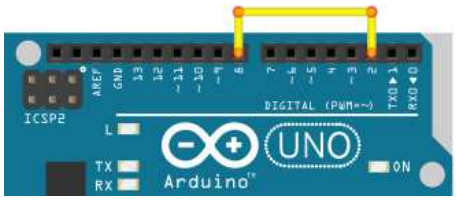
Une façon un peu plus compliquée de détruire les sorties numériques est de respecter la limite des 40 mA par ports, mais de dépasser la limite des 200 mA que peut débiter la carte Arduino. Si une résistance de 125 Ω relie un port numérique à la masse, 40 mA circulent quand le port est basculé en HIGH, ce qui n'abime pas le port (même si il vaut mieux ne pas trop s'approcher des 40 mA en pratique). Mais si six ports numériques basculent en même temps, le courant total délivré par la carte dépassera les 200 mA, et le microcontrôleur sera endommagé.



NE FAITES JAMAIS CE MONTAGE !

CE QU'IL NE FAUT PAS FAIRE N°3 : UN COURT-CIRCUIT

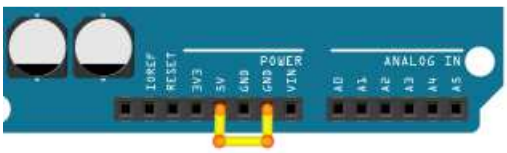
Si on relie deux ports numériques l'un à l'autre, tous deux définit en sortie, l'un étant basculé en HIGH, l'autre en LOW. Un court-circuit s'établit entre les deux ports, et la carte brule.



NE FAITES JAMAIS CE MONTAGE !

CE QU'IL NE FAUT PAS FAIRE N°4 : UN AUTRE COURT-CIRCUIT

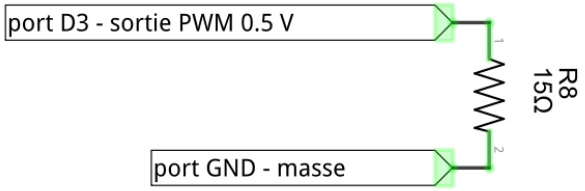
N'importe quel court-circuit est dangereux pour la carte : relier le port 5 V (ou le port 3.3 V) directement au port GND, ou relier le port 5 V au port 3.3 V.



NE FAITES JAMAIS CE MONTAGE !

CE QU'IL NE FAUT PAS FAIRE N°5 : UN COURANT TROP FORT MALGRE TOUT

Une sortie analogique peut être détruite de la même façon qu'une sortie numérique, si un courant plus élevé que 40 mA est demandé. La sortie D3 est utilisée pour produire une tension de valeur moyenne 0.5 V, il faut quand même une résistance de 125 Ω minimum entre ce port et le port GND : la tension moyenne est de 0.5 V, mais la tension réelle varie sans arrêt entre 0 et 5 V.

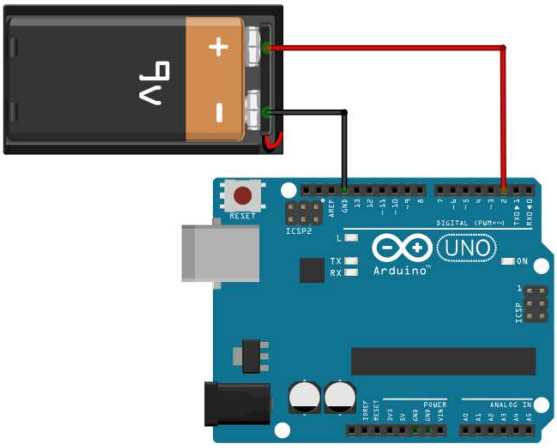


Une résistance de 15 Ω n'est pas suffisante pour limiter le courant en dessous de 40 mA, même si une tension de 0.5 V seulement est demandée.

NE FAITES JAMAIS CE MONTAGE !

CE QU'IL NE FAUT PAS FAIRE N°6 : UNE TROP FORTE TENSION

Il suffit de brancher une tension supérieure à 5 V sur un port de la carte (numérique ou analogique).



NE FAITES JAMAIS CE MONTAGE !

LE LANGAGE DE PROGRAMMATION

On trouvera toutes les instructions et des exemples très clairs sur :
http://arduino.cc/en/Reference/HomePage

STRUCTURE D'UN PROGRAMME

//Declaration des variables :

```
int led = 13; // fixe la variable led à 13
byte bytea ; // entier de 0 à 255
int integer1 ; // entier -32,768 à 32,767
unsigned int uInteger1 ; // 0 à 65,535
float mesure1 ; // reel
boolean flag=true; // booleen true ou false

int tableauInt[6] ; //tableau d'entiers a 6 entrées, numérotées de 0 à 5
```

// Initialisation des ports de la carte qui seront utilisés et initialisation du port série

```
void setup() {       // cette séquence est réalisée une seule fois, au début
 Serial.begin(9600);  // initialise le port série
 pinMode(3, OUTPUT);    // initialise la voie 3 comme sortie
 pinMode(4, INPUT);     // initialise la voie 4 comme entrée
  pinMode(led, INPUT);   // si led est une constante de valeur 13, initialise la
voie 13 comme entrée
} // fin de la procédure setup()
```

// programme principal : une boucle qui tourne à l'infini:

```
void loop() {
instructions du programme à mettre ici;
}
```

INSTRUCTIONS LES PLUS UTILES

Attention : la capitalisation des lettres est importante !

Imposer des tensions aux sorties de la carte (Ecrire) :
`digitalWrite(3, HIGH); //` impose la valeur HIGH au port 3 défini comme OUTPUT (autrement dit, envoie 5 V à la sortie du port D3). On peut mettre LOW et dans ce cas, c'est 0 V.

`analogWrite(6,100) ; //` impose la valeur 100 au port D6 défini comme OUTPUT (il s'agit d'une sortie analogique PWM). Cette valeur est définie entre 0 et 255 (255 vaut 5 V et 0 vaut 0 V donc 100 correspondra ici à une tension moyenne en sortie de D6 de 1,96 V).

Lire des tensions aux entrées de la carte :
`buttonState = digitalRead(4); //` lit la valeur HIGH ou LOW du port D4 défini comme INPUT (ici buttonState a été défini comme boolean)

`val = analogRead(5); //` lit la valeur analogique au port 5 défini comme INPUT (valeur entière comprise entre 0 et 1023)

affecter une valeur à un tableau
```
readings[3] = 1 ;
```

transformation d'un entier en réel ;
```
nombrereel = float(nombreentier) ;
```

exemples de calculs simples (faire attention aux types réel ou entier):
```
Vmes= average*Vref/1024;
temp = (Rmes/100-1)/3.85e-3;
```

attente en millisecondes:
```
delay(1);
```

mesure du temps écoulé depuis le lancement du programme en millisecondes:
`time = millis();` (définir time comme unsigned long)

envoi sur port série de chaine de caractère :
```
Serial.print("Mesure : ");
Serial.print(variable) ; // ça envoie la valeur de variable
Serial.println(" toto "); // Envoi d'une chaine avec retour à la ligne (car « println » et pas « print » indispensable en fin de ligne:
```

une boucle for (ici de 0 à 99) :
```
for (int index=0; index < 100; index++){
  instructions de la boucle à executer ;
}
```
autre exemple:
```
for(int x = 2; x < 100; x = x * 1.5){
  instructions de la boucle à executer ;
}
```

une boucle while
(ici qui s'execute tant que index est <3, et à chaque tour on ajoute 1)
```
while (index< 3) {
 index = index + 1 ;
  }
 do {
   instruction à faire;
   }
```

une condition if (ici par exemple sur la valeur d'un boolean buttonState:
```
  if (buttonState == HIGH) {
instruction à écrire ici si buttonState vaut HIGH ;
  }
  else {
instruction à écrire ici si buttonState vaut LOW ;
  }
```

LA SEANCE DE TRAVAIL : DECOUVRIR ARDUINO

Vous allez découvrir les capacités d'Arduino en l'utilisant. Voici une série de tâches à réaliser pour cela. Profitez-en pour vous familiariser avec la programmation et aux montages électroniques, cela vous sera utile pour les séances suivantes. N'hésitez pas à consulter internet, mais gardez un esprit critique sur les informations que vous y trouverez.

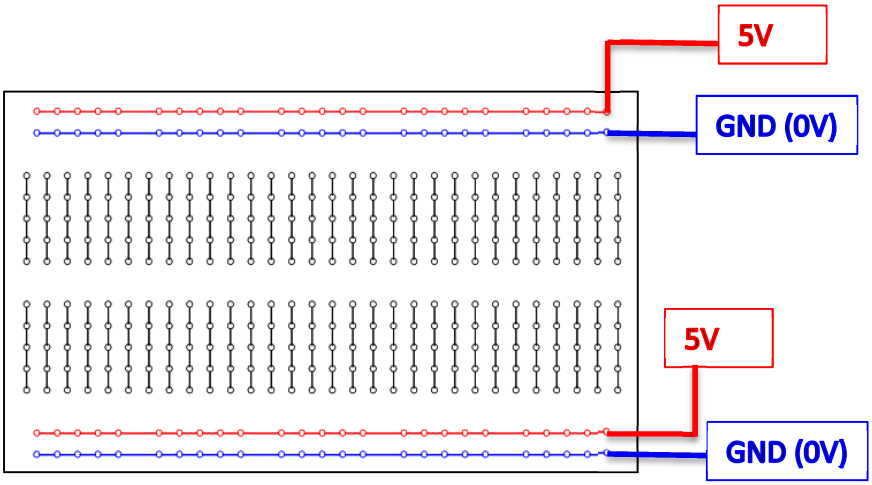
Enfin, pensez à prendre des notes au fur et à mesure, car vous ne vous souviendrez pas de tout à la prochaine séance…

AU TOUT DEBUT

- Se connecter sur sa session, lancer l'IDE Arduino.
- Spécifier le type de carte Arduino utilisée dans Outils\Type de carte (vérifier si votre carte est une UNO ou une MEGA)
- Brancher la carte Arduino via l'entrée USB la plus à droite sur face avant PC.
- Spécifier le port com utilisé en allant dans Outils\Port (un des ports est intitulé Arduino). Si un doute, demander à l'enseignant ou vérifier dans panneau de config / systeme / materiel / Gestionnaire de périph (dire ok quand il dit qu'on n'a pas les autorisations).

LA CONNECTIQUE : MICROCONTROLEUR ARDUINO ET BREADBOARD

Le breadboard : c'est une plateforme pour connecter facilement différents éléments. Voici à quoi il ressemble et les connections qui s'y cachent:



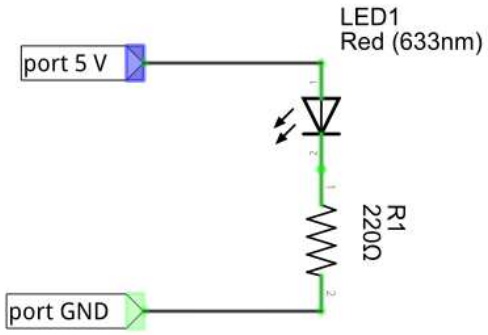
Par convention, on choisit en général de relier la ligne bleue à 0 V (la masse « GND ») et la ligne rouge à 5 V.

Pour s'entraîner à des montages électriques simples, avant même de programmer la carte Arduino, réalisez les petits exercices suivants :

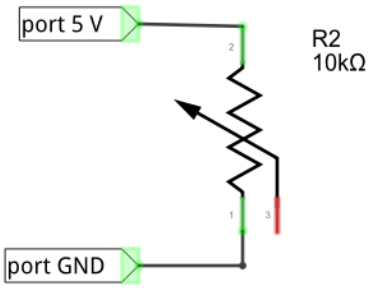
1) Vérifier avec le multimètre les connections du breadboard de la figure ci-dessus.

2) Utiliser la carte Arduino pour créer une tension de 5 V entre la zone rouge et la zone bleue du breadboard et mesurer cette tension avec un multimètre (Attention de ne pas court-circuiter la carte en reliant directement GND et 5 V). Faire pareil mais cette fois avec la tension 3.3 V de la carte.

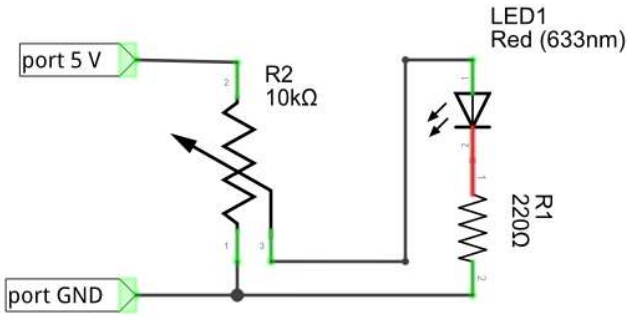
3) Construire un montage pour allumer une LED (n'oubliez pas d'insérer une résistance de 220 Ω qui limitera le courant à 20 mA).



4) Construire un montage pour alimenter un potentiomètre qui délivre une tension variable. Vérifier au multimètre.



5) Utiliser un potentiomètre pour allumer une LED progressivement.



PROGRAMMER LE MICROCONTROLEUR ARDUINO

Pour créer un programme sur le microcontrôleur, procéder ainsi à chaque fois :

- lancer le programme initOrsay avec le câble USB branché (ça va initialiser proprement toutes les voies en entrée et effacer les effets du programme précédent) ;
- débrancher le câble USB, réaliser votre montage électrique et votre programme, le faire valider par l'enseignant si vous avez un doute ;
- rebrancher le câble USB puis téléverser le programme arduino ;
- si ça marche, dessinez sur vos cahiers le circuit électrique ou prenez une photo pour garder la mémoire du circuit.

1) **Un premier test** (programmation de base)

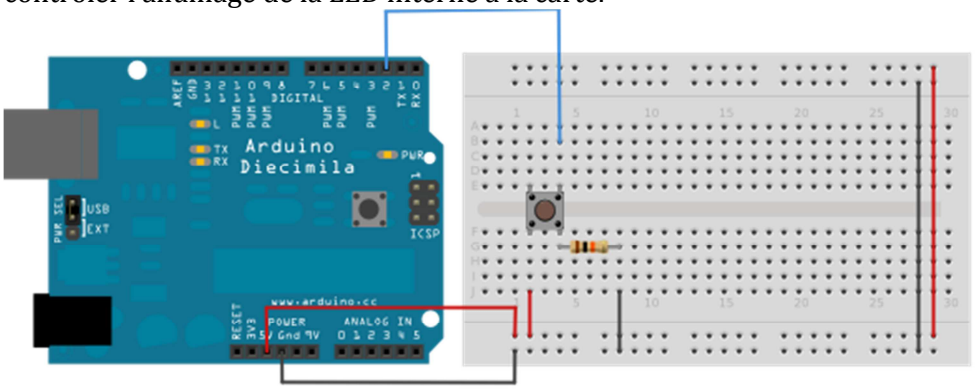
Lancer le programme Fichier / Exemples / Basics / Blink et vérifier que la LED interne de la carte arduino clignote bien. Modifier le délai de clignotement dans le programme et vérifier.

2) **Allumer une LED extérieure** (sortie numérique)

À partir du même programme Blink, faites cette fois clignoter une vraie LED via le breadboard.

3) **Utiliser un interrupteur** (entrée numérique)

Réaliser le circuit correspondant au programme Button pour utiliser l'interrupteur et contrôler l'allumage de la LED interne à la carte.



4) Utiliser un interrupteur pour contrôler une LED extérieure

Modifier le montage précédent pour cette fois contrôler l'allumage d'une LED extérieure avec l'interrupteur.

5) Lire une tension constante (entrée analogique / écriture et lecture du port série)

Utiliser la fonction AnalogRead pour lire une tension (utiliser la tension 3.3 V de la carte par exemple). Pour cela, s'inspirer de la syntaxe du programme Fichier / Exemples / Analog / AnalogInOutSerial. Envoyer cette valeur lue sur le port série pour l'afficher à l'écran et vérifier la valeur lue via Outils\Moniteur serie. Afficher également la valeur en volt.

6) Lire une tension pilotée avec un potentiomètre

Modifier le circuit électrique précédent pour cette fois avoir une tension réglable par un potentiomètre et vérifier là encore la tension lue à l'écran.

7) **Créer une tension PWM** (sortie analogique PWM)

Utiliser une sortie PWM pour créer une tension de 1 V (s'inspirer de l'exemple AnalogWriteMega). La lire à la fois avec un multimètre et avec une entrée analogique de la carte. Comparer les résultats.

8) Piloter une LED avec une tension PWM

Utiliser une sortie PWM pour créer une tension qui pilote l'éclairage d'une LED extérieure.

RECUPERER ET TRACER LES DONNEES AVEC LABVIEW

Utilisez le programme labview déjà fourni « Communication Arduino Vers Labview.vi » pour lire les datas lors d'une des expériences Arduino précédentes.

Attention, il ne faut pas lancer le programme labview si le moniteur Série est en marche ou si vous téléversez un programme dans la carte Arduino (car les deux programmes utilisent le même port série).

Pensez aussi à écrire dans le programme Arduino vos datas sous la forme d'un tableau composé de nombres séparés par des « ; » et avec un retour à la ligne en fin de chaque ligne, par exemple :

4 ; 7
6 ; 3.2
7 ; 2
…

Vérifier que vous arrivez à lire les données et à les sauver avec labview puis à relire ces données avec un autre programme (Kaleidagraph ou Exel).

E. QUE MESURE-T-ON VRAIMENT AVEC LE MICROCONTROLEUR ?

En faisant des mesures appropriées de tension, testez

- la digitalisation de la carte ;
- le bruit de la carte ;
- le temps d'acquisition minimum ;
- l'intérêt ou non de moyenner une mesure.

Facultatif : faites un programme qui mesure un thermomètre Pt100 en fonction du temps, et étudier la précision et la sensibilité de votre système.