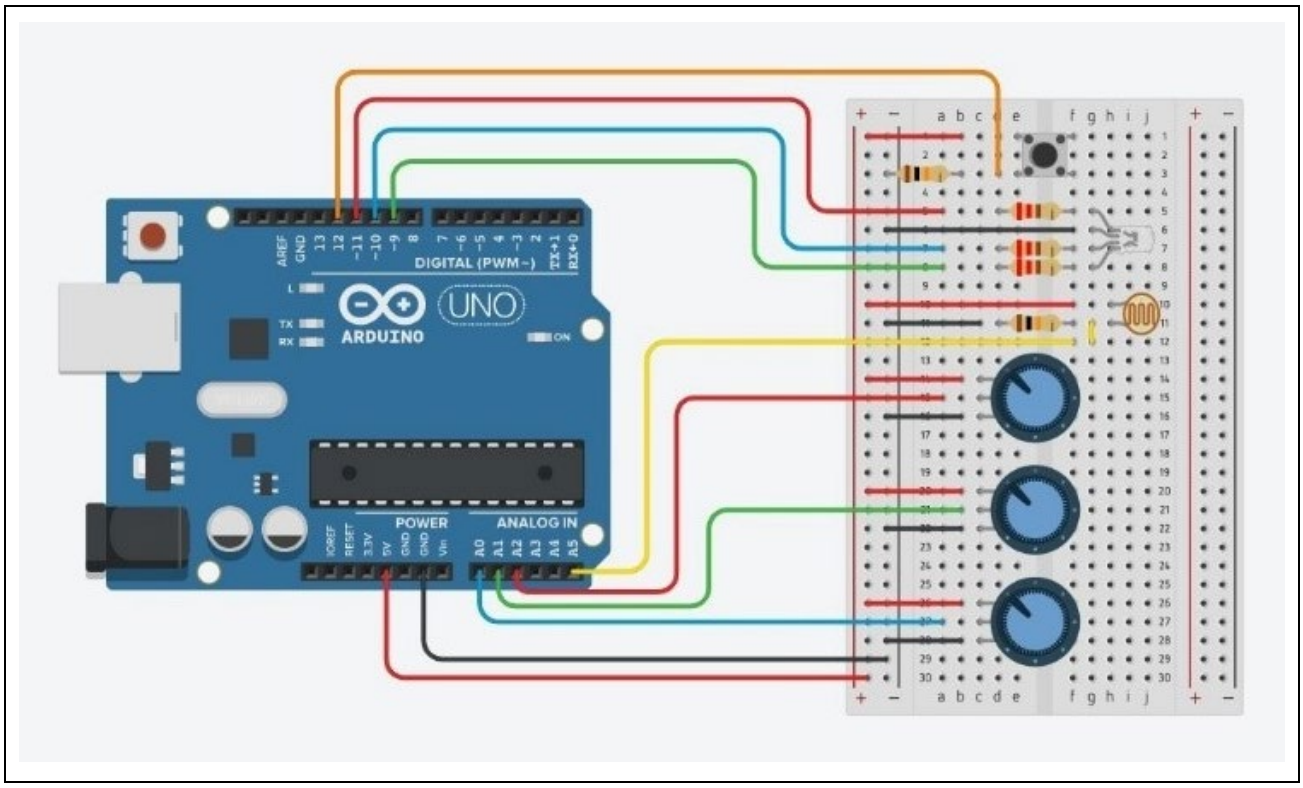
Projet 2 - DEL RVB : Entrées et Sorties analogiques

Après avoir vu le principe de fonctionnement des entrées et des sorties numériques de l'Arduino, avec ce nouveau circuit, nous allons nous intéresser aux entrées et sorties analogiques de la platine.



- Liste des composants :

. 1 DEL RVB

. 3 résistances de 220 Ω (résistances des DELs)

. 2 résistances de 10 kΩ (résistances du bouton-poussoir et de la photorésistance)

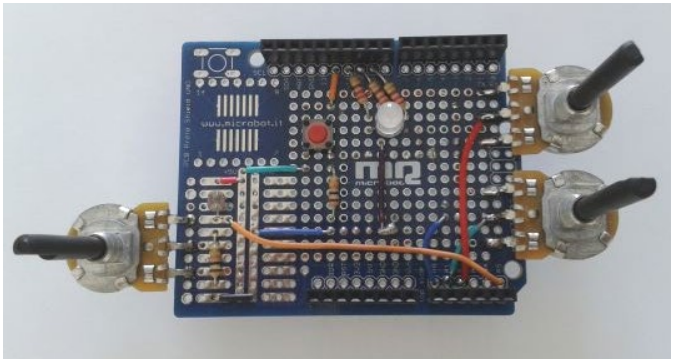
. 3 potentiomètres de 10 kΩ

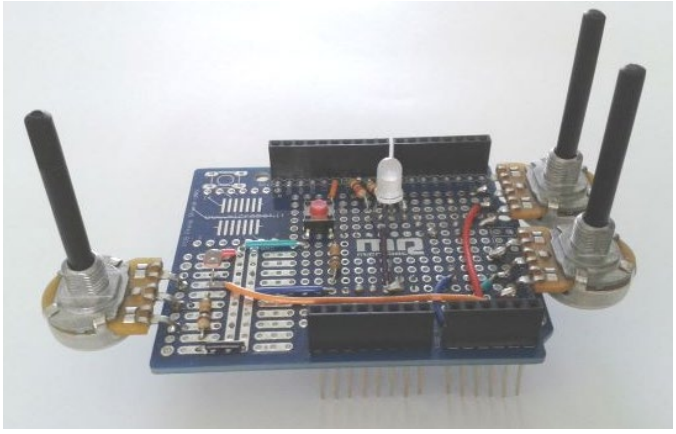
. 1 photorésistance

. 1 bouton poussoir

. 1 plaque d'essai

. Fils de connexion





Le circuit sur un "shield" pour Arduino Uno

- Protocole de communication:

. Firmata standard

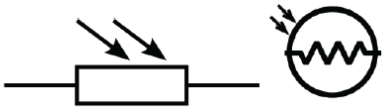
Rappels :

La photorésistance

Une photorésistance est un composant électronique dont la résistance en Ohm (Ω) varie en fonction de l'intensité lumineuse. Plus la luminosité est élevée, plus la résistance est basse. On peut donc l'utiliser comme capteur lumineux pour :

- Mesure de la lumière ambiante.
- Détecteur de lumière dans une pièce.
- Suiveur de lumière dans un robot.
- Détecteur de passage.
- …

Ses symboles électroniques sont les suivants :

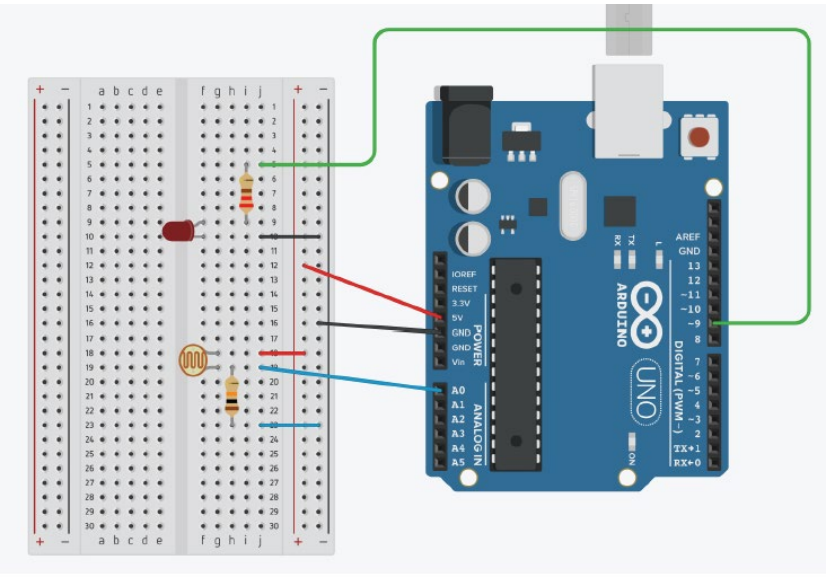


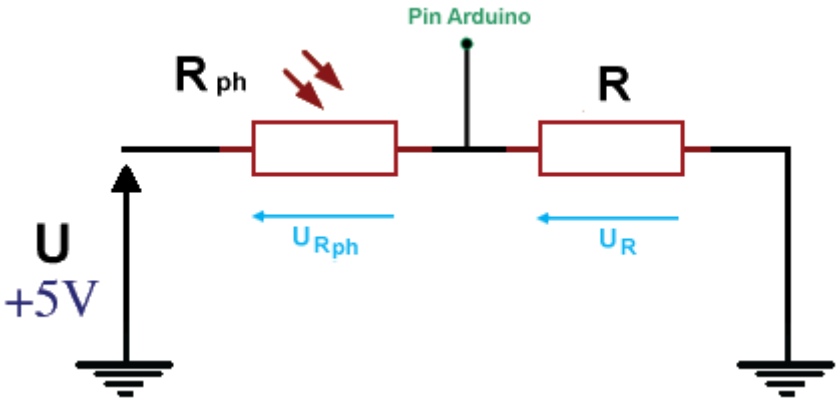
L'avantage des photorésistances est qu'elles sont très bon marché. Par contre, leur réaction à la lumière est différente pour chaque photorésistance, même quand elles ont été produites dans le même lot. On peut ainsi noter une différence de résistance de plus de 50% entre deux photorésistances du même modèle pour la même luminosité. On ne peut donc pas l'utiliser pour une mesure précise de la lumière. Par contre, elles sont idéales pour mesurer des changements simples de la luminosité.

On utilise la photorésistance dans un montage (Cf. ci-dessous), avec une résistance fixe de 10 kΩ, qu'on appelle un **diviseur de tension**.

La photorésistance est alimentée en 5V depuis l'Arduino. Le point entre les deux résistances est relié à une broche analogique de l'Arduino et on mesure la tension de cette broche par la fonction « **analogRead (broche)** ».

Tout changement de la tension mesurée est dû à la photorésistance puisque c'est la seule qui change dans ce circuit, en fonction de l'intensité lumineuse qu'elle reçoit.





D'après la loi d'additivité des tensions dans un circuit en série :

$$U = U_{Rph} + U_R = (R_{ph} + R)\, I$$

$$U_R = U - U_{Rph} = U - R_{ph}\, I$$

U_R est la tension appliquée sur l'entrée analogique de l'Arduino. Quand l'intensité lumineuse reçue par la photorésistance augmente, R_{ph} diminue, donc U_R augmente, et au contraire quand la luminosité diminue, R_{ph} augmente et U_R diminue.

On peut exprimer Ur en fonction de U :

$$U = U_{Rph} + U_R = (R_{ph} + R)\, I$$

Donc : $$I = \frac{U}{(R_{ph} + R)}$$

Et : $$U_R = R\, I = \frac{R}{(R_{ph} + R)}\, U$$

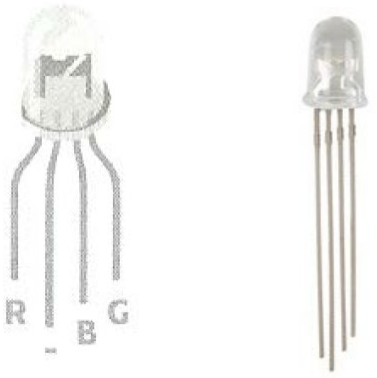
C'est la raison pour laquelle on appelle ce montage un diviseur de tension.

. La DEL RVB

Une DEL RVB (RVB pour Rouge-Vert-Bleu), n'est rien d'autre qu'un ensemble de 3 DELs, une rouge une verte et une bleue rassemblées dans un seul et même boitier.

Une DEL RVB a 4 broches : une commune à l'ensemble des DELs et une pour chaque couleur de la DEL. La broche commune pourra, selon les modèles, être le + (anode commune) ou le - (cathode commune).

La DEL ci-dessous a pour broche commune la cathode :



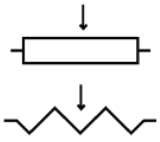
. Le potentiomètre

Le potentiomètre est une résistance variable. C'est le bouton de réglage du volume que l'on retrouve sur une radio. La plupart des potentiomètres sont soit rotatifs, soit linéaires.

Potentiomètre rotatif

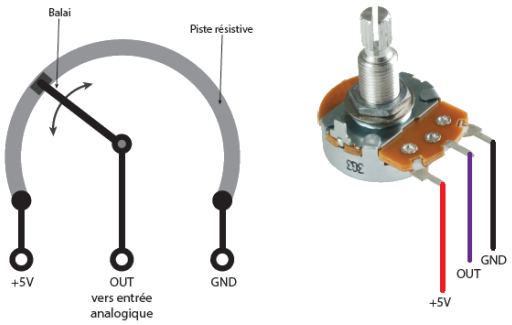
Potentiomètre linéaire

Voici les symboles électroniques (européen dessus et américain dessous) du potentiomètre :

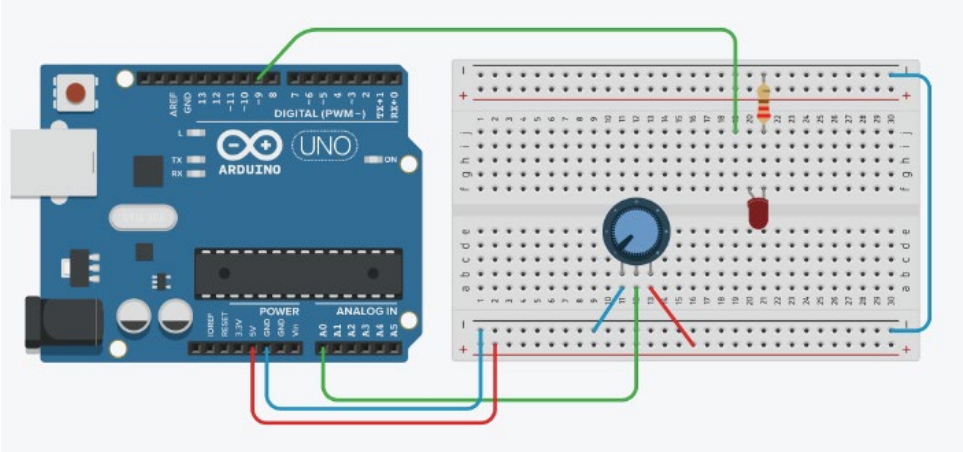


Comme toute résistance, le potentiomètre modifie la tension d'un circuit. On va donc l'utiliser principalement comme entrée (input) sur une broche analogique (A0 à A5) de l'Arduino.

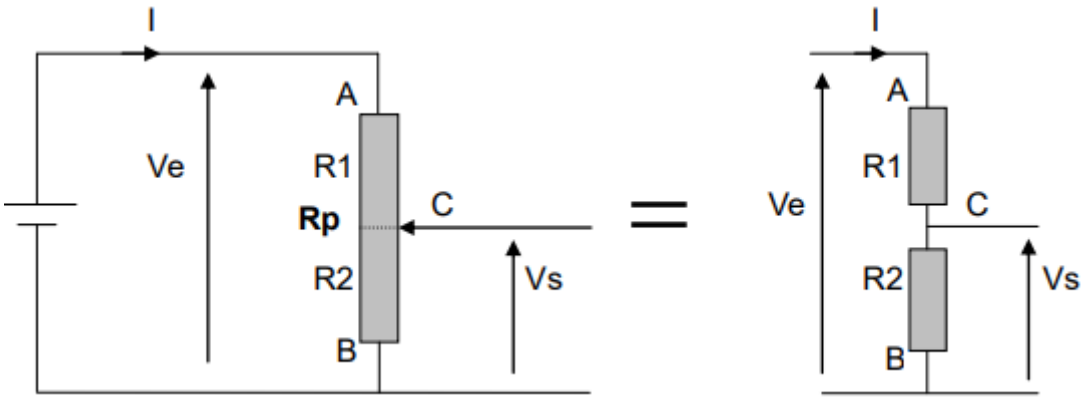
Les potentiomètres ont en général trois broches :



Les broches extérieures se connectent sur l'alimentation +5V et sur la terre, alors que la broche centrale envoie le signal sur la broche d'entrée analogique de l'Arduino, comme sur le circuit ci-dessous :



Dans ce montage, le potentiomètre de résistance totale Rp est utilisé en pont diviseur de tension :



On a : **Ve = (R1 + R2) I**

$$\mathbf{I = \frac{Ve}{(R_1 + R_2)}}$$

Et : $\mathbf{Vs = R_2\, I = \frac{R_2}{(R_1 + R_2)}\, Ve = \frac{R_2}{Rp}\, Ve}$ **(car $R_p = R_1 + R_2$)**

On peut remplacer le rapport **R_2 / Rp** par la position du curseur comprise entre 0 (position B) et 1 (position A).

Dans ce cas, la relation devient :

Vs = α Ve (avec α la position du curseur : 0 ≤ α ≤ 1)

Vs, qui est la tension appliquée sur une entrée analogique de l'Arduino, varie donc entre 0 et +5V en fonction de la position du curseur du potentiomètre.

De façon à limiter le courant dans le circuit à 0,5 mA, on utilise généralement des potentiomètres de **10 kΩ**.

. Les entrées analogiques

Il y a six entrées analogiques notées de A0 à A5 en bas à droite de la carte.

Les tensions, entre 0 et 5V, appliquées sur ces broches, sont numérisées via un convertisseur analogique-numérique CAN ou ADC (Analog Digital Converter).

Le convertisseur des Arduino effectue une conversion sur 10 bits, c'est à dire qu'il convertit la tension en un nombre entier ayant une valeur de 0 à 1023.

0 correspond à une tension de 0 V et 1023 à une tension de 5V.

La résolution, c'est à dire la différence entre deux valeurs successives de la tension correspondant à une différence de 1 sur l'entier résultat de la conversion analogique-numérique, est donc d'environ 5mV (**5/1024 = 4,9 mV**).

Attention :

. Appliquer une tension supérieure à 5 volts ou inférieure à 0 volt sur une broche analogique endommagera immédiatement et définitivement votre carte Arduino,

. La mesure prend environ 100µs, cela fait un maximum de 10 000 mesures par seconde,

. Effectuer une mesure d'une tension sur une broche non connectée retourne des valeurs de l'ordre de 300 à 500, même s'il n'y a pas de signal,

. La précision de la mesure (conversion sur 10 bits) n'est pas modifiable. Celle-ci est de plus ou moins 1 (+ ou - 5 mV).

Remarque :

Les broches notées de A0 à A5 peuvent également être configurées en entrées et sorties numériques.

. Les sorties analogiques (PWM)

Il n'y a pas de sortie analogique à proprement parler sur un Arduino. Par contre, six des connecteurs numériques (les connecteurs 3, 5, 6, 9, 10 et 11) sont capables de simuler des sorties analogiques et fonctionnent donc comme telles pour délivrer une tension entre 0 et 5 V.

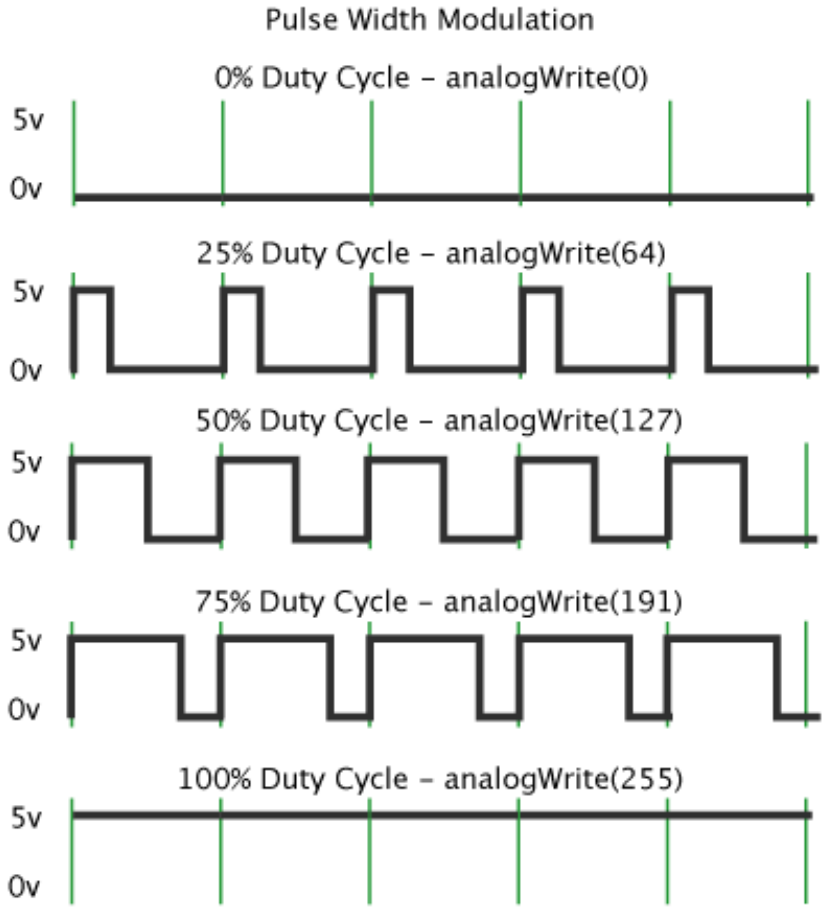
Elles sont marquées du symbole tilde ~ et du sigle PWM, qui veut dire Pulse Width Modulation (Modulation de Largeur d'Impulsion en français).

Le PWM fonctionne ainsi : comme il n'est possible que d'envoyer des informations binaires (haut ou bas, c'est à dire 5 V ou 0 V) sur ces broches, l'Arduino va faire varier la durée pendant laquelle ces deux valeurs sont appliquées afin d'obtenir la tension souhaitée.

L'Arduino génère donc un signal carré caractérisé par deux paramètres (Cf. ci-dessous) :

. L'amplitude du signal est de 5V ou de 0V,

. Le rapport entre la durée où la tension est à 5V et celle où elle est à 0V (ce rapport est appelé Duty cycle et est exprimé en %).



La fréquence du signal PWM est de 490 Hz, ce qui est suffisamment rapide pour que l'on puisse dire que l'amplitude du signal d'une sortie PWM est égale à la valeur moyenne du signal carré généré :

Tension sortie analogique (en V) = 5 x (Duty Cycle/100)

Exemple :

Duty Cycle à 0% : Tension sortie analogique = 5 x 0 = 0 V

Duty Cycle à 25% : Tension sortie analogique = 5 x 0,25 = 1,25 V

Duty Cycle à 50% : Tension sortie analogique = 5 x 0,5 = 2,5 V

Duty Cycle à 75% : Tension sortie analogique = 5 x 0,75 = 3,75 V

Duty Cycle à 100% : Tension sortie analogique = 5 x 1 = 5 V

Remarques :

En langage ARDUINO IDE, la fonction "**analogWrite ()**" permet de générer un signal analogique sur une sortie PWM.

Elle prend deux arguments :

. Le premier est le numéro de la broche sur laquelle on veut générer la PWM
. Le second argument représente la valeur du rapport cyclique à appliquer.

Cependant, on n'exprime pas cette valeur en pourcentage, mais avec un nombre entier compris entre 0 et 255.

Le rapport cyclique s'exprime de 0 à 100 % en temps normal. Cependant, dans cette fonction il s'exprimera de 0 à 255 (sur 8 bits). Ainsi, pour un rapport cyclique de 0% nous enverrons la valeur 0, pour un rapport de 50% on enverra 127 et pour 100% ce sera 255. Les autres valeurs sont bien entendu considérées de manière proportionnelle entre les deux.

. La programmation

1. Gestion des entrées et sorties analogiques en langage Arduino

- Pour lire la valeur de la tension d'une entrée analogique, on utilise la fonction :

analogRead()

. Syntaxe :

analogRead(broche_analogique)

. Paramètres :

broche_analogique : le numéro de la broche sur laquelle il faut convertir la tension analogique appliquée (0 à 5 sur la plupart des cartes Arduino)

. Valeur retournée :

valeur int (0 à 1023) correspondant au résultat de la mesure effectuée

. Remarque :

Il n'est pas nécessaire de déclarer les broches A0 à A5 en entrée avec la fonction **pinMode()** pour lire la tension appliquée sur celles-ci.

- Pour utiliser une broche de l'Arduino en sortie analogique (mode PWM), il faut au préalable la déclarer en sortie avec la fonction **pinMode()**.

La tension de la broche déclarée en sortie analogique est réglable en modifiant le rapport cyclique du signal PWM à l'aide de la fonction :

analogWrite()

.Syntaxe :

analogWrite(broche, valeur)

.Paramètres :

- broche: la broche utilisée pour "écrire" l'impulsion. Cette broche devra être une broche ayant la fonction PWM. Par exemple, sur la UNO, ce pourra être une des broches 3, 5 ,6 ,9 ,10 ou 11.

- valeur: rapport cyclique du signal PWM (proportion de l'onde carrée qui est au niveau HAUT) : entre 0 (0% HAUT donc toujours au niveau BAS) et 255 (100% HAUT donc toujours au niveau HAUT).

2. Gestion des entrées et sorties analogiques avec le protocole de communication "Firmata Standard"

- Pour lire la valeur de la tension d'une entrée analogique (par exemple, la broche A0), il faut la déclarer au préalable en entrée analogique avec la commande suivante :

pinA0 = board.get_pin('a:0:i')

La syntaxe est "**a**" pour analogique, "**0**" est le numéro de la broche A0, "**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 analogique :

```
def AnalogInput(board,pin):
   AnalogInputPin=board.get_pin('a:'+ str(pin) +':i')
   return AnalogInputPin
```

La syntaxe pour déclarer la broche A0 en entrée analogique est alors plus simple :

pinA0 = AnalogInput(board,0)

Ensuite, on utilise la fonction "**read()**" pour lire la valeur de la tension de la broche :

pinA0.read()

Attention :

. La fonction "**read()**" pour une broche déclarée en entrée analogique retourne une valeur décimale entre 0 et 1 (alors qu'en langage Arduino, "analogRead()" retourne un entier entre 0 et 1023).

. Il faut également utiliser un itérateur afin d'éviter de saturer la communication série entre l'Arduino et l'ordinateur hôte.

- Pour utiliser une broche de l'Arduino en sortie analogique (mode PWM), il faut au préalable la déclarer en créant un objet "**pinPWM**" avec la commande suivante :

pinPWM = board.get_pin('d:numéro_de_pin:p')

La syntaxe est "**d**" pour digital, "**numéro_de_pin**" est le numéro de la broche (pour rappel, les broches supportant le PWM sont les broches 3, 5, 6, 9, 10, 11 sur l'Arduino Uno), "**p**" pour PWM 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 sortie analogique :

```
def AnalogOutput(board,pin):
   AnalogOutputPin=board.get_pin('d:'+ str(pin) +':p')
   return AnalogOutputPin
```

La syntaxe pour déclarer, par exemple, la broche N°11 en sortie analogique est alors plus simple:

pinPWM = AnalogOutput(board,11)

La tension de la broche N°11 déclarée en sortie analogique est réglable en modifiant le rapport cyclique du signal PWM entre **0** (0 %) et **1** (100%) :

board.digital[11].write(rapport cyclique)

→ si rapport cyclique = 0, alors la tension est de 0 V
→ si rapport cyclique = 1, alors la tension est de 5 V
→ si rapport cyclique = 0.5, alors la tension est de 2,5 V

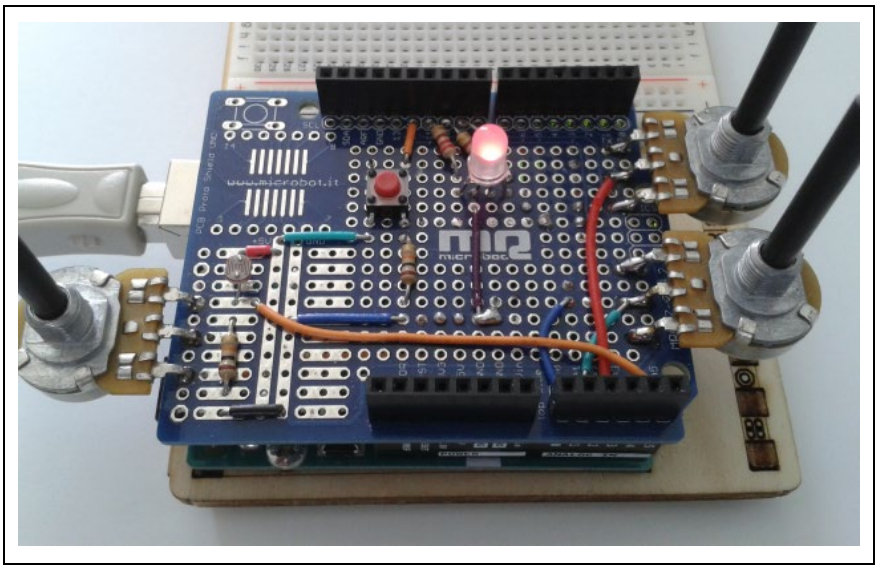
On peut également définir une fonction pour modifier le rapport cyclique d'une broche déclarée en sortie analogique :

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

Ainsi, l'instruction pour appliquer une tension de 2,5V sur la sortie "11" déclarée en sortie analogique, de l'objet "board", est :

AnalogWrite(board, 11, 0.5)

- Activité 1 : Contrôler la luminosité d'une DEL avec un bouton-poussoir



Contrairement aux sorties numériques qui ne peuvent avoir que deux valeurs 0 ou 1 (0 ou 5V), une sortie analogique (ou plutôt PWM) permet d'obtenir une tension entre 0 et 5 V. les broches 3, 5, 6, 9, 10 et 11 peuvent être configurés en sortie analogique.

C'est pour cela que les anodes de la DEL RVB (à cathode commune) de notre circuit sont connectées sur les broches :

- 9 pour la DEL rouge
- 11 pour la DEL verte
- 10 pour la DEL Bleue

Nous allons utiliser ces sorties pour alimenter une DEL (DEL rouge, verte ou bleue de la DEL RVB) et faire varier sa luminosité suivant ce principe de fonctionnement :

- la DEL étant éteinte, si on appuie sur le bouton poussoir, la diode s'allume.

- On règle la luminosité de la DEL en maintenant le bouton poussoir appuyé, du moins au plus lumineux (broche de la DEL à +0V) jusqu'à un maximum (broche de la DEL à +5V).

- Quand le maximum de la luminosité est atteint et que le bouton poussoir est maintenu appuyé, la luminosité revient au minimum (broche de la DEL à 0V).

- La DEL étant allumée, si on appuie sur le bouton poussoir, elle s'éteint.

Le code pourra être modifié pour voir l'influence des variables (choix de la DEL, durée d'appui pour éteindre la DEL).

. Programme en Python (Projet2\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 = 11
PinButton = 12
ValButton = 0
OldValButton = 0
State = 0
Brightness = 0
StartTime = 0
Deltatime = 0


# Connexion à l'Arduino

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)


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

PinDigitalInput = DigitalInput(board, PinButton)
PinAnalogOutput = AnalogOutput(board, PinLED)
ArduinoIterator = 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 = PinDigitalInput.read()
        time.sleep(.01)
        if ValButton == 1 and OldValButton == 0:
            StartTime = time.time()
            State = 1 - State
        if ValButton == 1 and OldValButton == 1:
            Deltatime = time.time() - StartTime
            if Deltatime > 0.5:
                if Brightness < 100:
                    Brightness=Brightness+1
                else:
                    Brightness=0
                time.sleep(.005)
        OldValButton = ValButton
        if State == 1:
            PinAnalogOutput.write(float(Brightness/100))
        else:
            PinAnalogOutput.write(0)

    except KeyboardInterrupt:
            PinAnalogOutput.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 :

. **PinLED = 11** (constante correspondant au n° de la broche sur laquelle la DEL choisie 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)

. **Brightness = 0** (variable correspondant à la luminosité de la DEL)

. 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 :

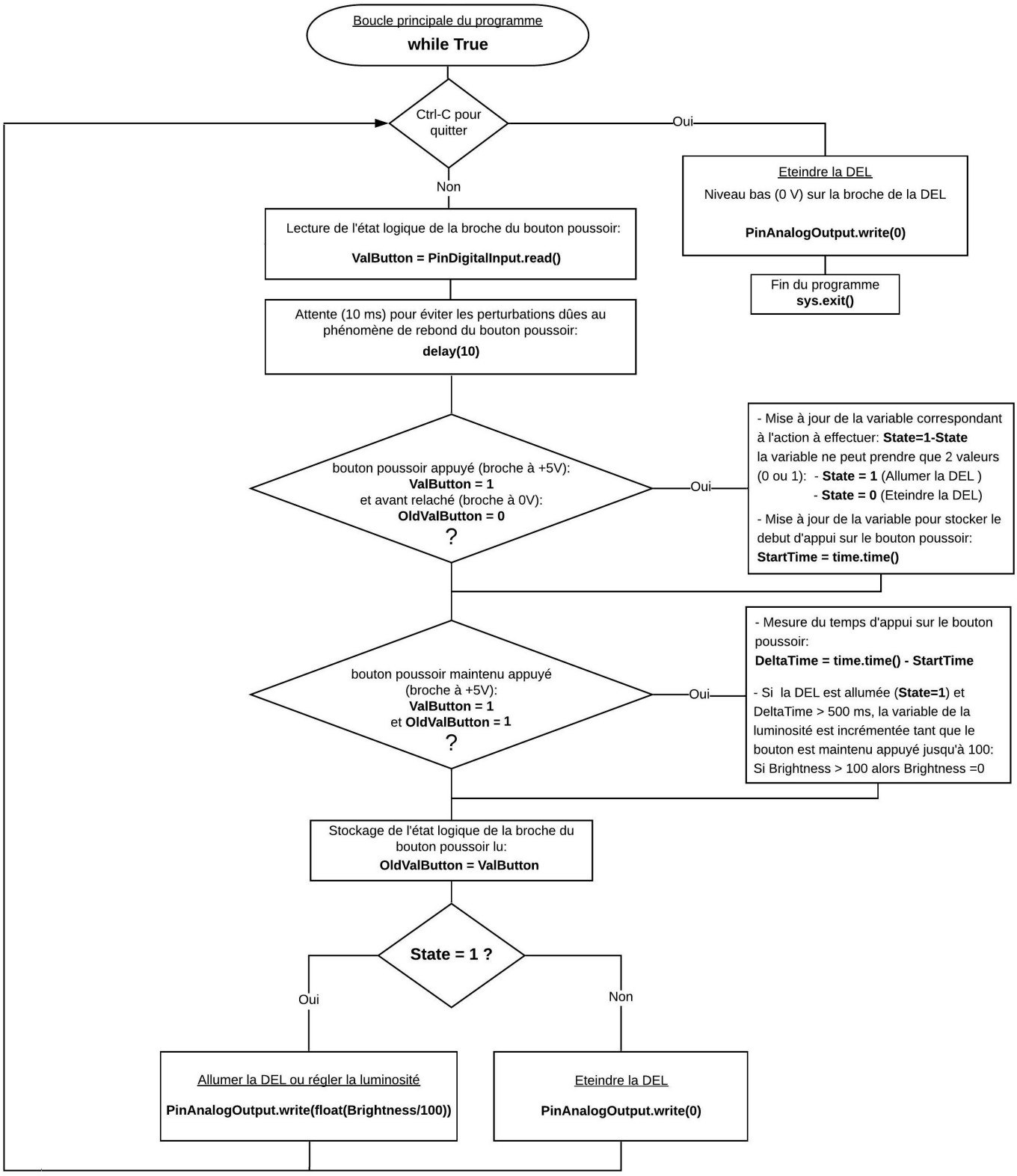
InputPin = DigitalInput(board, PinButton),

- Déclaration de la broche de la DEL en sortie analogique :

PinAnalogOutput = AnalogOutput(board, PinLED),

- Attente de 500 ms pour le lancement de l’itérateur.

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

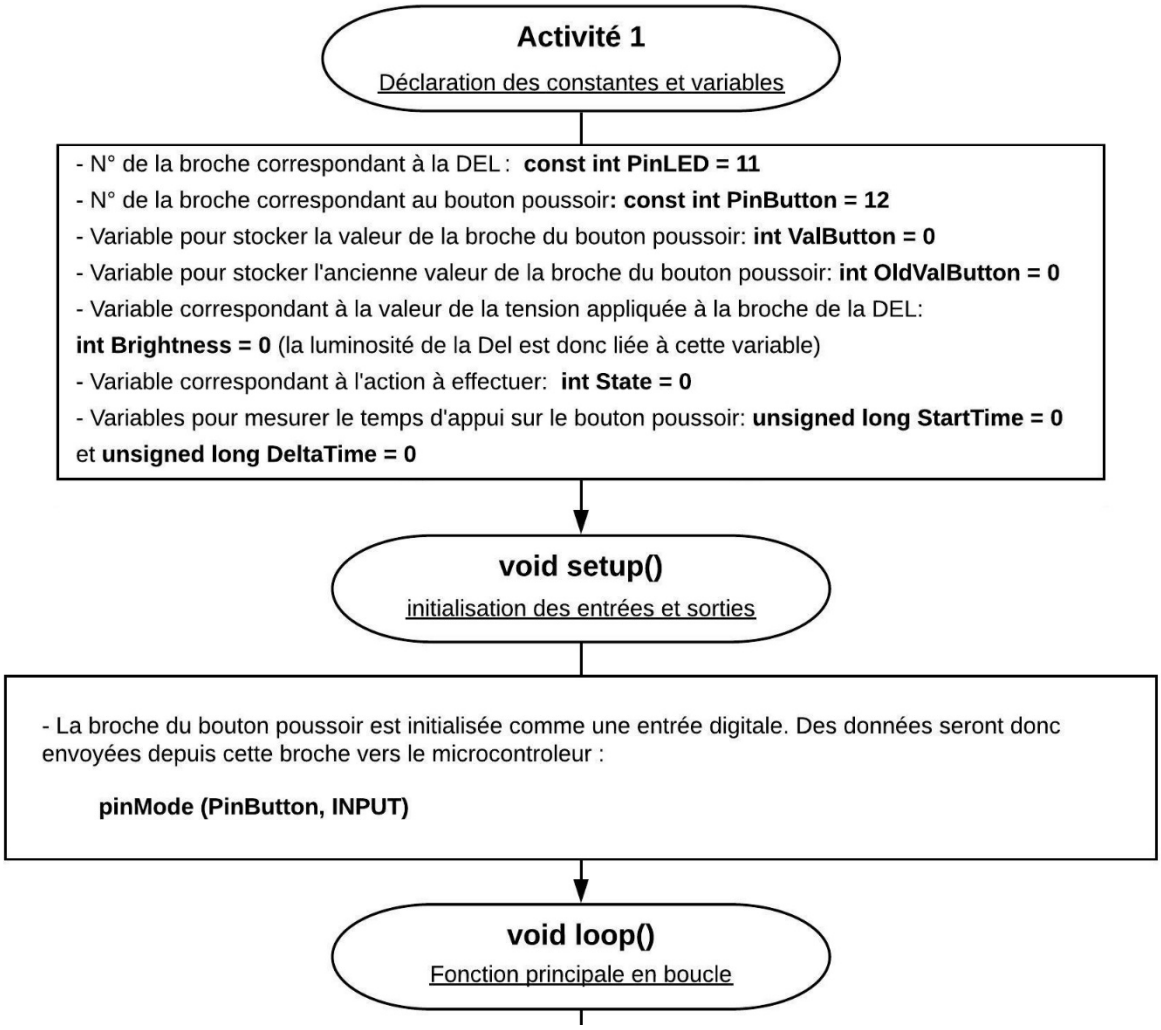


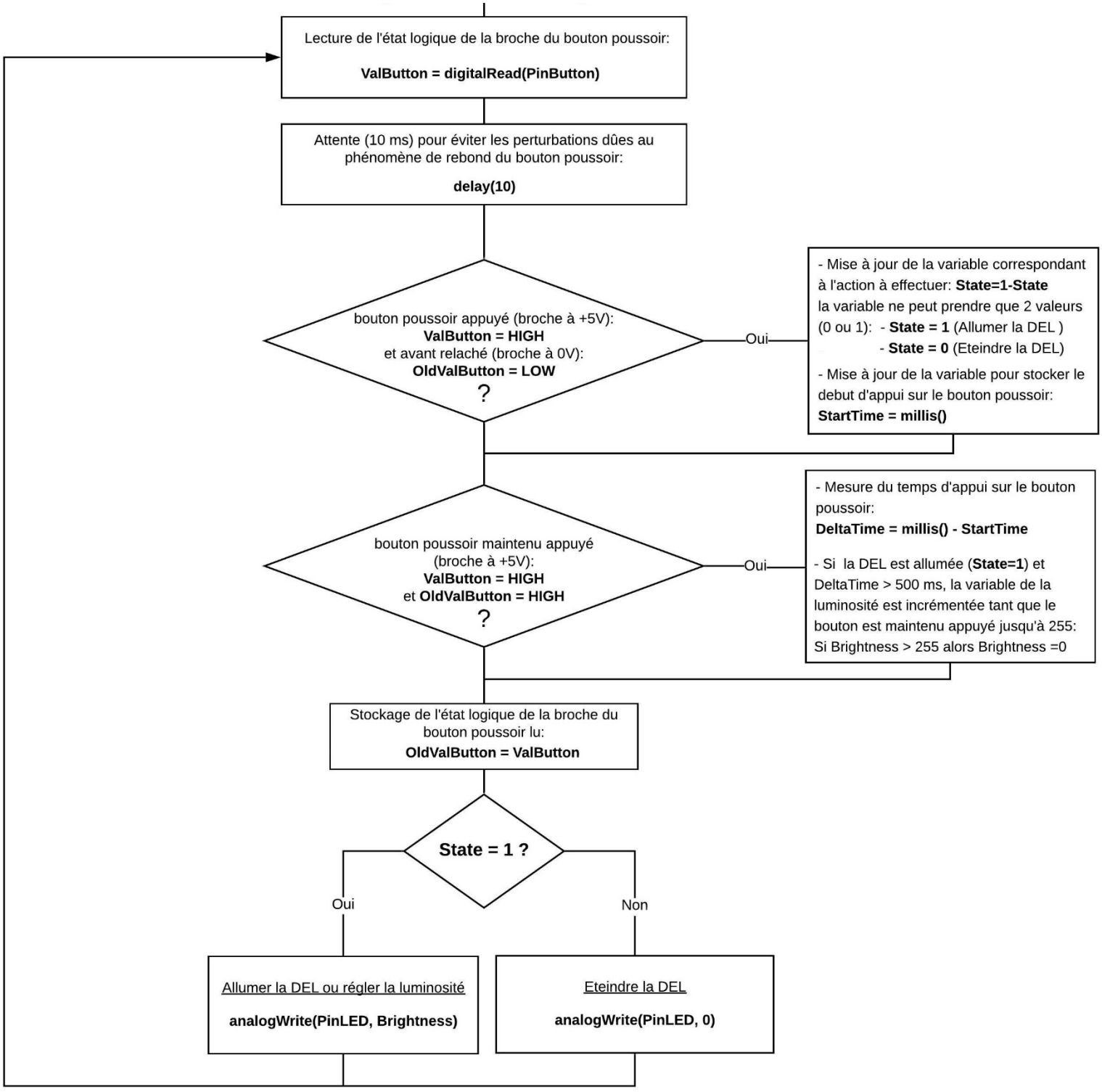
. Programme en langage Arduino (Projet2\Activity1\INO\Activity1.ino)

Activity1

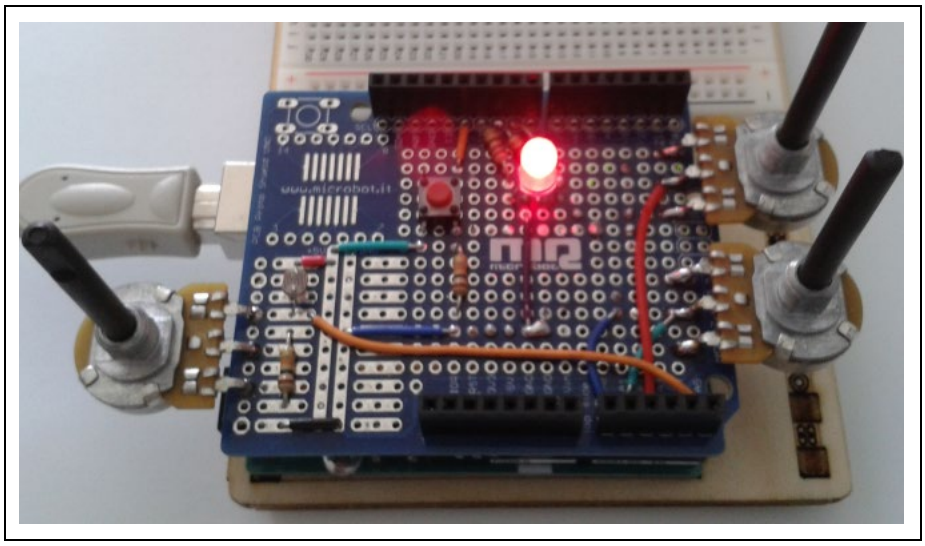
```
// Déclaration des constantes et variables

const int PinLED = 11;
const int PinButton = 12;

int ValButton = 0;
int OldValButton = 0;
int State = 0;
int Brightness = 0;
unsigned long StartTime = 0;
unsigned long DeltaTime = 0;


// Initialisation des entrées et sorties

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


// Fonction principale en boucle

void loop() {
  ValButton = digitalRead(PinButton);
  delay(10);
  if ((ValButton == HIGH) && (OldValButton == LOW)) {
    State=1-State;
    StartTime= millis();
  }
  if ((ValButton == HIGH) && (OldValButton == HIGH)) {
    DeltaTime = millis() - StartTime;
    if (State == 1 && DeltaTime > 500) {
      Brightness = Brightness + 1;
      delay(10);
      if (Brightness > 255) {
        Brightness = 0;
      }
    }
  }
  OldValButton = ValButton;
  if (State == 1) {
    analogWrite(PinLED, Brightness);
  }
  else {
    analogWrite(PinLED, 0);
    Brightness = 0;
  }
}
```

Déroulement du programme :




Lecture de l'état logique de la broche du bouton poussoir:
ValButton = digitalRead(PinButton)
Attente (10 ms) pour éviter les perturbations dûes au phénomène de rebond du bouton poussoir:
delay(10)
bouton poussoir appuyé (broche à +5V):
ValButton = HIGH
et avant relaché (broche à 0V):
OldValButton = LOW
?
Oui
- Mise à jour de la variable correspondant à l'action à effectuer: State=1-State
la variable ne peut prendre que 2 valeurs (0 ou 1): - State = 1 (Allumer la DEL)
- State = 0 (Eteindre la DEL)
- 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 la DEL est allumée (State=1) et DeltaTime > 500 ms, la variable de la luminosité est incrémentée tant que le bouton est maintenu appuyé jusqu'à 255:
Si Brightness > 255 alors Brightness =0
Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Oui
Non
Allumer la DEL ou régler la luminosité
analogWrite(PinLED, Brightness)
Eteindre la DEL
analogWrite(PinLED, 0)

- Activité 2 : Clignotement d'une DEL à une allure fixée par une entrée analogique



Maintenant que nous maîtrisons le fonctionnement d'une sortie analogique, nous allons aborder, dans cette activité, l'étude des entrées analogiques de l'Arduino.

Ces entrées (A0 à A5), qui peuvent être également utilisées en entrées digitales, sont capables de mesurer la tension réelle, entre 0 et 5V, qui leur est appliquée. On utilisera ces entrées pour les acquisitions avec des capteurs qui délivrent une tension entre 0 et 5 V suivant ce qu'ils mesurent.

Ici, le capteur utilisé est une photorésistance, dont la résistance varie en fonction de l'intensité lumineuse qu'elle reçoit. C'est donc un capteur résistif.
La sortie de la photorésistance est branchée sur une des entrées analogiques de la carte Arduino (entrée A5).

L'objectif est de faire clignoter une DEL à une fréquence dépendant de la lumière ambiante.
Pour cela, on va faire varier le délai entre 2 allumages de la DEL en fonction de la tension de l'entrée A5 et donc de l'intensité lumineuse reçue par la photorésistance.
Quand l'intensité lumineuse reçue par la photorésistance diminue, la fréquence de clignotement augmente

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

. Programme en Python (**Projet2\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 = 11
PinSensor = 5
ValSensor = 0


# Connexion à l'Arduino

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)


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

ArduinoIterator = Iterateur(board)
PinAnalogInput = AnalogInput(board, PinSensor)
time.sleep (0.5)

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

# Boucle principale du programme

while True:
    try:
        ValSensor = PinAnalogInput.read()
        DigitalWrite(board, PinLED, 1)
        time.sleep(ValSensor)
        DigitalWrite(board, PinLED, 0)
        time.sleep(ValSensor)

    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 = 11** (constante correspondant au n° de la broche sur laquelle la DEL choisie est connectée)

. **PinSensor = 5** (cst correspondant au n° de la broche A5 sur laquelle la photorésistance est connectée)

. **ValSensor = 0** (variable pour stocker la valeur de la tension de la broche de la photorésistance)

. **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)

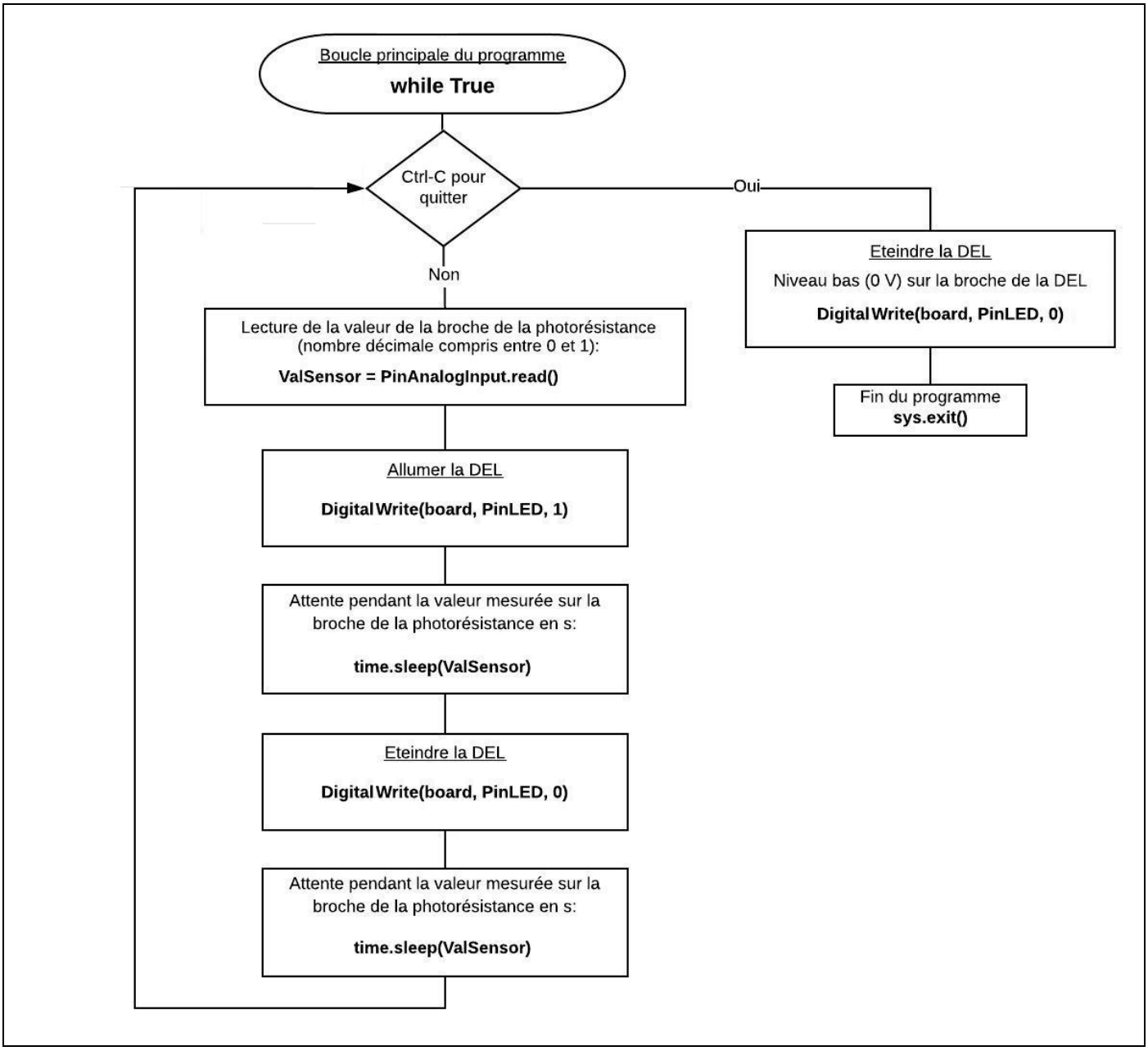
. Si la connexion à l'Arduino est réussie :

- Lancement de l'itérateur : **ArduinoIterateur = Iterateur(board)**,
- Déclaration de la broche de la photorésistance en entrée analogique :

PinAnalogInput = AnalogInput(board, PinSensor)

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

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

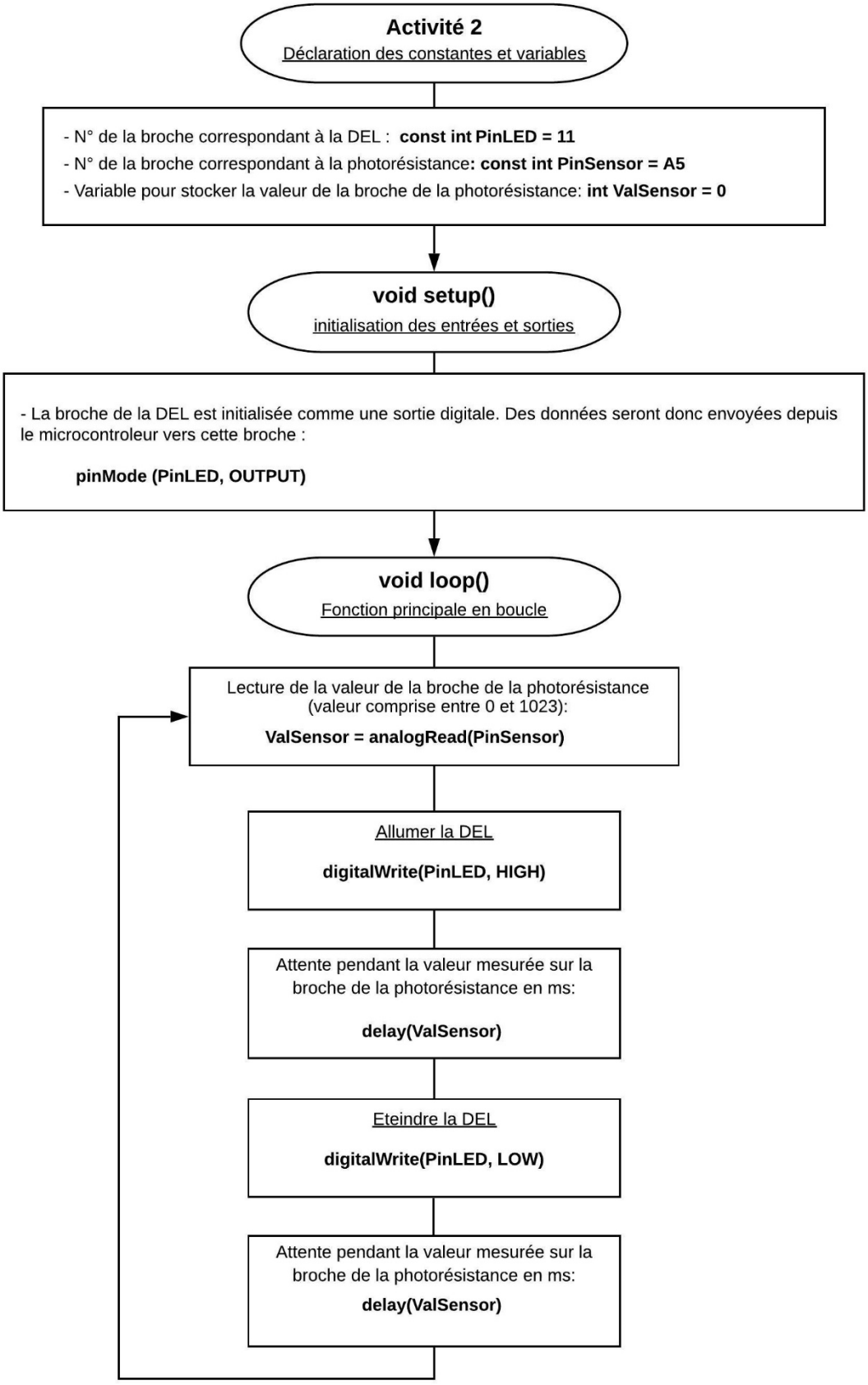

Boucle principale du programme
while True
Ctrl-C pour quitter
Oui
Non
Eteindre la DEL
Niveau bas (0 V) sur la broche de la DEL
DigitalWrite(board, PinLED, 0)
Fin du programme
sys.exit()
Lecture de la valeur de la broche de la photorésistance (nombre décimale compris entre 0 et 1):
ValSensor = PinAnalogInput.read()
Allumer la DEL
DigitalWrite(board, PinLED, 1)
Attente pendant la valeur mesurée sur la broche de la photorésistance en s:
time.sleep(ValSensor)
Eteindre la DEL
DigitalWrite(board, PinLED, 0)
Attente pendant la valeur mesurée sur la broche de la photorésistance en s:
time.sleep(ValSensor)

. **Programme en langage Arduino (Projet2\Activity2\INO\Activity2.ino)**

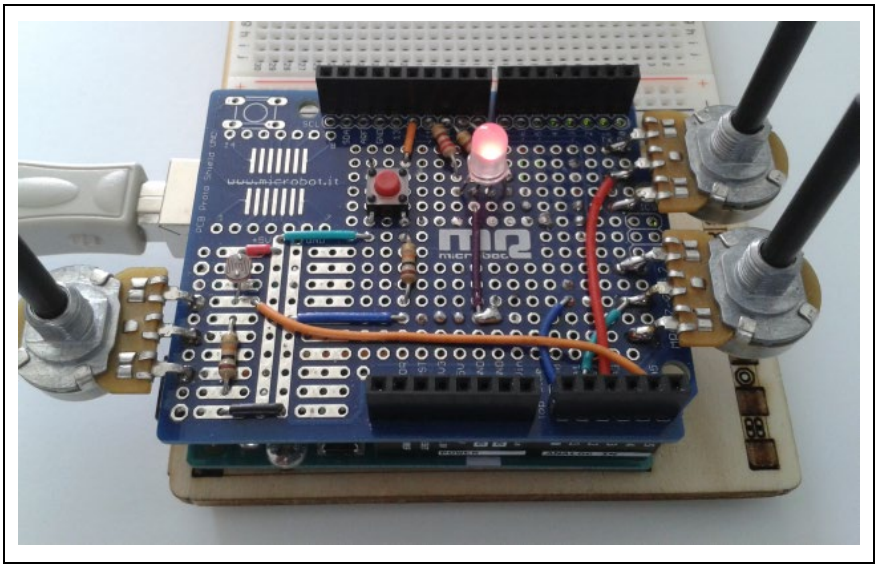
Activity2

```
// Déclaration des constantes et variables

const int PinLED = 11;
const int PinSensor = A5;
int ValSensor = 0;


// Initialisation des entrées et sorties

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


// Fonction principale en boucle

void loop() {
  ValSensor = analogRead(PinSensor);
  digitalWrite(PinLED, HIGH);
  delay(ValSensor);
  digitalWrite(PinLED, LOW);
  delay(ValSensor);
}
```

Déroulement du programme :



- Activité 3 : Réglage de la luminosité d'une DEL à un niveau fixé par une entrée analogique



Dans cette activité, selon le même principe que l'activité précédente, on va faire varier la luminosité d'une DEL en fonction de l'intensité lumineuse reçue par la photorésistance (la luminosité de la DEL sera inversement proportionnelle à l'intensité lumineuse reçue) :

- La DEL est allumée ou éteinte en appuyant sur le bouton-poussoir,
- La luminosité de la DEL varie en fonction de la tension de l'entrée A5.

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

. Programme en Python (**Projet2\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 = 11
PinButton = 12
PinSensor = 5
ValSensor = 0
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

ArduinoIterator = Iterateur(board)
PinDigitalInput = DigitalInput(board, PinButton)
PinAnalogInput = AnalogInput(board, PinSensor)
PinAnalogOutput = AnalogOutput(board, PinLED)
time.sleep (0.5)

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


# Boucle principale du programme

while True:
    try:
        ValButton = PinDigitalInput.read()
        ValSensor = PinAnalogInput.read()
        time.sleep(.01)

        if ValButton == 1 and OldValButton == 0:
            State = 1 - State

        OldValButton = ValButton

        if State == 1:
            PinAnalogOutput.write(1 - ValSensor)
        else:
            PinAnalogOutput.write(0)

    except KeyboardInterrupt:
        PinAnalogOutput.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.

- Déclaration des constantes et variables :

. **PinLED = 11** (constante correspondant au n° de la broche sur laquelle la DEL choisie est connectée)

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

. **PinSensor = 5** (cst correspondant au n° de la broche A5 sur laquelle la photorésistance est connectée)

. **ValSensor = 0** (variable pour stocker la valeur de la tension de la broche de la photorésistance)

. **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)**,

- Déclaration de la broche de la photorésistance en entrée analogique :

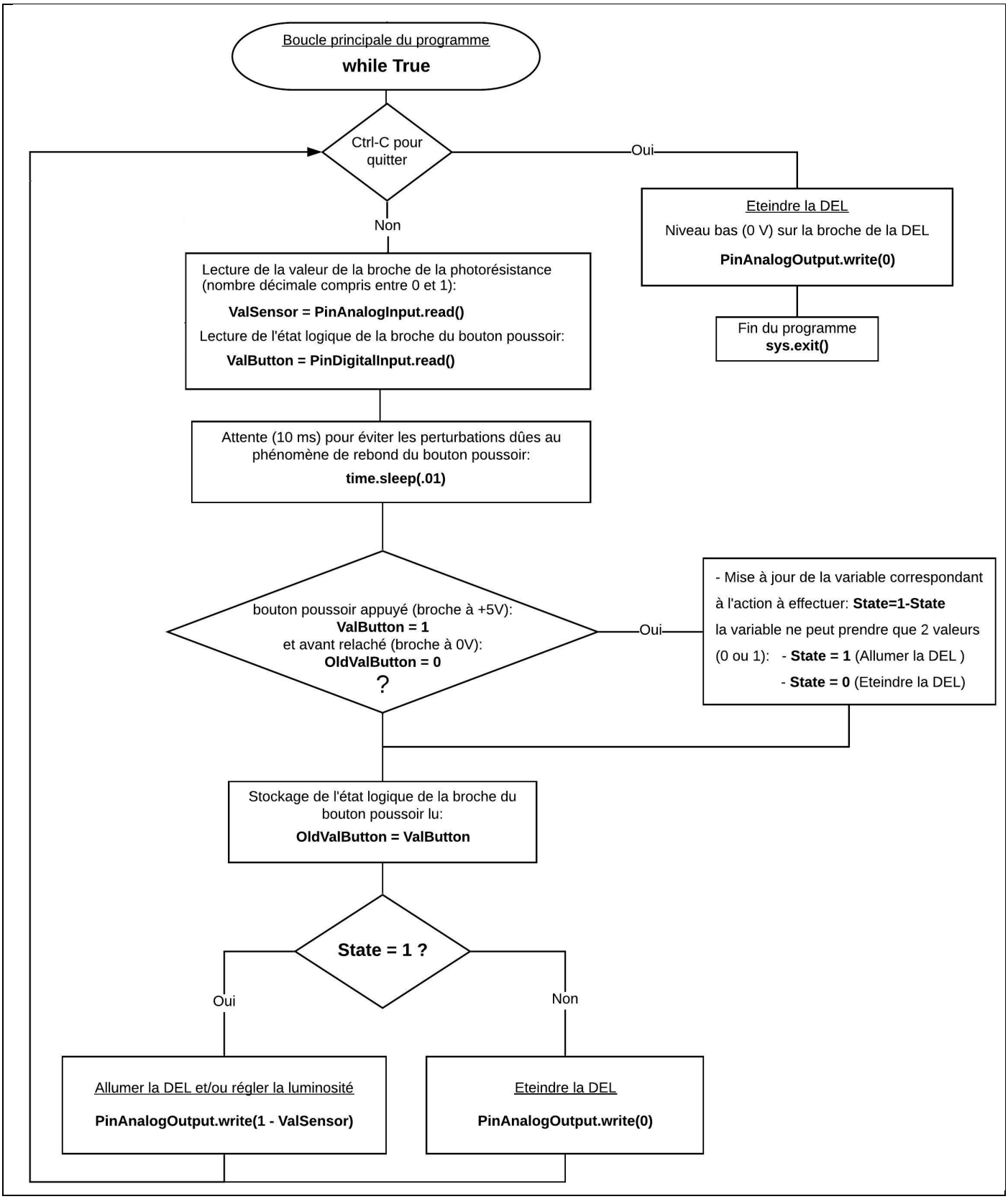
PinAnalogInput = AnalogInput(board, PinSensor)

- Déclaration de la broche de la DEL en sortie analogique :

PinAnalogOutput = AnalogOutput(board, PinLED),

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

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

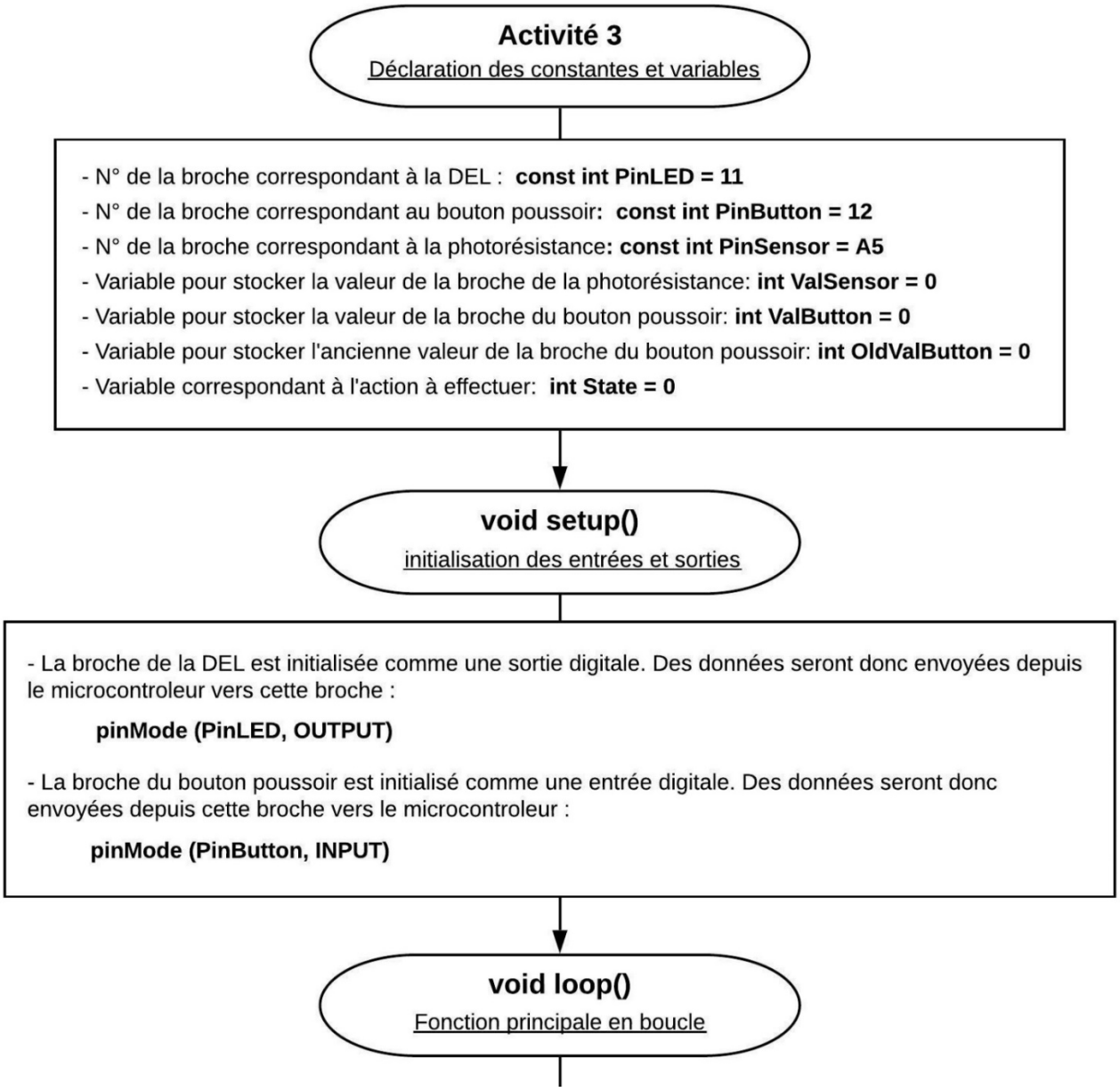


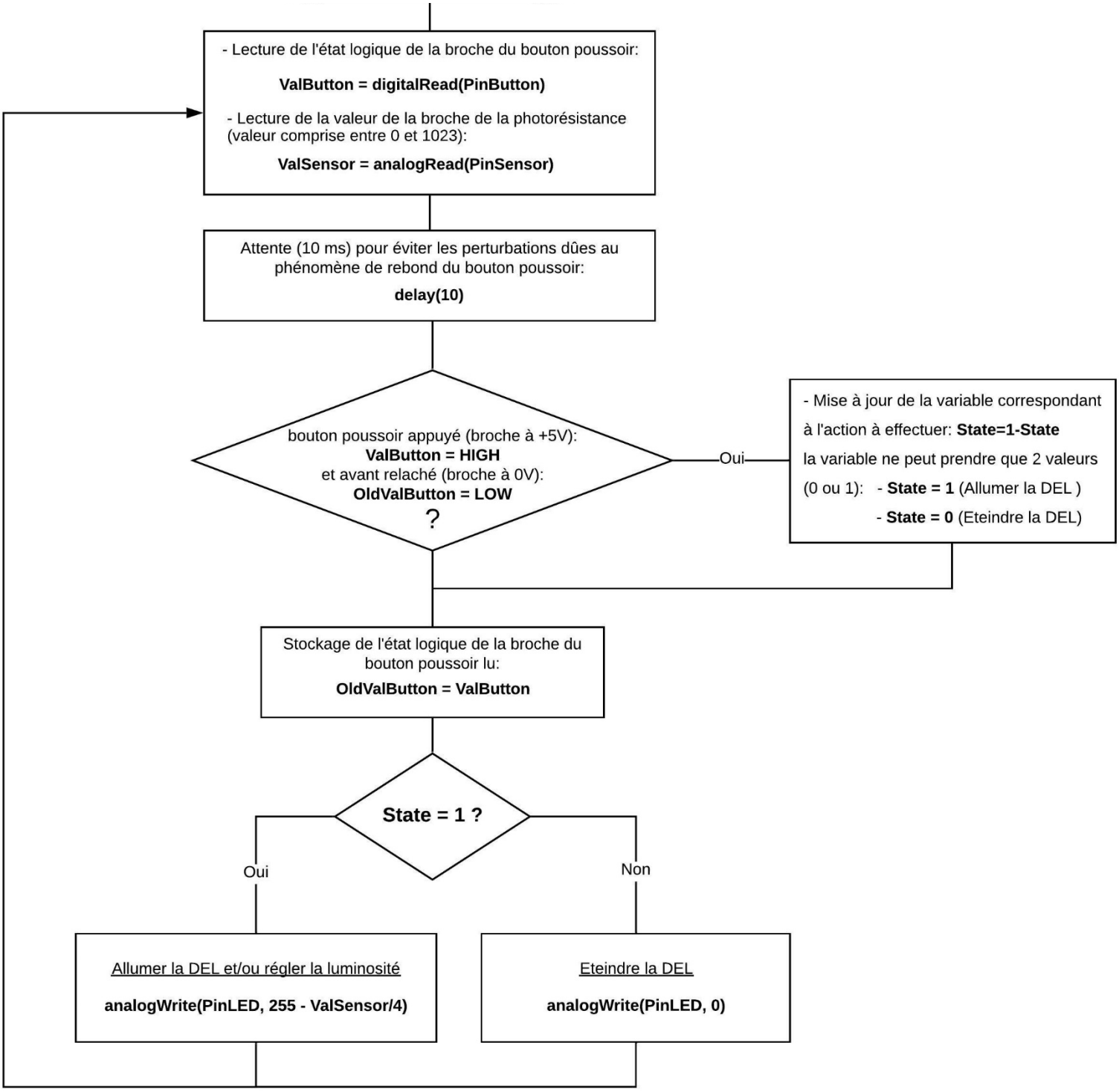
. Programme en langage Arduino (Projet2\Activity3\INO\Activity3.ino)

Activity3

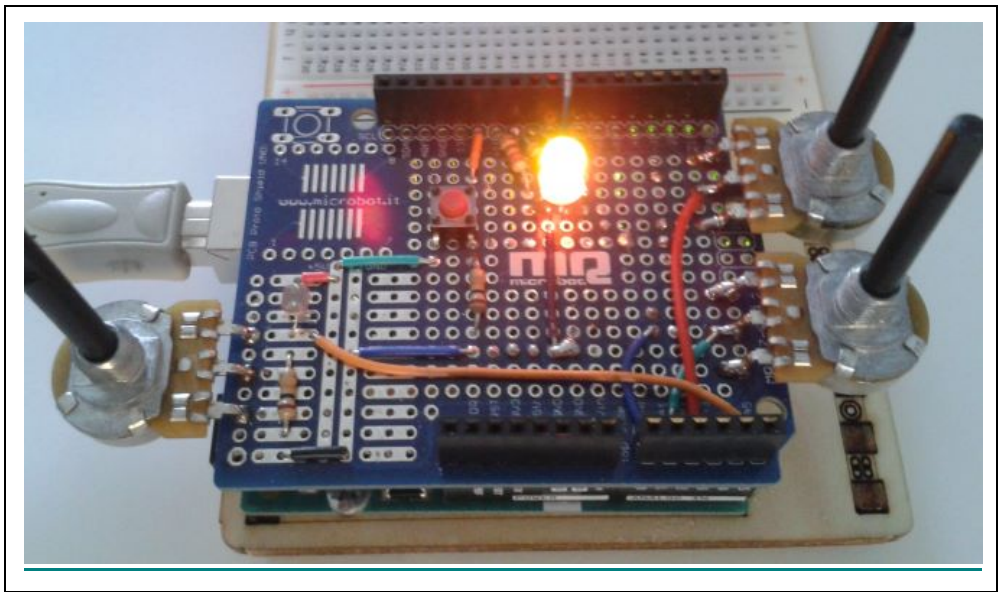
```
// Déclaration des constantes et variables

const int PinLED = 11;
const int PinSensor = A5;
const int PinButton = 12;

int ValButton = 0;
int ValSensor = 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);
  ValSensor = analogRead(PinSensor);
  delay(10);

  if ((ValButton == HIGH) && (OldValButton == LOW)) {
  State = 1 - State;
  }

  OldValButton = ValButton;

  if (State == 1) {
    analogWrite(PinLED, 255 - ValSensor/4);
  }
  else {
    analogWrite(PinLED, 0);
  }
}
```

Déroulement du programme :




- Lecture de l'état logique de la broche du bouton poussoir:
ValButton = digitalRead(PinButton)
- Lecture de la valeur de la broche de la photorésistance (valeur comprise entre 0 et 1023):
ValSensor = analogRead(PinSensor)
Attente (10 ms) pour éviter les perturbations dûes au phénomène de rebond du bouton poussoir:
delay(10)
bouton poussoir appuyé (broche à +5V):
ValButton = HIGH
et avant relaché (broche à 0V):
OldValButton = LOW
?
Oui
- Mise à jour de la variable correspondant à l'action à effectuer: State=1-State
la variable ne peut prendre que 2 valeurs (0 ou 1): - State = 1 (Allumer la DEL)
- State = 0 (Eteindre la DEL)
Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Oui
Non
Allumer la DEL et/ou régler la luminosité
analogWrite(PinLED, 255 - ValSensor/4)
Eteindre la DEL
analogWrite(PinLED, 0)

- Activité 4 : DEL RVB - Synthèse additive des couleurs



Cette activité est une application directe des activités précédentes. Nous allons utiliser 3 potentiomètres respectivement connectés aux entrées analogiques A2, A1 et A0 de l'Arduino pour régler les luminosités des DELs Rouge, Verte et Bleue d'une DEL RVB à cathode commune.

En effet, une DEL RVB à cathode commune dispose de 4 broches, 1 cathode et 3 anodes. Chaque anode correspond à une couleur (Rouge, Vert et Bleu).
En modulant les signaux sur les anodes, il est possible d'obtenir de multiples couleurs avec la DEL RVB. C'est le principe de la synthèse additive des couleurs des pixels des écrans d'ordinateur ou de télévision.

Le code de l'activité permet :

- Dans un premier temps, d'allumer la DEL RVB en appuyant sur le bouton poussoir,
- de faire varier la luminosité des DELs en fonction de la tension des entrées A2, A1 et A0,
- d'éteindre la DEL RVB en appuyant de nouveau sur le bouton poussoir.

Rappel sur la modélisation informatique de la couleur - Synthèse additive des couleurs

. Système RGB (Red, Green, Blue) ou RVB (Rouge, Vert, Bleu)

C'est le principe de la synthèse additive des couleurs des écrans d'ordinateurs.

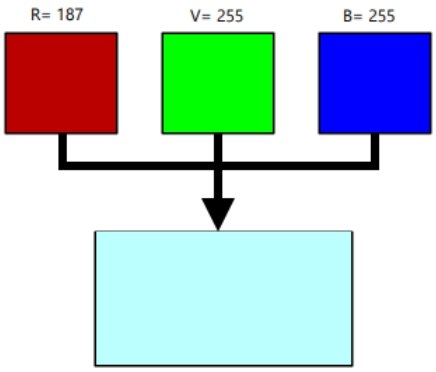
Toute couleur est obtenue en ajoutant différentes quantités de rouge, de vert et de bleu qui sont les seules couleurs dont on dispose à la base.

En effet, Les écrans d'ordinateur (cathodique, LCD), et d'une manière générale les systèmes de formation d'image numérique, fonctionnent sur le principe du mélange additif. Chaque pixel de l'écran est constitué de trois cellules, une verte, une bleue et une rouge. L'intensité lumineuse émise par ces cellules est ajustée pour produire la couleur voulue

Les images numériques destinées à l'affichage sur ces écrans sont codées en RVB. Elles comportent une couche rouge, une couche verte et une couche bleue. Chaque couche, le plus souvent codée sur 8 bits (valeurs de 0 à 255) représente le niveau d'intensité qui doit être délivré par la cellule correspondante sur l'écran.

Ces quantités de rouge, de vert et de bleu peuvent être exprimées sous forme de pourcentage (entre 0 et 1) ou sous forme de nombres (généralement compris entre 0 et 255) :

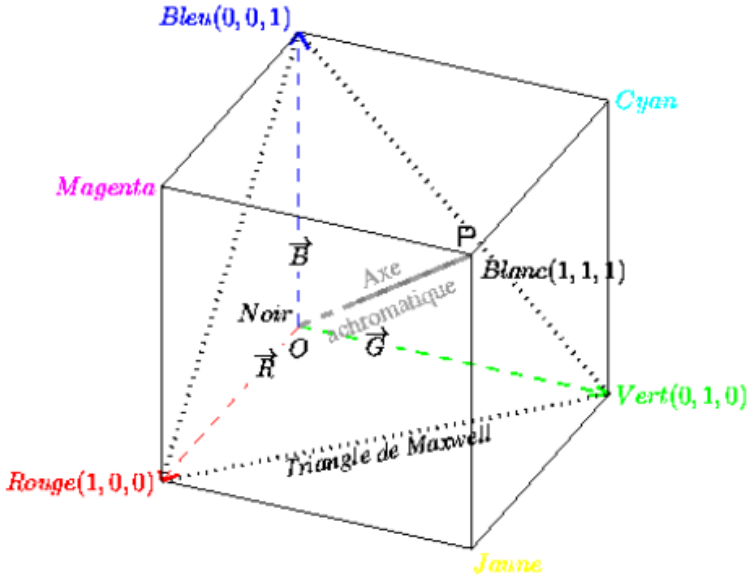
	R	V	B
Noir	0	0	0
Bleu	0	0	255
Vert	0	255	0
Cyan	0	255	255
Rouge	255	0	0
Magenta	255	0	255
Jaune	255	255	0
Blanc	255	255	255



Si on note R, V et B les trois couleurs primaires utilisées, une couleur C peut s'écrire comme la combinaison linéaire :

$\vec{C} = r\vec{R} + v\vec{V} + b\vec{B}$ (r, v, b sont les intensités relatives variables des trois couleurs)

Le système RGB peut être représenté sous la forme d'un cube (espace des couleurs et triangle de Maxwell) :

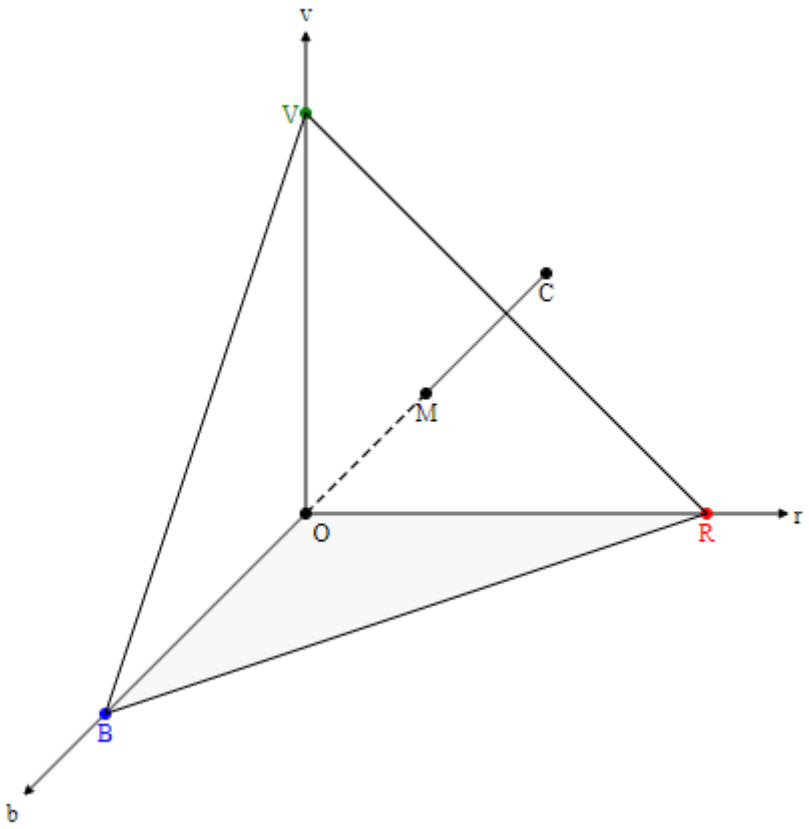


Les coefficients réels (r, v, b) sont positifs. Par convention, on peut les choisir dans l'intervalle [0,1].

Pour représenter géométriquement les couleurs, on représente les primaires par une base orthonormée et on place le noir à l'origine O. L'ensemble des couleurs est alors contenu dans un cube de côté 1. Le point P (1,1,1) est le sommet du cube opposé à O.

Soit C (r, g, b) un point représentant une couleur. Lorsqu'on multiplie tous les coefficients (r, g, b) par une même constante, on ne change pas la qualité d'une couleur, que l'on appelle sa chromaticité, mais seulement sa luminosité. Ainsi, toutes les couleurs de la droite OC sont perçues avec la même chromaticité. En particulier, les points de la diagonale OP sont des points neutres, c'est-à-dire des gris, obtenus par un mélange égal des trois primaires. Le point O (0,0,0) est le noir, le point P (1,1,1) est le blanc.

Considérons le plan d'équation r+v+b=1 qui passe par les points R (1,0,0), V (0,1,0) et B (0,0,1). Ces 3 points définissent dans ce plan un triangle appelé triangle de Maxwell. L'intersection de la droite OC avec ce triangle est le point M. La position de ce point dans le triangle suffit à déterminer la chromaticité de la couleur. D'un point de vue chromatique, toutes les couleurs sont donc représentables dans le triangle de Maxwell. Les coordonnées trichromatiques (r, v, b) vérifient alors r+v+b=1.

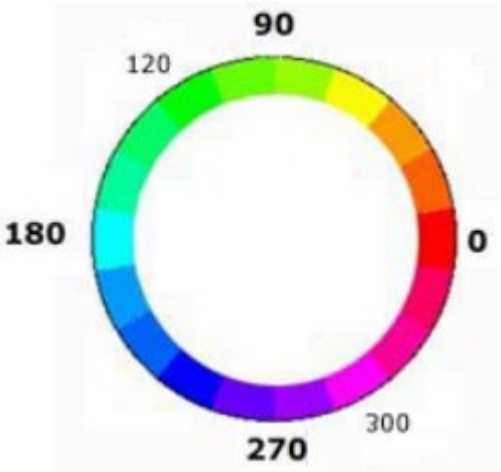


. <u>Système TSL (ou HSL)</u>

Toute couleur est décrite par :

- une Teinte (Hue)
- une Saturation (Saturation)
- une Luminance (Luminance)

La teinte permet de déterminer la couleur souhaitée à partir des couleurs rouge, vert et bleu. Elle est exprimée par un nombre qui est sa position angulaire sur le cercle chromatique. La teinte est donc un angle dans l'intervalle [0,360] :



Ex : rouge : 0 ; vert : 120 ; magenta : 300.

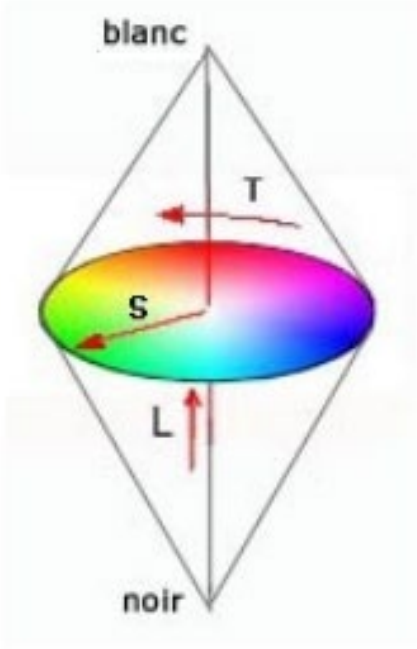
La saturation mesure l'intensité ou la pureté d'une couleur, c'est-à-dire le pourcentage de couleur pure par rapport au blanc. La saturation permet donc de distinguer une couleur vive d'une couleur pastel.

La saturation est représentée sur le rayon du cercle, par un pourcentage de pureté : elle est maximale sur le cercle (100% ou 1) et minimale au centre (0 = gris)

La luminance permet de définir la part de noir ou de blanc dans la couleur désirée (couleur claire ou sombre).

L'ensemble des couleurs est représenté à l'intérieur d'un double cône. La luminance varie sur l'axe vertical du double cône (axe des gris) du noir en bas au blanc, en haut.

La luminance est exprimée par un pourcentage : de 0% (noir) à 100% (blanc).



En résumé :

- Une augmentation de la luminance d'une couleur la fait tendre vers le blanc.

- Une diminution de sa luminance la fait tendre vers le noir.

- Une diminution de la saturation fera tendre cette couleur vers le gris (axe du double cône).

. Conversion RGB/TSL

On définit la luminance comme le maximum de r, g, b. Voici, l'algorithme qui permet de calculer la teinte (T), la saturation (S) et la luminance (L) :

$$Max = max(r, g, b)$$
$$Min = min(r, g, b)$$
$$C = Max - Min$$
$$L = Max$$
$$S = \frac{C}{L}$$
$$T = 60\frac{v - b}{C} \ [360] \ si \ \ Max = r$$
$$T = 120 + 60\frac{b - r}{C} \ \ si \ \ Max = v$$
$$T = 240 + 60\frac{r - v}{C} \ \ si \ \ Max = b$$

Si C=0, il s'agit d'une couleur neutre (trois couleurs égales) pour laquelle la teinte n'est pas définie et la saturation nulle. Les valeurs de S et L sont dans l'intervalle [0,1]. La teinte est un angle dans l'intervalle [0,360].

. Conversion TSL/RGB

D'après l'algorithme de conversion RGB/TSL, on a C=LS et Min=L-C et suivant l'intervalle dans lequel la teinte T se trouve, on calcul les valeurs de r, v, et b :

$T \in 300, 360$	$r = L$	$v = L - C$	$b = v + C(360 - T)/60$
$T \in 0, 60$	$r = L$	$b = L - C$	$v = b + C(T/60)$
$T \in 60, 120$	$v = L$	$b = L - C$	$r = b + C(120 - T)/60$
$T \in 120, 180$	$v = L$	$r = L - C$	$b = r + C(T - 120)/60$
$T \in 180, 240$	$b = L$	$r = L - C$	$v = r + C(240 - T)/60$
$T \in 240, 300$	$b = L$	$v = L - C$	$r = v + C(T - 240)/60$

. Programme en Python (**Projet2\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 = 11
PinLEDB = 10
PinLEDV = 9
PinPOTR = 2
PinPOTV = 1
PinPOTB = 0
PinButton = 12

BrightnessR = 0
BrightnessB = 0
BrightnessV = 0

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

ArduinoIterator = Iterateur(board)

PinDigitalInput = DigitalInput(board, PinButton)

PinAnalogInputR = AnalogInput(board, PinPOTR)
PinAnalogInputV = AnalogInput(board, PinPOTV)
PinAnalogInputB = AnalogInput(board, PinPOTB)

PinAnalogOutputR = AnalogOutput(board, PinLEDR)
PinAnalogOutputV = AnalogOutput(board, PinLEDV)
PinAnalogOutputB = AnalogOutput(board, PinLEDB)

time.sleep (0.5)

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


# Boucle principale du programme

while True:
    try:
        ValButton = PinDigitalInput.read()
        time.sleep(.01)

        if ValButton == 1 and OldValButton == 0:
            State = 1 - State

        OldValButton = ValButton
```

```
        if State==1:
            BrightnessR = PinAnalogInputR.read()
            BrightnessV = PinAnalogInputV.read()
            BrightnessB = PinAnalogInputB.read()

            PinAnalogOutputR.write(1 - BrightnessR)
            PinAnalogOutputV.write(1 - BrightnessV)
            PinAnalogOutputB.write(1 - BrightnessB)
        else:
            PinAnalogOutputR.write(0)
            PinAnalogOutputV.write(0)
            PinAnalogOutputB.write(0)

    except KeyboardInterrupt:
        PinAnalogOutputR.write(0)
        PinAnalogOutputV.write(0)
        PinAnalogOutputB.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 = 11** (constante correspondant au n° de la broche sur laquelle la DEL rouge est connectée)

. **PinLEDV = 9** (constante correspondant au n° de la broche sur laquelle la DEL verte est connectée)

. **PinLEDB = 10** (constante correspondant au n° de la broche sur laquelle la DEL bleue est connectée)

. PinPOTR = 2 (constante correspondant au n° de la broche sur laquelle le potentiomètre de le DEL rouge est connecté)

. PinPOTV = 1 (constante correspondant au n° de la broche sur laquelle le potentiomètre de le DEL verte est connecté)

. PinPOTB = 0 (constante correspondant au n° de la broche sur laquelle le potentiomètre de le DEL bleue est connecté)

. **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)

. **BrightnessR = 0** (variable correspondant à la luminosité de la DEL rouge)

. BrightnessB = 0 (variable correspondant à la luminosité de la DEL bleue)

. BrightnessV = 0 (variable correspondant à la luminosité de la DEL verte)

. **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)

- Déclaration des broches des potentiomètres en entrées analogiques :

PinAnalogInputR = AnalogInput(board, PinPOTR)

PinAnalogInputV = AnalogInput(board, PinPOTV)

PinAnalogInputB = AnalogInput(board, PinPOTB)

- Déclaration des broches de la DEL RVB en sorties analogiques :

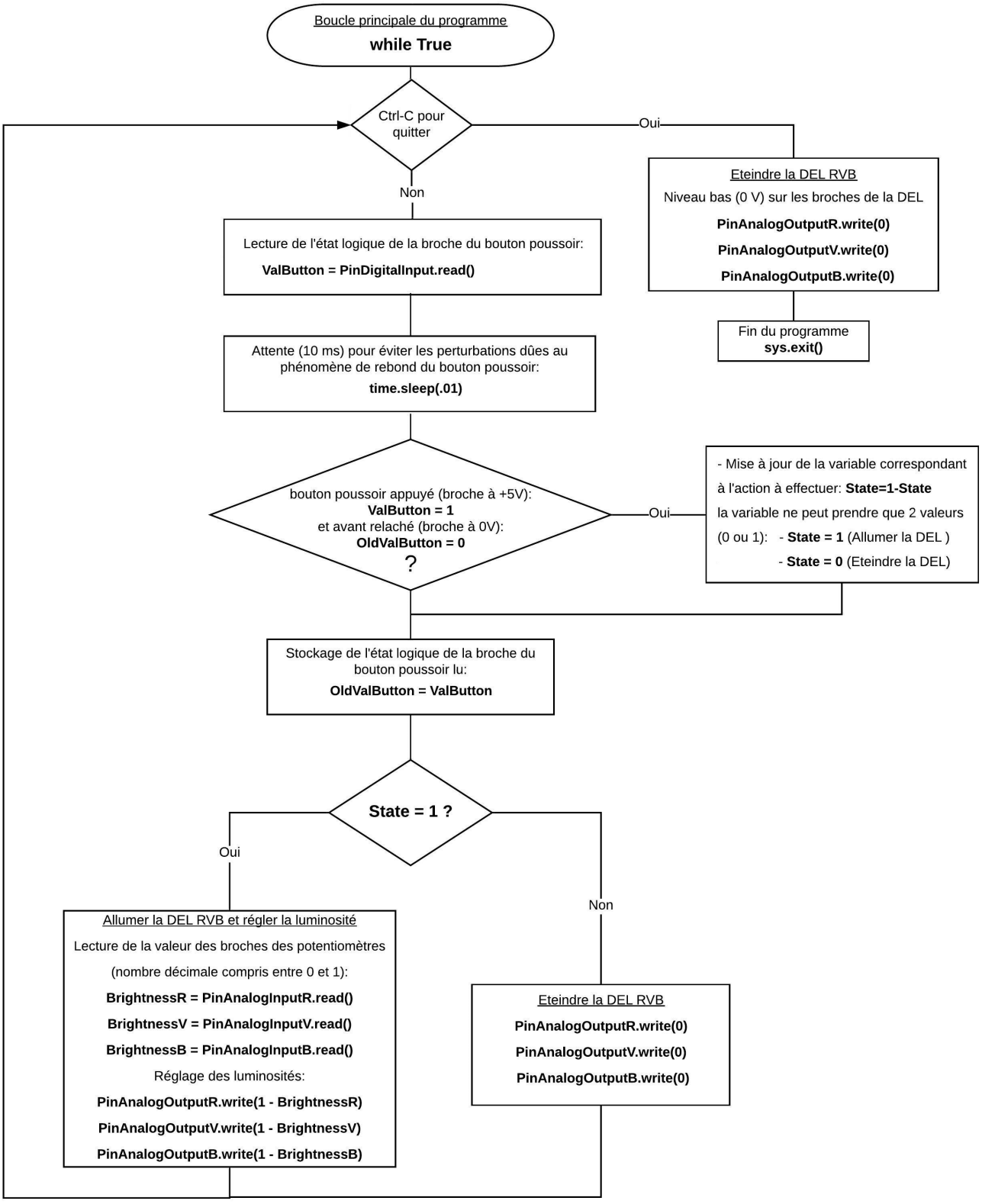
PinAnalogOutputR = AnalogOutput(board, PinLEDR)

PinAnalogOutputV = AnalogOutput(board, PinLEDV)

PinAnalogOutputB = AnalogOutput(board, PinLEDB)

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

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

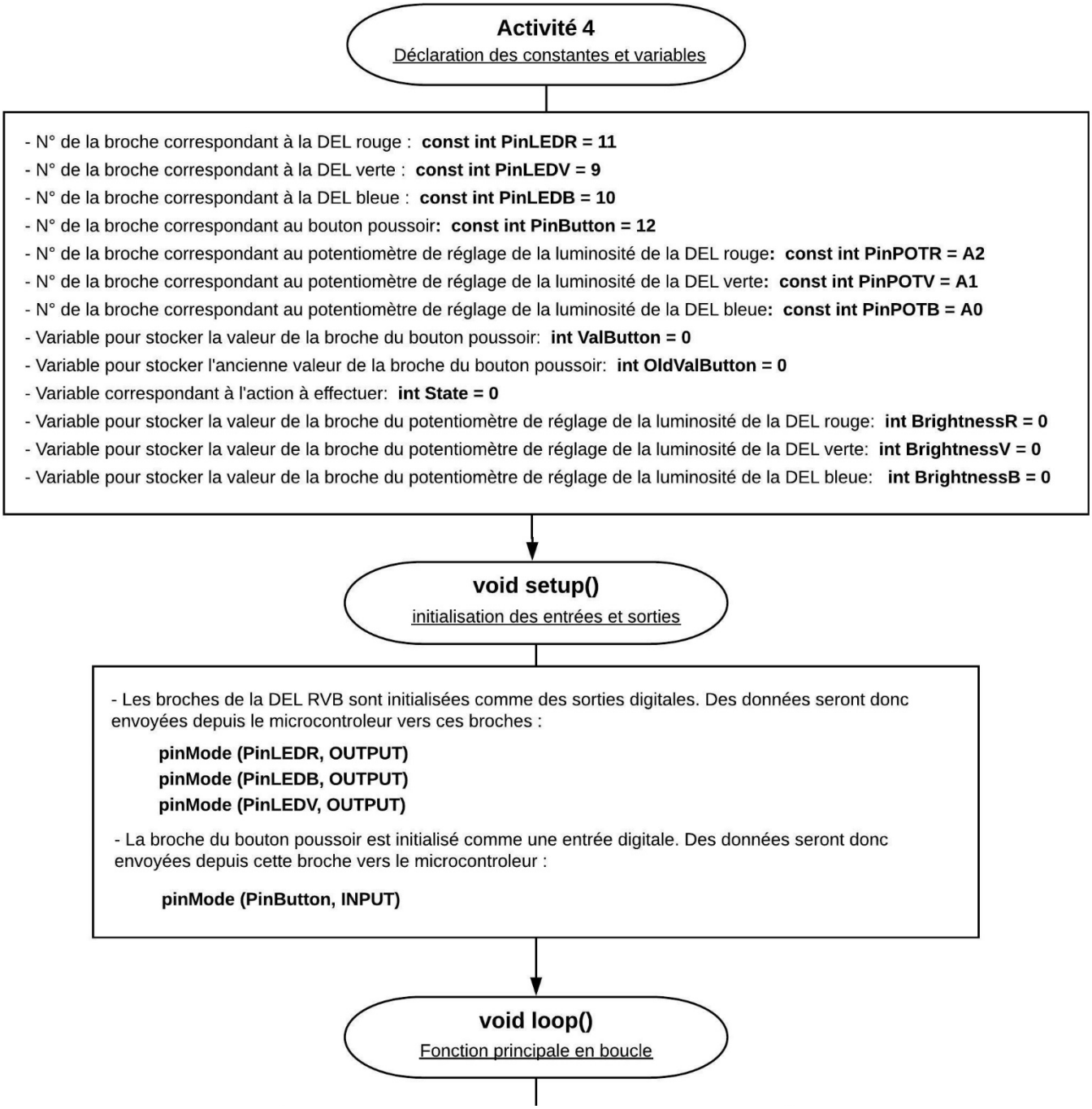


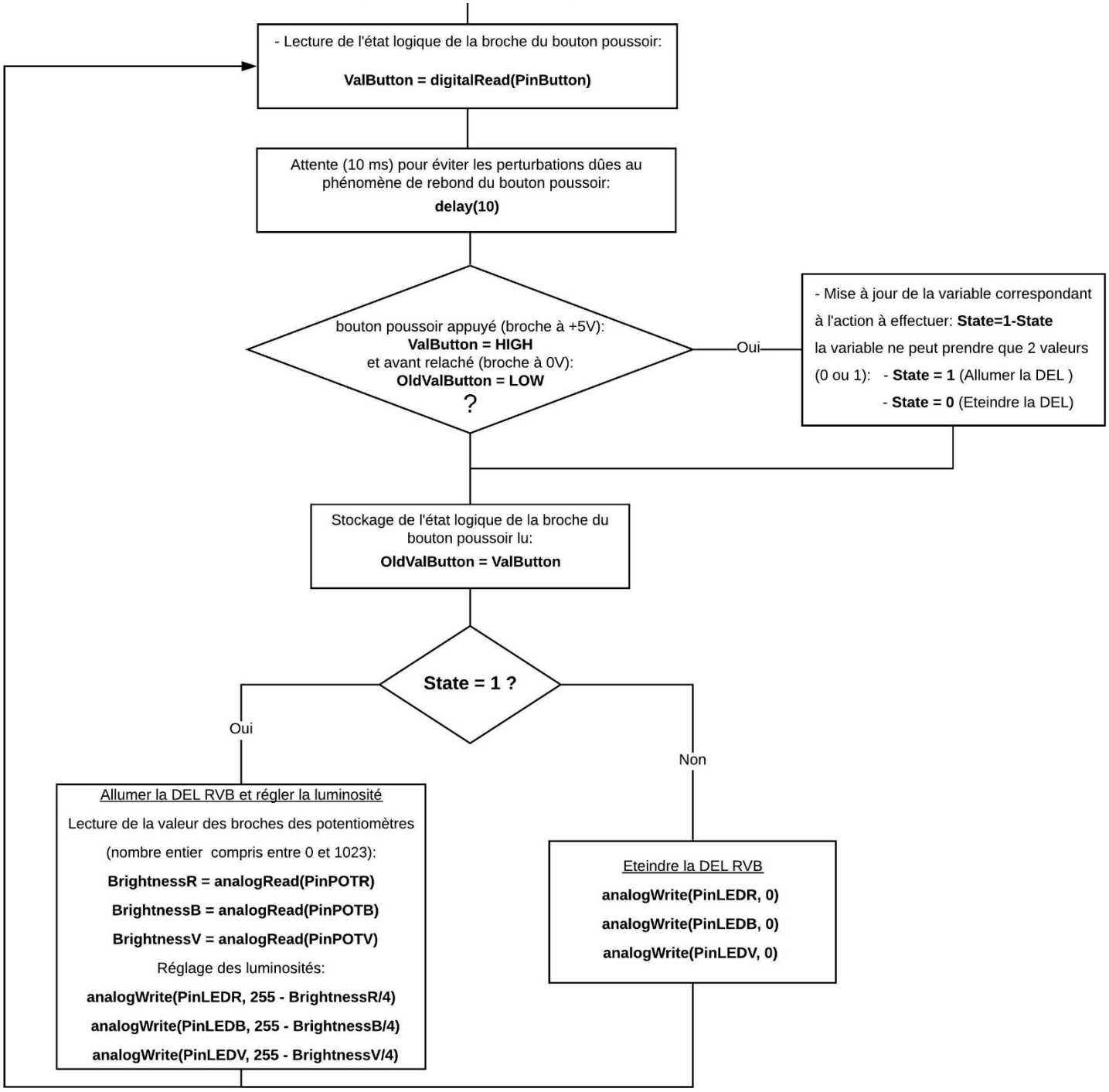
. Programme en langage Arduino (Projet2\Activity4\INO\Activity4.ino)

Activity4

```
// Déclaration des constantes et variables

const int PinLEDR = 11;
const int PinLEDB = 10;
const int PinLEDV = 9;
const int PinPOTR = A2;
const int PinPOTB = A0;
const int PinPOTV = A1;
const int PinButton = 12;

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

int BrightnessR = 0;
int BrightnessB = 0;
int BrightnessV = 0;


// Initialisation des entrées et sorties

void setup() {
  pinMode (PinLEDR, OUTPUT);
  pinMode (PinLEDB, OUTPUT);
  pinMode (PinLEDV, 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) {
    BrightnessR = analogRead(PinPOTR);
    BrightnessB = analogRead(PinPOTB);
    BrightnessV = analogRead(PinPOTV);
    analogWrite(PinLEDR, 255 - BrightnessR/4);
    analogWrite(PinLEDB, 255 - BrightnessB/4);
    analogWrite(PinLEDV, 255 - BrightnessV/4);
  }
  else {
    analogWrite(PinLEDR, 0);
    analogWrite(PinLEDB, 0);
    analogWrite(PinLEDV, 0);
  }
}
```

Déroulement du programme :




- Lecture de l'état logique de la broche du bouton poussoir:
ValButton = digitalRead(PinButton)
Attente (10 ms) pour éviter les perturbations dûes au phénomène de rebond du bouton poussoir:
delay(10)
bouton poussoir appuyé (broche à +5V):
ValButton = HIGH
et avant relaché (broche à 0V):
OldValButton = LOW
?
Oui
- Mise à jour de la variable correspondant à l'action à effectuer: **State=1-State**
la variable ne peut prendre que 2 valeurs (0 ou 1): - **State = 1** (Allumer la DEL)
- **State = 0** (Eteindre la DEL)
Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Oui
Non
Allumer la DEL RVB et régler la luminosité
Lecture de la valeur des broches des potentiomètres (nombre entier compris entre 0 et 1023):
BrightnessR = analogRead(PinPOTR)
BrightnessB = analogRead(PinPOTB)
BrightnessV = analogRead(PinPOTV)
Réglage des luminosités:
analogWrite(PinLEDR, 255 - BrightnessR/4)
analogWrite(PinLEDB, 255 - BrightnessB/4)
analogWrite(PinLEDV, 255 - BrightnessV/4)
Eteindre la DEL RVB
analogWrite(PinLEDR, 0)
analogWrite(PinLEDB, 0)
analogWrite(PinLEDV, 0)