

Broches numériques en entrée : utiliser les capteurs numériques ON/OFF.



Ateliers Arduino

par X. HINAULT

www.mon-club-elec.fr



Tous droits réservés – 2012.

Ce document légèrement payant est soumis au droit d'auteur et est réservé à l'usage personnel.

Afin d'encourager la production de supports didactiques de qualité, ce document est légèrement payant.

La licence d'utilisation est attribuée pour un usage personnel uniquement, dans le cercle familial. Mise en ligne et diffusion non autorisées.

Si vous n'êtes pas le détenteur de la licence attribuée pour l'usage de ce document, soyez sympa, merci d'acheter votre exemplaire personnel ici : <https://monclubelec.dpdcart.com/>

Pour tout problème lié à l'utilisation de ce document, veuillez envoyer une copie ici : support@mon-club-elec.fr

Pour obtenir tout autres types de licence d'utilisation (enseignement, commercial, etc...), veuillez contacter l'auteur ici : support@mon-club-elec.fr

Vous avez constaté une erreur ? une coquille ? N'hésitez pas à nous le signaler à cette adresse : support@mon-club-elec.fr

Truc d'utilisation : visualiser ce document en mode diaporama dans le visionneur PDF. Navigation avec les flèches HAUT / BAS ou la souris.

En mode fenêtre, activer le panneau latéral vous facilitera la navigation dans le document. Bonne lecture !

Lancer également le logiciel Arduino et connecter votre carte Arduino afin de pouvoir tester au fur et à mesure les codes d'exemples !

1. Intro

L'objectif ici est :

- de rappeler les principes d'utilisation d'une broche numérique en entrée
- de découvrir le principe d'utilisation d'un capteur analogique (ON/OFF)
- de montrer le principe de comptage d'un événement
- de montrer le principe de comptage d'une fréquence de survenue d'un événement

... afin d'être capable d'écrire des programmes permettant d'interagir avec Arduino à partir de capteurs numériques ON/OFF.

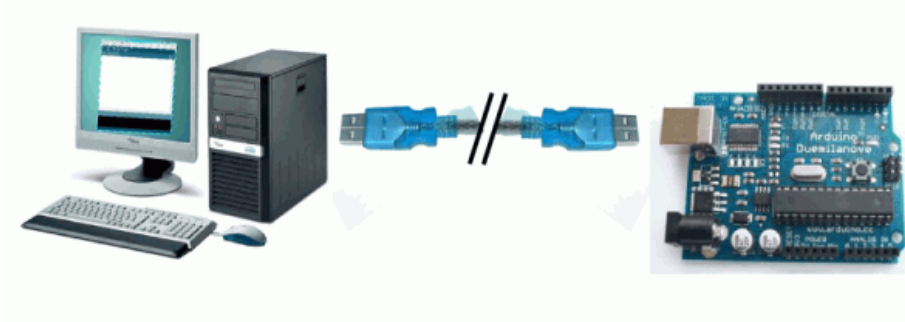


Prêt ? C'est parti !

2. Matériel nécessaire pour les ateliers Arduino

Pour cet atelier, vous aurez besoin de tout ou partie des éléments suivants pour pouvoir réaliser les exemples proposés :

De l'espace de développement Arduino

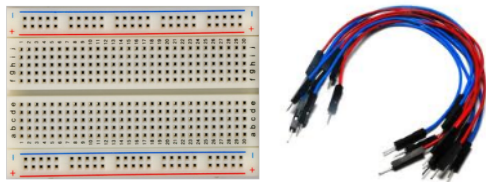


L'espace de développement Arduino associe :

- un ordinateur sous Windows, Mac Os X ou Gnu/Linux (Ubuntu)
- avec le logiciel Arduino installé (voir : <http://www.arduino.cc/>)
- un câble USB
- une carte Arduino UNO ou équivalente.

disponible chez : <http://shop.snootlab.com/> ou <http://www.gotronic.fr/>

Du nécessaire pour réaliser des montages sans soudure

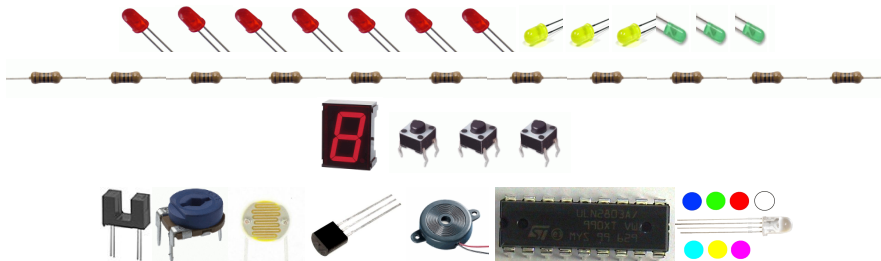


Pour réaliser des montages sans soudure, vous aurez besoin :

- d'une plaque d'essai ou breadboard moyenne (450 points)
- de quelques câbles souples (ou jumpers) mâle/mâle

disponible chez : <http://www.gotronic.fr/>

De quelques composants de base



Pour vous simplifier la vie, nous avons négocié ce kit pour vous !

Vous pouvez commander ce kit complet directement en 1 clic chez notre partenaire

<http://www.gotronic.fr/> avec le code express **701710**

GO TRONIC
ROBOTIQUE ET COMPOSANTS ÉLECTRONIQUES

Pour plus de détails, voir : http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS

Pour les ateliers Arduino niveau débutant, vous devrez idéalement disposer des composants suivants :

- des LEDs 5mm Rouges(x20), Vertes (x5) et 3 Jaunes (x5)
- digit à cathode commune rouge 13mm (x1)
- Résistances (1/4w - 5%) de 270 Ohms (x20), 4,7K Ohms (x1), 1K Ohms (x1)
- mini bouton-poussoir (x3)
- Opto-fourche (x 1)
- Résistance variable linéaire 10K (x 1)
- Photo-résistance 7mm (x 1)
- Capteur de température LM35DZ (-55/+150°C - 10mV/°C) (x 1)
- Capsule son piézoélectrique (x 1)
- ULN 2803A (CI amplificateur 8 voies, 500mA/ voie) (x 1)
- LED 5mm multicolore RVB cathode commune (x 1)

3. Rappel : Une broche numérique ne peut avoir que 2 états : HAUT ou BAS, « y'a ou y'a pas » !

Une broche numérique, dans un circuit numérique est un point du circuit matérialisé par une broche métallique dans le cas d'un circuit intégré ou d'une carte électronique.

Une broche numérique va être caractérisée par son **état** ou niveau de tension : elle va pouvoir se trouver dans **2 états possibles seulement** :

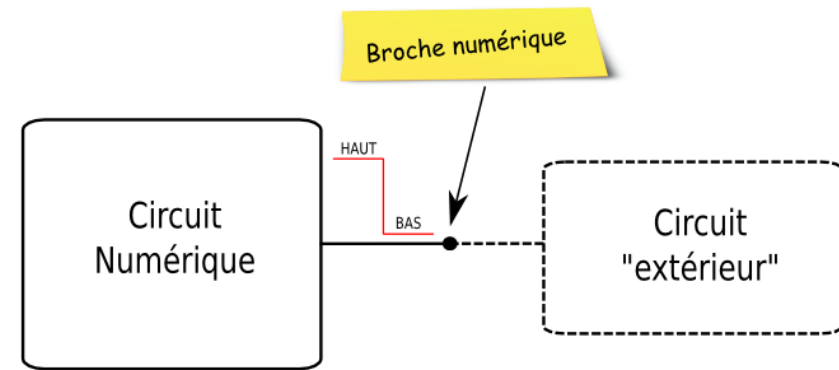
- soit au niveau **HAUT** (=5V), symbolisé par **1** ou **HIGH**
- soit au niveau **BAS** (=0V), symbolisé par **0** ou **LOW**
- noter qu'à **un instant quelconque, la broche se trouve obligatoirement dans l'un de ces 2 états.**

Pour reprendre l'image d'une vanne « tout ou rien » :

- soit il y a de l'eau qui circule
- soit il n'y en n'a pas

Pour reprendre l'image d'un « interrupteur » :

- soit il y a de la lumière,
- soit il n'y en n'a pas...



4. Rappel : Une broche numérique est caractérisée par son SENS : en SORTIE ou en ENTREE !

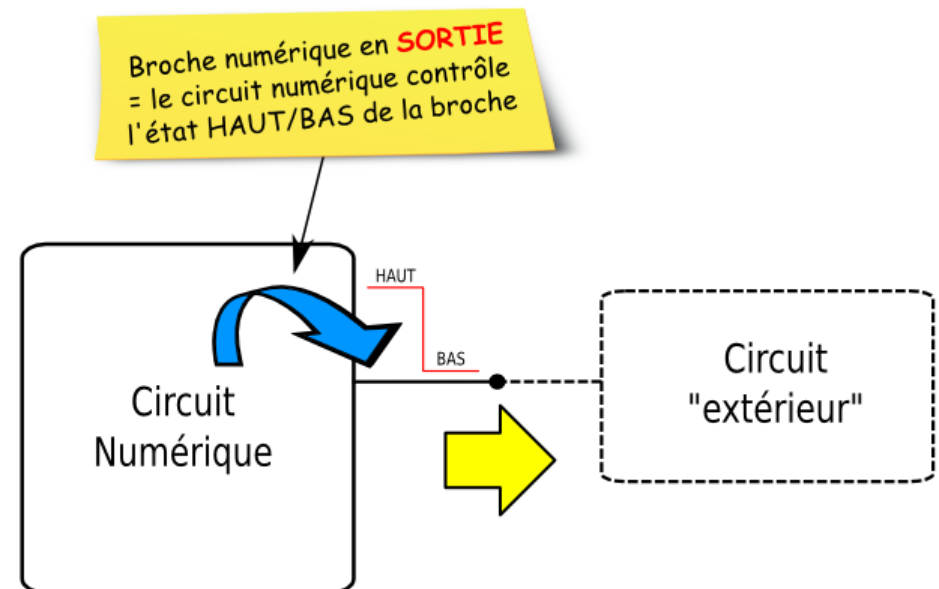
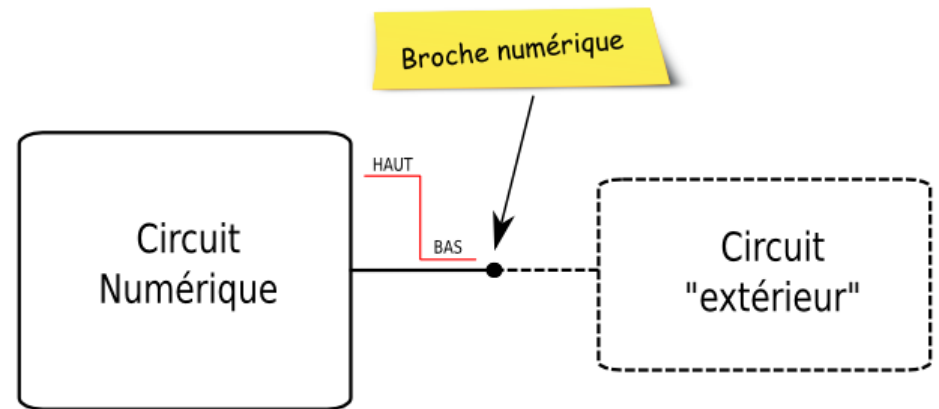
Une broche numérique est également caractérisée par son **sens** :

- la broche est dite en **sortie** d'un circuit numérique lorsque c'est **le circuit numérique qui contrôle l'état Haut/Bas de la broche**. On pourra symboliser le sens de la broche numérique en sortie par une flèche sortante.
- la broche est dite en **entrée** d'un circuit numérique lorsque le circuit numérique « reçoit » (ou « subit ») son état. **L'état Haut/Bas de la broche numérique est contrôlé par « l'extérieur »**. On pourra symboliser le sens de la broche numérique en entrée par une flèche entrante.
- noter qu'une broche numérique qui est en sortie d'un circuit numérique est en entrée du circuit extérieur auquel elle est connectée... et inversement... !

Usage avancé : en interne, dans un microprocesseur, une broche numérique est associée :

- à un bit de donnée (case unitaire mémoire) qui va permettre de fixer/lire son état
- à un bit de sens qui va définir son mode de fonctionnement

Techniquement, il existe par ailleurs plusieurs technologies de broches numériques (TTL, CMOS,...) qui ont des définitions différentes des niveaux de tension HAUT et BAS. Seules des broches compatibles entre-elles pourront être utilisées/connectées ensemble. Arduino est très souple de ce point de vue et est compatible avec la plupart des technologies E/S !



5. Rappel : Les broches numériques de la carte Arduino

La carte Arduino de base (la UNO, la Duemilanove, etc..) est une carte numérique qui possède **20 broches d'Entrée/Sortie numérique** (notées E/S) numérotées **de 0 à 19** !

Les broches 0 à 19 sont potentiellement utilisables en broches E/S !

Cependant, certaines broches ne doivent pas, dans la mesure du possible être utilisées en broches E/S :

- les **broches 0 et 1** sont utilisées par la communication USB donc, les utiliser pourrait perturber cette communication. En pratique, ne pas les utiliser.
- les **broches 14 à 19** ont un double rôle : elles peuvent également être utilisées en tant que broches analogiques pour réaliser des mesures. Donc, si possible, ne pas les utiliser en broches numériques... mais si on est obligé, on peut le faire !

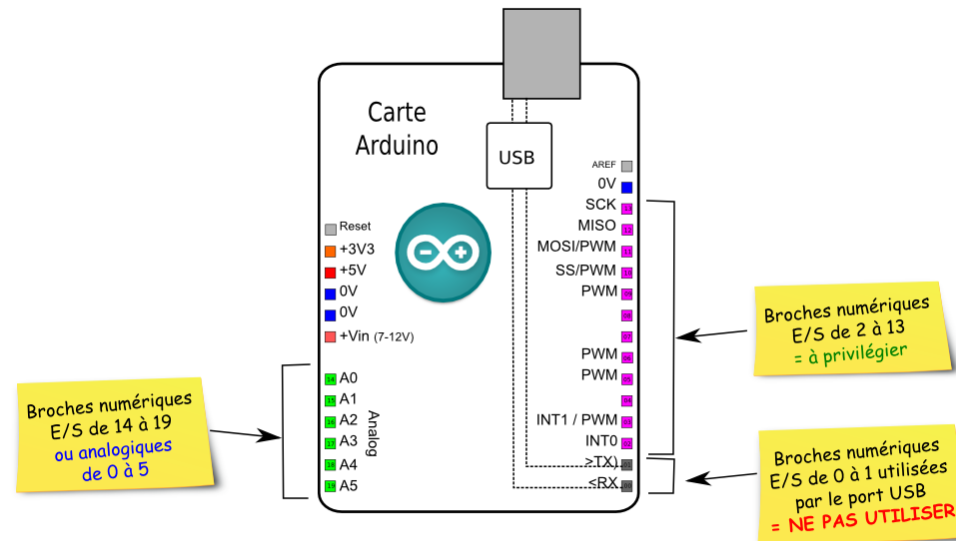
A savoir : les broches numériques 14 à 19 sont numérotées de 0 à 5 (ou désignées par A0, A1, A2, A3 et A5) lorsqu'elles sont utilisées en tant que broches analogiques.

Remarquer également que la plupart des broches numériques ont des fonctions particulières potentielles qui seront présentées au fur et à mesure de leur utilisation. *A titre indicatif, les fonctions disponibles sont la génération d'impulsion, la communication SPI, la communication I2C, les interruptions externes...*

D'un point de vue électrique, retenir que :

- chaque broche numérique E/S peut supporter 40 mA d'intensité en sortie ou en entrée
- L'ensemble des broches numériques E/S ne doit pas dépasser 200mA en entrée ou en sortie !

Usage avancé : pour des projets nécessitant de nombreuses broches E/S (= mal conçu?), la carte Arduino Mega dispose de plus d'une 50aine de broches E/S !



6. Rappel: Les instructions du langage Arduino pour la gestion des broches numériques

On a donc vu qu'une broche numérique E/S est caractérisée par :

- son sens : **ENTREE** ou **SORTIE**
- son état : HAUT ou BAS

Fixer le sens E/S d'une broche numérique

La première instruction à connaître est l'instruction `pinMode(broche, mode)` qui sert à fixer le sens (ou mode de fonctionnement) d'une broche numérique E/S avec :

- broche : le numéro de la broche de 0 à 19
- mode : une des constantes prédéfinies suivantes :
 - **OUTPUT** : pour un fonctionnement en **sortie**
 - **INPUT** : pour un fonctionnement en **entrée**
- Cette fonction est à utiliser dans la fonction `setup()`

Fixer le niveau HAUT/BAS d'une broche numérique

Si la broche est configurée en **SORTIE**, on pourra contrôler son état (ou « écrire » sur la broche) à l'aide de la fonction `digitalWrite(broche, etat)` avec :

- broche : le numéro de broche de 0 à 19
- valeur : une des constantes prédéfinies suivantes :
 - **HIGH** : pour mettre la broche au niveau HAUT (5V)
 - **LOW** : pour mettre la broche au niveau BAS (0V)

Si la broche est configurée en **ENTREE**, on pourra « lire » son état à l'aide de la fonction `int digitalWrite(broche)` avec :

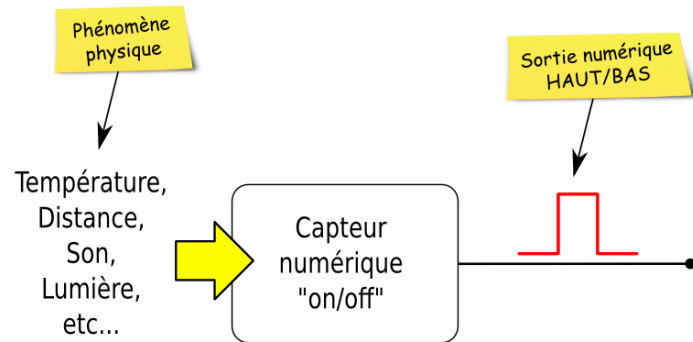
- broche : le numéro de broche de 0 à 19
- int : la valeur renvoyée par la fonction de type int

	ETAT HAUT	ETAT BAS
BROCHE EN SORTIE <code>pinMode(broche, OUTPUT);</code>	<code>digitalWrite(broche, HIGH);</code>	<code>digitalWrite(broche, LOW);</code>
BROCHE EN ENTREE <code>pinMode(broche, INPUT);</code>	<code>digitalRead(broche);</code>	<code>digitalRead(broche);</code>

7. Introduction aux capteurs numériques ON/OFF

Principe de fonctionnement

Un capteur numérique ON/OFF (ou « tout ou rien ») est un dispositif qui détecte l'existence ou non d'un événement ou phénomène physique, renvoie une information sous une forme « OUI/NON » et donc sous la forme d'un niveau HAUT ou BAS sur sa broche de sortie.

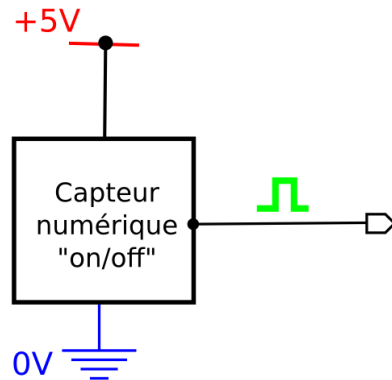


Par exemple, la broche de sortie d'un détecteur de ligne numérique « ON/OFF » sera à HAUT si une ligne est détectée, à BAS si aucune ligne n'est détectée, ou l'inverse.

Brochage type

Le brochage type d'un capteur numérique (ou « ON/OFF » ou « tout ou rien ») utilisable avec Arduino est assez simple (et logique !) :

- deux broches d'alimentation de 0V et de +5V
- une broche de sortie numérique ON/OFF

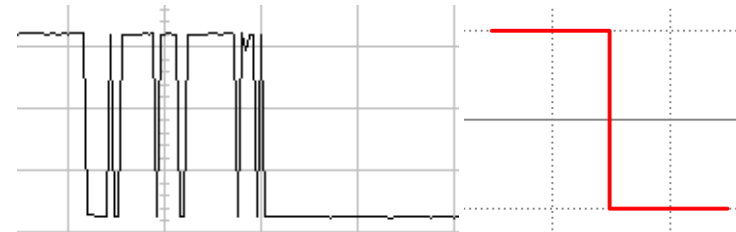


Comparatif bouton poussoir et capteur numérique ON/OFF

Un bouton poussoir est un capteur ON/OFF mais « mécanique », ce qui entraîne un certain nombre de problématiques spécifiques qui ont été abordées dans l'atelier consacré aux boutons poussoirs :

- nécessité d'un « rappel au plus » (ou « au moins ») de la broche numérique laissée non connectée,
- nécessité d'une pause « anti-rebond » lors de la lecture de l'état du bouton poussoir.

Un capteur numérique ON/OFF ne présente pas ces problèmes et a une transition HAUT/BAS nette et franche :



A gauche : bouton poussoir, à droite : capteur numérique

Comparatif capteur numérique ON/OFF et capteur analogique


Un capteur numérique ON/OFF se distingue d'un capteur analogique (voir l'atelier consacré aux entrées analogiques) à plusieurs niveaux :

- le capteur analogique est plus précis et donne une information fine (jusqu'à 1023 niveaux) alors que le capteur numérique ON/OFF fournit une information simple et binaire : « OUI » ou « NON »
- un capteur numérique ON/OFF est plus simple à gérer et plus rapide à tester par la simple lecture de l'état d'une broche numérique.
- Un capteur numérique ON/OFF peut également être utilisé pour déclencher des interruptions (voir l'atelier consacré aux interruptions)

Remarque : On peut facilement utiliser et transformer un capteur analogique comme un capteur numérique en fixant un seuil au niveau du programme. « Qui peut le plus peut le moins... »

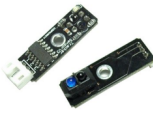
8. Quelques exemples de capteurs numériques ON/OFF

Opto-coupleur en fourche

	Ce capteur livré brut est facile à utiliser avec 2 résistances complémentaires. Un opto-coupleur en fourche associe face à face une photo-diode et un photo-transistor : un objet passant dans la fente coupe le faisceau lumineux et sera détecté. Utilisable à grande vitesse (détection d'objet en rotation). Existe aussi en version large. Intéressant pour des comptages de vitesse de rotation (anémomètre, moteurs, etc...)
---	--

Dispo ici : <http://www.gotronic.fr/art-interrupteur-optique-lth301-07-2325.htm>

Capteur de ligne DFRobot

	Ce capteur livré monté et prêt à connecter, livré avec son câble . Il associe une photo-diode et un photo-transistor qui permettent la détection d'une ligne. Sensible, efficace et facile à utiliser. Idéalement par 3 pour optimiser le centrage sur la ligne avec un robot mobile. Niveau logique bas pour le NOIR, niveau logique haut pour le BLANC.
---	--

Dispo ici : <http://www.zartronic.fr/module-suiveur-de-ligne-pour-arduino-p-129.html>

Capteur détecteur de mouvement à infra-rouge

	Ce capteur livré monté et prêt à connecter permet de détecter le mouvement d'un être vivant dans son « champ de vision ». Nécessite de s'acclimater à la pièce au préalable. Assez sensible. Alim : 5-12V. Conseillé de l'alimenter en 12V.
---	--

Dispo ici : <http://www.watterott.com/en/PIR-Motion-Sensor>

Détecteur numérique de vibration

	Ce capteur livré monté et prêt à connecter est sensé détecter les vibrations... Utiliser une capsule piézo-électrique simple plutôt...
---	---

Dispo ici : <http://www.zartronic.fr/capteur-num%C3%A9rique-de-vibrations-p-106.html>

Détecteur numérique d'inclinaison (Tilt sensor)

	Ce capteur livré monté et prêt à connecter est sensé détecter l'inclinaison... capricieux... Utiliser plutôt un accéléromètre analogique...
---	--

Dispo ici : <http://www.zartronic.fr/capteur-dinclinaison-num%C3%A9rique-tilt-compatible-arduino-p-105.html>

Détecteur numérique capacitif de contact

	Ce capteur livré monté et prêt à connecter
---	---

Dispo ici : <http://www.alpha-crucis.com/fr/capteurs/3449-capacitive-touch-sensor-3700386600302.html>

Purchased by Franck Ourion, franck.ourion@univ-lorraine.fr #6280170

Atelier Arduino : Broches numériques en entrée : utiliser les capteurs numériques ON/OFF.

9. Utiliser un capteur numérique ON/OFF en pratique

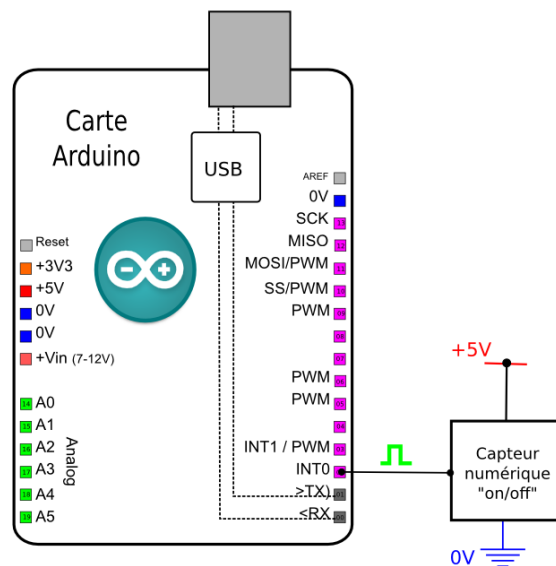
Utiliser un capteur numérique ON/OFF en pratique

Comme on l'a vu, le brochage type d'un capteur numérique (ou « ON/OFF » ou « tout ou rien ») est assez simple et logique :

- deux broches d'alimentation de 0V et de +5V
- une broche de sortie numérique ON/OFF

En pratique, on peut utiliser n'importe quel capteur numérique très simplement en connectant :

- les deux broches d'alimentation de 0V et de +5V au 0V et au +5V de la carte Arduino,
- la broche de sortie numérique ON/OFF sur une broche numérique de la carte Arduino qui sera **configurée en entrée**.

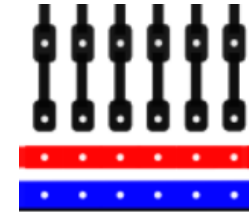


Bon à savoir

Dans les faits, plusieurs fabricants proposent leur capteurs numérique avec une connectique à 3 broches différentes de celle des capteurs analogiques, avec une simple inversion de l'ordre des broches, probablement pour des raisons marketing... Faire attention au brochage donc...

Utilisation de plusieurs capteurs numériques

Si on y réfléchit bien, chaque capteur a besoin d'une broche de +5V, d'une broche de 0V et d'une broche numérique. D'où l'idée de mettre le bus d'alimentation en regard des broches numériques pour pouvoir facilement connecter un capteur numérique sur un connecteur droit à 3 broches.



Il existe pour cela des shields tout prêts. On pourra aussi faire cela avec un shield de prototypage. On encore, créer soi-même son « mini-shield » permettant de réaliser cela.

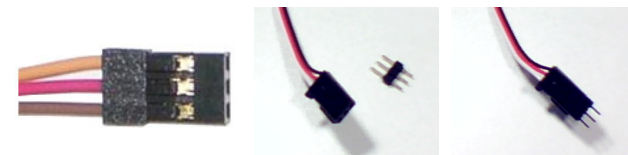


Exemple d'un mini-shield à connecteurs droits 0V / +5V / Broche
réalisé par www.mon-club-elec.fr

Connectique utile

Pour utiliser facilement un capteur numérique, on utilisera un câble 3 brins :

- soit celui fournit par le fabricant avec le capteur
- soit tout simplement un cordon dit « de servomoteur » qui dispose d'un connecteur à 3 broches femelles, dit « JR mâle » (facile à utiliser soit avec des jumpers mâle/mâle ou encore un connecteur droit 3 contacts)



10. Fiche composant : découvrir le transistor et le photo-transistor

Description

En électronique, le transistor est un composant semi-conducteur (il en existe 2 types dits PNP ou NPN) dans un petit boîtier qui dispose de 3 broches :

- la base B qui reçoit une intensité de déclenchement I_b
- le collecteur C qui laisse entrer une intensité I_c proportionnelle à I_b
- l'émetteur E qui laisse sortir une intensité valant $I_e = I_c + I_b$



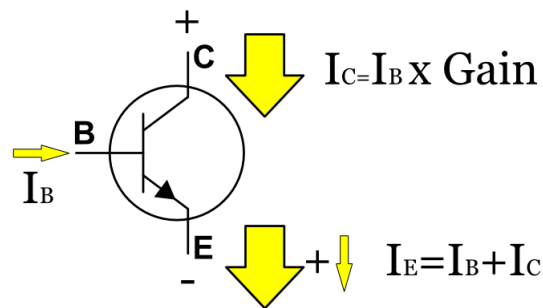
A savoir :

Le transistor est un composant essentiel, qui date des années 40, et qui a révolutionné l'électronique et permis l'apparition de l'électronique numérique et des ordinateurs. Les processeurs des ordinateurs actuels possèdent des millions de transistors miniaturisés !!

Le micro-contrôleur de votre carte Arduino lui-même intègre environ 500 000 transistors !

Principe de fonctionnement

Le principe fondamental de fonctionnement d'un transistor est le suivant : une petite intensité circulant sur la broche de la base va provoquer la circulation d'une intensité importante proportionnelle entre le collecteur et l'émetteur.



Le Transistor NPN

Pour faire simple, on peut dire qu'un transistor est un « multiplicateur » d'intensité : il multiplie l'intensité de la base et l'intensité résultante circule entre le collecteur et l'émetteur. Le coefficient multiplicateur est appelé **gain**.

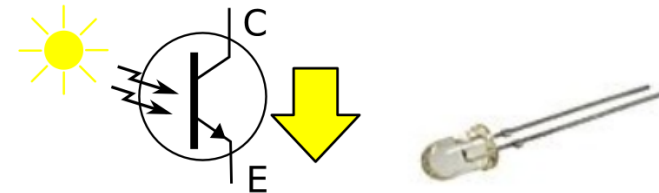
Modes de fonctionnement d'un transistor

Le transistor est un composant qui peut être utilisé aussi bien en mode analogique que « numérique » :

- **en mode analogique**, la variation d'intensité sur la base se répercute immédiatement en variation d'intensité du collecteur. C'est ce principe qui est à la base des amplificateurs audio et autres appareils de radio (dont lui vient d'ailleurs le nom de transistor).
- **en mode « numérique » ou « ON/OFF » appelé également mode saturé** : dès qu'une intensité est présente sur la base, le courant de collecteur est d'emblée maximal. L'absence de courant sur la base ne laisse passer aucun courant de collecteur. C'est une sorte d'interrupteur à commande électrique. C'est ce mode de fonctionnement qui est à la base de tous les circuits logiques et numériques.

Une variante du transistor : le photo-transistor

Dans ce composant, la broche de la base est remplacée par une zone sensible à la lumière infra-rouge. Le photo-transistor n'a donc que 2 broches :



Le principe de fonctionnement est le suivant : une intensité lumineuse présente sur la zone photo-sensible va provoquer la circulation d'un courant de collecteur qui sera proportionnel à l'intensité lumineuse reçue.

Le photo-transistor pourra être utilisé soit en mode analogique ou saturé.

Les transistors avec Arduino en pratique

Afin de ne pas compliquer inutilement les montages, en pratique, on n'utilisera quasiment pas les transistors « bruts » avec Arduino, mais plutôt des circuits les utilisant tels que le circuit intégré ULN 2803 qui intègre 8 étages d'amplification ON/OFF et ne nécessite aucun composant externe.

Par contre, on utilisera le photo-transistor, utilisé au sein des opto-coupleurs, comme nous allons le voir par la suite.

Remarque : l'étude des transistors et de leur utilisation est un domaine passionnant et qui peut faire l'objet de livres entiers. Ici, nous en parlons uniquement pour introduire le photo-transistor. Si vous voulez approfondir, voir notamment : <http://fr.wikipedia.org/wiki/Transistor>

11. Fiche composant : découvrir l'opto-coupleur en fourche

Description

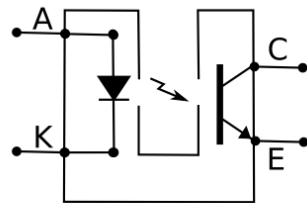
L'opto-coupleur en fourche est un composant qui associe en fait 2 composants différents qui sont positionnés face à face dans un même boîtier (2 broches par composant soit 4 broches en tout) :

- d'une part une photo-diode ou LED infra-rouge qui fonctionne comme une LED classique et émet une lumière invisible dite infra-rouge,
- d'autre part un photo-transistor infra-rouge utilisé ici pour détecter la présence de la lumière infra-rouge (patte courte = Emetteur).



Schéma interne

Le schéma théorique de l'optocoupleur est le suivant :



Opto-coupleur fourche
(Ex: LTH301-7)

- On retrouve d'une part la **LED infra-rouge signalée par les lettres A et K** sur le boîtier de l'opto-coupleur correspondant à l'anode (A = +) et la cathode (K = - = patte courte)
- On retrouve d'autre part le **photo-transistor signalé par les lettres C et E** sur le boîtier de l'opto-coupleur correspondant au collecteur (C = +) et à l'émetteur (E = - = patte courte).

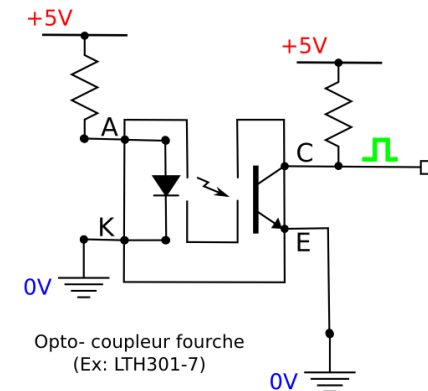
Principe de fonctionnement

- Lorsque la LED infra-rouge est allumée, la base du photo-transistor est éclairée et le photo-transistor laisse passer le courant.
- Lorsque la LED infra-rouge est éteinte, ou si un objet se trouve dans la fente, la base du photo-transistor ne laisse passer aucun courant.

Le montage type

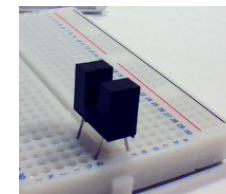
L'utilisation de ce composant nécessite en fait la réalisation de 2 circuits :

- tout d'abord, le circuit de la LED infra-rouge, qui s'utilise comme une LED standard. On pourra donc se contenter de mettre une résistance en série avec LED pour qu'elle soit allumée. **Comme vu précédemment, si on désire une intensité de 13mA dans la LED, on utilisera, d'après la loi d'ohm, une résistance de $R = U/I = 3,5V/0,013A = 270 \text{ Ohms}$.**
- le circuit du photo-transistor qui sera ici utilisé en mode saturé, autrement dit :
 - si pas d'objet dans la fente = lumière IR présente, alors la tension du collecteur vaudra 0V
 - si objet présent dans la fente = pas de lumière IR, alors la tension du collecteur vaudra 5V
 - pour obtenir ce résultat, on se contente d'utiliser une résistance de quelques milliers d'Ohms entre le collecteur et le +5V. **En pratique, on utilisera 4,7KOhms avec un LTH301-7.**

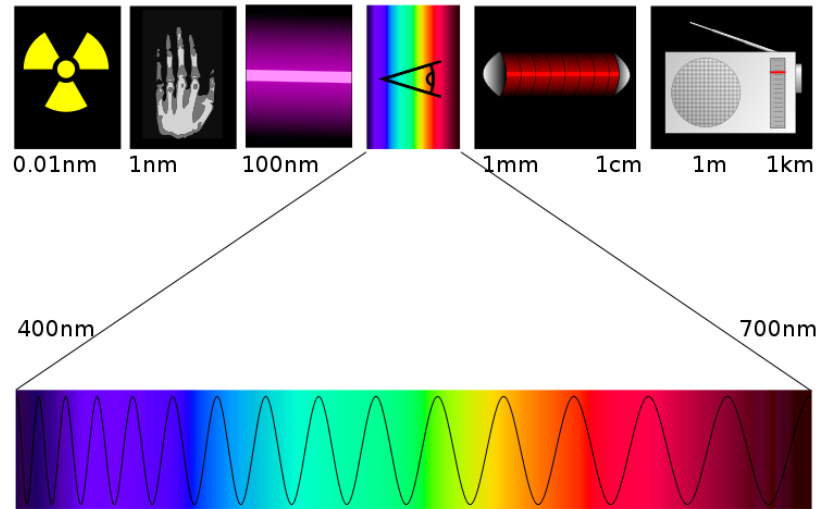


Principe d'utilisation sur une plaque d'essai

L'opto-coupleur s'utilise simplement, à cheval sur le rail central :

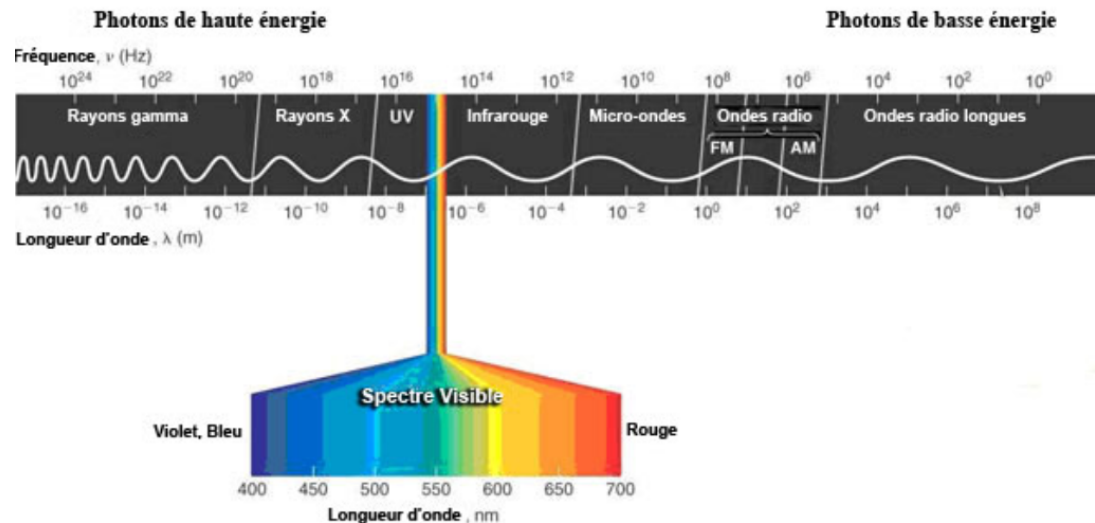


12. Pour info : le spectre des ondes électro-magnétiques et de la lumière visible



<http://fr.wikipedia.org/wiki/Fichier:Spectre.svg>

La lumière, tout comme les ondes radio ou les micro-ondes sont des ondes dites « électro-magnétiques »



source : <http://www.lampexpress.fr/images/ampoules-fiche-technique/spectre-lumiere.jpg>

La lumière visible ne représente qu'une toute petite partie de l'ensemble des ondes électro-magnétiques.

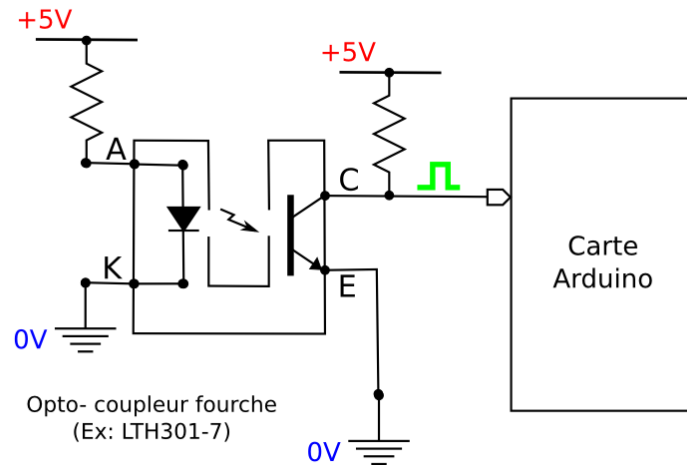
La lumière infra-rouge, à laquelle est sensible le photo-transistor, est une lumière invisible à l'oeil nu, de même que la lumière ultra-violette.

Purchased by Franck Ourion, franck.ourion@univ-lorraine.fr #6280170

Atelier Arduino : Broches numériques en entrée : utiliser les capteurs numériques ON/OFF.

13. Utiliser un opto-coupleur en fourche en tant que capteur numérique : le montage

On reprend ici simplement le montage type de l'opto-coupleur vu précédemment. On connecte l'émetteur sur une broche analogique de la carte Arduino :



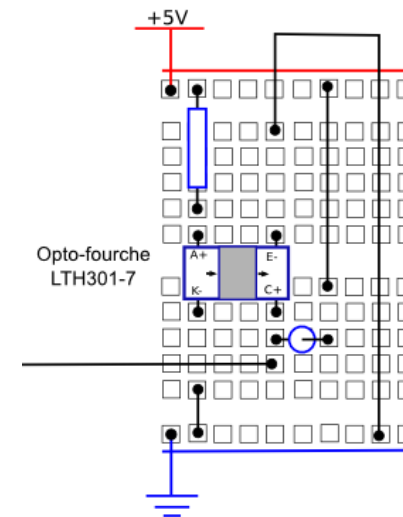
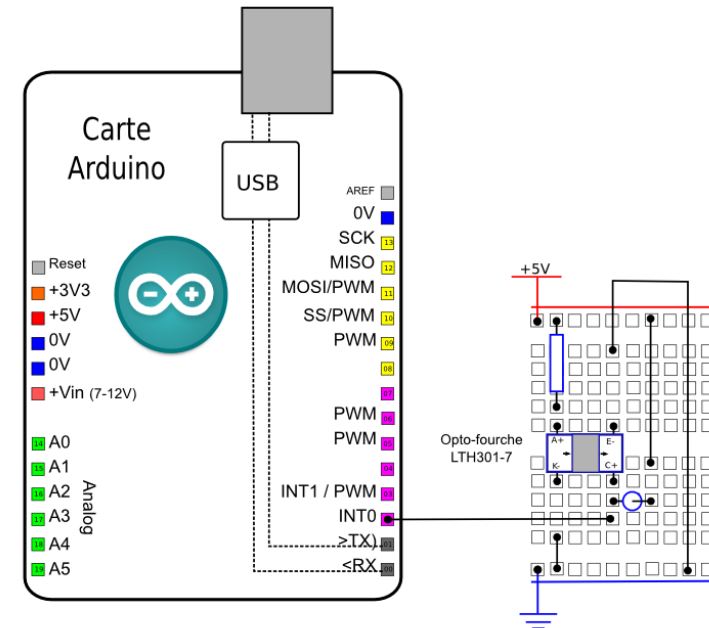
Comprendre comment ça marche

- Lorsqu'un objet est présent dans la fente, aucune lumière n'est détectée par le photo-transistor et donc aucune intensité ne circule dans le collecteur. La tension de la résistance en série vaut donc $U = R \times I = 0 \text{ V}$
- Lorsqu'aucun objet n'est présent dans la fente, la lumière est détectée par le photo-transistor et donc le courant circule dans le collecteur. La tension de la résistance en série vaut donc $U = R \times I \sim 5 \text{ V}$

Truc de repérage :

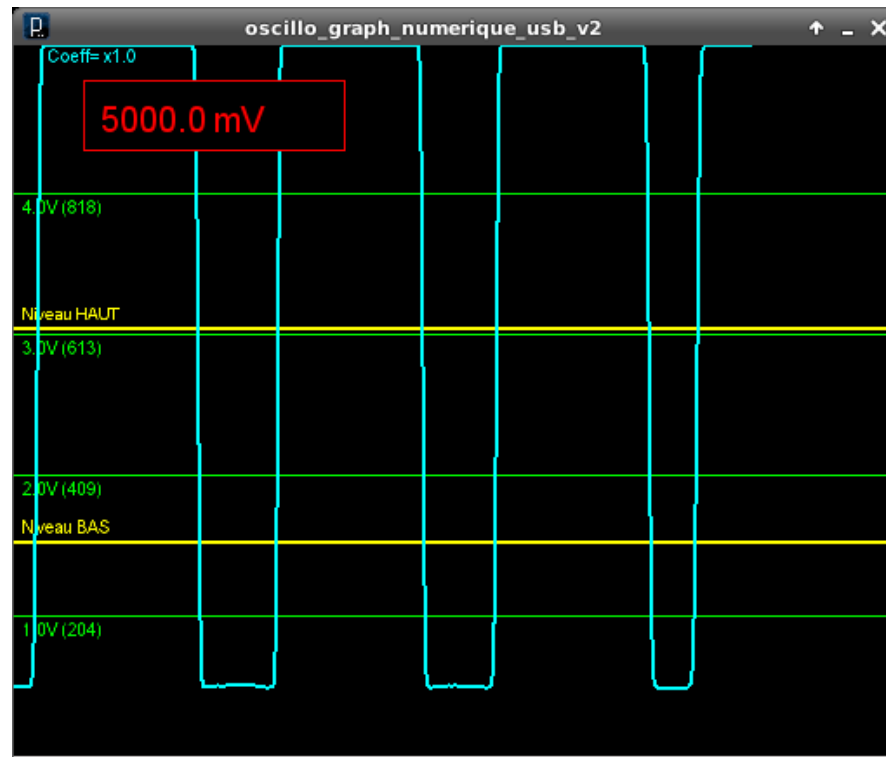
Pour la LED, la broche courte est la cathode et la longue l'anode, pour le photo-transistor, la broche courte est l'émetteur et la longue le collecteur.

Truc pratique : pour vérifier que la LED s'allume bien, enlever l'opto-coupleur et remplacez-le par une LED normale. Si elle s'allume, tout est bien connecté. Ensuite, remettre l'opto-coupleur.

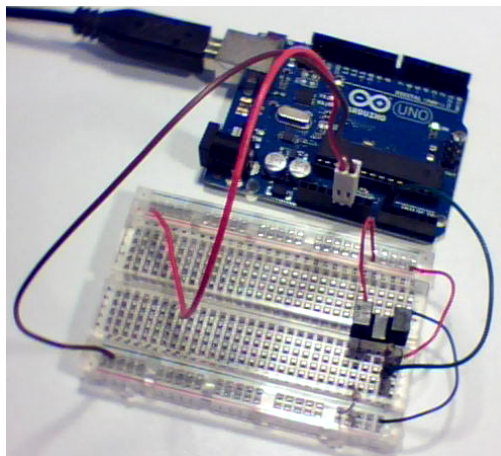


14. Pour info : Visualisation de la sortie de l'opto-coupleur.

Pour info, voici la visualisation dans une interface Processing de la sortie collecteur de l'opto-coupleur du montage précédent :



A chaque passage à 5V = présence d'un objet dans la fente !



Purchased by Franck Ourion, franck.ourion@univ-lorraine.fr #6280170

Atelier Arduino : Broches numériques en entrée : utiliser les capteurs numériques ON/OFF.

15. Détecter le passage d'un objet dans la fente d'un opto-coupleur : le programme

Nous allons commencer par un programme très simple mais qui va nous permettre de détecter la présence d'un objet dans la fente.

Entete déclarative

On va déclarer à ce niveau :

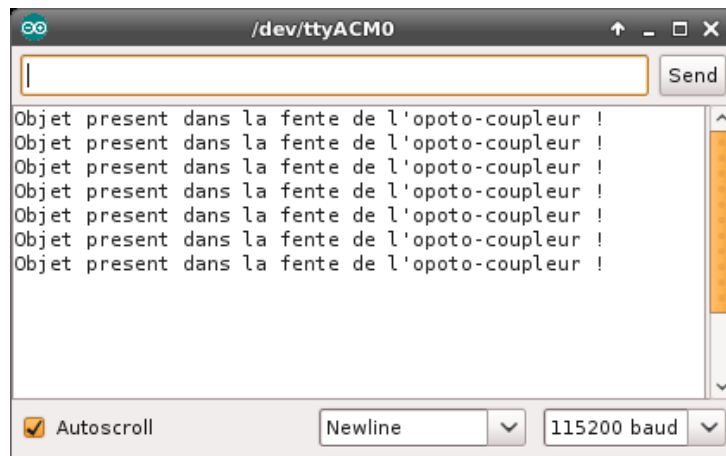
- 1 constante désignant la broche utilisée, appelée ici OPTO
- 1 constante int appelée PRESENCE et initialisée à 1.

Fonction `setup()`

- On initialise la communication série à 115200 bauds avec l'instruction `Serial.begin(debit)`
- on configure la broche en entrée avec l'instruction `pinMode(broche, INPUT)`

Fonction `loop()`

- À l'aide d'une condition `if()` on teste si un objet est présent en lisant l'état de la broche avec la fonction `digitalRead()`
- Si un objet est présent :
 - on envoie un message sur le port Série.
 - on réalise une courte pause entre 2 prises en compte



Fonctionnement

- Lancer le Terminal Série (Tools > Serial Monitor) et régler la vitesse de transmission sur 115200 bauds.
- Placer un objet dans la fente : le message apparaît !

```
//--- entete déclarative = déclarer ici variables et constantes globales

const int OPTO=2; // broche de l'optocoupleur

const int PRESENCE=1; // constante valeur broche lors de la présence
objet

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    Serial.begin(115200); // vitesse communication série

    pinMode(OPTO, INPUT); // broche en entrée

} // fin de la fonction setup()

//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    if(digitalRead(OPTO)==PRESENCE) { // si présence d'un objet
        Serial.println("Objet present dans la fente de l'opoto-
coupleur !"); // message
        delay(500); // pause entre 2 prises en compte
    } // fin if

} // fin de la fonction loop()
```


16. Compter les passages d'un objet dans un optocoupleur : le programme

Bien, c'est un bon début. Une fois que l'on sait faire ça, on peut envisager de compter le nombre de passages d'un objet dans la fente. Pour obtenir ce résultat, on va utiliser une variable de comptage tout simplement. Voyons cela...

Entete déclarative

On va déclarer à ce niveau :

- 1 constante désignant la broche utilisée, appelée ici OPTO
- 1 constante int appelée PRESENCE et initialisée à 1.
- 1 variable de type **int** pour stocker le nombre de passage
- 1 variable **boolean** pour mémoriser le dernier état pris en compte. En effet, si un objet reste dans la fente, il ne faut pas qu'il soit pris en compte plusieurs fois...

Fonction **setup()**

- On initialise la communication série à 115200 bauds avec l'instruction **Serial.begin**(debit)
- on configure la broche en entrée avec l'instruction **pinMode**(broche, **INPUT**)

Fonction **loop()**

- À l'aide d'une condition **if()** : on teste si un objet est présent en lisant l'état de la broche avec la fonction **digitalRead()**
- A l'aide d'une seconde condition, on teste si au dernier passage, aucun objet n'était présent et donc dernier état n'était pas « **true** »
- Si un objet est présent et qu'il ne s'agit pas du même objet :
 - on incrémente la variable de comptage et mise à true du témoin
 - on envoie un message sur le port Série.
 - on réalise une courte pause entre 2 prises en compte
- Sinon, on réinitialise la variable témoin à « **false** »

Fonctionnement

- Lancer le Terminal Série (Tools > Serial Monitor) et régler la vitesse de transmission sur 115200 bauds.
- Placer un objet dans la fente : le message apparaît ! Si le même objet reste dans la fente, il n'est pas pris en compte 2 fois.

```
//--- entete déclarative = déclarer ici variables et constantes globales

const int OPTO=2; // broche de l'optocoupleur

const int PRESENCE=1; // constante valeur broche lors de la présence
objet

int nombrePassage=0; // variable de comptage du nombre de passage

boolean temoinPassage=false; // variable temoin presence objet

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    Serial.begin(115200); // vitesse communication série

    pinMode(OPTO, INPUT); // broche en entrée

} // fin de la fonction setup()

//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    if (digitalRead(OPTO)==PRESENCE){ // si présence d'un objet et aucun
objet avant

        if (temoinPassage==false) { // si dernier etat = absence d'objet

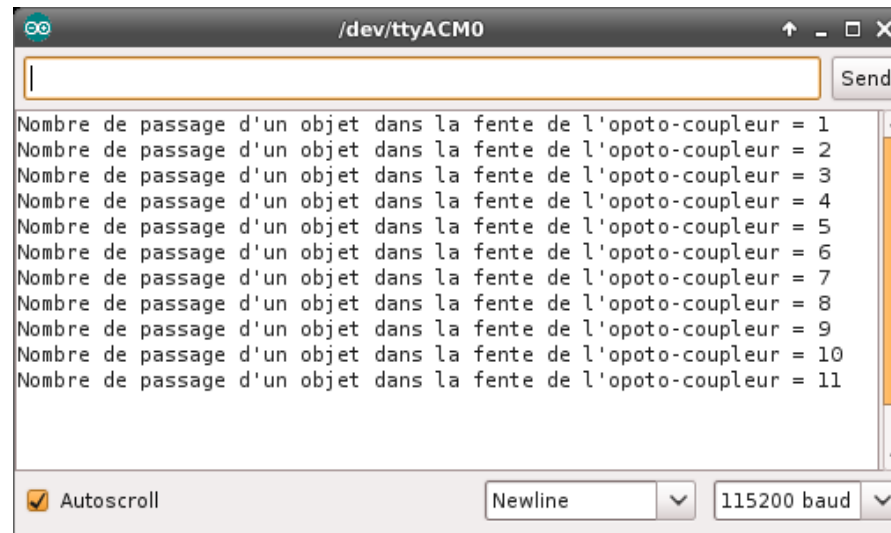
            temoinPassage=true; // mémorise presence objet
            nombrePassage=nombrePassage+1; // ajoute un passage

            Serial.print("Nombre de passage d'un objet dans la fente de
l'opoto-coupleur = "); // message
            Serial.println(nombrePassage);
            delay(500); // pause entre 2 prises en compte

        } // fin if temoinPassage

    } // fin if
    else { // si pas d'objet
        temoinPassage=false; // mémorise absence d'objet
    } // fin else

} // fin de la fonction loop()
```

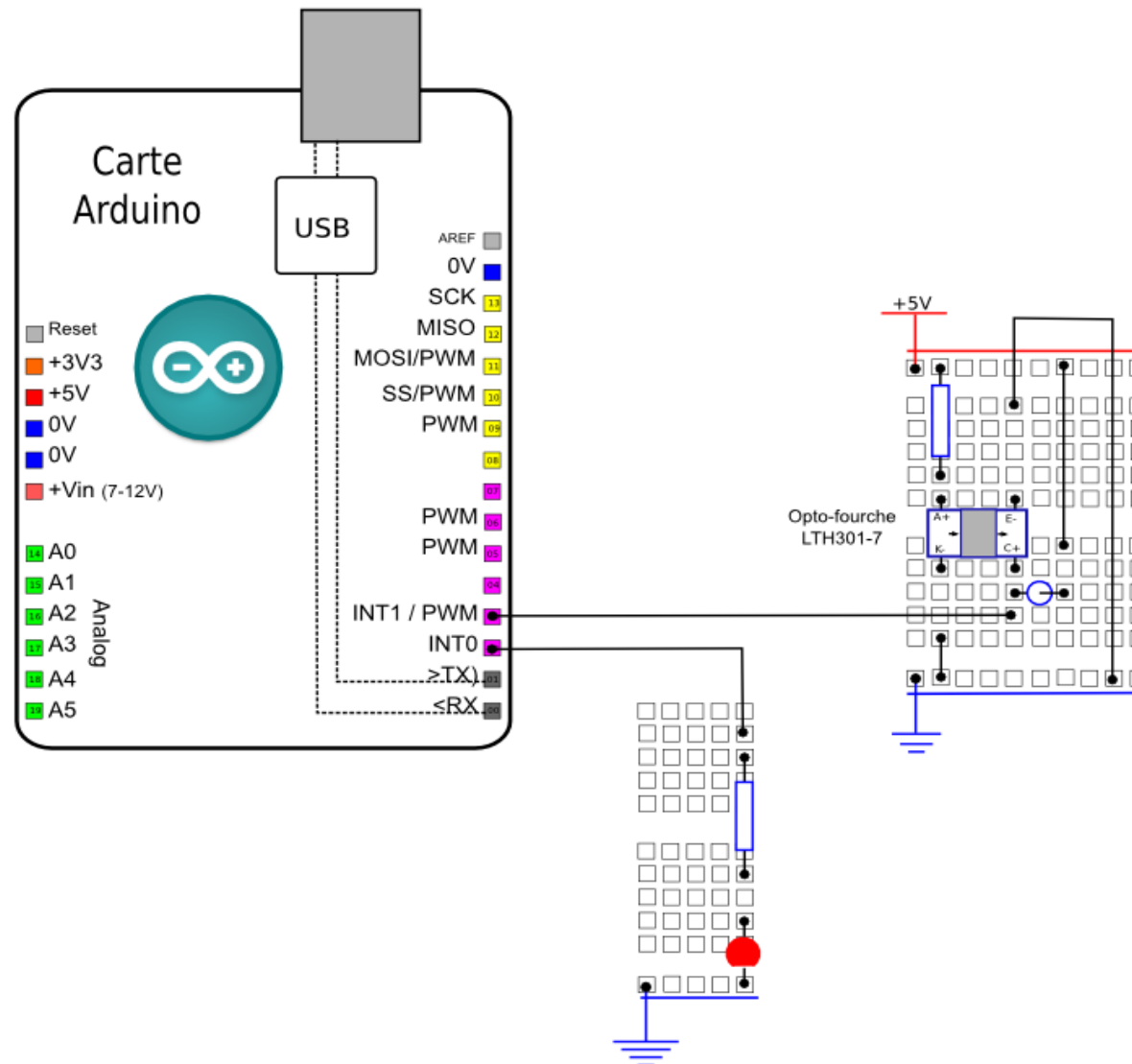


Purchased by Franck Ourion, franck.ourion@univ-lorraine.fr #6280170

Atelier Arduino : Broches numériques en entrée : utiliser les capteurs numériques ON/OFF.

17. La détection d'un objet dans un opto-coupleur allume une LED : le montage

Rien de très compliqué pour vous à présent : on reprend le montage précédent mais en utilisant la broche 3 pour la sortie collecteur de l'opto-coupleur. On connecte par ailleurs une LED et sa résistance sur la broche 2. Voici le schéma du montage :



18. La détection d'un objet dans un opto-coupleur allume une LED : le programme

On continue sur notre lancée : là encore, vous devriez y arriver sans trop de difficultés. Ce programme va allumer la LED lorsqu'un objet est présent dans la fente de l'opto-coupleur. L'intérêt ici est de montrer une nouvelle fois l'utilisation simultanée d'une broche en entrée et une en sortie.

Entete déclarative

On va déclarer à ce niveau :

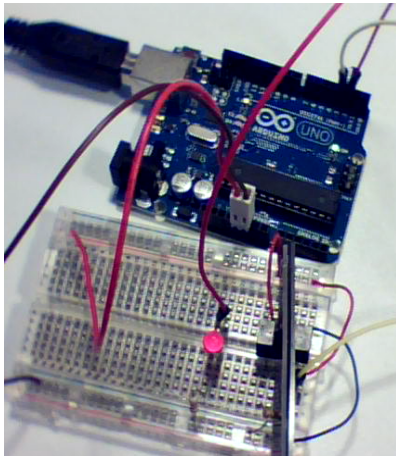
- 1 constante désignant la broche utilisée, appelée ici OPTO
- 1 constante **int** appelée PRESENCE et initialisée à 1.
- 1 constante de broche pour la LED

Fonction **setup()**

- on configure la broche de l'optocoupleur en entrée avec l'instruction **pinMode**(broche, **INPUT**)
- on configure la broche de la LED en sortie avec l'instruction **pinMode**(broche, **OUTPUT**)

Fonction **loop()**

- À l'aide d'une condition **if()** on teste si un objet est présent en lisant l'état de la broche avec la fonction **digitalRead()**
- Si un objet est présent, on allume la LED, sinon on l'éteint.



Fonctionnement

- Placer un objet dans la fente : la LED s'allume ! Enlevez-le : la LED s'éteint !

Purchased by Franck Ourion, franck.ourion@univ-lorraine.fr #6280170

Atelier Arduino : Broches numériques en entrée : utiliser les capteurs numériques ON/OFF.

```
//--- entete déclarative = déclarer ici variables et constantes globales

const int OPTO=3; // broche de l'optocoupleur
const int PRESENCE=1; // constante valeur broche lors de la présence
objet
const int LED=2; // broche de la LED

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    pinMode(OPTO, INPUT); // broche en entrée
    pinMode(LED, OUTPUT); // broche en sortie

} // fin de la fonction setup()

//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    if(digitalRead(OPTO)==PRESENCE) { // si présence d'un objet
        digitalWrite(LED,HIGH); // allume la LED
    } // fin if
    else { // sinon
        digitalWrite(LED,LOW); // éteint la LED
    } // fin else

} // fin de la fonction loop()
```

19. Stratégie de programmation : comptage de fréquence

Notion de fréquence

Avant de passer à la suite, prenons le temps de réfléchir à la notion de fréquence, plus exactement au comptage de la fréquence de survenue d'un évènement.

Par exemple, imaginons que l'on veuille compter le nombre de tours par seconde d'un axe en rotation. Pour réaliser cette mesure, on va avoir besoin de 2 choses :

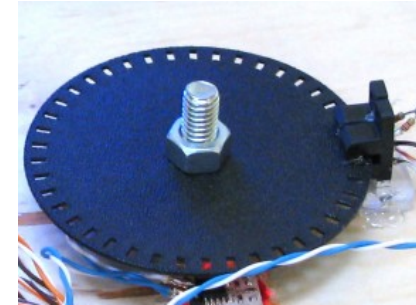
- d'une base temps fixe, autrement dit une durée précise pendant laquelle on va compter la survenue d'un évènement
- d'un « compteur » qui va permettre de comptabiliser tous les évènements qui sont survenus pendant la durée du comptage.



La fréquence de survenue de l'évènement vaudra :
fréquence = nombres d'évènements / durée de comptage

Comptage des événements

Imaginons que l'on veuille compter le nombre fois où un évènement survient dans un certain délai. Par exemple, si l'on veut compter la vitesse de rotation d'un moteur ou d'un axe, on pourra compter le nombre fois où l'objet en rotation est détecté dans un opto-coupleur. A ce stade, on sait faire comme on l'a vu dans un programme précédent : il suffit d'incrémenter une variable.



Exemple de comptage en rotation par opto-coupleur

Fixer un délai de comptage

Pour fixer un délai de comptage, on va se baser sur l'instruction Arduino `millis()` qui renvoie à tout moment le nombre de millisecondes écoulées depuis la mise sous tension de l'Arduino.

Pour fixer un délai de comptage fixe, on va utiliser 2 variables :

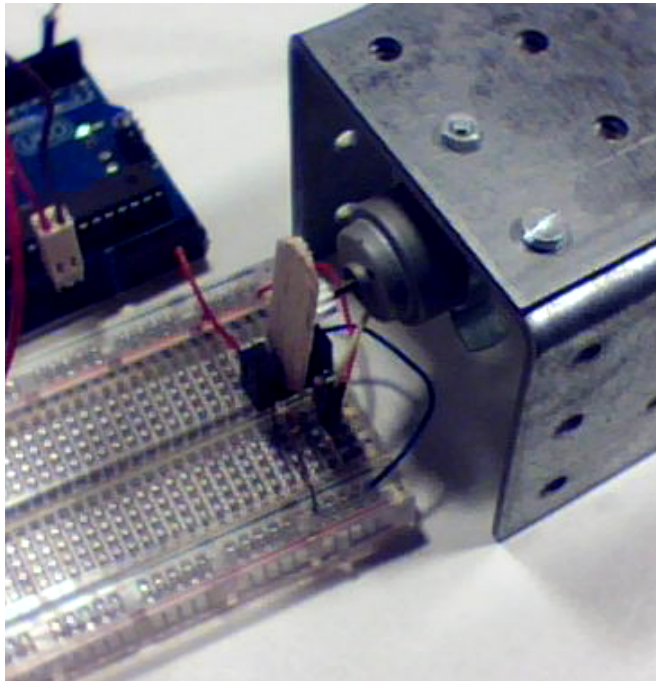
- une pour mémoriser la dernière valeur de `millis()` prise en compte
- une pour fixer le délai de comptage et permettre d'évaluer si le délai voulu s'est écoulé.

La stratégie de programmation va consister à :

- mémoriser la valeur de `millis()`
- tester à chaque passage de `loop()` si le délai de comptage est écoulé
- si oui :
 - exécuter les instructions voulues
 - remettre à zéro les variables de comptages
 - mémoriser la nouvelle valeur de `millis()`
- et ainsi de suite...

20. Mesurer la vitesse de rotation d'un moteur : la mécanique

- Côté électronique, on va utiliser le même montage que vu précédemment (optocoupleur seul connecté sur la broche 2).
- Côté mécanique, on va ici utiliser un simple moteur à courant continu (CC) 6V sur l'axe duquel on va fixer une languette de bois. On positionne le moteur de façon à ce que la languette de bois passe dans l'opto-coupleur à chaque rotation, ce qui va permettre de compter la vitesse de rotation du moteur. **Noter que le moteur est alimenté ici par sa propre alimentation (bloc secteur ou pile) .**



Le moteur est fixé de manière à ce qu'une languette de bois fixée sur l'axe passe dans la fente de l'opto-coupleur...

21. Mesurer la vitesse de rotation d'un moteur : le programme

Le moteur dont on veut mesurer la vitesse va tourner à plusieurs milliers de tours par minute... et sans notre Arduino, il serait impossible de mesurer cette vitesse ! Nous allons utiliser ici un délai de comptage de 1000ms (= 1 seconde !) pour simplifier les choses. Voyons comment coder cela...

Entête déclarative

- On va déclarer à ce niveau :
- 1 constante désignant la broche utilisée, appelée ici OPTO
- 1 constante int appelée PRESENCE et initialisée à 1.
- 1 variable de type **int** pour stocker le nombre de passage
- 1 variable **boolean** pour mémoriser le dernier état pris en compte. En effet, si un objet reste dans la fente, il ne faut pas qu'il soit pris en compte plusieurs fois...
- 1 variable **long** pour stocker la dernière valeur de **millis()** prise en compte et une variable **int** pour fixer le délai de comptage.

```
//--- entete déclarative = déclarer ici variables et constantes globales

const int OPTO=2; // broche de l'optocoupleur
const int PRESENCE=1; // constante valeur broche lors de la présence
objet

int nombrePassage=0; // variable de comptage du nombre de passage
boolean temoinPassage=false; // variable temoin presence objet

long millis0=0; // variable de mémorisation de millis() courant
int delai=1000; // delai de comptage en millisecondes
```

Fonction **setup()**

- on initialise la communication série avec l'instruction **Serial.begin(vitesse)**. On utilisera 115200 bauds.
- on configure la broche de l'optocoupleur en entrée avec l'instruction **pinMode(broche, INPUT)**

```
//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    Serial.begin(115200); // vitesse communication série

    pinMode(OPTO, INPUT); // broche en entrée

} // fin de la fonction setup()
```


Fonction `loop()`

Gestion du comptage des événements

- À l'aide d'une condition `if()` : on teste si un objet est présent en lisant l'état de la broche avec la fonction `digitalRead()`
- A l'aide d'une seconde condition, on teste si au dernier passage, aucun objet n'était présent et donc dernier état n'était pas « `true` »
- Si un objet est présent et qu'il ne s'agit pas du même objet :
 - on incrémente la variable de comptage et mise à `true` du témoin
 - on envoie un message sur le port Série.
 - on réalise une courte pause entre 2 prises en compte
- Sinon, on réinitialise la variable témoin à « `false` »

Gestion du délai de comptage

- A l'aide d'une condition `if()`, on teste à chaque passage de `loop()` si le délai voulu s'est écoulé.
- Si c'est le cas (c'est à dire toutes les secondes) :
 - on affiche le nombre de tours par seconde, en n'oubliant pas de diviser par 2, car à chaque tour, la languette entraîne 2 événements,
 - on affiche ensuite le nombre de tour par minute correspondant,
 - on ré-initialise la variable de comptage des événements
 - on mémorise le `millis()` courant pour lancer une nouvelle période de comptage.

Remarque :

Ce programme est facilement adaptable en changeant simplement la valeur de la variable `delai` pour augmenter ou réduire la durée du comptage.

Ce programme pourra servir de base également pour un anémomètre par exemple ou tout autre comptage de vitesse de rotation axiale.

```
//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    if (digitalRead(OPT0)==PRESENCE){ // si présence d'un objet et aucun
    objet avant

        if (temoinPassage==false) { // si dernier etat = absence d'objet

            temoinPassage=true; // mémorise presence objet
            nombrePassage=nombrePassage+1; // ajoute un passage

            //Serial.print("Nombre de passage d'un objet dans la fente de
l'opoto-coupleur = "); // message
            //Serial.println(nombrePassage);
            //delay(500); // pause entre 2 prises en compte

        } // fin if temoinPassage

    } // fin if
    else { // si pas d'objet
        temoinPassage=false; // mémorise absence d'objet
    } // fin else

    //--- gestion du calcul de la fréquence ---
    if ( millis()-millis0>delai) { // si le delai de comptage est écoulé

        Serial.print("Nombre de tours par seconde = ");
        nombrePassage=nombrePassage/2; // 2 passages par tour
        Serial.print(nombrePassage);
        Serial.print(" soit ");
        Serial.print(nombrePassage*60);
        Serial.println(" tours par minute !");

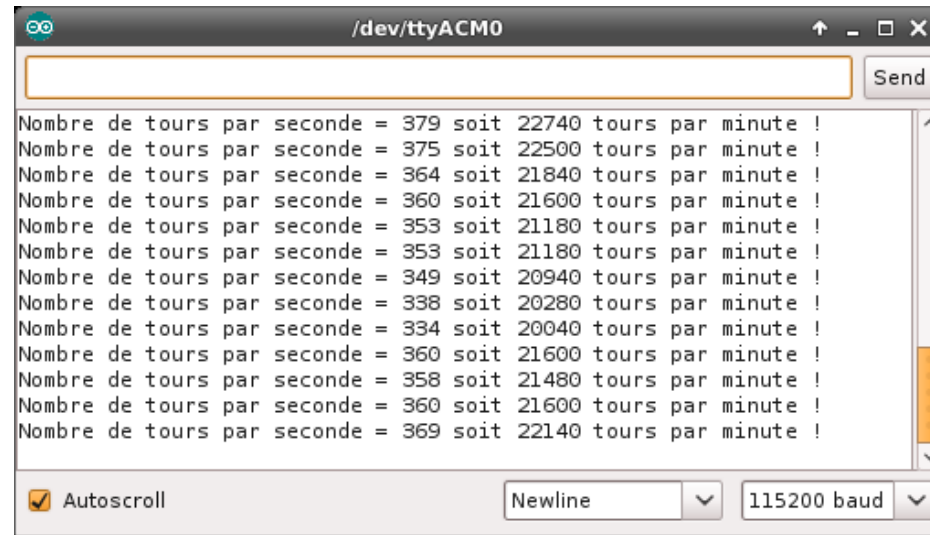
        nombrePassage=0; // réinitialise variable de comptage
        millis0=millis(); // réinitialise millis0

    } // fin if millis()

} // fin de la fonction loop()
```


Fonctionnement du programme

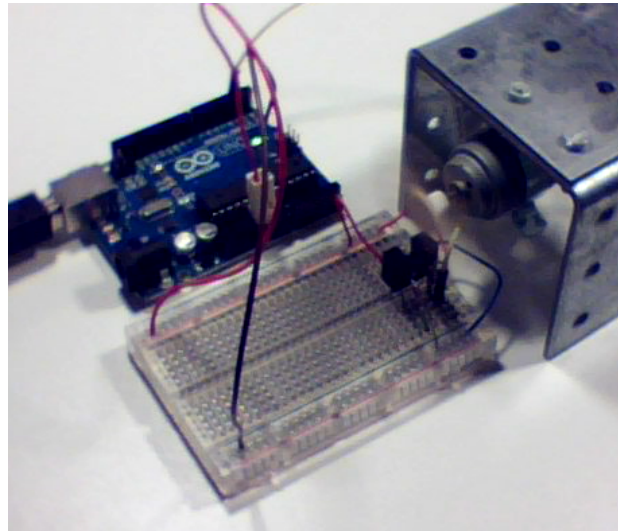
- Ouvrir le Terminal Série (Tools > Serial Monitor) et fixer le débit à la même valeur que celle utilisée pour l'instruction `Serial.begin(vitesse)`. Ici, **115200** bauds.
- Mettre le moteur sous tension et.... magie ! Arduino vous donne la vitesse de rotation du moteur : dans mon cas, **plus de 20 000 tours par minute** ! Excusez du peu...



The screenshot shows the Arduino Serial Monitor window titled "/dev/ttyACM0". The window has a text input field at the top with a "Send" button. Below it, a list of 12 lines of text displays the number of rotations per second and per minute. At the bottom, there are checkboxes for "Autoscroll" (checked), a "Newline" dropdown menu, and a baud rate dropdown menu set to "115200 baud".

```
Nombre de tours par seconde = 379 soit 22740 tours par minute !
Nombre de tours par seconde = 375 soit 22500 tours par minute !
Nombre de tours par seconde = 364 soit 21840 tours par minute !
Nombre de tours par seconde = 360 soit 21600 tours par minute !
Nombre de tours par seconde = 353 soit 21180 tours par minute !
Nombre de tours par seconde = 353 soit 21180 tours par minute !
Nombre de tours par seconde = 349 soit 20940 tours par minute !
Nombre de tours par seconde = 338 soit 20280 tours par minute !
Nombre de tours par seconde = 334 soit 20040 tours par minute !
Nombre de tours par seconde = 360 soit 21600 tours par minute !
Nombre de tours par seconde = 358 soit 21480 tours par minute !
Nombre de tours par seconde = 360 soit 21600 tours par minute !
Nombre de tours par seconde = 369 soit 22140 tours par minute !
```

Le moteur à courant continu tourne à pleine vitesse... et la vitesse s'affiche dans le Terminal Série !



Purchased by Franck Ourion, franck.ourion@univ-lorraine.fr #6280170

Atelier Arduino : Broches numériques en entrée : utiliser les capteurs numériques ON/OFF.

22. Les éléments du langage Arduino étudiés dans cet atelier

Structure

Variables et constantes

Fonctions

Constantes prédéfinies

- [HIGH](#) | [LOW](#)
- [INPUT](#) | [OUTPUT](#)

Entrées/Sorties numériques

- [pinMode](#)(broche, mode)
- [digitalWrite](#)(broche, valeur)
- int [digitalRead](#)(broche)

La documentation complète du langage Arduino en français est disponible ici :
http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.ReferenceMaxi

23. *A présent, vous devriez être capable :*

- d'écrire des programmes permettant d'interagir avec Arduino à l'aide de capteurs numériques
- de réaliser des comptages d'évènements
- de réaliser des comptages de fréquence

Table des matières

Broches numériques en entrée : utiliser les capteurs numériques ON/OFF.

Intro |

Matériel nécessaire pour les ateliers Arduino |

Rappel : Une broche numérique ne peut avoir que 2 états : HAUT ou BAS, « y'a ou y'a pas » ! |

Rappel : Une broche numérique est caractérisée par son SENS : en SORTIE ou en ENTREE ! |

Rappel : Les broches numériques de la carte Arduino |

Rappel: Les instructions du langage Arduino pour la gestion des broches numériques |

Introduction aux capteurs numériques ON/OFF |

Quelques exemples de capteurs numériques ON/OFF |

Utiliser un capteur numérique ON/OFF en pratique |

Fiche composant : découvrir le transistor et le photo-transistor |

Fiche composant : découvrir l'opto-coupleur en fourche |

Pour info : le spectre des ondes électro-magnétiques et de la lumière visible |

Utiliser un opto-coupleur en fourche en tant que capteur numérique : le montage |

Pour info : Visualisation de la sortie de l'opto-coupleur. |

Détecter le passage d'un objet dans la fente d'un opto-coupleur : le programme |

Compter les passages d'un objet dans un optocoupleur : le programme |

La détection d'un objet dans un opto-coupleur allume une LED : le montage |

La détection d'un objet dans un opto-coupleur allume une LED : le programme |

Stratégie de programmation : comptage de fréquence |

Mesurer la vitesse de rotation d'un moteur : la mécanique |

Mesurer la vitesse de rotation d'un moteur : le programme |

Les éléments du langage Arduino étudiés dans cet atelier |

A présent, vous devriez être capable : |

Bravo !
vous avez terminé cet atelier Arduino !



Prêt pour la suite ? Retrouvez de nombreux autres thèmes d'ateliers Arduino ici :

http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS