

**Broches numériques en entrée : utiliser le bouton poussoir.**



# Ateliers Arduino

par X. HINAULT

[www.mon-club-elec.fr](http://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'avez pas payé 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](mailto: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](mailto: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](mailto: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 comprendre les principes d'utilisation d'une broche numérique en entrée
- de comprendre les notions de « rappel au plus » et de pause « anti-rebond »
- d'apprendre à utiliser le bouton poussoir avec les broches numériques en entrée

... afin d'être capable d'écrire des programmes permettant d'interagir avec Arduino à partir d'appui sur des boutons poussoirs.

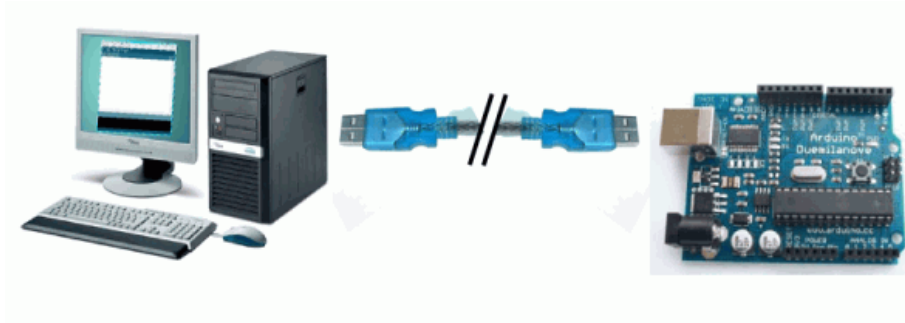


**Prêt ? C'est parti !**

## 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

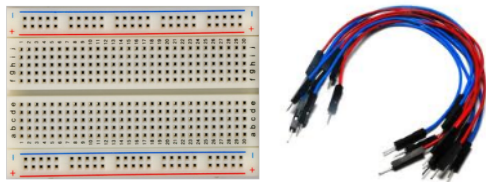


L'espace de développement Arduino associe :

- 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

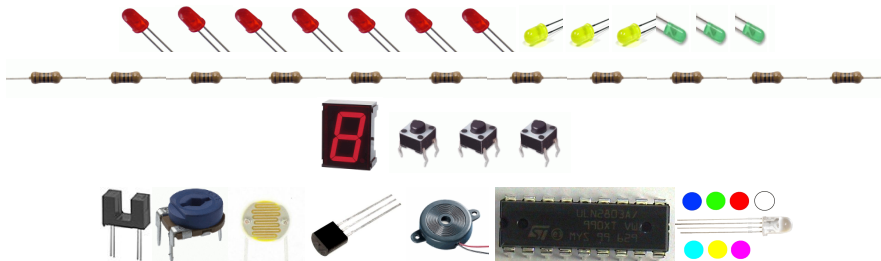


Pour réaliser des montages sans soudure, vous aurez besoin :

- 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](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** :

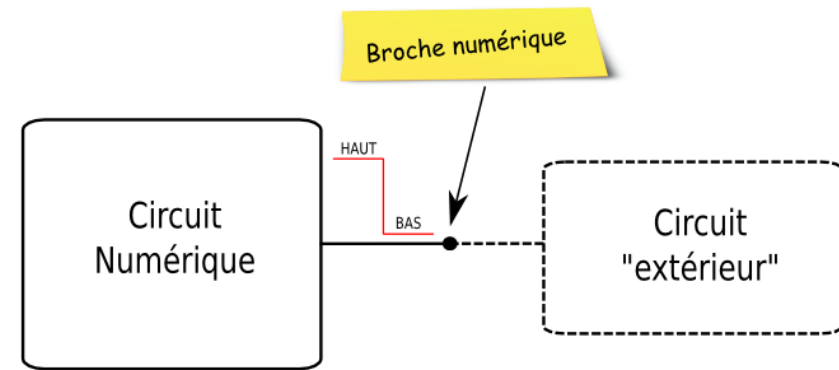
- soit au niveau **HAUT** (=5V), symbolisé par **1** ou **HIGH**
- soit au niveau **BAS** (=0V), symbolisé par **0** ou **LOW**
- noter qu'à **un instant quelconque, la broche se trouve obligatoirement dans l'un de ces 2 états.**

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 !

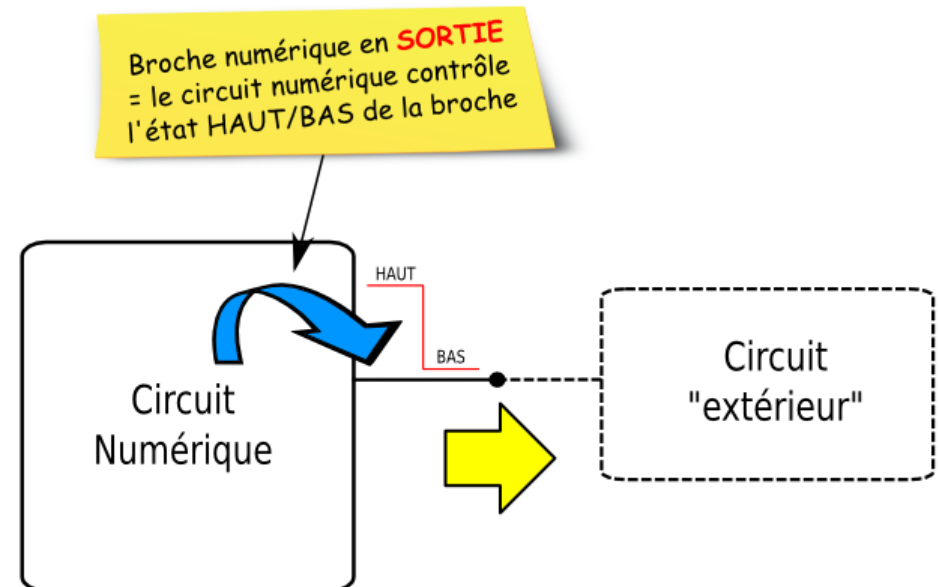
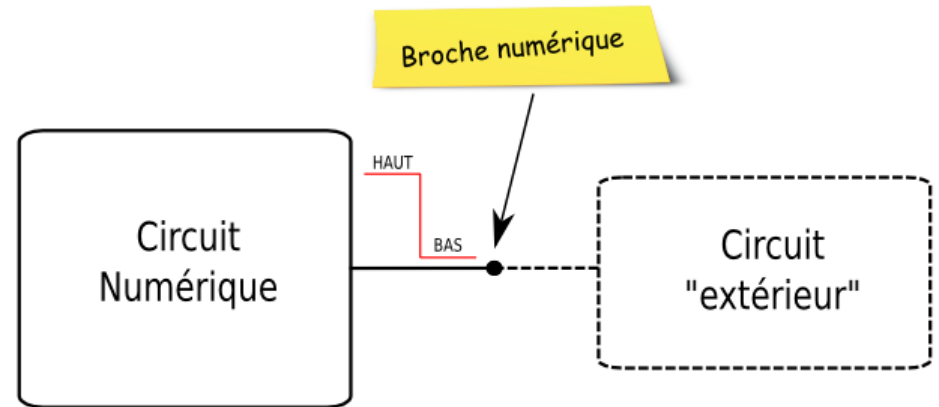
Une broche numérique est également caractérisée par son **sens** :

- 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 !

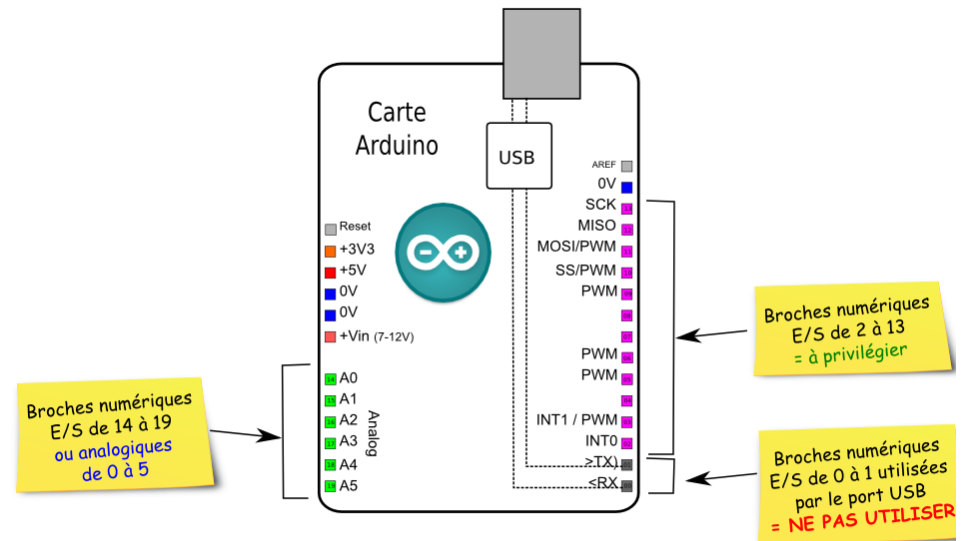
A savoir : les broches numériques 14 à 19 sont numérotées de 0 à 5 (ou désignées par A0, A1, A2, A3 et A5) lorsqu'elles sont utilisées en tant que broches analogiques.

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 !

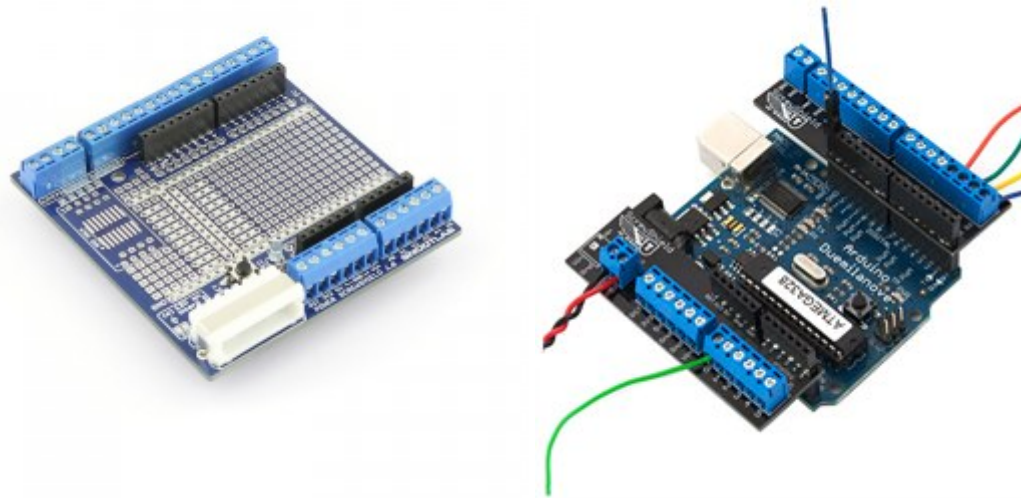


## 6. *Truc technique : les broches E/S de la carte Arduino sur borniers à vis avec un screwshield.... !*

### **Bon à savoir :**

Il existe des shields (carte d'extension) de la carte Arduino qui permettent de dédoubler les broches de la carte Arduino sur des borniers à vis : **les « screwshields » !**

**Très pratique pour finaliser des montages en « dur ».**



2 exemples de « screwshields » : le Powerscrewshield de Snootlab et le un screwshield en 2 parties.

## 7. 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
<b>BROCHE EN SORTIE</b> <code>pinMode(broche, OUTPUT);</code>	<code>digitalWrite(broche, HIGH);</code>	<code>digitalWrite(broche, LOW);</code>
<b>BROCHE EN ENTREE</b> <code>pinMode(broche, INPUT);</code>	<code>digitalRead(broche);</code>	<code>digitalRead(broche);</code>



## 8. Découvrir le bouton poussoir

### Description

Le bouton poussoir est un composant très simple : il s'agit d'un contacteur déclenché par l'appui sur un bouton :

- lorsque le bouton est appuyé, le contact est établi et le courant passe.
- lorsque le bouton est relâché, le contact n'est pas établi et le courant ne passe pas.

Un bouton poussoir miniature dispose de 4 broches typiquement qui sont connectées 2 à 2 : à l'appui, le contact est établi entre les 2 broches « à plat » lorsque le bouton est tourné vers soi.



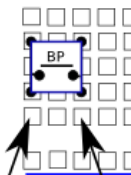
### Schéma théorique

Le schéma théorique du bouton poussoir est très simple :



### Principe d'utilisation sur une plaque d'essai

Le bouton poussoir s'enfiche à cheval entre 2 colonnes sur la plaque d'essai, le « plat » des broches tourné vers le bus d'alimentation : lorsque l'on appuie sur le bouton poussoir, le contact est établi entre les 2 colonnes.



Les 2 colonnes sont connectées entre-elles lorsque le bouton poussoir est appuyé.

Pour utiliser facilement le bouton poussoir sur une plaque d'essai, il peut être utile d'aplatir légèrement les pattes avec une pince de façon à ce qu'il s'insère facilement sur la plaque d'essai.

## 9. Principe d'utilisation d'un bouton poussoir avec une carte Arduino : notion de « rappel au plus ».

### Principe général

Comme on l'a déjà dit, une broche numérique utilisée en entrée va permettre de « lire » l'état de la broche.

D'où l'idée, très simple, de connecter un bouton poussoir entre la broche en entrée et le 0V afin d'interagir avec la carte Arduino.

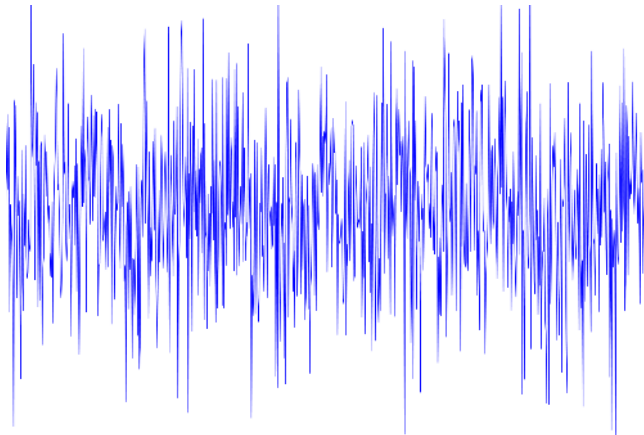
Ainsi :

- lorsque le bouton est appuyé, le contact est établi et la broche sera connectée au 0V.
- lorsque le bouton est relâché, le contact n'est pas établi et la broche ne sera pas connectée au 0V

### Etat d'une broche numérique en entrée non connectée...

Si vous êtes attentif, vous allez vous poser cette question : « dans quel état sera la broche numérique en entrée lorsqu'elle ne sera connectée à rien, c'est à dire lorsque le bouton poussoir sera relâché ? »

En effet, lorsque le bouton poussoir est relâché, la broche numérique en entrée restera non connectée et lorsque Arduino lira son état, quel sera-t-il ?? On pourrait intuitivement penser que dans ce cas la broche sera à 1... En fait, dans cette situation, la broche va se comporter comme une petite antenne... et va osciller en permanence au gré des interférences électro-magnétiques ambiantes. Si on enregistrerait l'état de la broche, on obtiendrait quelque chose comme ça :

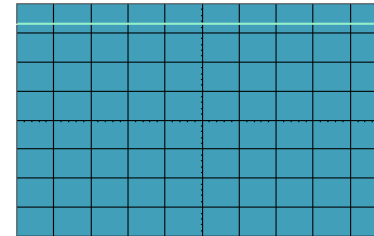


### La solution : une résistance de « rappel au plus »

Vous comprenez que dans ces conditions, il sera impossible d'analyser l'état de la broche numérique en entrée !!

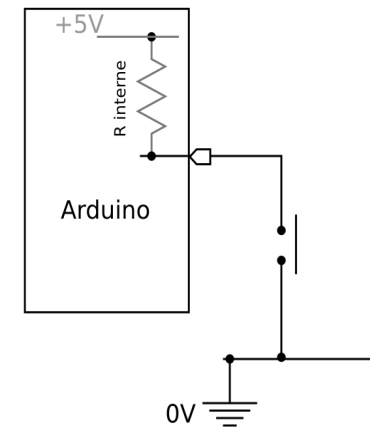
Pour contourner ce problème, on « attache » au +5V (ou au 0V) la broche numérique en entrée à l'aide d'une résistance assez élevée, de l'ordre de 10KOhms (la valeur exacte n'a pas grande importance, seul l'ordre de grandeur est à respecter...) : on appelle cela le « rappel au plus ».

Du coup, lorsque la broche ne sera pas connectée, elle sera parfaitement maintenue à 5V et l'enregistrement de la broche non-connectée deviendra :



### Truc : Utiliser les résistances de « rappel au plus » internes

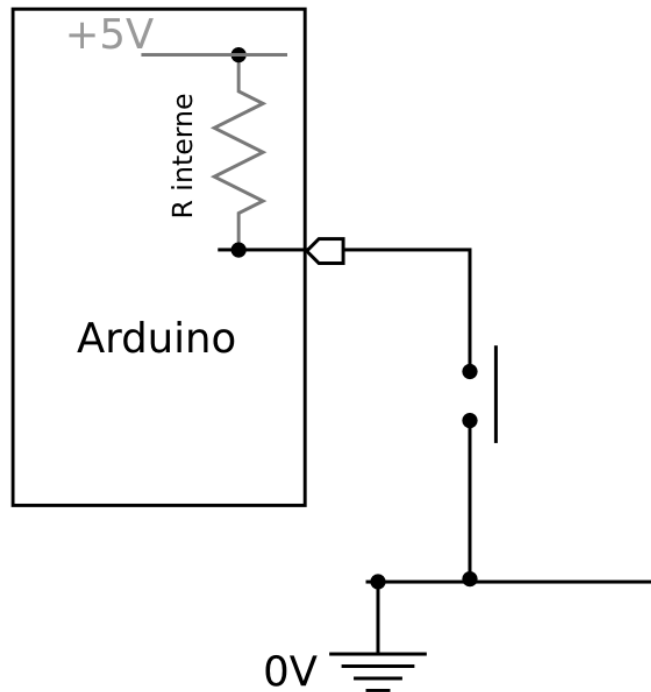
Les concepteurs de microprocesseur savent que vous allez utiliser un bouton poussoir sur une broche en entrée et pour résoudre ce problème, ils ont prévus en interne des résistances de rappel au plus (ou pull-up) internes. Arduino dispose de telles résistances : elles pourront être activées par le programme.



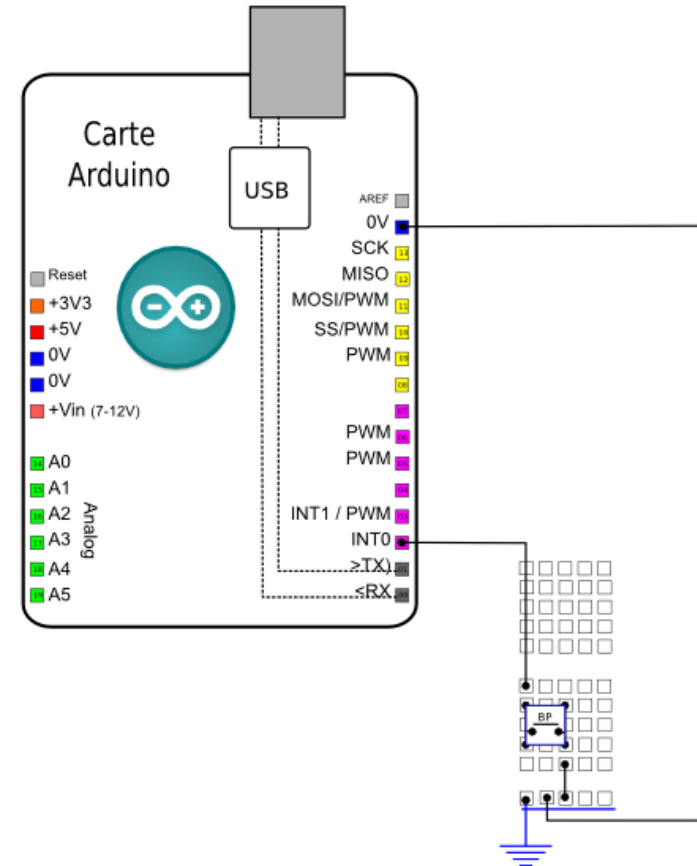
## 10. Broche ES en entrée : utiliser un bouton poussoir : Le montage

Le montage à réaliser pour utiliser un bouton poussoir avec la carte Arduino est simple si l'on utilise les résistances de rappel au plus internes comme on va le faire ici.

Il suffit donc de connecter le bouton poussoir (BP) d'une part au 0V et d'autre part à la broche numérique voulue de la carte Arduino selon :



**Noter que la résistance interne est représentée ici en grisé.**  
Il est tout à fait possible de réaliser le même montage en externe mais cela surcharge vite le câblage.



**A noter : il est déconseillé d'utiliser la broche 13 en entrée en raison de la présence d'une LED et de sa résistance attachées à cette broche sur la carte Arduino.**

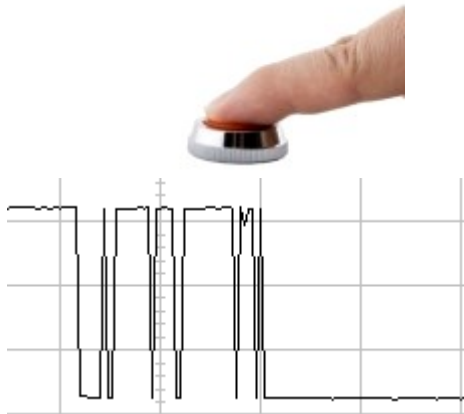
## 11. Notion de rebond et de pause anti-rebond

### Pour comprendre

Lorsque l'on appuie sur un bouton poussoir, au niveau microscopique, le contact n'est pas immédiat. Tout se passe un peu comme une balle qui rebondit avant de rester posée au sol : lorsque l'on appuie sur le bouton poussoir, le contact « rebondit » à plusieurs reprises avant d'être permanent.



D'un point de vue électrique, si on enregistrerait ce qui se passe lorsque l'on appuie sur un bouton poussoir, on aurait quelque chose comme ça :



La broche connectée au bouton poussoir passerait une première fois à l'état BAS puis redeviendrait HAUT (rebond) puis à nouveau BAS puis HAUT (rebond) et ainsi de suite avant de se stabiliser pour rester BAS en permanence.

### Le problème...

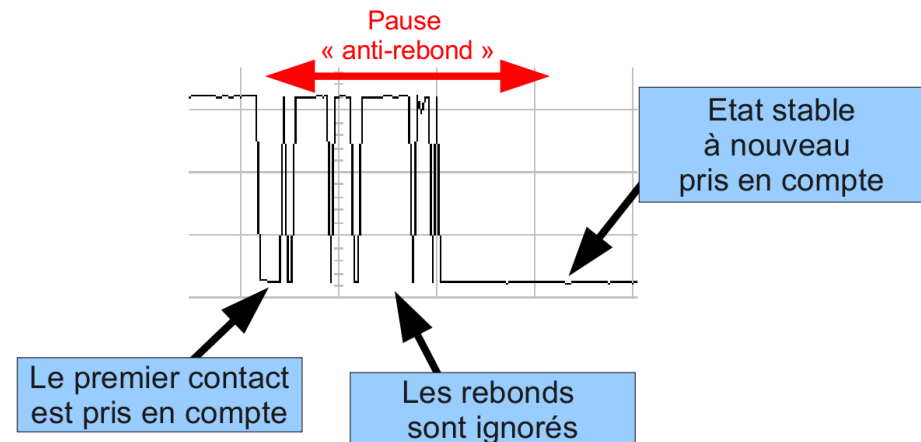
On comprend aisément ici le problème rencontré : alors que l'on aura appuyé qu'une seule fois sur le bouton, et que l'on s'attend à ce qu'il soit considéré comme appuyé 1 seule fois, la broche numérique en entrée va changer d'état à plusieurs reprises se comportant comme si le bouton avait été appuyé puis relâché à plusieurs reprises... c'est l'effet rebond.

### En pratique, utiliser une pause anti-rebond

Pour contourner ce problème, on va utiliser ce que l'on appelle une pause anti-rebond :

- on prendra en compte uniquement le premier changement d'état de la broche numérique connectée au bouton poussoir
- puis **on ne lira pas l'état de la broche pendant un certain délai (c'est la pause « anti-rebond »)**, de l'ordre que 100ms par exemple, de façon à laisser le temps au bouton poussoir de se stabiliser
- puis on lira à nouveau l'état du bouton poussoir...
- et ainsi de suite.

De cette façon, les changements ne seront bel et bien pris en compte qu'une seule fois et le rebond sera ignoré.



## 12. Broche ES en entrée : Visualiser l'appui sur un bouton poussoir dans le Terminal Série

Nous allons commencer par un programme très simple mais qui va nous permettre d'apprendre à utiliser un bouton poussoir : nous allons visualiser les appuis sur un bouton poussoir par un message dans le Terminal Série.

### Entete déclarative

On va déclarer à ce niveau :

- 1 constante désignant la broche utilisée, appelée ici BP
- 1 constante int appelée APPUI et initialisée à 0.

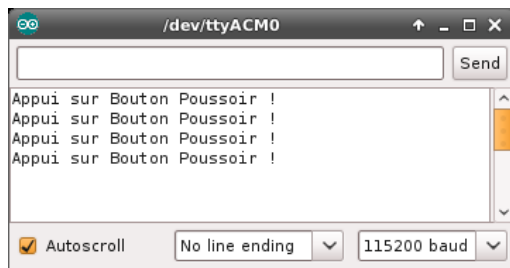
### Fonction **setup()**

- 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)`
- **ON ACTIVE LE « RAPPEL AU PLUS » interne de la broche en écrivant la valeur HIGH sur la broche, bien qu'elle ait été configurée en entrée.**

### Fonction **loop()**

- À l'aide d'une condition `if()` on teste si le bouton poussoir est bien appuyé en lisant son état avec la fonction `digitalRead()`
- Si le bouton poussoir est bien appuyé :
  - on envoie un message sur le port Série
  - on réalise une courte pause anti-rebond

**A RETENIR : ON ACTIVE LE « RAPPEL AU PLUS » interne d'une broche configurée en entrée en écrivant la valeur HIGH sur la broche, bien qu'elle soit configurée en entrée.**



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

const int BP=2; // broche su BP

const int APPUI=0; // constante valeur broche à l'appui sur BP

//---lafonctionsetup():exécutéeaudébutetlseulefois
void setup() {

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

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

  digitalWrite(BP, HIGH); // ACTIVATION DU RAPPEL AU PLUS INTERNE de la
  broche en entrée

} // fin de la fonction setup()

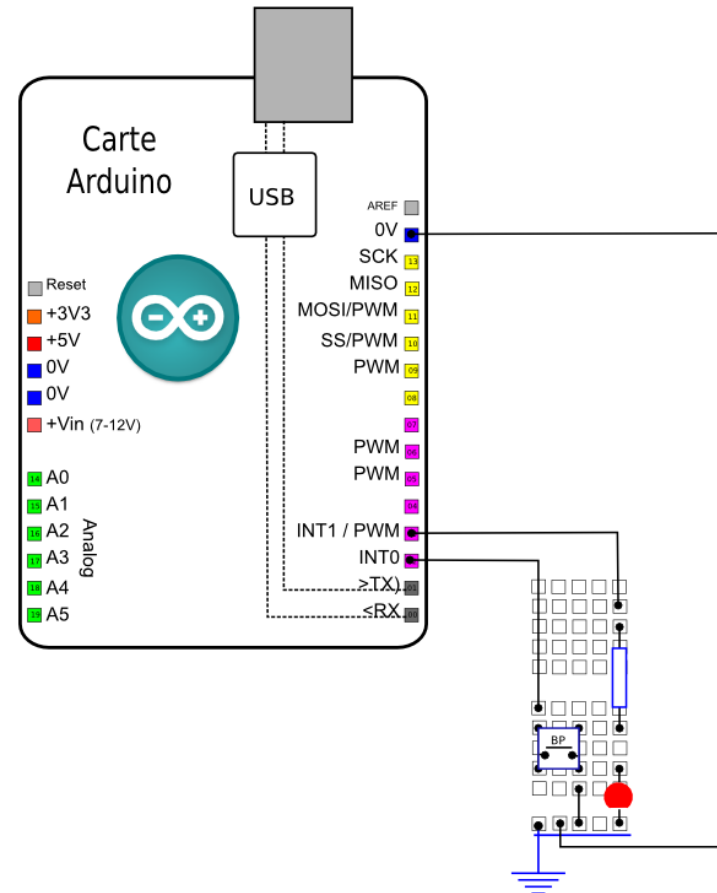
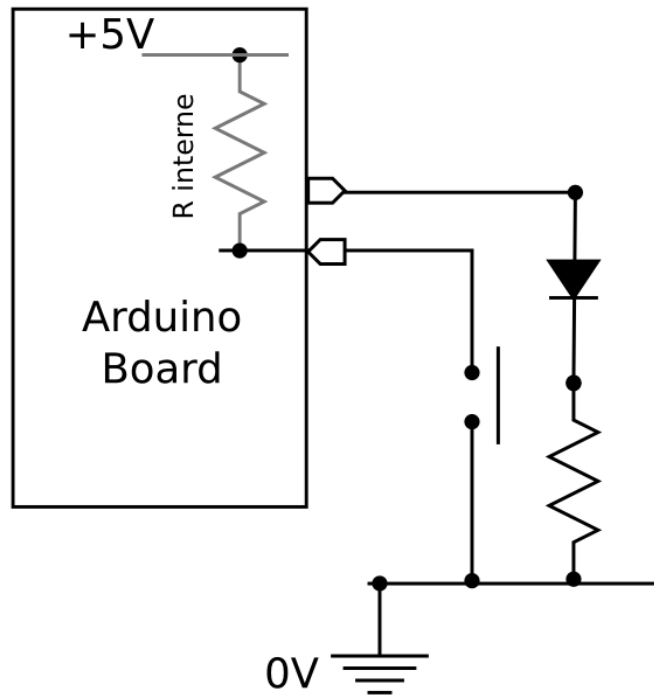
//---lafonctionloop():exécutéeensuiteenbouclesansfin
void loop() {

  if(digitalRead(BP)==APPUI) { // si appui
    Serial.println("Appui sur Bouton Poussoir !"); // message
    delay(250); // pause anti-rebond
  } // fin if

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

### 13. Utiliser un bouton poussoir et une LED : le montage.

Le montage à réaliser pour utiliser un bouton poussoir et une LED avec la carte Arduino consiste à combiner le montage d'un bouton poussoir et d'une LED : vous savez faire !



## 14. Un BP et une LED : La LED reflète l'état du BP

Cette fois, nous allons écrire un programme de façon à ce que la LED soit à tout moment dans l'état du bouton poussoir. Rien de très compliqué.

### Entete déclarative

On va déclarer à ce niveau :

- 1 constante désignant la broche utilisée pour la LED, appelée ici LED
- 1 constante désignant la broche utilisée pour le bouton poussoir, appelée ici BP
- 1 constante int appelée APPUI et initialisée à 0.
- 1 variable int pour mémoriser l'état du BP

### Fonction **setup()**

- on configure la broche de la LED en sortie avec l'instruction `pinMode(broche, OUTPUT)`
- on configure la broche du bouton poussoir en entrée avec l'instruction `pinMode(broche, INPUT)`
- **ON ACTIVE LE « RAPPEL AU PLUS » interne de la broche en écrivant la valeur HIGH sur la broche**, bien qu'elle ait été configurée en entrée.

### Fonction **loop()**

- À l'aide d'une condition `if()` on teste si le bouton poussoir est bien appuyé en lisant son état avec la fonction `digitalRead()`
- Si le bouton poussoir est bien appuyé :
  - on allume la LED
  - sinon, on éteint la LED
  - on réalise une courte pause anti-rebond (pas indispensable ici)

```
// --- Déclaration des constantes ---
const int APPUI=0; // constante état du BP - appui sur niveau bas

const int BP=2; //déclaration constante de broche
const int LED=3; //déclaration constante de broche

// --- Déclaration des variables globales ---

int ETAT_BP=0; // variable état BP

// La fonction setup() est exécutée en premier et 1 seule fois, au
démarrage du programme

void setup() { // debut de la fonction setup()

// ----- Broches en sortie -----
pinMode(LED, OUTPUT); //met la broche en sortie

// ----- Broches en entrée -----
pinMode(BP, INPUT); //met la broche en entree

// ----- Activation du rappel au + interne des broches en entrée si
nécessaire -----
digitalWrite(BP, HIGH) ; // activation du pullup de la broche en entrée

} // fin de la fonction setup()

// la fonction loop() s'exécute sans fin en boucle

void loop(){ // debut de la fonction loop()

// lire l'état du BP et mémoriser la valeur
ETAT_BP=digitalRead(BP);

// tester l'état du BP mémorisé et allumer la LED si appuyé
if (ETAT_BP==APPUI){ // si BP appuyé

digitalWrite (LED,1); // allume la LED

} // fin if

//.. sinon éteindre la LED
else { // sinon = BP obligatoirement relâché car seulement 2 états
possibles
digitalWrite (LED,0); //éteint la LED
}

} // fin de la fonction loop() - le programme recommence au début de la
fonction loop sans fin
```

## 15. Un BP et une LED : le BP en minuteur

Elaborons un petit peu ce programme de façon à ce qu'un appui sur le bouton poussoir allume la LED un certain temps puis l'éteigne à la façon d'une minuterie.

### Entete déclarative

On va déclarer à ce niveau :

- 1 constante désignant la broche utilisée pour la LED, appelée ici LED
- 1 constante désignant la broche utilisée pour le bouton poussoir, appelée ici BP
- 1 constante int appelée APPUI et initialisée à 0.

### Fonction **setup()**

- on configure la broche de la LED en sortie avec l'instruction `pinMode(broche, OUTPUT)`
- on configure la broche du bouton poussoir en entrée avec l'instruction `pinMode(broche, INPUT)`
- **ON ACTIVE LE « RAPPEL AU PLUS » interne de la broche en écrivant la valeur HIGH sur la broche**, bien qu'elle ait été configurée en entrée.

### Fonction **loop()**

- À l'aide d'une condition `if()` on teste si le bouton poussoir est bien appuyé en lisant son état avec la fonction `digitalRead()`
- Si le bouton poussoir est bien appuyé :
  - on allume la LED avec l'instruction `digitalWrite(broche, HIGH)`
  - puis on réalise une pause de durée voulue à l'aide de l'instruction `delay(duree)`

```
// --- Déclaration des constantes ---
const int APPUI=0; // constante état du BP - appui sur niveau bas

const int BP=2; //déclaration constante de broche
const int LED=3; //déclaration constante de broche

// --- Déclaration des variables globales ---

int etatLED=0; // variable état BP

// La fonction setup() est exécutée en premier et 1 seule fois, au
démarrage du programme

void setup() { // debut de la fonction setup()

// ----- Broches en sortie -----
pinMode(LED, OUTPUT); //met la broche en sortie

// ----- Broches en entrée -----
pinMode(BP, INPUT); //met la broche en entree

// ----- Activation du rappel au + interne des broches en entrée si
nécessaire -----
digitalWrite(BP, HIGH) ; // activation du pullup de la broche en entrée

} // fin de la fonction setup()

// la fonction loop() s'exécute sans fin en boucle

void loop(){ // debut de la fonction loop()

if (digitalRead(BP)==APPUI) { // si le BP est appuyé

digitalWrite(LED,HIGH); //allume la LED
delay (5000); // attend 5 secondes - le BP est inactif pendant ce temps
digitalWrite (LED,LOW); // éteint la LED

} // fin si BP appuyé

} // fin de la fonction loop() - le programme recommence au début de la
fonction loop sans fin
```



## 16. Un BP et une LED : l'appui sur le BP inverse la LED

A présent, nous allons écrire un programme de façon à ce qu'un appui sur le bouton poussoir inverse l'état de la LED.

### Entete déclarative

On va déclarer à ce niveau :

- 1 constante désignant la broche utilisée pour la LED, appelée ici LED
- 1 constante désignant la broche utilisée pour le bouton poussoir, appelée ici BP
- 1 constante int appelée APPUI et initialisée à 0.
- 1 variable int pour mémoriser l'état de la LED.

### Fonction **setup()**

- on configure la broche de la LED en sortie avec l'instruction `pinMode(broche, OUTPUT)`
- on configure la broche du bouton poussoir en entrée avec l'instruction `pinMode(broche, INPUT)`
- **ON ACTIVE LE « RAPPEL AU PLUS » interne de la broche en écrivant la valeur HIGH sur la broche**, bien qu'elle ait été configurée en entrée.

### Fonction **loop()**

- À l'aide d'une condition `if()` on teste si le bouton poussoir est bien appuyé en lisant son état avec la fonction `digitalRead()`
- Si le bouton poussoir est bien appuyé, on inverse l'état de la LED par un jeu de condition testant la variable d'état de la LED.

```
// --- Déclaration des constantes ---
const int APPUI=0; // constante état du BP - appui sur niveau bas

const int BP=2; //déclaration constante de broche
const int LED=3; //déclaration constante de broche

// --- Déclaration des variables globales ---

int etatLED=0; // variable état BP

// La fonction setup() est exécutée en premier et 1 seule fois, au
démarrage du programme

void setup() { // debut de la fonction setup()

// ----- Broches en sortie -----
pinMode(LED, OUTPUT); //met la broche en sortie

// ----- Broches en entrée -----
pinMode(BP, INPUT); //met la broche en entree

// ----- Activation du rappel au + interne des broches en entrée si
nécessaire -----
digitalWrite(BP, HIGH) ; // activation du pullup de la broche en entrée

} // fin de la fonction setup()

// la fonction loop() s'exécute sans fin en boucle

void loop(){ // debut de la fonction loop()

if (digitalRead(BP)==APPUI) { // si appui sur le BP

if (etatLED==0) etatLED=1; else etatLED=0; // inverse la variable
etatLED
delay(250); // pause anti-rebond
}

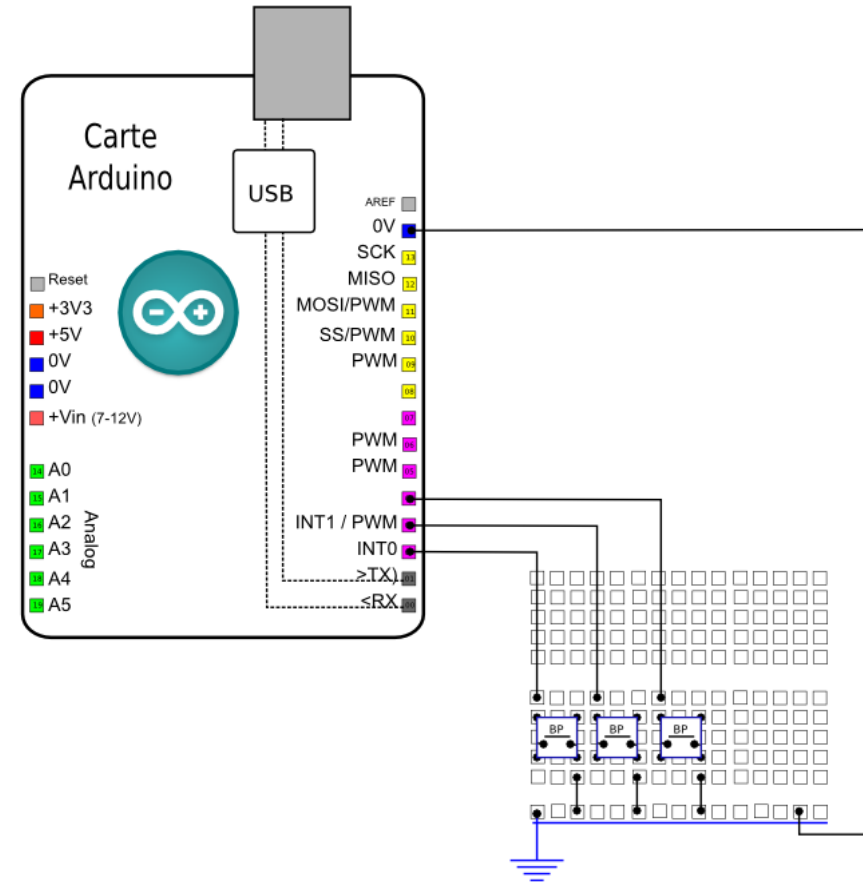
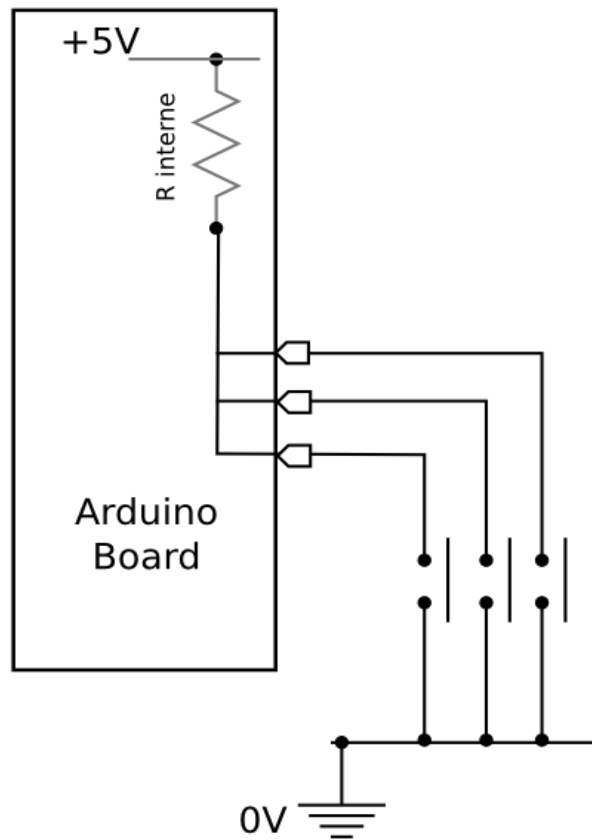
//met la LED dans l'état de la variable etatLED
if (etatLED==1) { // si la variable vaut 1
digitalWrite(LED,HIGH); // la LED est allumée
}
else { // sinon, càd si la variable vaut 0
digitalWrite(LED,LOW); // la LED est éteinte
}

} // fin de la fonction loop() - le programme recommence au début de la
fonction loop sans fin
```

## 17. Utiliser 3 boutons poussoir : le montage

Une situation beaucoup plus utile en pratique : utiliser 3 boutons poussoir, typiquement l'un +, l'autre – et le 3ème OK. Ceci permet de réaliser des réglages de valeur notamment.

Le montage est une réplication x3 du montage avec un bouton poussoir :



## 18. BP x 3 : incrémenter/décrémenter une variable avec affichage série

Une des utilisations les plus utiles des boutons poussoirs est le réglage de valeurs : avec 3 BP, on peut augmenter (ou incrémenter), diminuer (ou décrémenter) et valider une variable. On pourra de la sorte interagir avec Arduino pour régler une valeur de déclenchement ou une vitesse de clignotement.

Ici, on va uniquement voir le principe d'utilisation des 3 BP pour modifier une variable.

### Entete déclarative

On va déclarer à ce niveau :

- 3 constante désignant les 3 broches utilisées pour les bouton poussoir, appelée ici bpPLUS, bpMOINS, bpOK.
- 1 constante int appelée APPUI et initialisée à 0.
- 1 variable représentant une valeur à régler.

### Fonction setup()

- On initialise la communication série à 115200 bauds avec l'instruction `Serial.begin`(debit)
- on configure les broches des boutons poussoir en entrée avec l'instruction `pinMode`(broche, `INPUT`)
- **ON ACTIVE LE « RAPPEL AU PLUS » interne des broches en écrivant la valeur HIGH sur chaque broche.**

### Fonction loop()

- À l'aide d'une condition `if()` on teste si chaque bouton poussoir est appuyé en lisant son état avec la fonction `digitalRead()`
- Si le bouton poussoir PLUS est appuyé, on ajoute 1 à la variable
- Si le bouton poussoir MOINS est appuyé, on retranche 1 à la variable
- Si le bouton OK est appuyé, on affiche la variable sur le port Série

Un des défauts de cette façon de faire est de devoir lire en permanence l'état des boutons poussoir pour ne pas rater un appui : si on fait faire autre chose à Arduino, on risque de rater des appuis. **Pour contourner ce problème, on utilisera une interruption** (on verra ça plus tard...)

```
//---Déclarationdesconstantesutiles---
const int APPUI=LOW; // constante pour tester état BP

//---DéclarationdesconstantesdesbrochesE/Snumériques---

const int bpMOINS=2; // Constante pour la broche 2
const int bpPLUS=3; // Constante pour la broche 3
const int bpOK=4; // Constante pour la broche 3

//---Déclarationdesvariablesglobales---

int valeur=0; // variable à régler
int maxi=20; // valeur maximale
int mini=0; // valeur minimale

//Lafonctionsetup()estexécutéeenpremieretlseulefois

void setup() { // debut de la fonction setup()

//---iciinstructionsàexécuterlseulefoisaudémarrageduprogramme---

//-----Initialisationfonctionnalitésutilisées-----

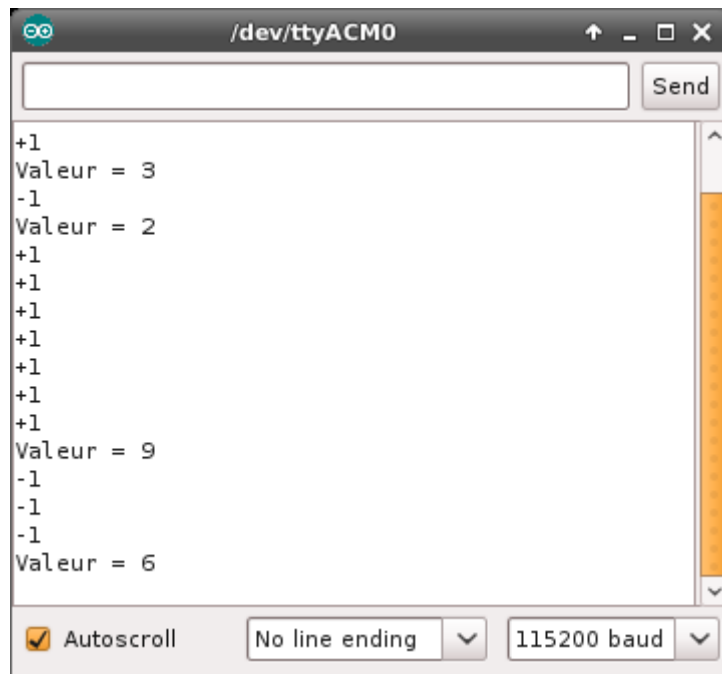
Serial.begin(115200); // initialise connexion série matérielle à 115200
bauds
//IMPORTANT:réglerleterminalcôtéPCaveclamêmevaleurdetransmission

//-----Brochesensortiesnumériques-----

//-----Brochesenentréesnumériques-----
pinMode (bpMOINS,INPUT); // Broche BP MOINS configurée en entrée
pinMode (bpPLUS,INPUT); // Broche BP PLUS configurée en entrée
pinMode (bpOK,INPUT); // Broche BP OK configurée en entrée

//-----Activationsibesoindurappelau+
(pullup)desbrochesenentréesnumériques-----
digitalWrite (bpMOINS,HIGH); // Rappel au + activé sur la broche BP
MOINS configurée en entrée
digitalWrite (bpPLUS,HIGH); // Rappel au + activé sur la broche BP PLUS
configurée en entrée
digitalWrite (bpOK,HIGH); // Rappel au + activé sur la broche BP OK
configurée en entrée

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



```
//la fonction loop() s'exécute sans fin
void loop(){ // debut de la fonction loop()

  if (digitalRead(bpPLUS)==APPUI) {

    valeur=valeur+1;
    if (valeur>maxi) valeur=maxi; // valeur maxi

    Serial.println("+1");
    delay(250); // pause anti-rebond

  }

  if (digitalRead(bpMOINS)==APPUI) {

    valeur=valeur-1;
    if (valeur<mini) valeur=mini; // valeur mini

    Serial.println("-1");
    delay(250); // pause anti-rebond

  }

  if (digitalRead(bpOK)==APPUI) {

    Serial.print("Valeur = ");
    Serial.println(valeur);
    delay(250); // pause anti-rebond

  }

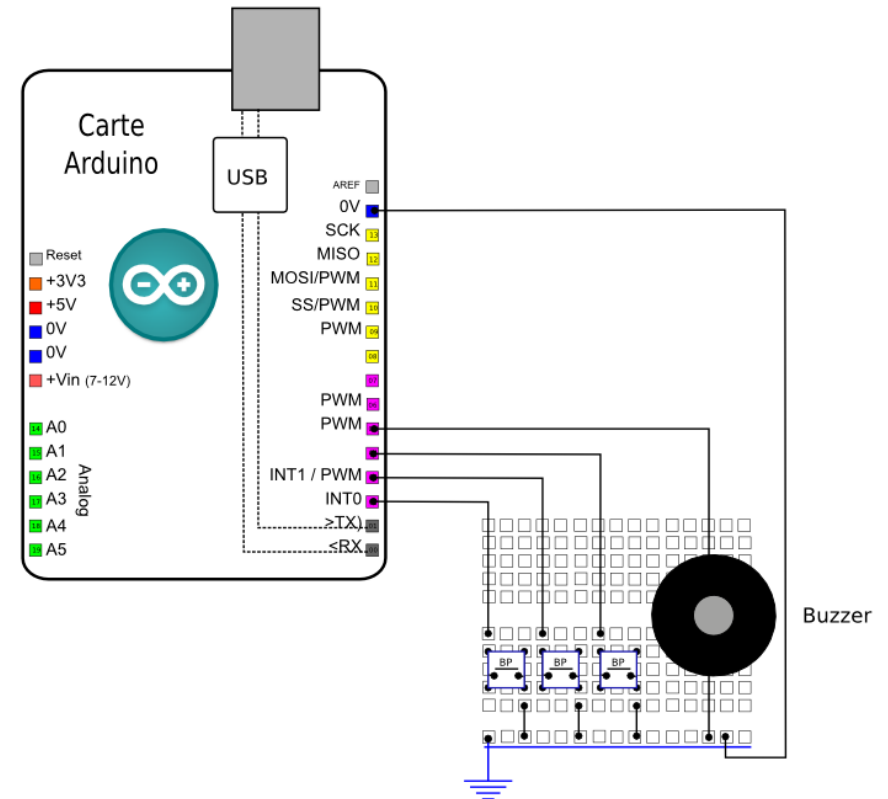
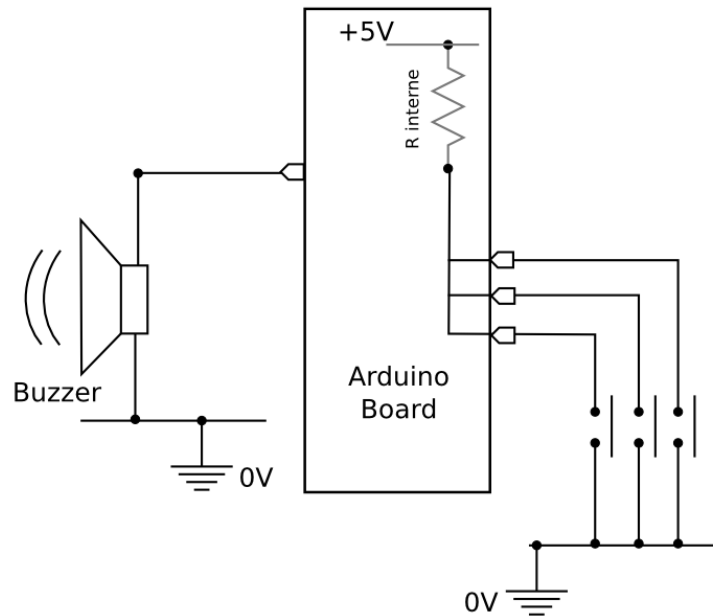
}

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

## 19. BP x 3 : Mini orgue – 3 notes : le montage

Cette fois, on va utiliser simultanément un buzzer piézo-électrique pour simuler un « mini-orgue » à 3 notes.

Le montage est une réplification x3 du montage avec un bouton poussoir associé à celui de l'utilisation du buzzer piézo-électrique :



## 20. BP x 3 : Mini orgue – 3 notes : le programme

Dans ce programme, on va déclencher la génération d'une note lors de l'appui sur un BP donné, et ce pour 3 notes différentes.

### Entete déclarative

On va déclarer à ce niveau :

- 3 constante désignant les 3 broches utilisées pour les bouton poussoir, appelée ici bpDO, bpRE, bpMI.
- 1 constante désignant la broche utilisée en sortie avec le buzzer
- 1 constante int appelée APPUI et initialisée à 0.
- 3 constantes de notes

### Fonction **setup()**

- on configure la broches du bouton poussoir en sortie avec l'instruction **pinMode** (broche, **OUTPUT**)
- on configure les broches des boutons poussoir en entrée avec l'instruction **pinMode**(broche, **INPUT**)
- **ON ACTIVE LE « RAPPEL AU PLUS » interne des broches en écrivant la valeur HIGH sur chaque broche.**

```
//---Déclarationdesconstantesutiles---
constint APPUI=LOW; // constante pour tester état BP

const int bpDO=2; // Constante pour la broche 2
const int bpRE=3; // Constante pour la broche 3
const int bpMI=4; // Constante pour la broche 3

const int PIEZO=5; // constante pour la broche du piézo

//---Déclarationdesvariablesglobales---

int D0=262; // variable de la fréquence note
int RE=294; // variable de la fréquence note
int MI=330; // variable de la fréquence note

void setup() { // debut de la fonction setup()

//-----Initialisationfonctionnalitésutilisées-----

//-----Brochesensortiesnumériques-----
pinMode (PIEZO,OUTPUT); // Broche BP MOINS configurée en entrée

//-----Brochesenentréesnumériques-----
pinMode (bpDO,INPUT); // Broche BP MOINS configurée en entrée
pinMode (bpRE,INPUT); // Broche BP PLUS configurée en entrée
pinMode (bpMI,INPUT); // Broche BP OK configurée en entrée

//-----Activationsibesoindurappelau+
// (pullup)desbrochesenentréesnumériques-----
digitalWrite (bpDO,HIGH); // Rappel au + activé sur la broche BP MOINS
// configurée en entrée
digitalWrite (bpRE,HIGH); // Rappel au + activé sur la broche BP PLUS
// configurée en entrée
digitalWrite (bpMI,HIGH); // Rappel au + activé sur la broche BP OK
// configurée en entrée

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

### (suite) Fonction **loop()**

- À l'aide d'une condition **if()** on teste si chaque bouton poussoir est appuyé en lisant son état avec la fonction **digitalRead()**
- Si le bouton poussoir DO est appuyé, on active la note DO avec l'instruction **tone(broche, note)**, sinon, on stoppe la note avec l'instruction **noTone(broche)**
- Idem pour les 2 autres notes.

```
void loop(){ // debut de la fonction loop()

  if (digitalRead(bpD0)==APPUI) { // si appui BP D0

    tone(PIEZ0,D0); // joue note D0
    delay(50); // pause anti-rebond

  }
  else if (digitalRead(bpRE)==APPUI) { // sinon si appui BP RE

    tone(PIEZ0,RE); // joue note RE
    delay(50); // pause anti-rebond

  }
  else if (digitalRead(bpMI)==APPUI) { // sinon, si appui BP MI
    tone(PIEZ0,MI); // joue note MI
    delay(50); // pause anti-rebond
  }
  else { // sinon
    noTone(PIEZ0); // sinon stoppe son
  }

}

} // fin de la fonction loop() - le programme recommence au début de la
fonction loop sans fin
```

## 21. 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](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.ReferenceMaxi)



## **22. *A présent, vous devriez être capable :***

- d'écrire des programmes permettant d'interagir avec Arduino à partir d'appui sur un ou plusieurs boutons poussoirs.

# Table des matières

Broches numériques en entrée : utiliser le bouton poussoir.

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 |

Truc technique : les broches E/S de la carte Arduino sur borniers à vis avec un screwshield.... ! |

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

Découvrir le bouton poussoir |

Principe d'utilisation d'un bouton poussoir avec une carte Arduino : notion de « rappel au plus ». |

Broche ES en entrée : utiliser un bouton poussoir : Le montage |

Notion de rebond et de pause anti-rebond |

Broche ES en entrée : Visualiser l'appui sur un bouton poussoir dans le Terminal Série |

Utiliser un bouton poussoir et une LED : le montage. |

Un BP et une LED : La LED reflète l'état du BP |

Un BP et une LED : le BP en minuteur |

Un BP et une LED : l'appui sur le BP inverse la LED |

Utiliser 3 boutons poussoir : le montage |

BP x 3 : incrémenter/décrémenter une variable avec affichage série |

BP x 3 : Mini orgue – 3 notes : le montage |

BP x 3 : Mini orgue – 3 notes : 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](http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS)