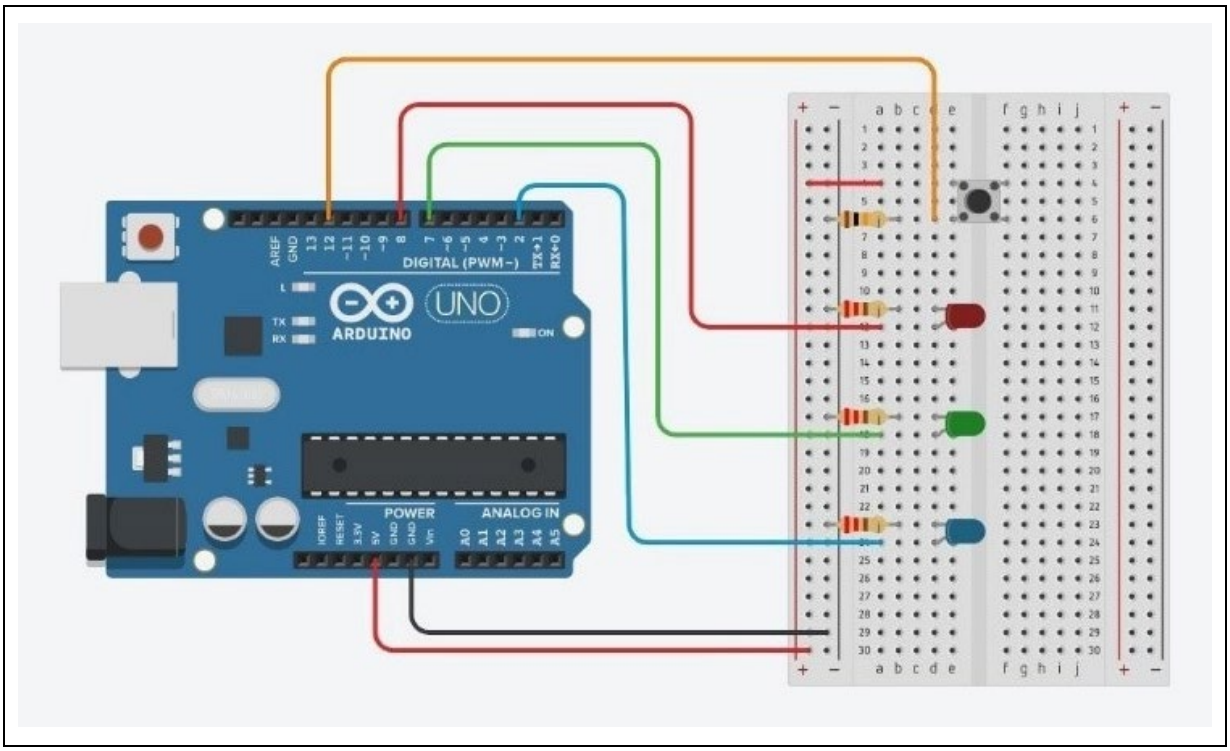
Projet 1 - Premiers pas : DEL & Bouton poussoir

L'objectif de l'étude de ce premier circuit est de comprendre le principe de fonctionnement des entrées et sorties numériques de l'Arduino à travers 4 activités, programmées en Python et en langage Arduino.



- Liste des composants :

. 1 DEL rouge

. 1 DEL verte

. 1 DEL bleue

. 3 résistances de 220 Ω
(résistances des DELs)

. 1 bouton poussoir

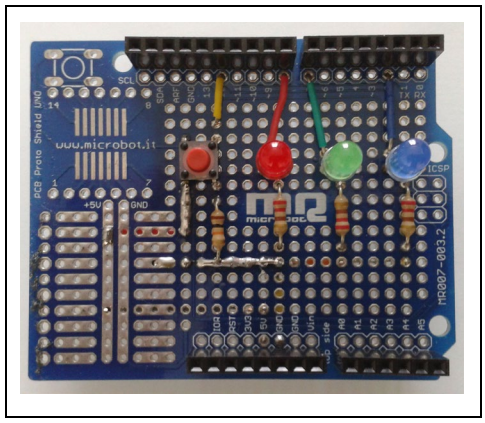
. 1 résistance de 10 kΩ
(résistance du bouton poussoir)

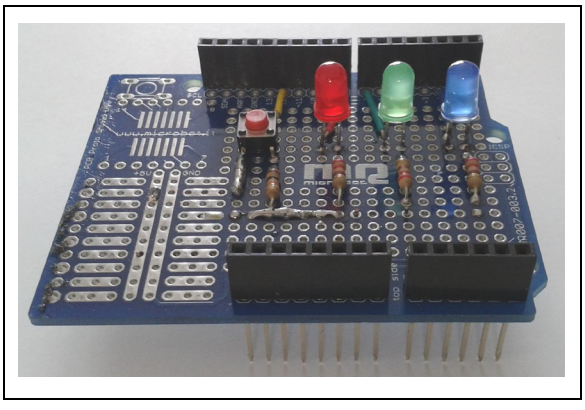
. 1 plaque d'essai

. Fils de connexion

- Protocole de communication :

. Firmata standard





Le circuit sur un "shield" pour Arduino Uno

Rappels :

Avant de voir en détail les différentes activités basées sur le circuit ci-dessus, nous allons faire quelques rappels sur l'électronique, l'Arduino et la programmation :

. Petit rappel sur l'électricité

L'électricité est un déplacement d'électrons dans un milieu conducteur.
Pour que ces électrons se déplacent tous dans un même sens, il faut qu'il y ait une différence du nombre d'électrons entre les deux extrémités du circuit électrique.
Pour maintenir cette différence du nombre d'électrons, on utilise un générateur (pile, accumulateur, alternateur...)

La différence de quantité d'électrons entre deux parties d'un circuit s'appelle la **différence de potentiel** et elle se mesure en **Volts (V)**et se note *U.*

Le débit d'électrons dans le conducteur correspond à *l'intensité*, aussi appelée *courant*. Elle se mesure en *Ampères* (A) et se note *I.*

La puissance électrique se note *P* et se mesure en *Watts* (W). Elle exprime la quantité de courant (I), transformée en chaleur ou en mouvement.

La puissance P est le produit de la tension U et de l'intensité I.

$$P = U \cdot I$$

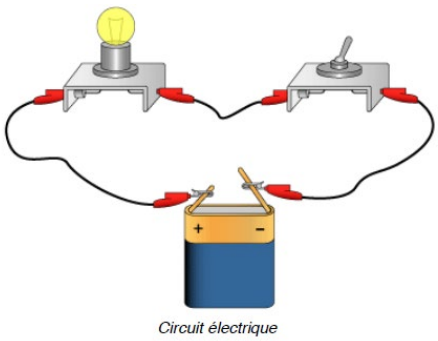
. Le circuit électrique

Un circuit électrique est une association de dipôles (générateur, résistances, ampoules, …) reliés par des conducteurs.

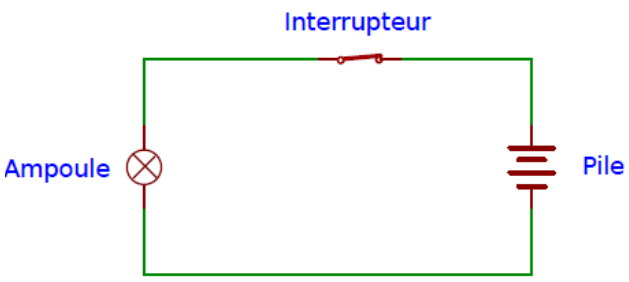
Pour qu'un courant circule, il faut que le circuit soit fermé et qu'il contienne un générateur (une pile par exemple). Il circule du pôle (+) au pôle (-) du générateur.

Une pile est constituée d'un milieu contenant de nombreux électrons en trop, et d'un second milieu en manque d'électrons. Quand on relie les deux pôles de la pile (le + et le -) avec un fil électrique (le conducteur), les électrons vont alors se déplacer du milieu riche en électrons vers le milieu pauvre.

Si on place une lampe électrique entre les deux, le passage des électrons va générer de la lumière.



Circuit électrique

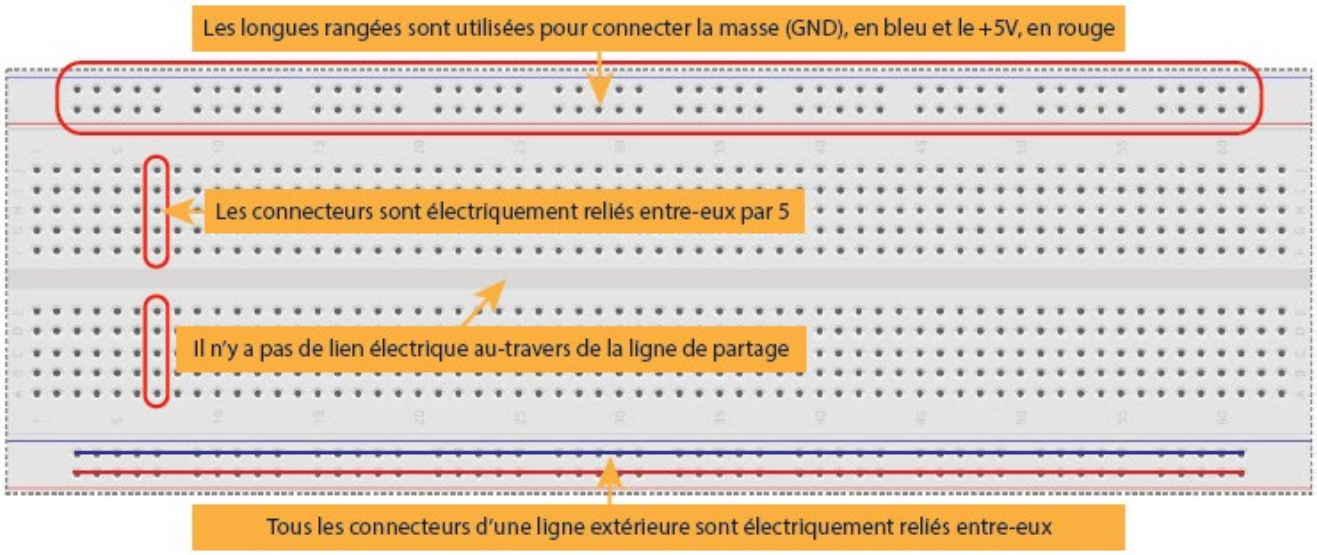


Dans le circuit ci-dessus, lorsque l'interrupteur est enclenché, on dit que le circuit est fermé. Les électrons vont alors se déplacer et l'énergie de ce déplacement pourra être exploitée pour allumer une lampe ou faire fonctionner un moteur, par exemple.
Lorsque l'interrupteur est déclenché, on dit que le circuit est ouvert. Le pôle positif n'étant alors plus relié au pôle négatif, les électrons ne peuvent plus se déplacer.

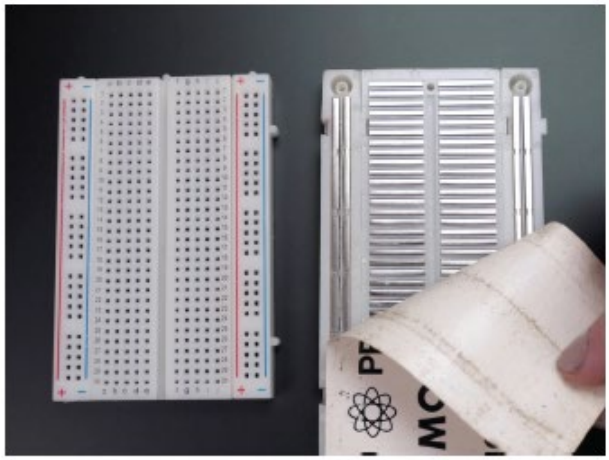
Dans les circuits électriques que nous allons étudier, L'Arduino servira d'alimentation électrique du circuit, comme une pile.

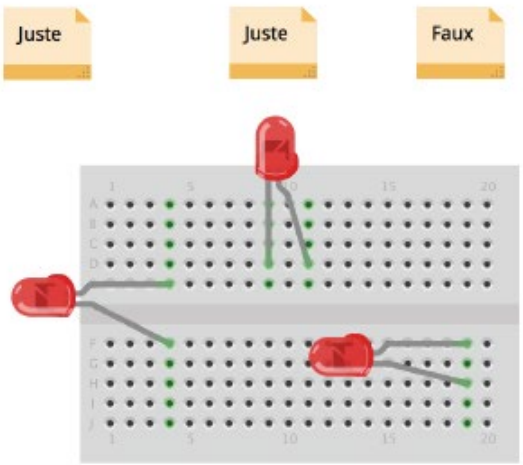
. La platine d'expérimentation

Une platine d'expérimentation (appelée breadboard) permet de réaliser des prototypes de montages électroniques sans soudure et donc de pouvoir réutiliser les composants.



Tous les connecteurs dans une rangée de 5 sont reliés entre eux. Donc si on branche deux éléments dans un groupe de cinq connecteurs, ils seront reliés entre eux. Il en est de même des alignements de connecteurs rouges (pour l'alimentation) et bleus (pour la terre).





Les composants doivent ainsi être placés à cheval sur des connecteurs qui n'ont pas de liens électriques entre eux.

. Les résistances

Une **résistance** est un composant électronique ou électrique dont la principale caractéristique est d'opposer une plus ou moins grande résistance (mesurée en ohms : Ω) à la circulation du courant électrique.

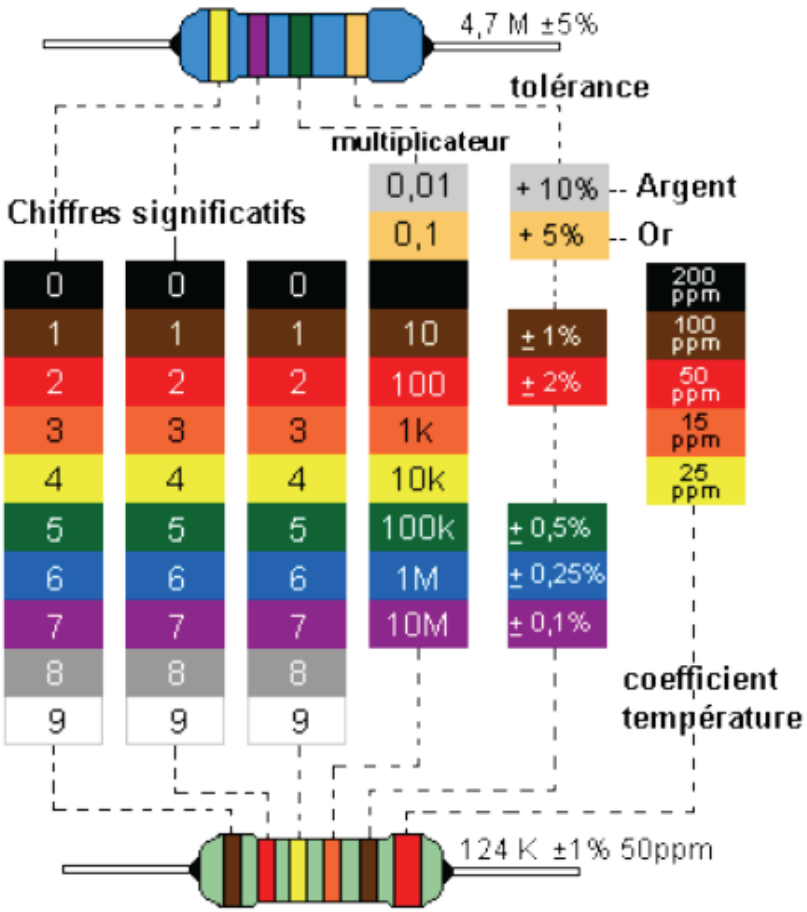
Ainsi, pour une tension fixe, plus la résistance est faible, plus le courant la traversant est fort. Cette proportion est vérifiée par la loi d'Ohm :

$$U = R \times I \text{ et donc } R = U/I \text{ et } I = U/R$$

Une résistance est un milieu peu conducteur. Les électrons peinent à s'y déplacer. Leur énergie se dissipe alors en général sous forme de chaleur. C'est ce principe utilisé pour les bouilloires électriques ou les ampoules à filaments.

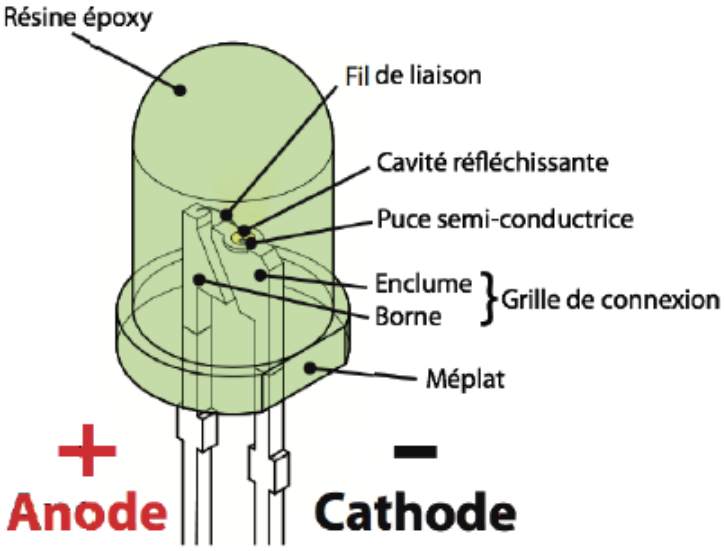
La résistance est schématisée de cette manière :

La valeur de la résistance se mesure en Ohms (Ω). La valeur d'une résistance est déterminée par ses bandes de couleurs :

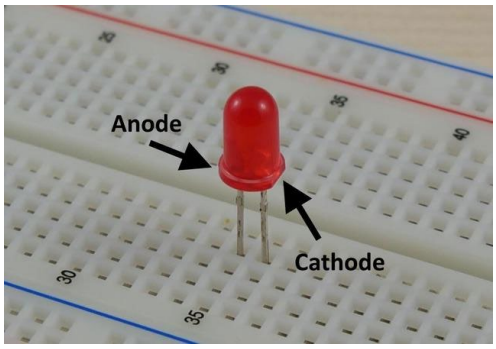


. Les diodes

La diode électroluminescente, aussi appelée DEL (ou LED en anglais), a la particularité de ne laisser passer le courant électrique que dans un sens.
Le courant électrique ne peut traverser la diode que dans le sens de *l'anode* vers la *cathode*.



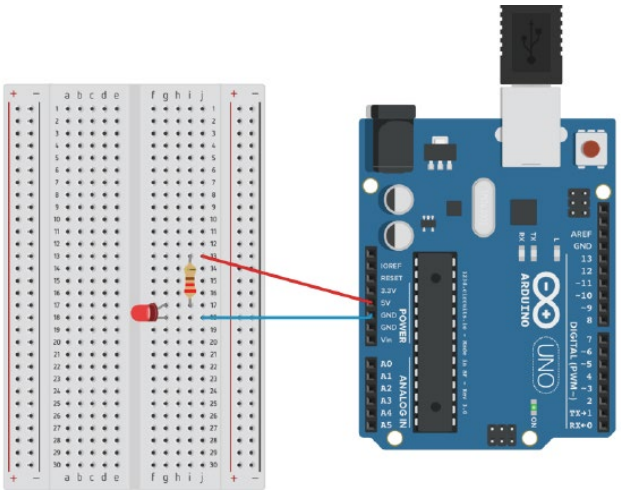
On reconnaît l'anode, car il s'agit de la broche la plus longue. Lorsque les deux broches sont de même longueur, on peut distinguer l'anode de la cathode, par un méplat du côté de cette dernière.



Le symbole de la DEL est le suivant :

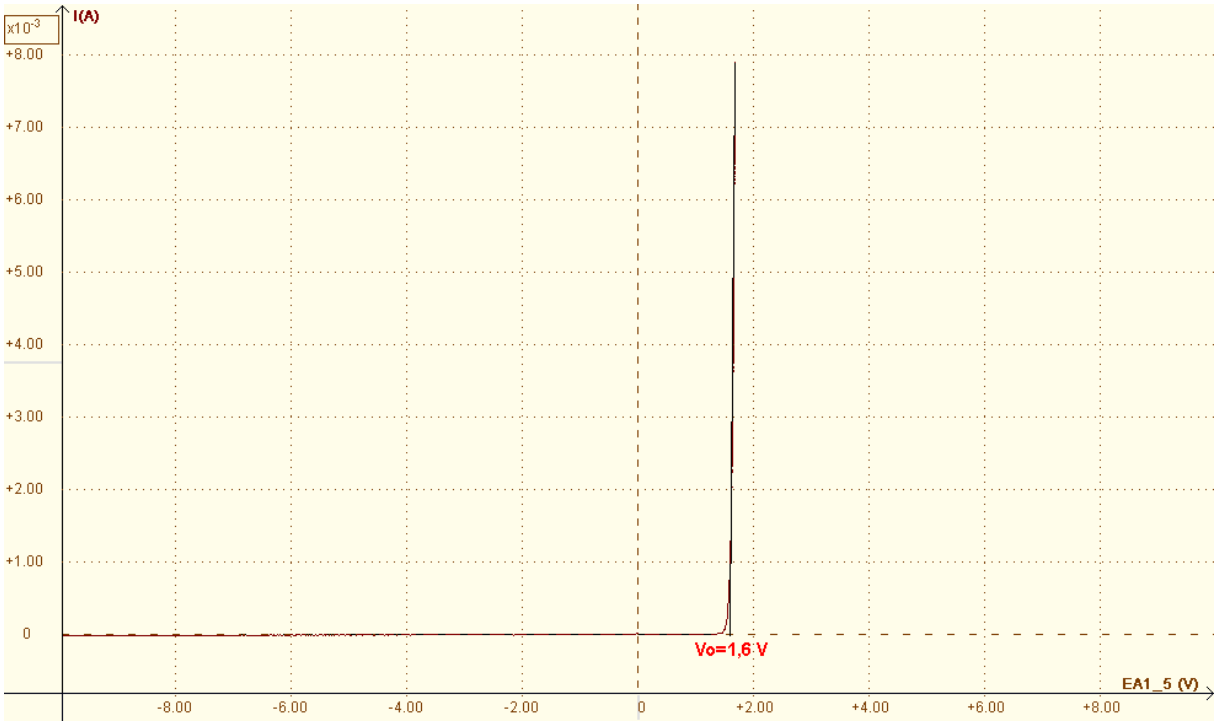
En utilisant divers matériaux semi-conducteurs, on fait varier la couleur de la lumière émise par la DEL et Il existe une grande variété de formes de DELs.

Attention : le courant produit par l'Arduino est trop important pour y brancher directement une DEL dessus.
L'utilisation d'une résistance est obligatoire, pour ne pas griller la DEL, comme dans le circuit ci-dessous :



Mais si la valeur de la résistance est trop grande, la DEL ne s'allumera pas et au contraire, si la valeur de la résistance n'est pas suffisante, la DEL grillera.

Si on trace la caractéristique d'une DEL, c'est-à-dire, le graphe représentant l'intensité I traversant la DEL en fonction de la tension U à ses bornes : **I=f(U)**, on remarque qu'en dessous d'une certaine valeur de tension, le courant ne passe pas (la DEL ne s'allume pas).
On dit que la DEL est bloquante en dessous d'une tension seuil et passante au-dessus.



Caractéristique courant / tension d'une DEL rouge

Pour que la DEL s'allume, il faut donc que la tension appliquée à ses bornes soit supérieure à sa tension seuil.
La tension seuil de la DEL, U_{LED}, dépend de sa couleur. Pour connaitre sa valeur, il suffit de consulter sa fiche technique (Datasheet) donné par le constructeur de la DEL.

En général, on aura :

. DEL Blanche : U_{LED} = 3,4 à 3,8 V

. DEL Bleue : U_{LED} = 3,2 à 3,6 V

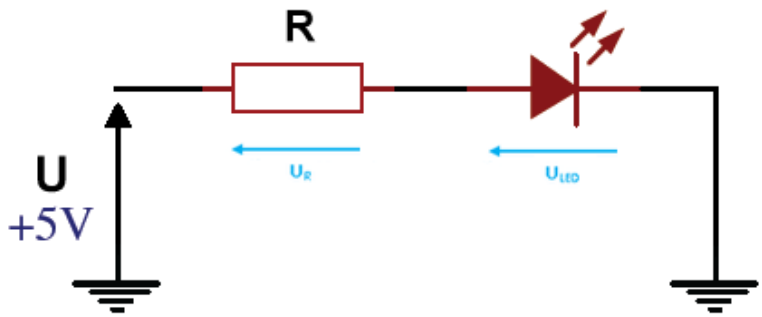
. DEL Verte : U_{LED} = 2,2 V

. DEL Rouge : U_{LED} = 1,6V à 2 V

. DEL Jaune : U_{LED} = 2,1 V

Et pour qu'une DEL fonctionne dans des conditions optimales, les constructeurs préconisent généralement un courant de **20 mA** (0,02 A) maximum.

Pour calculer la valeur de la résistance adéquate qu'il faut utiliser, on doit appliquer la loi d'Ohm au circuit suivant :



D'après la loi d'additivité des tensions dans un circuit en série, $\mathbf{U = U_R + U_{LED}}$

Donc : $\mathbf{U = Ri + U_{LED}}$

Et : $$R = \frac{U - U_{LED}}{I}$$

Par exemple :

Pour une LED rouge, U_{LED} = 1,6 V, avec une alimentation de 5 V et une Intensité de 20mA maximum, la résistance minimale est :

R = (5 - 1,6) / 0,02 = 170 Ω

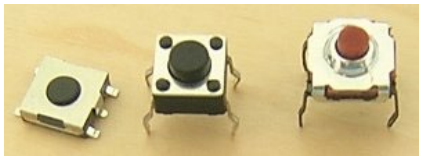
En prenant une résistance de **220 Ω**, nous sommes assurés de ne pas dépasser le courant maximal admissible par la DEL, et comme la DEL rouge est celle qui a une tension de seuil la plus basse, la résistance de 220 Ω peut être utilisée avec les autres DELs (l'intensité dans le circuit sera obligatoirement inférieure à 20 mA).

A retenir :

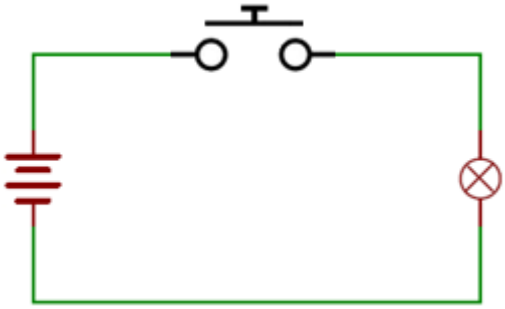
Si on utilise une DEL dans un circuit alimenté par un Arduino, il est impératif de placer une résistance, par défaut de 220 Ω, en série avec elle. Cette résistance est appelée, résistance de protection, de façon à limiter le courant qui la traverse à 20 mA maximum.

. Le bouton poussoir normalement ouvert

Un bouton poussoir normalement ouvert est un dispositif mécanique doté de 4 broches et d'une lamelle métallique qui met en contact toutes les broches lorsqu'on appui sur la tête du bouton. Un ressort de rappel ramène la tête du bouton lorsqu'il est relâché.



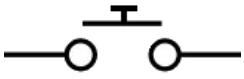
Un bouton poussoir normalement ouvert n'est jamais qu'un fil qui est connecté au circuit électrique ou non selon sa position :



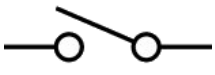
. **Relâché** : le courant ne passe pas dans le circuit électrique, le circuit est "ouvert ".
. **Appuyé** : le courant passe, on dit que le circuit est fermé.

Les mêmes effets peuvent être produits avec un interrupteur sauf que circuit reste fermé ou ouvert tant que la position de l'interrupteur n'a pas été changé.

Le bouton poussoir et l'interrupteur ne possèdent pas le même symbole pour les schémas électroniques :

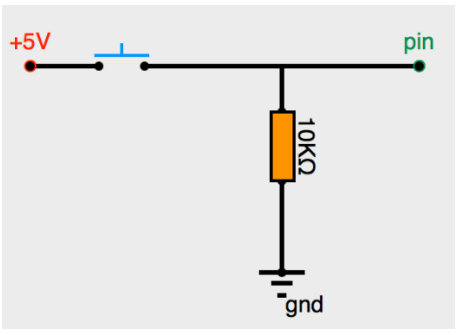


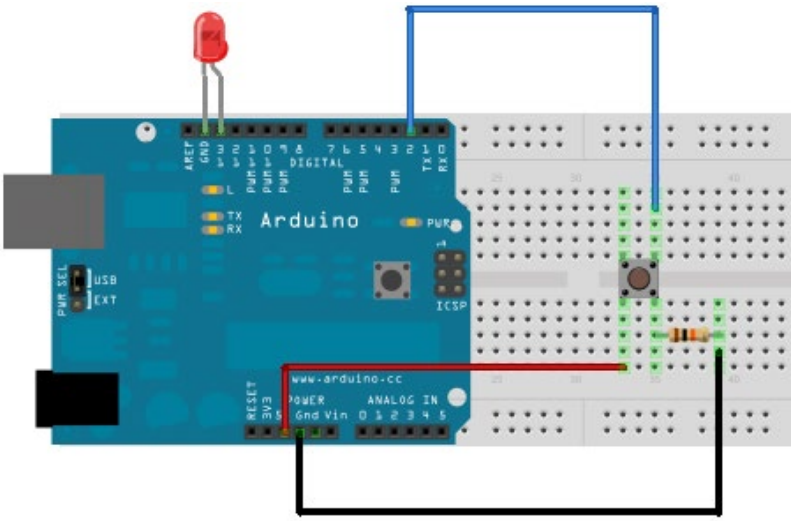
Bouton Poussoir NO



Interrupteur

Avec un Arduino, il est principalement utilisé pour envoyer une "impulsion de commande" avec ce circuit électrique :





Quand le bouton poussoir est appuyé, le potentiel de sa broche connectée à la broche 2 de l'Arduino passe à 5 V (le circuit est fermé) et quand il est relâché, celui-ci passe à 0 V (le circuit est ouvert).

On pourra alors demander à l'Arduino, d'allumer ou d'éteindre la DEL connectée sur sa broche 13 en fonction de la valeur du potentiel de la broche 2 de l'Arduino.

Avec le bouton poussoir, nous allons donner, à l'Arduino, l'ordre d'effectuer une action. D'où le terme "impulsion de commande"

<u>Attention</u> :

Il est impératif d'utiliser une résistance de 10 kΩ en série avec le bouton poussoir dans un montage "impulsion de commande".

Ainsi, le courant dans le circuit est très faible (I = U/R = 0,5 mA) quand le circuit est fermé (bouton poussoir appuyé), et il n'y a pas de risque pour l'Arduino.

. Les entrées et sorties numériques de l'Arduino

Il y a 14 entrées/sorties numériques notées de 0 à 13 sur l'Arduino Uno. Elles sont situées sur la grande rangée du haut de la carte. Ces connecteurs (ou broches) sont numériques car le signal sur ces broches ne connaît que deux états (niveau logique) : haut ou bas (1 ou 0). Électriquement, cela se traduit, respectivement, par une tension de 5 V ou 0 V.

Ces broches sont configurées en entrées ou sorties numériques par programmation :

. Configurées en sortie, elles ne peuvent délivrer que des niveaux logiques bas (0 V) et des niveaux logiques haut (5 V).

. Configurées en entrée, elles ne peuvent recevoir que des niveaux logiques bas (0 V) et niveaux logiques haut (5 V).

Attention :

. Les tensions appliquées sur les broches d'entrées/sorties numériques doivent être comprise entre 0 et 5 V. En dehors de ces limites, le microcontrôleur sera endommagé.

. Pour une broche configurée en entrée, toute tension inférieure à **0,3 x Vcc**, Vcc étant égale à 5V, soit **1,5 V**, sera comprise comme un niveau logique bas (0 V) et toute tension supérieure à **0,6 x Vcc**, soit **3 V**, sera comprise comme un niveau logique haut (5 V).

Entre les deux, c'est incertain. L'Arduino renverra de toutes façons un 0 ou un 1 mais de manière plus ou moins aléatoire.

. Pour une broche configurée en sortie, il est préférable de limiter l'intensité du courant dans le circuit électrique à **20 mA** et absolument nécessaire de ne pas dépasser **40 mA** sous peine de destruction de la sortie. On veillera aussi à ne pas dépasser une intensité totale de **200 mA** dans les circuits électriques reliés à ces broches.

. Il est déconseillé d'utiliser les broches 0 (RX) et 1 (TX) qui sont initialement prévues pour la communication série (broche 1 pour l'émission et broche 0 pour la réception des données) avec certains modules externes (par exemple, le module Bluetooth HC-05).

. La programmation

1. Le langage Arduino

Les programmes Arduino peuvent être divisés en trois parties principales : la structure, les valeurs (variables et constantes) et les fonctions. Le langage Arduino est basé sur les langages C/C++.

En ce qui concerne la structure du programme Arduino, ces deux fonctions sont obligatoires dans tout programme en langage Arduino :

setup()

La fonction setup() est appelée au démarrage du programme. Cette fonction est utilisée pour initialiser les variables, le sens des broches, les librairies utilisées. La fonction setup n'est exécutée qu'une seule fois, après chaque mise sous tension ou reset (réinitialisation) de la carte Arduino.

Syntaxe de la fonction :

```
void setup()
{
}
```

loop()

Après avoir créé une fonction setup(), qui initialise et fixe les valeurs de démarrage du programme, la fonction loop () (boucle en anglais) fait exactement ce que son nom suggère et s'exécute en boucle sans fin, permettant au programme de s'exécuter et de répondre. On utilise cette fonction pour contrôler activement la carte Arduino.

Syntaxe de la fonction :

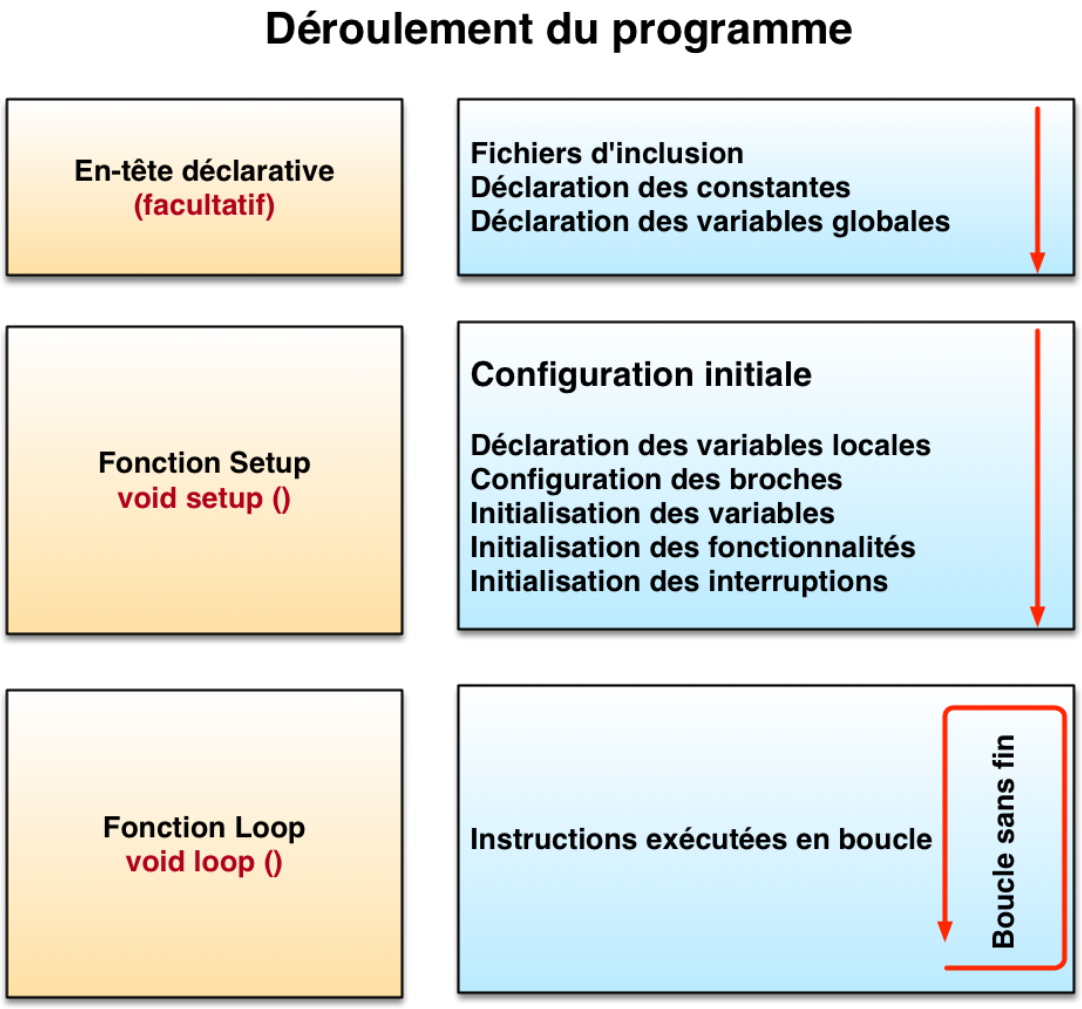
```
void loop()
{
}
```

Le programme se déroule donc de la façon suivante :

1. Prise en compte des instructions de la partie déclarative (Inclusion des librairies, déclaration des constantes et variables)

2. Exécution de la partie configuration (fonction setup( )),

3. Exécution de la boucle sans fin (fonction loop ( )): le code compris dans la boucle sans fin est exécuté indéfiniment.



. Gestion des entrées et sorties numériques :

- Les broches numériques de l'Arduino sont configurées en entrée ou en sortie à l'aide de la fonction :

pinMode()

. Syntaxe :

pinMode(broche, mode)

.Paramètres :

broche: le numéro de la broche de la carte Arduino dont le mode de fonctionnement (entrée ou sortie) doit être défini.

mode: soit INPUT (entrée en anglais) ou OUTPUT (sortie en anglais)

- Pour modifier l'état logique d'une sortie numérique, on utilise la fonction :

digitalWrite()

Met un niveau logique HIGH (HAUT en anglais) ou LOW (BAS en anglais) sur une broche numérique.

Sa tension est mise à la valeur correspondante : 5V pour le niveau HAUT, 0V (masse) pour le niveau BAS.

. Syntaxe :

digitalWrite(broche, valeur)

. Paramètres :

broche: le numéro de la broche de la carte Arduino

valeur : HIGH ou LOW (1 ou 0)

- Pour lire l'état logique d'une entrée numérique, on utilise la fonction :

digitalRead()

Lit l'état (= le niveau logique) d'une broche précise en entrée numérique, et renvoie la valeur HIGH (HAUT en anglais) ou LOW (BAS en anglais).

. Syntaxe :

digitalRead(broche)

. Paramètre :

broche: le numéro de la broche de la carte Arduino

. Valeur retournée :

Renvoie la valeur HIGH (1) ou LOW (0)

2. *le protocole de communication "Firmata Standard"*

Avec le protocole de communication **"Firmata Standard"**, la lecture et l'écriture sur les entrées et sorties de l'Arduino se font directement à partir d'un programme en Python.

Pour que le programme Python "donneur d'ordres" fonctionne, le chargement, dans la mémoire de l'Arduino, du code " **Firmata Standard** " doit être fait avant son lancement, à l'aide du logiciel "**IDE ARDUINO**".

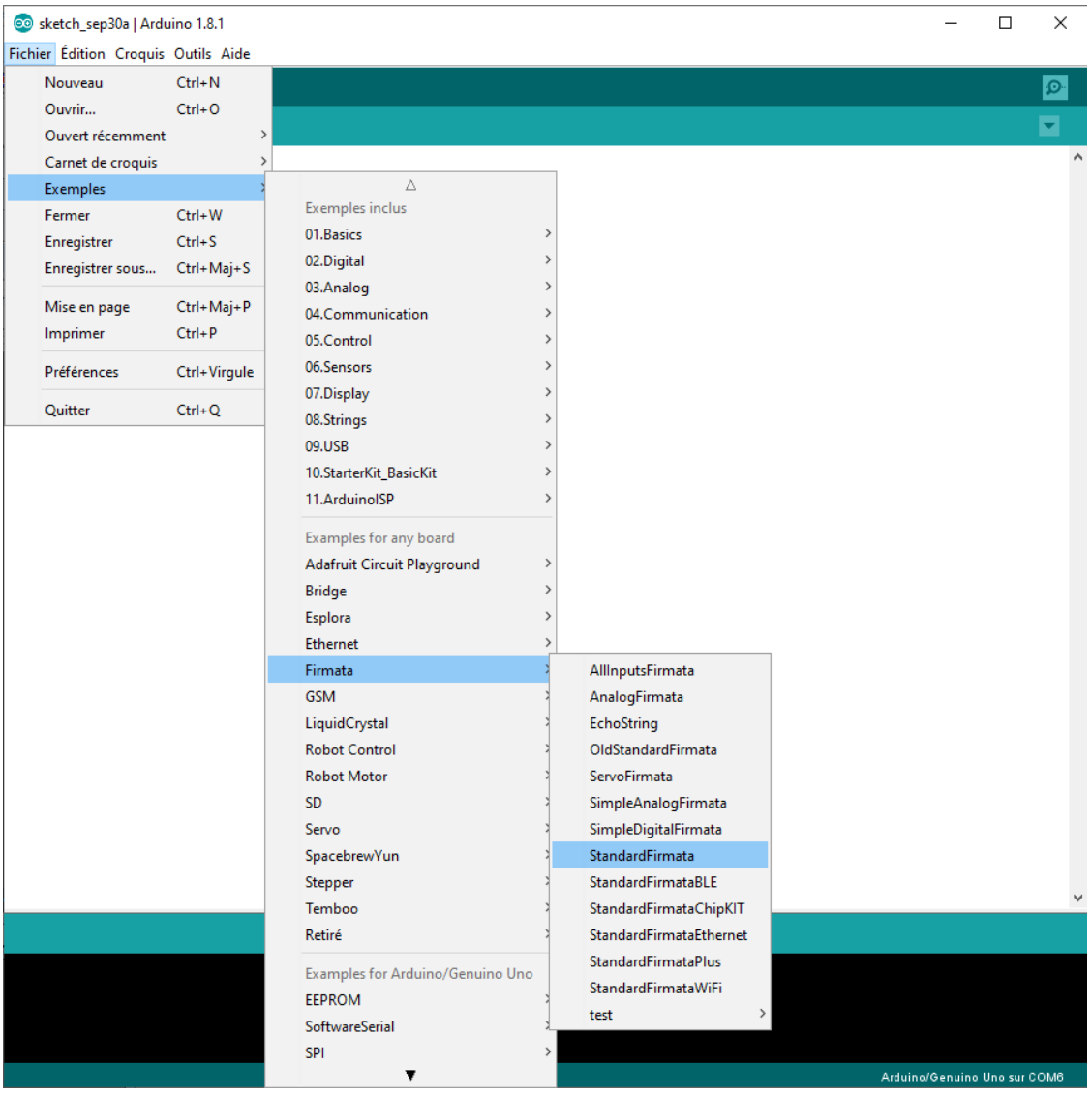
. Chargement du code "Firmata Standard" dans la mémoire de l'Arduino :

– Brancher l'Arduino via un port USB,

- Afin de charger la librairie "**Firmata standard**" sur l'ARDUINO, il faut lancer le logiciel "**IDE ARDUINO**", puis sélectionner :

Fichier > Exemples > Firmata > Standard Firmata,

- puis cliquer sur "**téléverser**".



. <u>Installation de la bibliothèque PyFirmata dans Python:</u>

Pour faire fonctionner, un programme en Python qui contrôle l'Arduino via le protocole de communication Firmata, Python doit disposer de la bibliothèque "**PyFirmata**". Celle-ci peut être installée via **"pip"**, à l'aide de la ligne de commande :

pip install pyfirmata

Pour utiliser la bibliothèque **"pyfirmata"** dans un programme python, il faut importer le module "**pyfirmata**", à l'aide de l'instruction : **import pyfirmata**

La connexion avec le microcontrôleur, via le port série, est réalisée avec la méthode **"Arduino"** du module **"pyfirmata"** en précisant le port COM sur lequel l'Arduino est connecté :

board = pyfirmata.Arduino(Port COM)

Une fois la connexion établie, il est possible d'interroger ou de modifier les entrées et sorties numériques ou analogiques de l'Arduino.

- <u>Gestion des sorties numériques</u>

Pour modifier l'état d'une sortie numérique (l'équivalent d'un "digitalWrite" en langage Arduino), la syntaxe est la suivante :

board.digital[numéro_de_pin].write(0_ou_1)

où :

- **"board"** est l'objet créé lors de l'appel de la méthode "Arduino" du module "pyfirmata",
- "**numéro_de_pin**" est le numéro de la sortie du microcontrôleur dont on veut modifier l'état,
- "**0_ou_1**" est le niveau logique souhaité.

Le plus simple est de créer une fonction qui sera appelée dans nos programmes en Python pour modifier l'état d'une sortie numérique :

```
def DigitalWrite(board,pin,val):
  board.digital[pin].write(val)
```

Ainsi, l'instruction pour mettre la sortie numérique "9", de l'objet "board", à l'état haut est :

DigitalWrite(board, 9, 1)

Remarque :
Contrairement à un programme Arduino, avec Pyfirmata, il n'est pas nécessaire de déclarer au préalable une sortie numérique avant de l'utiliser.

- Gestion des entrées numériques :

Pour lire l'état logique d'une broche numérique (par exemple, la broche N°5), il faut la déclarer au préalable en entrée avec la commande suivante :

pin5 = board.get_pin('d:5:i')

La syntaxe est "**d**" pour digital, "**5**" est le numéro de la broche, "**i**" pour input et **"board"** est l'objet créé lors de l'appel de la méthode "Arduino" du module "pyfirmata".

On peut définir une fonction déclarant plus facilement une broche en entrée numérique :

```
def DigitalInput(board,pin):
    DigitalInputPin=board.get_pin('d:'+ str(pin) +':i')
    return DigitalInputPin
```

La syntaxe pour déclarer la broche N°5 en entrée numérique est alors plus simple :

pin5 = DigitalInput(board,5)

Ensuite on pourra lire l'état logique de la broche au moyen de cette instruction :

valeur = pin5.read()

qui retourne "**1**" lorsque l'entrée est à **5 V**, et "**0**" lorsqu'elle est à **0 V**.

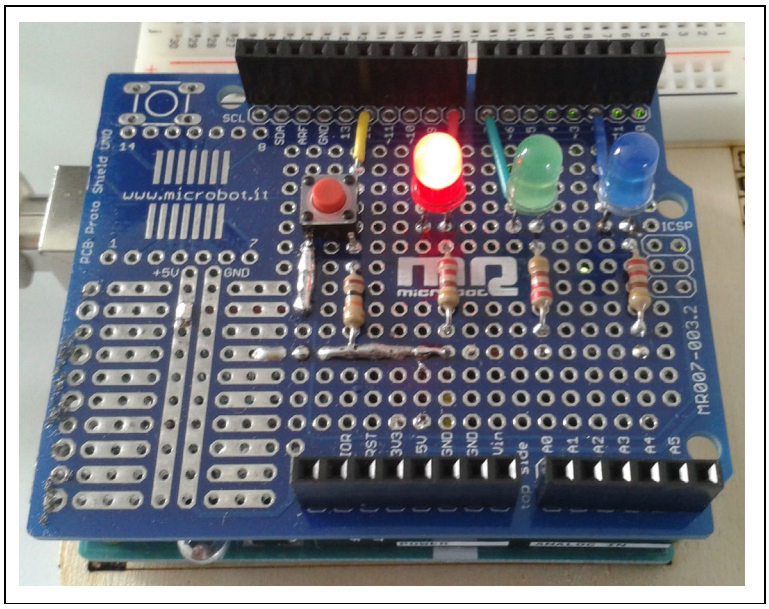
Attention :

. Les données lues transitent bien-sûr par la liaison série, et pour éviter qu'un trop grand nombre de mesures n'encombrent la communication série entre l'Arduino et l'ordinateur, Il faut aussi utiliser un itérateur prévu dans pyFirmata :

```
it = pyfirmata.util.Iterator(board)
it.start()
```

. L'itérateur doit est lancé après la connexion à l'Arduino.

- Activité 1 : Faire clignoter une DEL



Comme première activité, nous allons faire clignoter une DEL (rouge, verte, ou bleue), préalablement choisie, connectée sur une des broches 8, 7, ou 2, comme sur le schéma de câblage ci-dessus.

Cette activité a pour but l'apprentissage de l'utilisation des sorties digitales de l'Arduino qui ne peuvent prendre que 2 valeurs : **0 (niveau bas)** ou **1 (niveau haut)**, soit électriquement : **0 V ou +5 V**.
Donc, pour allumer la DEL, la broche de l'Arduino sur laquelle celle-ci est connectée, doit être au niveau haut (**+5V**) et pour l'éteindre, elle doit être au niveau bas (**0 V**).

Pour réaliser cette activité, on va demander à l'Arduino d'allumer une des 3 DELs (**donc d'appliquer un niveau haut sur la broche de la DEL**) pendant ½ seconde, puis de l'éteindre (**donc d'appliquer un niveau bas sur la broche de la DEL**) pendant une ½ seconde, puis à nouveau de l'allumer pendant ½ seconde et cela indéfiniment. De cette façon, on verra la DEL choisie clignoter.

Le code en Python ou en langage Arduino pourra être modifié pour voir l'influence des variables (durée d'allumage, d'extinction, numéro de la broche de la DEL).

. Programme en Python (Projet1\Activity1\PY\Activity1.py)

```
# Importations des librairies et définition des fonctions

from PyFirmataDef import *
from ConnectToArduino import *
import time


# Déclaration des constantes et variables

PinLED = 8


# Connexion à l'Arduino

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)

print("Connexion à l'arduino établie - Appuyez sur Ctrl-C pour quitter.\n")

# Boucle principale du programme

while True:
    try:
        DigitalWrite(board,PinLED,1)
        time.sleep(0.5)
        DigitalWrite(board,PinLED,0)
        time.sleep(0.5)

    except KeyboardInterrupt:
        DigitalWrite(board,PinLED,0)
        board.exit()
        sys.exit(0)
```

Déroulement du programme :

- Importation des librairies et définition de fonctions :

  . Le module **”ConnectToArduino.py”,** contenant les fonctions de connexion à l’Arduino via le protocole **”Firmata standard”**,

  . Le module "**PyFirmataDef.Py**" regroupant toutes les fonctions utiles à l’utilisation de "**PyFirmata**" (fonction de déclaration des entrées et sorties, d’itérateur, d’écritures…)

  . La bibliothèque **"time"** pour la gestion des temps de pause.

- Déclaration des constantes et variables :

  . **PinLED = 8** (constante correspondant au n° de la broche sur laquelle la DEL rouge est connectée)

  . **PortComArduino** (port COM sur lequel l’Arduino est connecté)

- Connexion à l'Arduino :

. Détection du port COM, tentative d'ouverture du port COM sélectionné et connexion à l'Arduino :

PortComArduino = SelectPortCOM()

board = OpenPortCom(PortComArduino)

- Boucle principale du programme (boucle "while True") :

. Niveau haut sur la broche de la DEL pendant ½ seconde :

DigitalWrite(board,PinLED,1)

time.sleep(0.5)

. Niveau bas sur la broche de la DEL pendant ½ seconde :

DigitalWrite(board,PinLED,0)

time.sleep(0.5)

- Fin du programme en appuyant sur Ctrl-C :

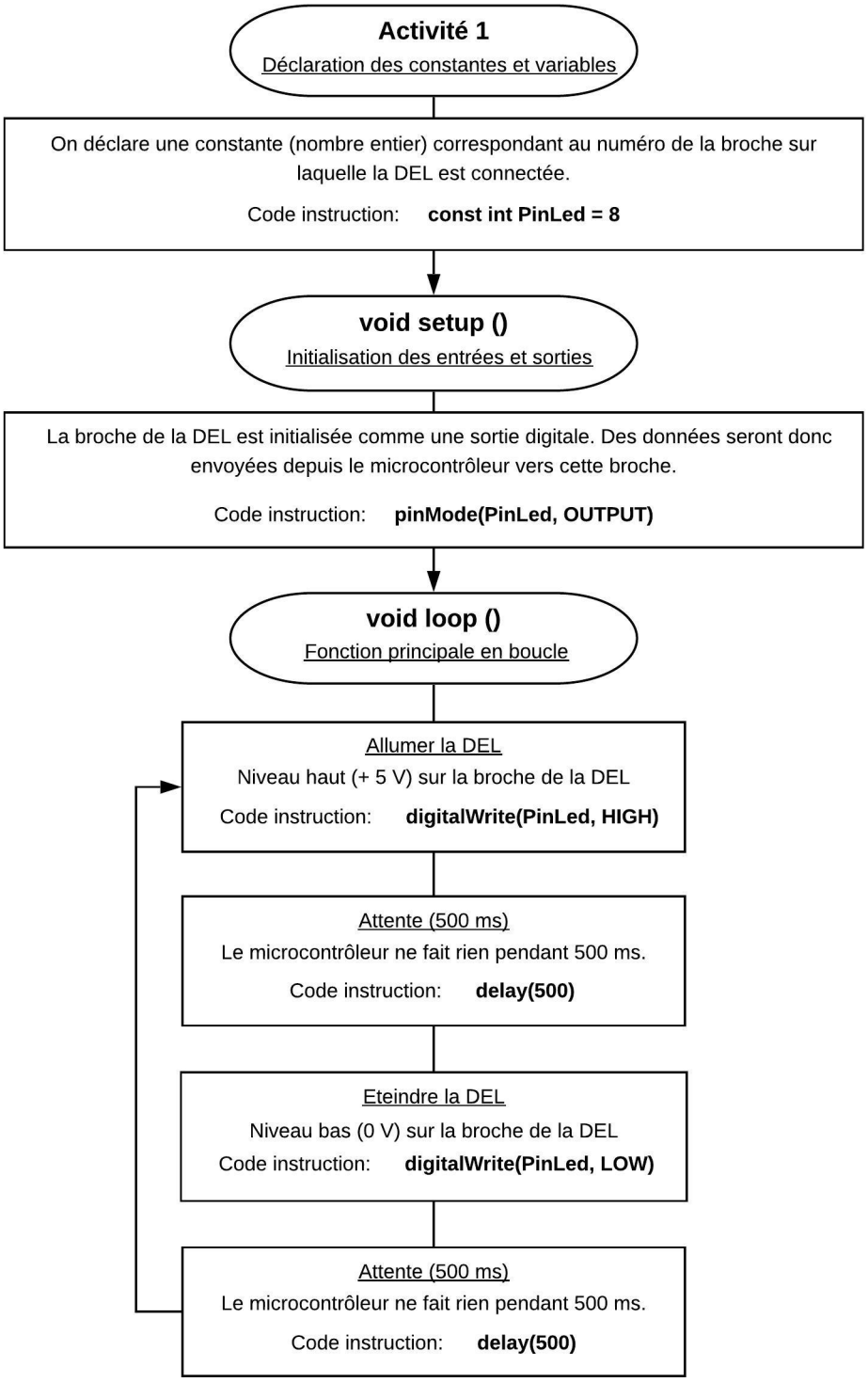
→ La DEL est éteinte et le port COM est fermé.

. Programme en langage Arduino (Projet1\Activity1\INO\Activity1.ino)

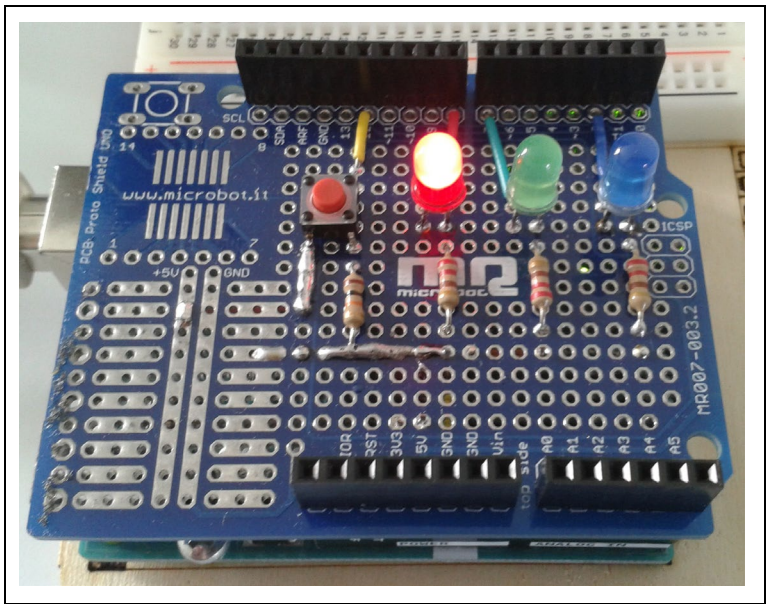
Activity1

```
// Déclaration des constantes et variables

const int PinLed = 8;


// Initialisation des entrées et sorties

void setup()
{
   pinMode(PinLed, OUTPUT);
}


// Fonction principale en boucle

void loop()
{
  digitalWrite(PinLed, HIGH);
  delay(500);
  digitalWrite(PinLed, LOW);
  delay(500);
}
```

Déroulement du programme :



- Activité 2 : Allumer une DEL avec un bouton-poussoir



Dans cette activité, la DEL, préalablement choisie, s'allume en appuyant sur le bouton poussoir et s'éteint si on le relâche. L'objectif est de se familiariser avec les entrées numériques de l'Arduino.

En effet, en appuyant sur le bouton poussoir, une tension de + 5V est appliquée sur la broche sur laquelle celui-ci est connecté. La broche est alors à un niveau haut. Si on relâche le bouton poussoir, le circuit électrique est ouvert, la tension sur la broche du bouton poussoir est alors de 0 V et passe à un niveau bas.

Si on demande à l'Arduino d'interroger l'état logique (**niveau haut ou bas**) de la broche du bouton poussoir qui a été déclaré comme une entrée numérique, on peut savoir si celui-ci est appuyé ou pas et donc lui donner l'ordre d'allumer ou d'éteindre la DEL.

Le code pourra être modifié pour voir l'influence des variables (numéro de la broche de la DEL).

. Programme en Python (**Projet1\Activity2\PY\Activity2.py**)

```
# Importations des librairies et définition des fonctions

from PyFirmataDef import *
from ConnectToArduino import *
import time

# Déclaration des constantes et variables

PinLED = 8
PinButton = 12
ValButton = 0


# Connexion à l'Arduino

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)


# Connexion à l'Arduino réussie - Déclaration des entrées et sorties

DigitalInputPin = DigitalInput(board, PinButton)
ArduinoIterateur = Iterateur(board)
time.sleep (0.5)

print("Connexion à l'arduino établie - Appuyez sur Ctrl-C pour quitter")


# Boucle principale du programme

while True:
    try:
        ValButton = DigitalInputPin.read()
        if ValButton == 1:
            DigitalWrite(board,PinLED,1)
        else:
            DigitalWrite(board,PinLED,0)

    except KeyboardInterrupt:
        DigitalWrite(board,PinLED,0)
        board.exit()
        sys.exit(0)
```

Déroulement du programme :

- Importation des librairies et définition de fonctions :

. Le module **”ConnectToArduino.py”,** contenant les fonctions de connexion à l’Arduino via le protocole **”Firmata standard”**,

. Le module "**PyFirmataDef.Py**" regroupant toutes les fonctions utiles à l’utilisation de "**PyFirmata**" (fonction de déclaration des entrées et sorties, d’itérateur, d’écritures…)

. La bibliothèque **"time"** pour la gestion des temps de pause.

- Déclaration des constantes et variables :

. **PinLED = 8** (constante correspondant au n° de la broche sur laquelle la DEL rouge est connectée)

. **PinButton = 12** (cst correspondant au n° de la broche sur laquelle le bouton poussoir est connecté)

. **ValButton = 0** (variable pour stocker la valeur de l'état logique de la broche du bouton poussoir)

. **PortComArduino** (port COM sur lequel l'Arduino est connecté)

- Connexion à l'Arduino :

. Détection du port COM, tentative d'ouverture du port COM sélectionné et connexion à l'Arduino :

PortComArduino = SelectPortCOM()

board = OpenPortCom(PortComArduino)

. Si la connexion à l'Arduino est réussie :

- Lancement de l'itérateur : **ArduinoIterateur = Iterateur(board)**,

- Déclaration de la broche du bouton poussoir en entrée digitale : **InputPin = DigitalInput(board, PinButton)**,

- Attente de 500 ms pour le lancement de l'itérateur

- Boucle principale du programme (boucle "while True") :

. Lecture de l'état logique de la broche du bouton poussoir :

ValButton = InputPin.read()

. La DEL est allumée ou éteinte suivant la valeur de ValButton :

```
if ValButton == 1:
    DigitalWrite(board,PinLED,1)
else:
    DigitalWrite(board,PinLED,0)
```

- Fin du programme en appuyant sur Ctrl-C :

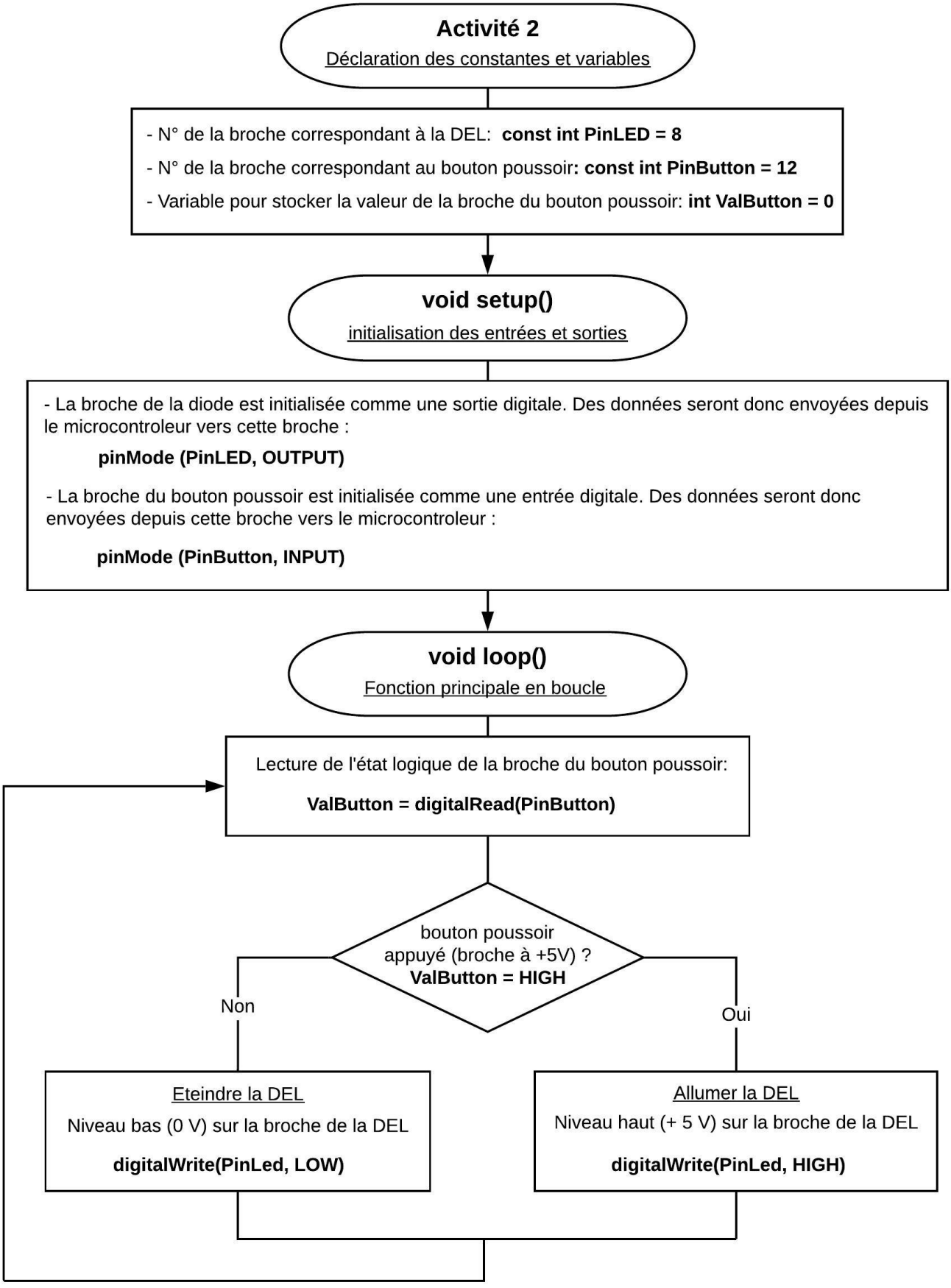
→ La DEL est éteinte et le port COM est fermé.

. Programme en langage Arduino (Projet1\Activity2\INO\Activity2.ino)

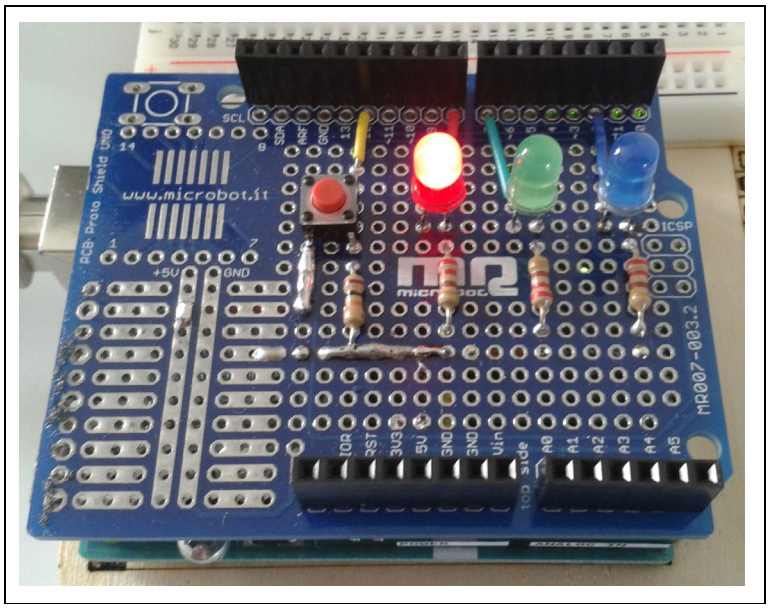
Activity2

```
// Déclaration des constantes et variables

const int PinLED = 8;
const int PinButton = 12;

int ValButton = 0;


// Initialisation des entrées et sorties

void setup() {

pinMode (PinLED, OUTPUT);
pinMode (PinButton, INPUT);

}


// Fonction principale en boucle

void loop() {

ValButton = digitalRead(PinButton);
if (ValButton == HIGH) {
  digitalWrite(PinLED, HIGH);
}
else {
digitalWrite(PinLED, LOW);
}
}
```

Déroulement du programme :



- Activité 3 : Allumer ou éteindre une DEL avec un bouton-poussoir



Dans cette activité, quand les DELs sont éteintes, si on appuie sur le bouton poussoir, une DEL préalablement choisie s'allume, mais contrairement à l'activité précédente la DEL ne s'éteint pas quand on relâche le bouton poussoir.

En effet, si une des DELs est allumée, elle s'éteint en appuyant à nouveau sur le bouton poussoir. Cette fois, le principe de fonctionnement du bouton poussoir est comme celui d'un interrupteur.

Pour réaliser cette activité, on va demander à l'Arduino d'interroger l'état logique (**niveau haut ou bas**) de la broche du bouton poussoir qui a été déclaré comme une entrée numérique. A l'aide de variables permettant de stocker les valeurs (actuelle et précédente) de cet état, l'Arduino pourra savoir quelle action effectuer (allumer ou éteindre la DEL) après l'appui sur le bouton poussoir.

Le code pourra être modifié pour voir l'influence des variables (numéro de la broche de la DEL).

. Programme en Python (**Projet1\Activity3\PY\Activity3.py**)

```
# Importations des librairies et définition des fonctions

from PyFirmataDef import *
from ConnectToArduino import *
import time


# Déclaration des constantes et variables

PinLED = 8
PinButton = 12
ValButton = 0
OldValButton = 0
State=0


# Connexion à l'Arduino

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)


# Connexion à l'Arduino réussie - Déclaration des entrées et sorties

DigitalInputPin = DigitalInput(board, PinButton)
ArduinoIterateur = Iterateur(board)
time.sleep (0.5)

print("Connexion à l'arduino établie - Appuyez sur Ctrl-C pour quitter")


# Boucle principale du programme

while True:
    try:
        ValButton = DigitalInputPin.read()
        time.sleep(0.01)
        if ValButton == 1 and OldValButton == 0:
            State = 1- State
        OldValButton = ValButton

        if State == 1:
            DigitalWrite(board,PinLED,1)
        else:
            DigitalWrite(board,PinLED,0)

    except KeyboardInterrupt:
        DigitalWrite(board,PinLED,0)
        board.exit()
        sys.exit(0)
```

Déroulement du programme :

- Importation des librairies et définition de fonctions :

. Le module **"ConnectToArduino.py",** contenant les fonctions de connexion à l'Arduino via le protocole **"Firmata standard"**,

. Le module "**PyFirmataDef.Py**" regroupant toutes les fonctions utiles à l'utilisation de "**PyFirmata**" (fonction de déclaration des entrées et sorties, d'itérateur, d'écritures…)

. La bibliothèque **"time"** pour la gestion des temps de pause.

- Déclaration des constantes et variables :

. **PinLED = 8** (constante correspondant au n° de la broche sur laquelle la DEL rouge est connectée)

. **PinButton = 12** (cst correspondant au n° de la broche sur laquelle le bouton poussoir est connecté)

. **ValButton = 0** (variable pour stocker la valeur de l'état logique de la broche du bouton poussoir)

. **OldValButton = 0** (variable pour stocker la valeur précédente de l'état logique de la broche du bouton poussoir)

. **State=0** (variable correspondant à l'action à effectuer)

. **PortComArduino** (port COM sur lequel l'Arduino est connecté)

- Connexion à l'Arduino :

. Détection du port COM, tentative d'ouverture du port COM sélectionné et connexion à l'Arduino :

PortComArduino = SelectPortCOM()

board = OpenPortCom(PortComArduino)

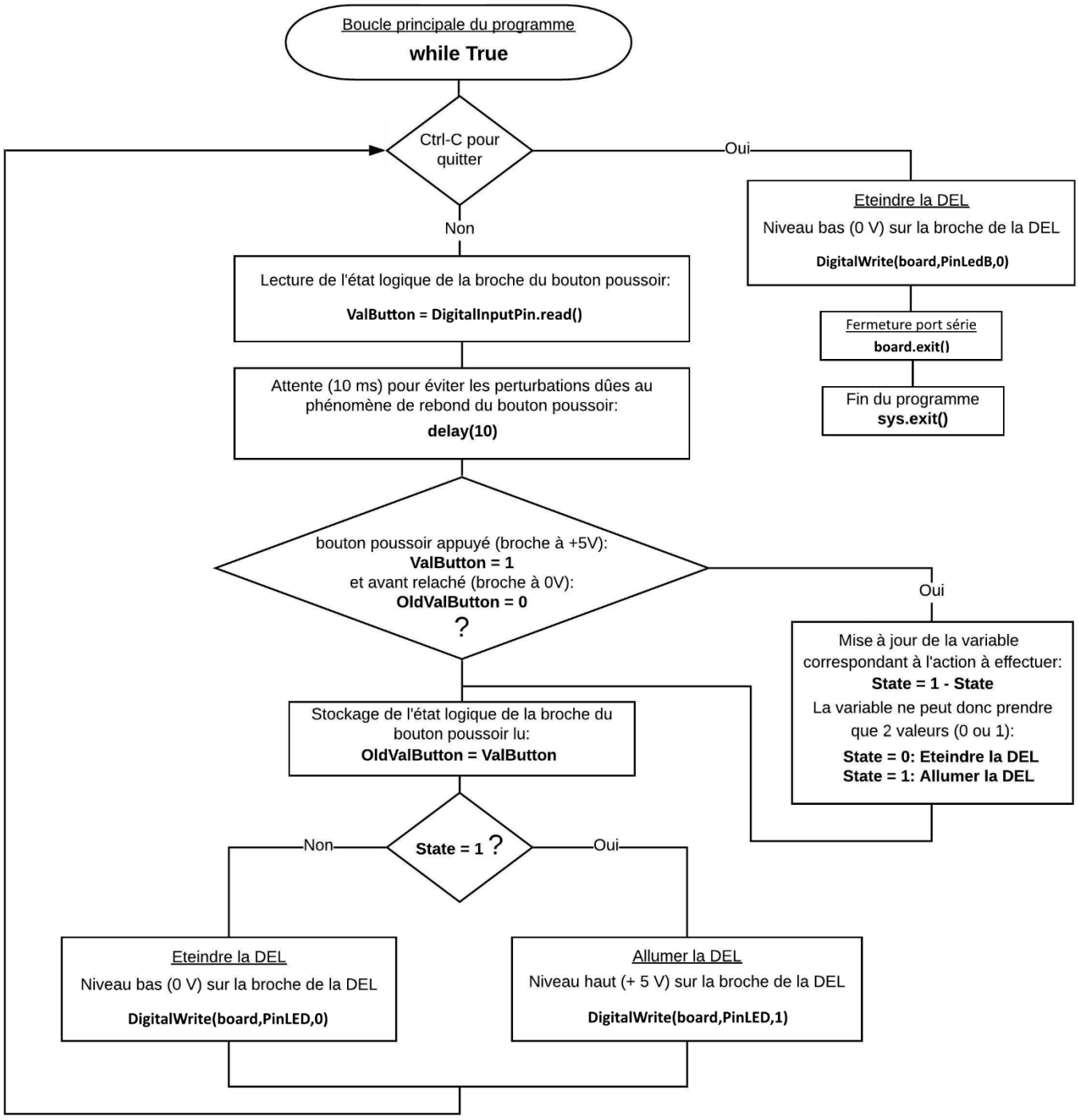
. Si la connexion à l'Arduino est réussie :

- Lancement de l'itérateur : **ArduinoIterateur = Iterateur(board)**,

- Déclaration de la broche du bouton poussoir en entrée digitale : **InputPin = DigitalInput(board, PinButton)**,

- Attente de 500 ms pour le lancement de l'itérateur

- Boucle principale du programme (boucle "while True") :

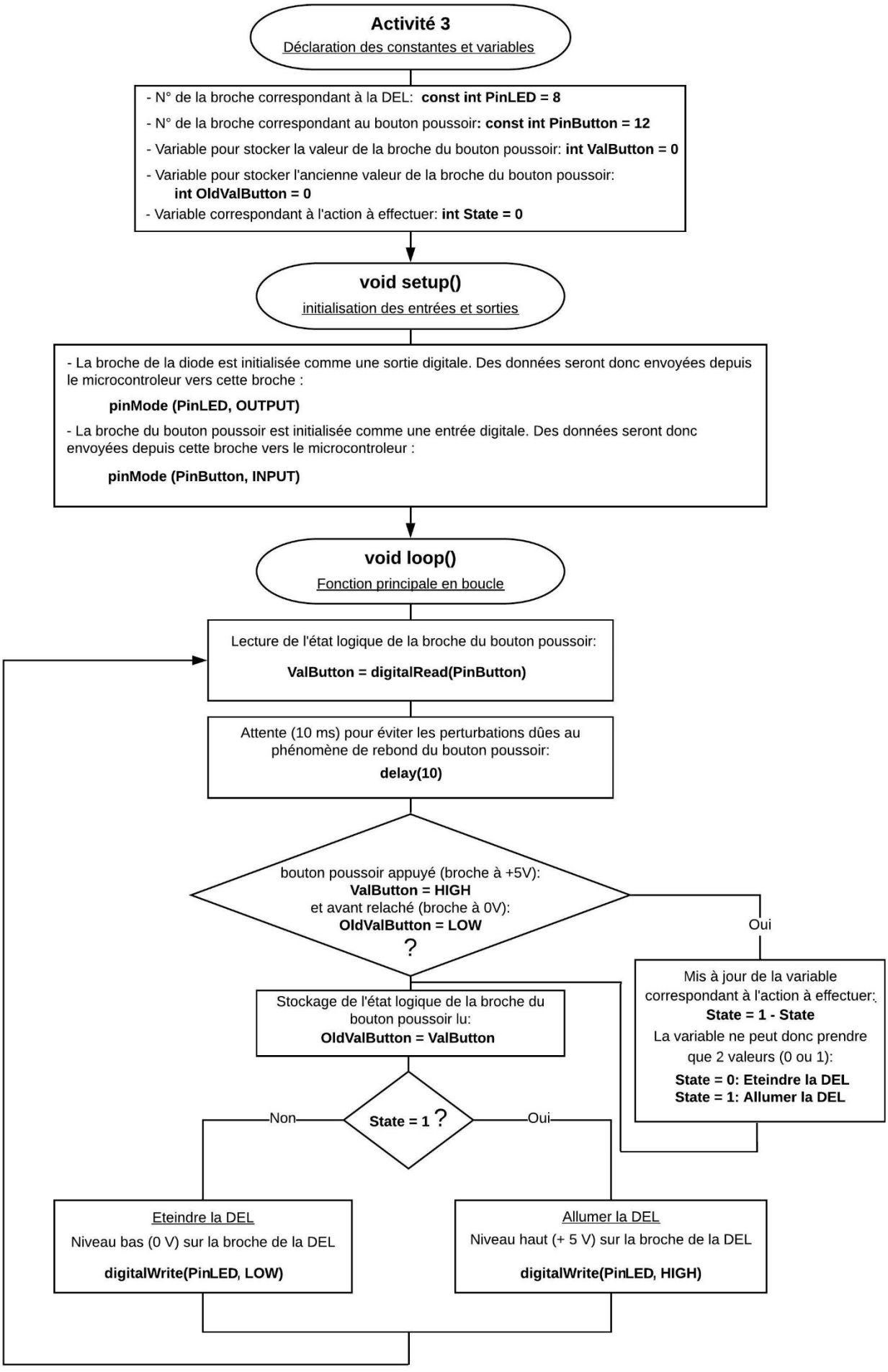

Boucle principale du programme
while True
Ctrl-C pour quitter
Oui
Non
Eteindre la DEL
Niveau bas (0 V) sur la broche de la DEL
DigitalWrite(board,PinLedB,0)
Fermeture port série
board.exit()
Fin du programme
sys.exit()
Lecture de l'état logique de la broche du bouton poussoir:
ValButton = DigitalInputPin.read()
Attente (10 ms) pour éviter les perturbations dûes au phénomène de rebond du bouton poussoir:
delay(10)
bouton poussoir appuyé (broche à +5V):
ValButton = 1
et avant relaché (broche à 0V):
OldValButton = 0
?
Oui
Mise à jour de la variable correspondant à l'action à effectuer:
State = 1 - State
La variable ne peut donc prendre que 2 valeurs (0 ou 1):
State = 0: Eteindre la DEL
State = 1: Allumer la DEL
Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Non
Oui
Eteindre la DEL
Niveau bas (0 V) sur la broche de la DEL
DigitalWrite(board,PinLED,0)
Allumer la DEL
Niveau haut (+ 5 V) sur la broche de la DEL
DigitalWrite(board,PinLED,1)

. Programme en langage Arduino (Projet1\Activity3\INO\Activity3.ino)

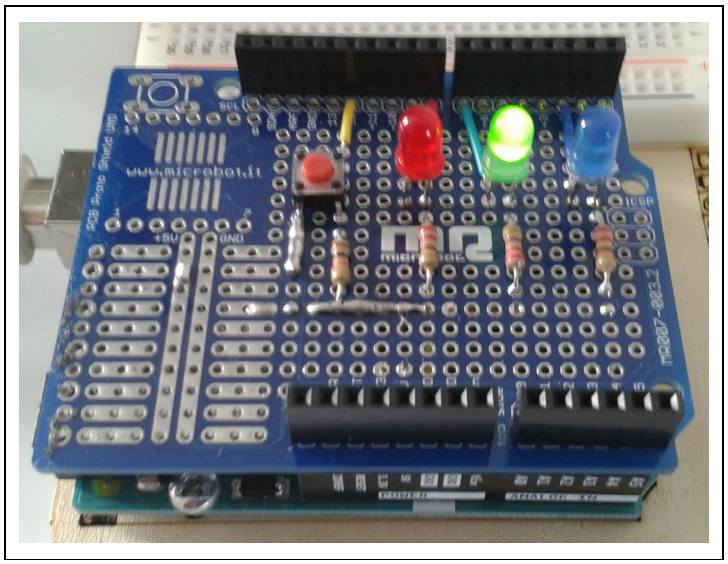
Activity3

```
// Déclaration des constantes et variables

const int PinLED = 8;
const int PinButton = 12;

int ValButton = 0;
int OldValButton = 0;
int State = 0;


// Initialisation des entrées et sorties

void setup() {
  pinMode (PinLED, OUTPUT);
  pinMode (PinButton, INPUT);
}


// Fonction principale en boucle

void loop() {
  ValButton = digitalRead(PinButton);
  delay(10);
  if ((ValButton == HIGH) && (OldValButton == LOW)) {
    State = 1 - State;
  }
  OldValButton = ValButton;
  if (State == 1) {
    digitalWrite(PinLED, HIGH);
  }
  else {
    digitalWrite(PinLED, LOW);
  }
}
```

Déroulement du programme :



- Activité 4 : Allumer en alternance ou éteindre 3 DELs avec un bouton-poussoir



Dans cette activité, l'allumage en alternance des DELs est géré par le bouton poussoir. Un premier appui sur le bouton allume la diode rouge, un deuxième appui allume la diode verte, un troisième appui allume la diode bleue et ainsi de suite…
Un appui prolongé sur le bouton éteint la DEL allumée.

Comme pour l'activité précédente, c'est à l'aide des variables permettant de stocker les valeurs (actuelle et précédente) de l'état logique de la broche du bouton poussoir, mais aussi d'une variable pour compter le nombre d'appui sur le bouton et de variables pour mesurer la durée d'appui, que l'Arduino pourra allumer ou éteindre les DELs

Le code pourra être modifié pour voir l'influence des variables (durée d'appui pour éteindre les DELS).

. Programme en Python (**Projet1\Activity4\PY\Activity4.py**)

```
# Importations des librairies et définition des fonctions

from PyFirmataDef import *
from ConnectToArduino import *
import time


# Déclaration des constantes et variables

PinLEDR = 8
PinLEDV = 7
PinLEDB = 2
PinButton = 12
ComptBtn = 0
ValButton = 0
OldValButton = 0
StartTime = 0
DeltaTime = 0


# Connexion à l'Arduino

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)


# Connexion à l'Arduino réussie - Déclaration des entrées et sorties

DigitalInputPin = DigitalInput(board, PinButton)
ArduinoIterateur = Iterateur(board)
time.sleep (0.5)

print("Connexion à l'arduino établie- Appuyez sur Ctrl-C pour quitter")
```

```
# Boucle principale du programme

while True:
    try:
        ValButton = DigitalInputPin.read()
        time.sleep(0.01)
        if ValButton == 1 and OldValButton == 0:
            StartTime=time.time()
            ComptBtn = ComptBtn +1
            if ComptBtn == 4: ComptBtn = 1
        if ValButton == 1 and OldValButton == 1:
            DeltaTime=time.time()-StartTime
            if DeltaTime > 0.5: ComptBtn = 0

        OldValButton = ValButton

        if ComptBtn == 0:
            board.digital[PinLEDR].write(0)
            board.digital[PinLEDV].write(0)
            board.digital[PinLEDB].write(0)
        elif ComptBtn == 1:
            board.digital[PinLEDR].write(1)
            board.digital[PinLEDV].write(0)
            board.digital[PinLEDB].write(0)
        elif ComptBtn == 2:
            board.digital[PinLEDR].write(0)
            board.digital[PinLEDV].write(1)
            board.digital[PinLEDB].write(0)
        elif ComptBtn == 3:
            board.digital[PinLEDR].write(0)
            board.digital[PinLEDV].write(0)
            board.digital[PinLEDB].write(1)

    except KeyboardInterrupt:
        board.digital[PinLEDR].write(0)
        board.digital[PinLEDV].write(0)
        board.digital[PinLEDB].write(0)
        board.exit()
        sys.exit(0)
```

Déroulement du programme :

- Importation des librairies et définition de fonctions :

. Le module **”ConnectToArduino.py”,** contenant les fonctions de connexion à l’Arduino via le protocole **”Firmata standard”**,

. Le module "**PyFirmataDef.Py**" regroupant toutes les fonctions utiles à l’utilisation de "**PyFirmata**" (fonction de déclaration des entrées et sorties, d’itérateur, d’écritures…)

. La bibliothèque **"time"** pour la gestion des temps de pause et de la durée d’appui sur le bouton-poussoir.

- Déclaration des constantes et variables :

. **PinLEDR = 8** (constante correspondant au n° de la broche sur laquelle la DEL rouge est connectée)

. **PinLEDV = 7** (constante correspondant au n° de la broche sur laquelle la DEL verte est connectée)

. **PinLEDB = 2** (constante correspondant au n° de la broche sur laquelle la DEL bleue est connectée)

. **PinButton = 12** (cst correspondant au n° de la broche sur laquelle le bouton poussoir est connecté)

. **ValButton = 0** (variable pour stocker la valeur de l'état logique de la broche du bouton poussoir)

. **OldValButton = 0** (variable pour stocker la valeur précédente de l'état logique de la broche du bouton poussoir)

. **ComptBtn = 0** (variable permettant de comptabiliser le nombre de fois que le bouton-poussoir est appuyé)

. StartTime = 0 (variable pour stocker l'heure de début d'appui sur le bouton-poussoir)

. DeltaTime = 0 (variable pour calculer la durée d'appui sur le bouton-poussoir)

. **PortComArduino** (port COM sur lequel l'Arduino est connecté)

- Connexion à l'Arduino :

. Détection du port COM, tentative d'ouverture du port COM sélectionné et connexion à l'Arduino :

PortComArduino = SelectPortCOM()

board = OpenPortCom(PortComArduino)

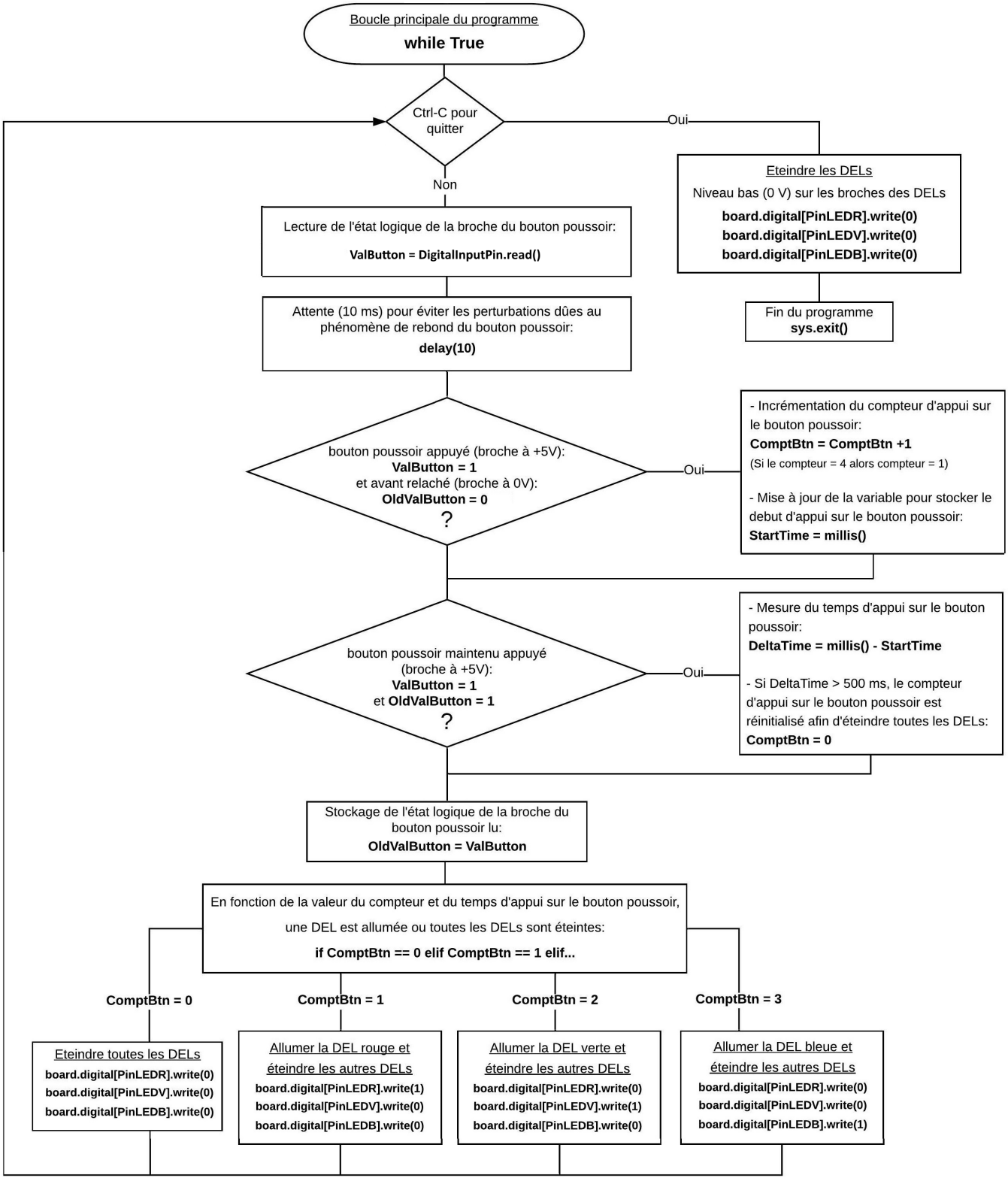
. Si la connexion à l'Arduino est réussie:

- Lancement de l'itérateur : **ArduinoIterateur = Iterateur(board)**,

- Déclaration de la broche du bouton poussoir en entrée digitale : **InputPin = DigitalInput(board, PinButton)**,

- Attente de 500 ms pour le lancement de l'itérateur

- Boucle principale du programme (boucle "while True") :


Boucle principale du programme
while True
Ctrl-C pour quitter
Oui
Non
Eteindre les DELs
Niveau bas (0 V) sur les broches des DELs
board.digital[PinLEDR].write(0)
board.digital[PinLEDV].write(0)
board.digital[PinLEDB].write(0)
Fin du programme
sys.exit()
Lecture de l'état logique de la broche du bouton poussoir:
ValButton = DigitalInputPin.read()
Attente (10 ms) pour éviter les perturbations dûes au phénomène de rebond du bouton poussoir:
delay(10)
bouton poussoir appuyé (broche à +5V):
ValButton = 1
et avant relaché (broche à 0V):
OldValButton = 0
?
Oui
- Incrémentation du compteur d'appui sur le bouton poussoir:
ComptBtn = ComptBtn +1
(Si le compteur = 4 alors compteur = 1)
- Mise à jour de la variable pour stocker le debut d'appui sur le bouton poussoir:
StartTime = millis()
bouton poussoir maintenu appuyé (broche à +5V):
ValButton = 1
et OldValButton = 1
?
Oui
- Mesure du temps d'appui sur le bouton poussoir:
DeltaTime = millis() - StartTime
- Si DeltaTime > 500 ms, le compteur d'appui sur le bouton poussoir est réinitialisé afin d'éteindre toutes les DELs:
ComptBtn = 0
Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
En fonction de la valeur du compteur et du temps d'appui sur le bouton poussoir, une DEL est allumée ou toutes les DELs sont éteintes:
if ComptBtn == 0 elif ComptBtn == 1 elif...
ComptBtn = 0
ComptBtn = 1
ComptBtn = 2
ComptBtn = 3
Eteindre toutes les DELs
board.digital[PinLEDR].write(0)
board.digital[PinLEDV].write(0)
board.digital[PinLEDB].write(0)
Allumer la DEL rouge et éteindre les autres DELs
board.digital[PinLEDR].write(1)
board.digital[PinLEDV].write(0)
board.digital[PinLEDB].write(0)
Allumer la DEL verte et éteindre les autres DELs
board.digital[PinLEDR].write(0)
board.digital[PinLEDV].write(1)
board.digital[PinLEDB].write(0)
Allumer la DEL bleue et éteindre les autres DELs
board.digital[PinLEDR].write(0)
board.digital[PinLEDV].write(0)
board.digital[PinLEDB].write(1)

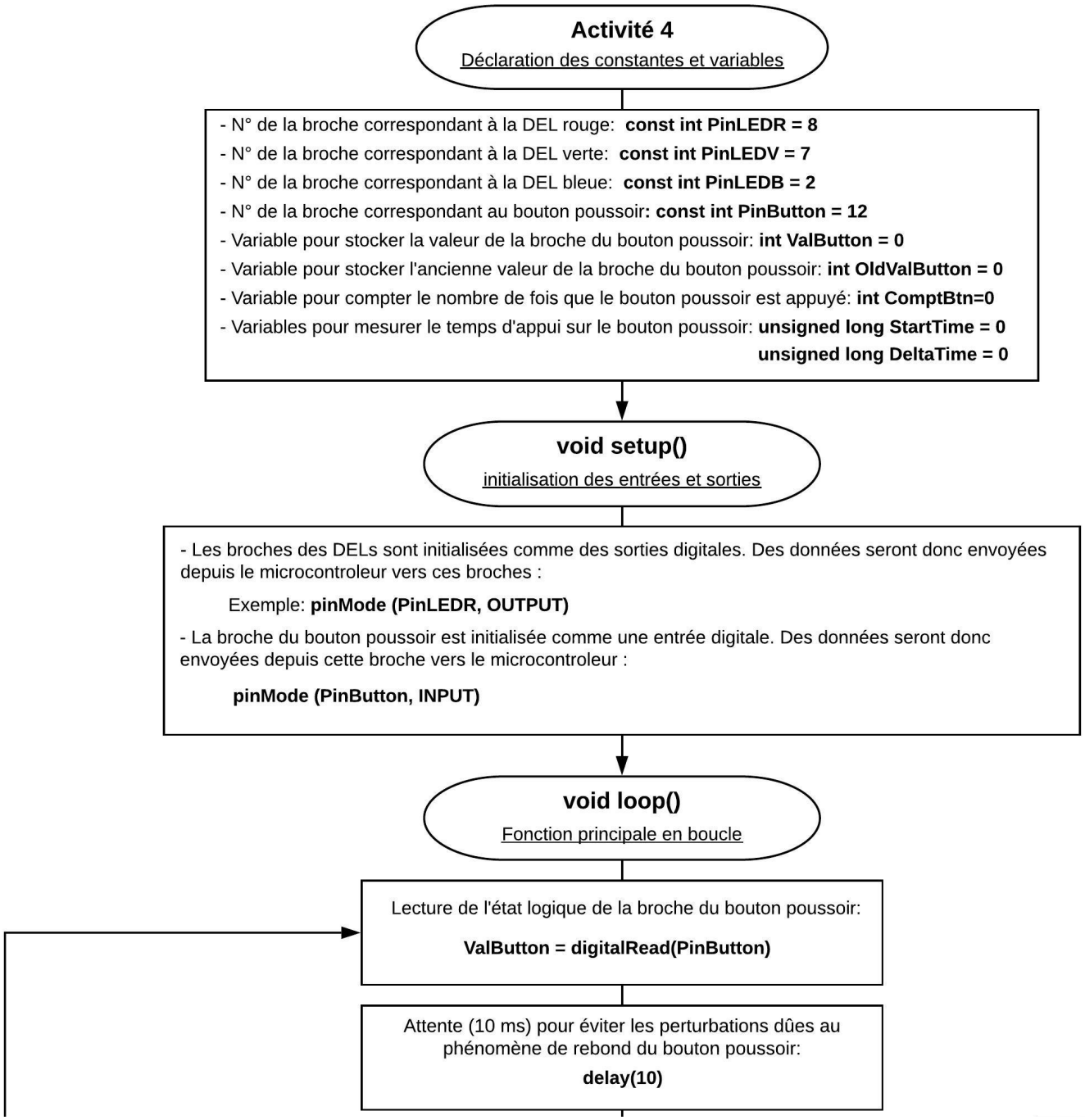
. Programme en langage Arduino (Projet1\Activity4\INO\Activity4.ino)

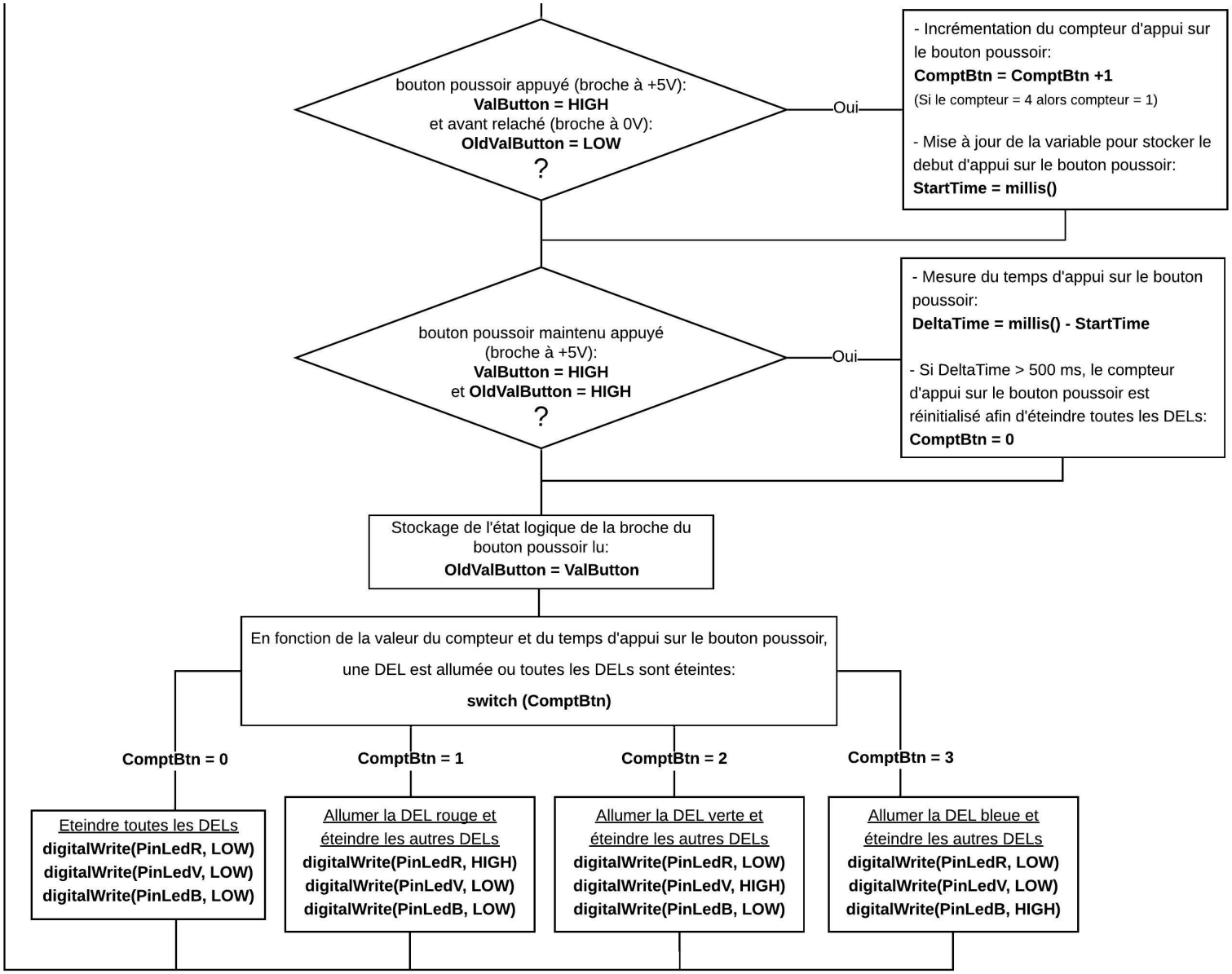
Activity4

```
// Déclaration des constantes et variables

const int PinLEDR = 8;
const int PinLEDV = 7;
const int PinLEDB = 2;
const int PinButton = 12;

int ValButton=0;
int OldValButton=0;
int ComptBtn=0;
unsigned long StartTime = 0;
unsigned long DeltaTime = 0;


// Initialisation des entrées et sorties

void setup() {
  pinMode (PinLEDR, OUTPUT);
  pinMode (PinLEDV, OUTPUT);
  pinMode (PinLEDB, OUTPUT);
  pinMode (PinButton, INPUT);
}


// Fonction principale en boucle

void loop() {
  ValButton = digitalRead(PinButton);
  delay(10);
  if ((ValButton == HIGH)&& (OldValButton == LOW)) {
    StartTime = millis();
    ComptBtn = ComptBtn +1;
    if (ComptBtn == 4) {
      ComptBtn = 1;
    }
  }

  if ((ValButton == HIGH) && (OldValButton == HIGH)) {
  DeltaTime = millis() - StartTime;
    if (DeltaTime > 500) {
      ComptBtn = 0;
    }
  }

  OldValButton = ValButton;
```

```
  switch (ComptBtn) {
    case 1 :
      digitalWrite(PinLEDR, HIGH);
      digitalWrite(PinLEDV, LOW);
      digitalWrite(PinLEDB, LOW);
    break;

  case 2 :
    digitalWrite(PinLEDR, LOW);
    digitalWrite(PinLEDV, HIGH);
    digitalWrite(PinLEDB, LOW);
  break;

  case 3 :
    digitalWrite(PinLEDR, LOW);
    digitalWrite(PinLEDV, LOW);
    digitalWrite(PinLEDB, HIGH);
  break;

  case 0 :
    digitalWrite(PinLEDR, LOW);
    digitalWrite(PinLEDV, LOW);
    digitalWrite(PinLEDB, LOW);
  break;
  }
}
```

Déroulement du programme :




bouton poussoir appuyé (broche à +5V):
ValButton = HIGH
et avant relaché (broche à 0V):
OldValButton = LOW
?
Oui
- Incrémentation du compteur d'appui sur le bouton poussoir:
ComptBtn = ComptBtn +1
(Si le compteur = 4 alors compteur = 1)
- Mise à jour de la variable pour stocker le debut d'appui sur le bouton poussoir:
StartTime = millis()
bouton poussoir maintenu appuyé (broche à +5V):
ValButton = HIGH
et OldValButton = HIGH
?
Oui
- Mesure du temps d'appui sur le bouton poussoir:
DeltaTime = millis() - StartTime
- Si DeltaTime > 500 ms, le compteur d'appui sur le bouton poussoir est réinitialisé afin d'éteindre toutes les DELs:
ComptBtn = 0
Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
En fonction de la valeur du compteur et du temps d'appui sur le bouton poussoir, une DEL est allumée ou toutes les DELs sont éteintes:
switch (ComptBtn)
ComptBtn = 0
ComptBtn = 1
ComptBtn = 2
ComptBtn = 3
Eteindre toutes les DELs
digitalWrite(PinLedR, LOW)
digitalWrite(PinLedV, LOW)
digitalWrite(PinLedB, LOW)
Allumer la DEL rouge et éteindre les autres DELs
digitalWrite(PinLedR, HIGH)
digitalWrite(PinLedV, LOW)
digitalWrite(PinLedB, LOW)
Allumer la DEL verte et éteindre les autres DELs
digitalWrite(PinLedR, LOW)
digitalWrite(PinLedV, HIGH)
digitalWrite(PinLedB, LOW)
Allumer la DEL bleue et éteindre les autres DELs
digitalWrite(PinLedR, LOW)
digitalWrite(PinLedV, LOW)
digitalWrite(PinLedB, HIGH)