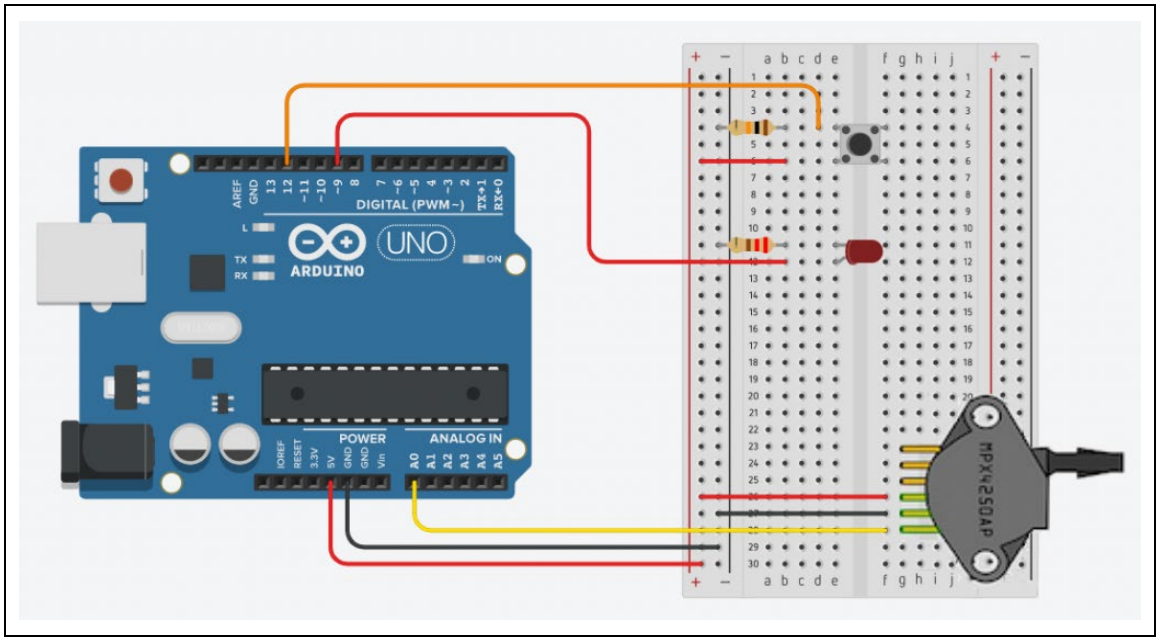
Projet 6 - Pression : Mesurer & Exploiter

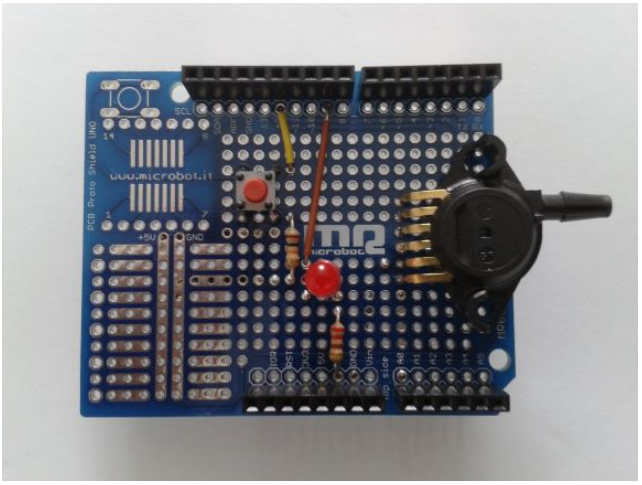
Dans le domaine des sciences, une autre grandeur physique fréquemment utilisée est la pression exprimée en Pascal (Pa).

Pour mesurer cette grandeur avec un Arduino, on utilisera le capteur de pression MPX4250AP dont la sensibilité est de 20 mv/kPa pour une plage de mesure allant de 20 à 250 kPa (soit, une tension de sortie du capteur entre 0,2 V et 4,8 V). Ce capteur est donc parfaitement adapté à une utilisation avec un Arduino.

Toutes les activités de mesure de pression et d'exploitation seront réalisées avec le circuit suivant :



- Liste des composants :



. 1 capteur de pression MPX4250AP

. 1 DEL rouge

. 1 résistance de 220 Ω (résistance de protection de la DEL)

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

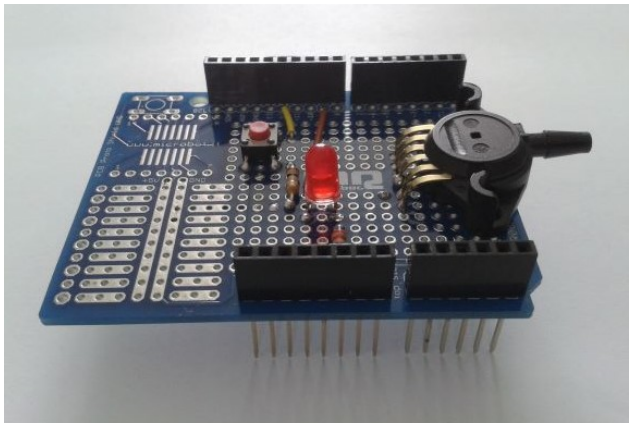
. 1 bouton poussoir

. 1 plaque d'essais

. Fils de connexion

- Protocole de communication:

. Firmata Standard



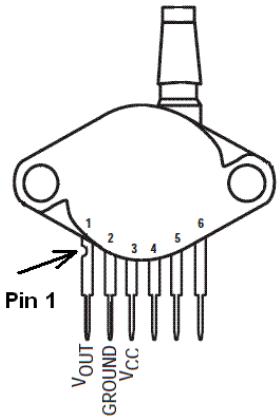
Le circuit sur un ”shield” pour Arduino Uno

Rappels :

. Capteur de pression MPX4250AP :

(Vue de dessus)

Le capteur de pression absolue MPX4250AP dispose de 6 broches, mais seules trois bornes sont utilisées pour le connecter à une carte Arduino :



(Vue de dessous)

VCC : +5 V
GROUND : masse
Vout : signal de sortie sur une entrée analogique de la carte.

Caractéristiques :

. Tension de fonctionnement : de 4,85V à 5,35V

. Courant circuit alimentation maximal : 10 mA

. Sensibilité : **20 mV / kPa**

. Plage de mesure : 20 à 250 kPa

. Tension de sortie (Vout) : 0,2 V à 4,8 V

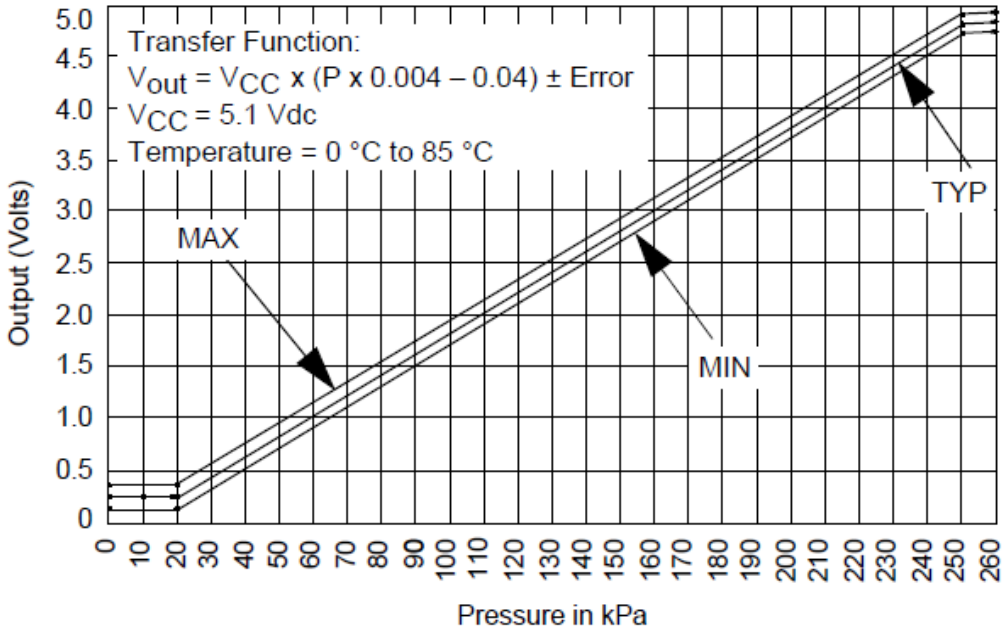
. Précision : ± 1,5% sur Vout

. Courant de charge maximal (signal de sortie) : 0,1 mA

. Temps de réponse : 1 ms

. Température de fonctionnement : 0 - 85 °C

La fiche technique du capteur, fourni par le constructeur, donne la courbe de transfert reliant la pression absolue P en kPa à la tension de sortie Vout en V :



On a donc :

$$V_{out} = 5{,}0 \times (P \times 0{,}004 - 0{,}04)$$

On obtient alors la pression absolue à partir de la mesure de la tension de sortie par :

$$P\ (en\ kPa) = \frac{V_{out}}{0,02} + 10$$

. **Firmata Standard**

Pour contrôler l'Arduino via le protocole de communication "**Firmata standard**", le programme Python utilise la bibliothèque "**PyFirmata**" (le sketch "**Firmata standard**" doit être téléversé dans la mémoire de l'Arduino au préalable).

Toutes les fonctions utiles à l'utilisation de "**PyFirmata**" (fonction de déclaration des entrées et sorties, d'itérateur, d'écritures…) ont été regroupées dans un fichier Python, nommé "**PyFirmataDef.Py**" que l'on peut importer dans tous les programmes, à condition que le fichier des fonctions soit dans le même dossier que le fichier du programme, avec l'instruction :

from PyFirmataDef import *

```
import pyfirmata


########## GESTION DES SORTIES NUMERIQUES ##########

# MODIFICATION DE L'ETAT LOGIQUE D'UNE SORTIE NUMERIQUE

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


########## GESTION DES ENTREES NUMERIQUES ##########

# DECLARATION D'UNE BROCHE EN ENTREE NUMERIQUE
# LECTURE DE L'ETAT LOGIQUE AVEC LA FONCTION DigitalInputPin.read()

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


########## GESTION DES SORTIES ANALOGIQUES ##########

# DECLARATION D'UNE BROCHE EN SORTIE ANALOGIQUE

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

# MODIFICATION DE LA VALEUR D'UNE SORTIE ANALOGIQUE

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


########## GESTION DES ENTREES ANALOGIQUES ##########

# DECLARATION D'UNE BROCHE EN ENTREE ANALOGIQUE
# LECTURE DE LA VALEUR DE L'ENTREE ANALOGIQUE AVEC LA FONCTION AnalogInputPin.read()

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


########## ITERATEUR ##########

# DECLARATION D'UN ITERATEUR POUR EVITER LA SATURATION DU PORT SERIE

def Iterateur(board):
    it = pyfirmata.util.Iterator(board)
    it.start()
    return it
```

. Connexion à l'Arduino avec le protocole de communication "Firmata Standard"

Dans les programmes de contrôle de l'Arduino par le protocole de communication "Firmata Satndard", le module "**ConnectToArduino.py**", contenant les fonctions de connexion avec l'Arduino, sera importé.

La connexion se déroulera en 2 étapes :

- Appel de la fonction de sélection du port COM du module "**ConnectToArduino.py**":

PortComArduino = SelectPortCOM()

Le nombre de port COM disponible est alors déterminé :

PortsCOM = list(serial.tools.list_ports.comports()

–> si nombre de port COM = 0 : message d'erreur,

–> si nombre de port COM = 1 : sélection de ce port COM pour la connexion,

–> si nombre de port COM > 1 : L'utilisateur doit sélectionner le bon port COM.

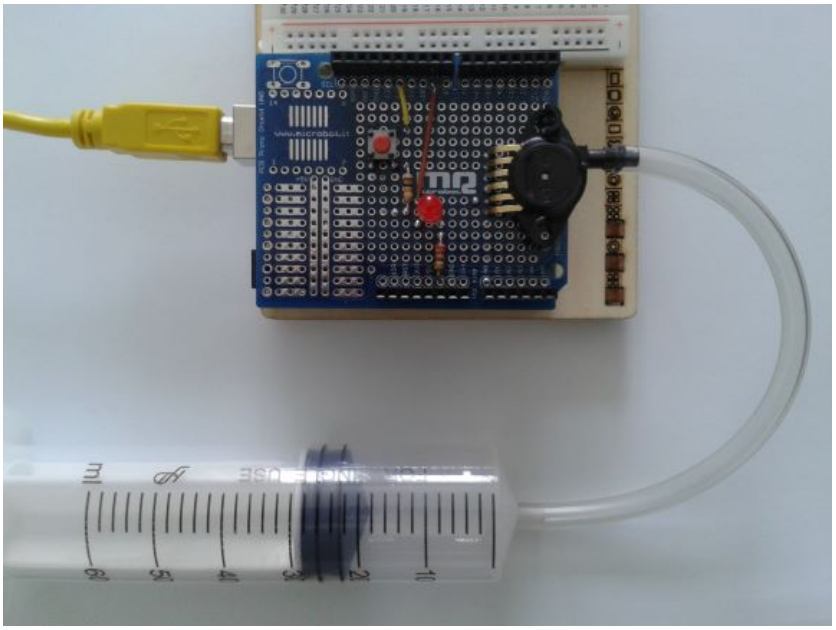
```
def SelectPortCOM():

    Nport=[]
    PortsCOM = list(serial.tools.list_ports.comports())
    for port_numero, description, address in PortsCOM:
        Nport.append(port_numero)


    if len(PortsCOM)== 0:
        print("Aucune carte Arduino n'a été détectée!")
        saisie = ""
        while saisie != "q":
            saisie = str(input("Entrez 'q' pour quitter: "))
        sys.exit()

    elif len(PortsCOM)== 1:
        PortComArduino = Nport[0]

    else:
        print("Liste des ports COM disponibles:\n")
        for i in range(len(PortsCOM)):
            print(i+1, ": ", PortsCOM[i])
        ChoixPort=False
        while ChoixPort==False:
            Choix = input("\nVeuillez indiquer le numéro du port de la carte Arduino:")
            try:
                Choix = int(Choix)
                assert Choix >= 1 and Choix <= len(PortsCOM)
                ChoixPort = True
            except AssertionError:
                print("Le numéro indiqué n'est pas entre 1 et", len(PortsCOM) ,"!")
                ChoixPort = False
            except:
                print("Vous n'avez pas saisi un numéro entre 1 et", len(PortsCOM) ,"!")
        PortComArduino = Nport[Choix-1]

    return PortComArduino
```

- Tentative d'ouverture du port COM sélectionné (**PortComArduino**) et de connexion à l'Arduino via le protocole **"Firmata Standard"** avec la fonction **"OpenPortCom"** du module **"ConnectToArduino.py"** :

board = OpenPortCom(PortComArduino)

```
def OpenPortCom(PortCom):

    try:
        board = pyfirmata.Arduino(PortCom)
        testConnect = board.get_firmata_version()

        assert testConnect != None

    except AssertionError:
        AffichMessageErreur(PortCom)

    except:
        AffichMessageErreur(PortCom)

    else:
        return board


def AffichMessageErreur(PortCom):

    print("Un problème s'est produit à l'ouverture du port.\n"
        "Vérifiez que le port utilisé par la carte Arduino est bien "+ PortCom + ",\n" +
        "et que le protocole de communication FIRMATA STANDARD a bien été téléversé.\n")
    saisie = ""
    while saisie != "q":
        saisie = str(input("Entrez 'q' pour quitter: "))
    sys.exit()
```

- Activité 1 : Mesure d'une pression absolue avec un capteur MPX4250AP



. Objectif

Dans cette activité, nous allons simplement mettre en œuvre l'utilisation d'un capteur de pression dont la sortie est reliée à l'entrée A0 de l'Arduino (Cf. circuit d'étude), pour afficher dans la fenêtre Python Shell ou le moniteur série, la tension mesurée et la pression correspondante après un appui sur le bouton poussoir. Un nouvel appui sur le bouton poussoir arrête les mesures.

La pression mesurée sera modifiée avec une seringue d'un volume utile de 60 mL fixée au capteur par l'intermédiaire d'un tuyau.
En déplaçant le piston, initialement placé sur la graduation 30 mL, on fera varier le volume de l'air enfermé dans le corps de la seringue et donc la pression appliquée sur le capteur.

. Le programme

Voici le code de l'activité en Python et en Langage Arduino :

. Programme en Python ("Projet6\Activity1\PY\Activity1.py")

```
# Importations des librairies et définition de fonctions

from PyFirmataDef import *
from ConnectToArduino import *
import time


# Déclaration des constantes et variables

PinSensor=0
PinButton= 12

ValSensor = 0
tension = 0
Pression = 0
OldPression = 0

ValButton = 0
OldValButton = 0
State = 0
OldState = 0


# Connexion à l'Arduino

print("\nConnexion à l'Arduino en cours...")

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)

InputPinSensor = AnalogInput(board, PinSensor)
InputPinButton = DigitalInput(board, PinButton)

ArduinoIterateur = Iterateur(board)
time.sleep(0.5)

print("\nConnexion à l'Arduino établie - Appuyez sur Ctrl-C pour quitter\n")
print("\nAppuyez sur le bouton poussoir pour commencer les mesures.\n");


# Boucle principale du programme

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

        OldValButton = ValButton;

        if State==1:
            if OldState == 0:
               print("\nMesure de la pression en cours.\n")
               print("Tension Entree A0 (en V) ; Pression (en kPa):")
               OldState=1

            ValSensor = InputPinSensor.read()
            tension = ValSensor*5.0
            Pression = (tension / 0.02) + 10

            if OldPression != Pression:
                print(round(tension,2), " ; ", round(Pression,1))
                OldPression = Pression

            time.sleep(0.1)

        else:
            if OldState == 1:
                print("\nFin des mesures.\n")
                OldState = 0

    except KeyboardInterrupt:
        board.exit()
        sys.exit(0)
```

Résultats dans la fenêtre Python Shell :

```
Connexion à l'Arduino en cours...

Connexion à l'Arduino établie - Appuyez sur Ctrl-C pour quitter


Appuyez sur le bouton poussoir pour commencer les mesures.


Mesure de la pression en cours.

Tension Entree A0 (en V) ; Pression (en kPa):
1.87  ;  103.3
1.86  ;  103.1
1.87  ;  103.3
1.86  ;  103.1
1.87  ;  103.3

Fin des mesures.
```

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 Satandard"**,

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

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

- Déclaration des constantes et variables :

. **PinSensor = 0** (cst correspondant au n° de la broche A0 sur laquelle le capteur de pression est connecté)

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

. **ValSensor = 0** (variable pour stocker la valeur de la broche du capteur de pression)

. **tension = 0** (variable correspondant à la valeur de la tension en V de la broche du capteur de pression)

. **Pression = 0** (variable correspondant à la pression en kPa calculée à partir de la valeur de la broche du capteur)

. **OldPression = 0** (variable correspondant à la pression en kPa calculée précédemment)

. **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)

. **OldState = 0** (variable pour stocker la valeur précédente de la variable **state**)

. **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 :

- Déclaration de la broche du capteur de pression en entrée analogique :

InputPinSensor = AnalogInput(board, PinSensor)

- Déclaration de la broche du bouton poussoir en entrée numérique :

InputPinButton = DigitalInput(board, PinButton)

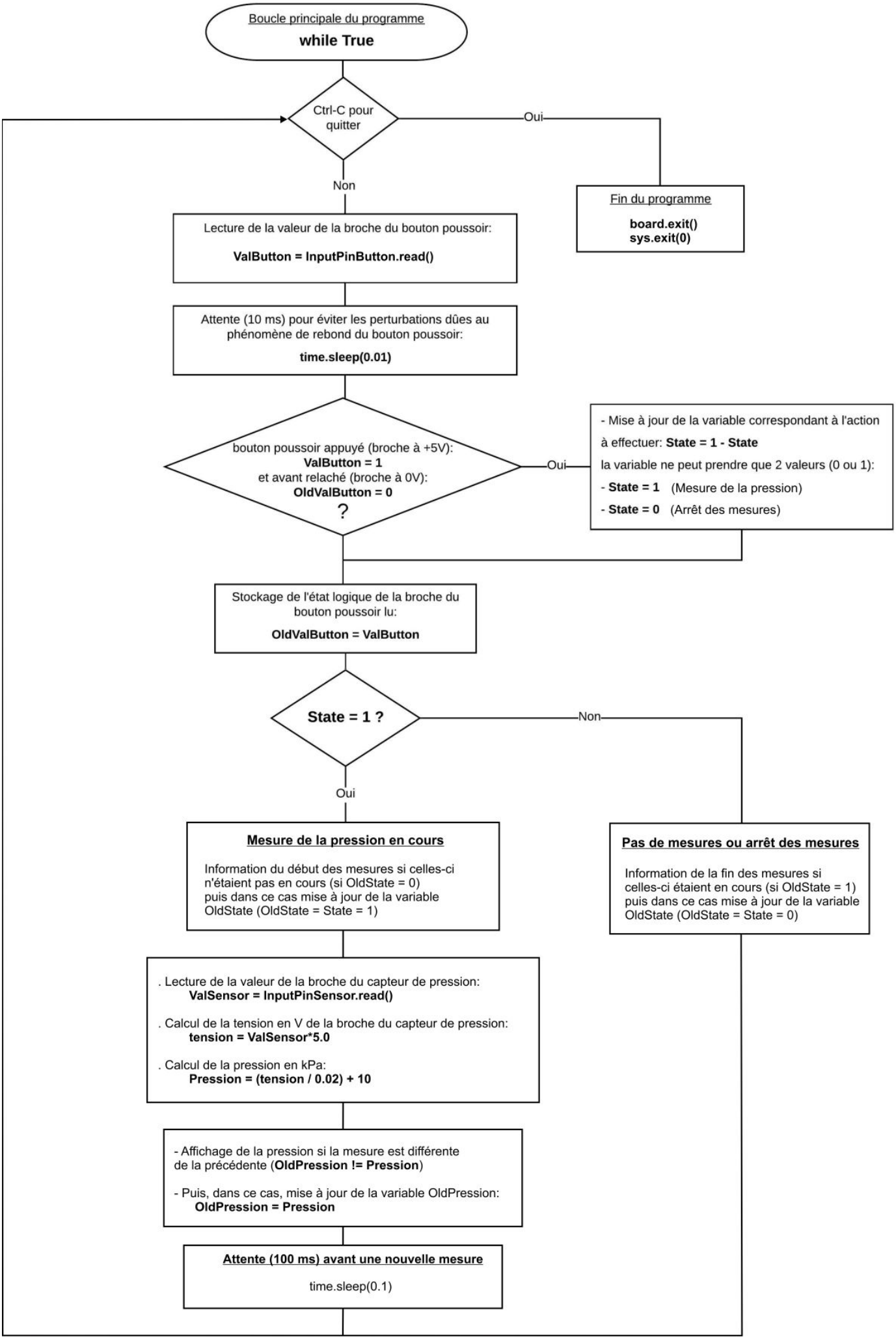
- Lancement de l'itérateur :

ArduinoIterateur = Iterateur(board)

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

time.sleep(0.5)

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

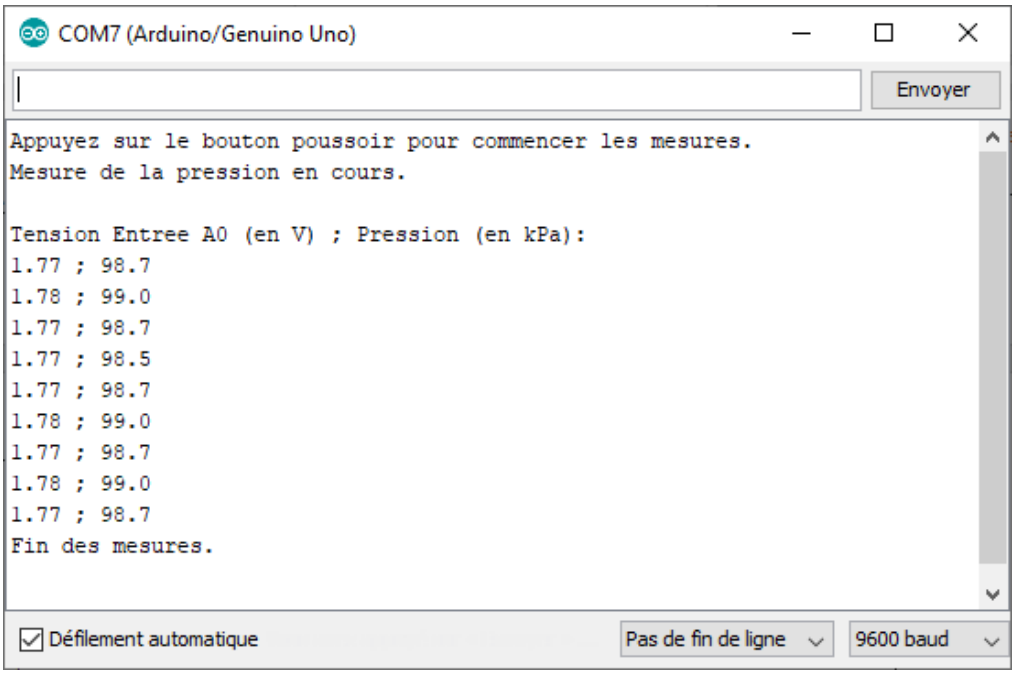

Boucle principale du programme
while True
Ctrl-C pour quitter
Oui
Non
Fin du programme
board.exit()
sys.exit(0)
Lecture de la valeur de la broche du bouton poussoir:
ValButton = InputPinButton.read()
Attente (10 ms) pour éviter les perturbations dûes au phénomène de rebond du bouton poussoir:
time.sleep(0.01)
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 prendre que 2 valeurs (0 ou 1):
- State = 1 (Mesure de la pression)
- State = 0 (Arrêt des mesures)
Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Non
Oui
Mesure de la pression en cours
Information du début des mesures si celles-ci n'étaient pas en cours (si OldState = 0) puis dans ce cas mise à jour de la variable OldState (OldState = State = 1)
Pas de mesures ou arrêt des mesures
Information de la fin des mesures si celles-ci étaient en cours (si OldState = 1) puis dans ce cas mise à jour de la variable OldState (OldState = State = 0)
. Lecture de la valeur de la broche du capteur de pression:
ValSensor = InputPinSensor.read()
. Calcul de la tension en V de la broche du capteur de pression:
tension = ValSensor*5.0
. Calcul de la pression en kPa:
Pression = (tension / 0.02) + 10
- Affichage de la pression si la mesure est différente de la précédente (OldPression != Pression)
- Puis, dans ce cas, mise à jour de la variable OldPression:
OldPression = Pression
Attente (100 ms) avant une nouvelle mesure
time.sleep(0.1)

. Programme en langage Arduino (”Projet6\Activity1\INO\Activity1.ino”)

Activity1

```
// Déclaration des constantes et variables

const int PinSensor=0;
const int PinButton= 12;

int ValSensor = 0;
float tension = 0.0;
float Pression = 0.0;
float OldPression = 0.0;

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


// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  pinMode(PinButton,INPUT);
  Serial.println("Appuyez sur le bouton poussoir pour commencer les mesures.");
}


// Fonction principale en boucle

void loop() {
  ValButton = digitalRead(PinButton);
delay(10);

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

if (State==1)
  {
  if (OldState == 0)
  {
   Serial.println("Mesure de la pression en cours.");
   Serial.println("");
   Serial.println ("Tension Entree A0 (en V) ; Pression (en kPa):");
   OldState=1;
  }
  ValSensor = analogRead(PinSensor);
  tension = (ValSensor/1023.0)*5.0;

  Pression = tension / 0.02 + 10;

  if (OldPression != Pression)
  {
    Serial.print(tension);
    Serial.print(" ; ");
    Serial.println(Pression,1);
    OldPression = Pression;
  }
  delay(500);
  }

  else
  {
    if (OldState == 1){
    Serial.println("Fin des mesures.");
    OldState = 0;}
  }
}
```

Résultats dans le moniteur série :



Déroulement du programme :

- Déclaration des constantes et variables :

. **const int PinSensor = 0** (broche du capteur de pression)
. const int PinButton = 12 (broche du bouton poussoir)

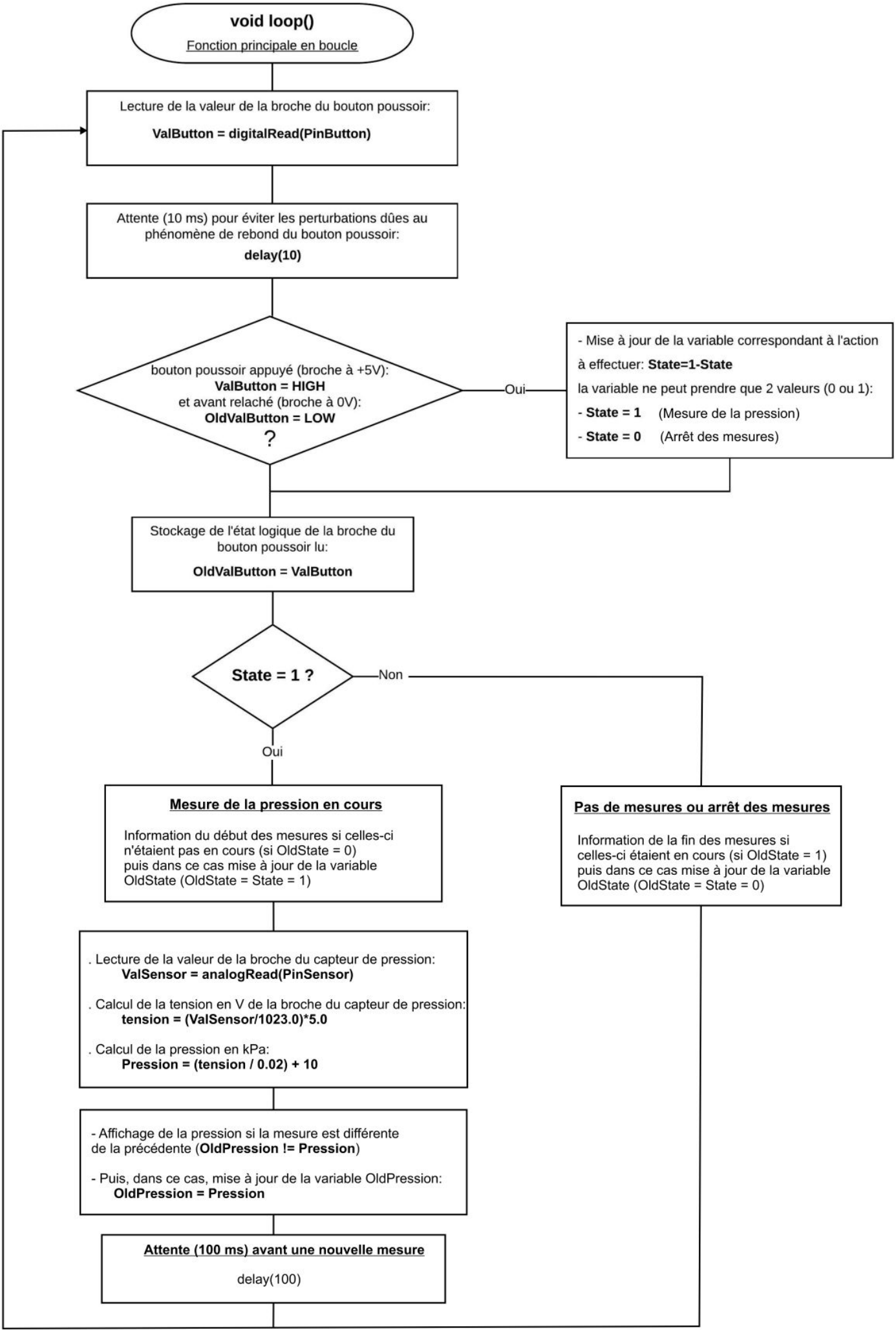
. **int ValSensor = 0** (variable nombre entier valeur broche du capteur)
. **float tension = 0.0** (variable nombre décimal calcul tension broche capteur)
. float Pression = 0.0 (variable nombre décimal calcul pression)
. float OldTPression = 0.0 (variable nombre décimal ancien calcul pression)

. int ValButton = 0 (variable nombre entier valeur broche bouton poussoir)
. int OldValButton = 0 (variable nombre entier ancienne valeur broche bouton poussoir)
. int State = (variable nombre entier pour action à effectuer)
. int OldState = 0 (variable nombre entier pour action effectuée précédemment)

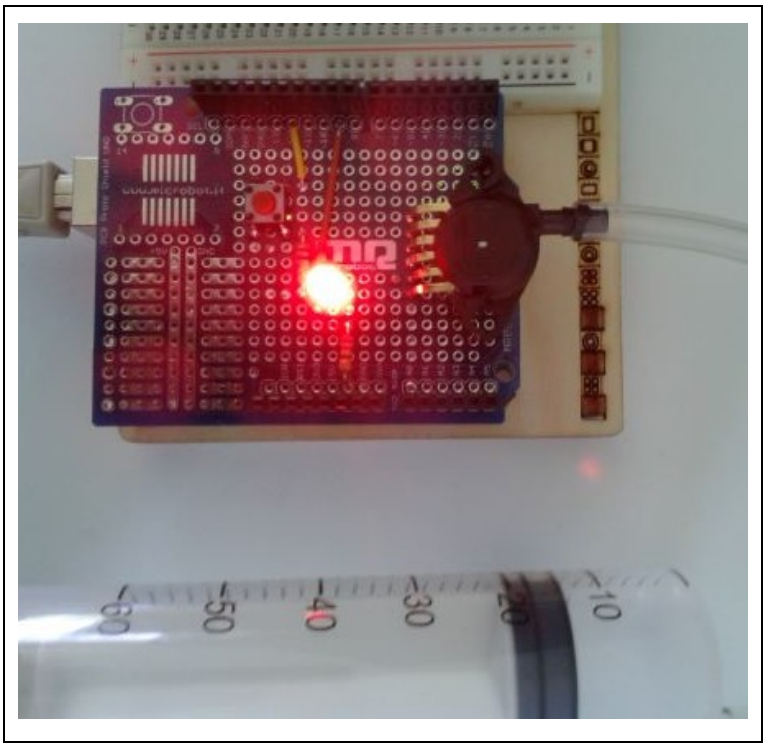
- Initialisation des entrées et sorties :

. Initialisation de la liaison série à un débit de 9600 bauds,
. Initialisation de la broche du bouton poussoir en entrée numérique.

- Fonction principale en boucle :


void loop()
Fonction principale en boucle
Lecture de la valeur 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 (Mesure de la pression)
- State = 0 (Arrêt des mesures)
Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Non
Oui
Mesure de la pression en cours
Information du début des mesures si celles-ci n'étaient pas en cours (si OldState = 0) puis dans ce cas mise à jour de la variable OldState (OldState = State = 1)
Pas de mesures ou arrêt des mesures
Information de la fin des mesures si celles-ci étaient en cours (si OldState = 1) puis dans ce cas mise à jour de la variable OldState (OldState = State = 0)
. Lecture de la valeur de la broche du capteur de pression:
ValSensor = analogRead(PinSensor)
. Calcul de la tension en V de la broche du capteur de pression:
tension = (ValSensor/1023.0)*5.0
. Calcul de la pression en kPa:
Pression = (tension / 0.02) + 10
- Affichage de la pression si la mesure est différente de la précédente (OldPression != Pression)
- Puis, dans ce cas, mise à jour de la variable OldPression:
OldPression = Pression
Attente (100 ms) avant une nouvelle mesure
delay(100)

- Activité 2 : Indicateur de pression



De façon à s'assurer que la pression mesurée par notre capteur MPX4250AP ne soit pas supérieure ou inférieure à la pression maximale (250 kPa) ou minimale (20 kPa) admissible, nous allons dans cette activité utiliser une DEL rouge qui sera allumée quand la pression est supérieure ou inférieure à des seuils à définir afin de prévenir de leurs dépassements.

La pression mesurée sera modifiée avec une seringue d'un volume utile de 60 mL fixée au capteur par l'intermédiaire d'un tuyau.
En déplaçant le piston, initialement placé sur la graduation 30 mL, on fera varier le volume de l'air enfermé dans le corps de la seringue et donc la pression appliquée sur le capteur.

La pression est mesurée après un appui sur le bouton poussoir. Un nouvel appui sur le bouton poussoir arrête les mesures.

. Le programme

Le code de l'activité en Python ou en langage Arduino pourra être modifié pour voir l'influence des variables (seuils de pression)

. Programme en Python ("Projet6\Activity2\PY\Activity2.py")

```
# Importations des librairies et définition de fonctions

from PyFirmataDef import *
from ConnectToArduino import *
import time


# Déclaration des constantes et variables

PinSensor = 0
PinButton = 12
PinLed = 9
PMax = 200
PMin = 55

ValSensor = 0
tension = 0
Pression = 0
OldPression = 0

ValButton = 0
OldValButton = 0
State = 0
OldState = 0


# Connexion à l'Arduino

print("\nConnexion à l'Arduino en cours...")

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)

InputPinSensor = AnalogInput(board, PinSensor)
InputPinButton = DigitalInput(board, PinButton)

ArduinoIterateur = Iterateur(board)
time.sleep(0.5)

print("\nConnexion à l'Arduino établie - Appuyez sur Ctrl-C pour quitter\n")
print("\nAppuyez sur le bouton poussoir pour commencer les mesures.\n");


# Boucle principale du programme

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

        OldValButton = ValButton;

        if State==1:
            if OldState == 0:
               print("\nMesure de la pression en cours.\n")
               print("Tension Entree A0 (en V) ; Pression (en kPa):")
               OldState=1
```

```
            ValSensor = InputPinSensor.read()
            tension = ValSensor*5.0
            Pression = (tension / 0.02) + 10

            if OldPression != Pression:
                print(round(tension,2), " ; ", round(Pression,1))
                OldPression = Pression

            if Pression>PMax or Pression<PMin:
                DigitalWrite(board,PinLed,1)
            else:
                DigitalWrite(board,PinLed,0)

            time.sleep(0.5)

        else:
            if OldState == 1:
                print("\nFin des mesures.\n")
                DigitalWrite(board,PinLed,0)
                OldState = 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 Satandard"**,

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

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

- Déclaration des constantes et variables :

. **PinSensor = 0** (cst correspondant au n° de la broche A0 sur laquelle le capteur de pression est connecté)

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

. **PinLed = 9** (cst correspondant au n° de la broche sur laquelle la DEL est connectée)

. **PMax = 200** (Pression maximale admissible en kPa)

. **PMin = 55** (Pression minimale admissible en kPa)

. **ValSensor = 0** (variable pour stocker la valeur de la broche du capteur de pression)

. **tension = 0** (variable correspondant à la valeur de la tension en V de la broche du capteur de pression)

. **Pression = 0** (variable correspondant à la pression en kPa calculée à partir de la valeur de la broche du capteur)

. **OldPression = 0** (variable correspondant à la pression en kPa calculée précédemment)

. **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)

. **OldState = 0** (variable pour stocker la valeur précédente de la variable **state**)

. **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 :

- Déclaration de la broche du capteur de pression en entrée analogique :

InputPinSensor = AnalogInput(board, PinSensor)

- Déclaration de la broche du bouton poussoir en entrée numérique :

InputPinButton = DigitalInput(board, PinButton)

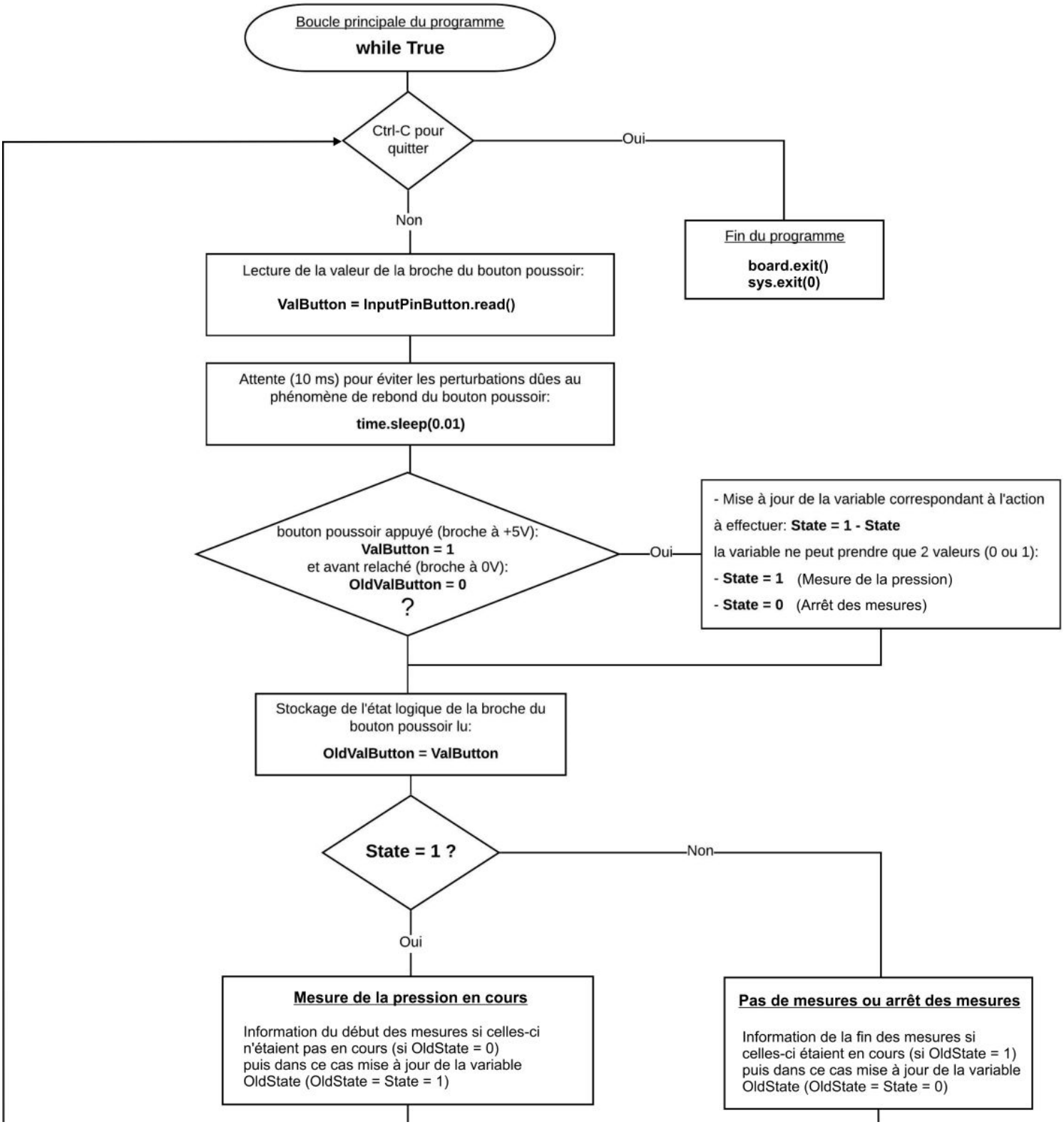
- Lancement de l'itérateur :

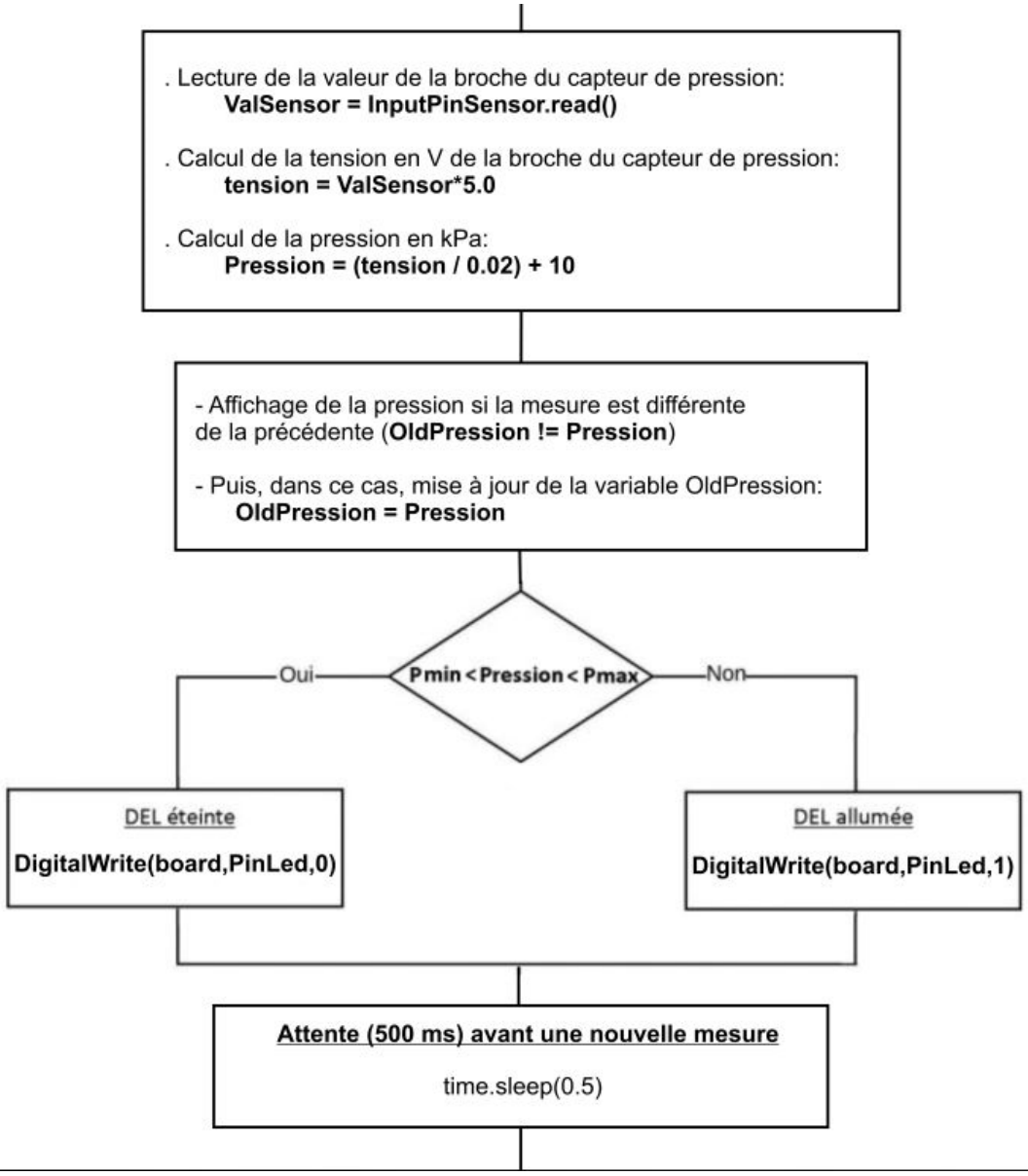
ArduinoIterateur = Iterateur(board)

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

time.sleep(0.5)

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


Boucle principale du programme
while True
Ctrl-C pour quitter
Oui
Non
Fin du programme
board.exit()
sys.exit(0)
Lecture de la valeur de la broche du bouton poussoir:
ValButton = InputPinButton.read()
Attente (10 ms) pour éviter les perturbations dûes au phénomène de rebond du bouton poussoir:
time.sleep(0.01)
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 prendre que 2 valeurs (0 ou 1):
- State = 1 (Mesure de la pression)
- State = 0 (Arrêt des mesures)
Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Non
Oui
Mesure de la pression en cours
Information du début des mesures si celles-ci n'étaient pas en cours (si OldState = 0) puis dans ce cas mise à jour de la variable OldState (OldState = State = 1)
Pas de mesures ou arrêt des mesures
Information de la fin des mesures si celles-ci étaient en cours (si OldState = 1) puis dans ce cas mise à jour de la variable OldState (OldState = State = 0)


. Lecture de la valeur de la broche du capteur de pression:
ValSensor = InputPinSensor.read()
. Calcul de la tension en V de la broche du capteur de pression:
tension = ValSensor*5.0
. Calcul de la pression en kPa:
Pression = (tension / 0.02) + 10
- Affichage de la pression si la mesure est différente de la précédente (OldPression != Pression)
- Puis, dans ce cas, mise à jour de la variable OldPression:
OldPression = Pression
Pmin < Pression < Pmax
Oui
Non
DEL éteinte
DigitalWrite(board,PinLed,0)
DEL allumée
DigitalWrite(board,PinLed,1)
Attente (500 ms) avant une nouvelle mesure
time.sleep(0.5)

. Programme en langage Arduino (”Projet6\Activity2\INO\Activity2.ino”)

Activity2

```
// Déclaration des constantes et variables

const int PinSensor = 0;
const int PinButton = 12;
const int PinLed = 9;
const int PMax = 200;
const int PMin = 55;

int ValSensor = 0;
float tension = 0.0;
float Pression = 0.0;
float OldPression = 0.0;

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


// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  pinMode(PinButton,INPUT);
  pinMode(PinLed,OUTPUT);
  Serial.println("Appuyez sur le bouton poussoir pour commencer les mesures.");
}


// Fonction principale en boucle

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

if (State==1)
{
  if (OldState == 0)
  {
   Serial.println("Mesure de la pression en cours.");
   Serial.println("");
   Serial.println ("Pression (en kPa):");
   OldState=1;
  }
  ValSensor = analogRead(PinSensor);
  tension = (ValSensor/1023.0)*5.0;

  Pression = tension / 0.02 + 10;
```

```
  if (OldPression != Pression)
  {
    Serial.println(Pression,1);
    OldPression = Pression;
  }

  if (Pression>PMax or Pression<PMin)
  {
    digitalWrite(PinLed,HIGH);
  }
  else {
    digitalWrite(PinLed,LOW);
  }
  delay(500);
}
else
  {
    if (OldState == 1){
    Serial.println("Fin des mesures.");
    digitalWrite(PinLed,LOW);
    OldState = 0;}
  }
}
```

Déroulement du programme :

- Déclaration des constantes et variables :

. **const int PinSensor = 0** (broche du capteur de pression)
. const int PinButton = 12 (broche du bouton poussoir)
. **const int PinLed = 9** (broche de la DEL)
. const int PMax = 200 (constante nombre entier valeur pression max en kPa)
. **const int PMin = 55** (constante nombre entier valeur pression min en kPa)

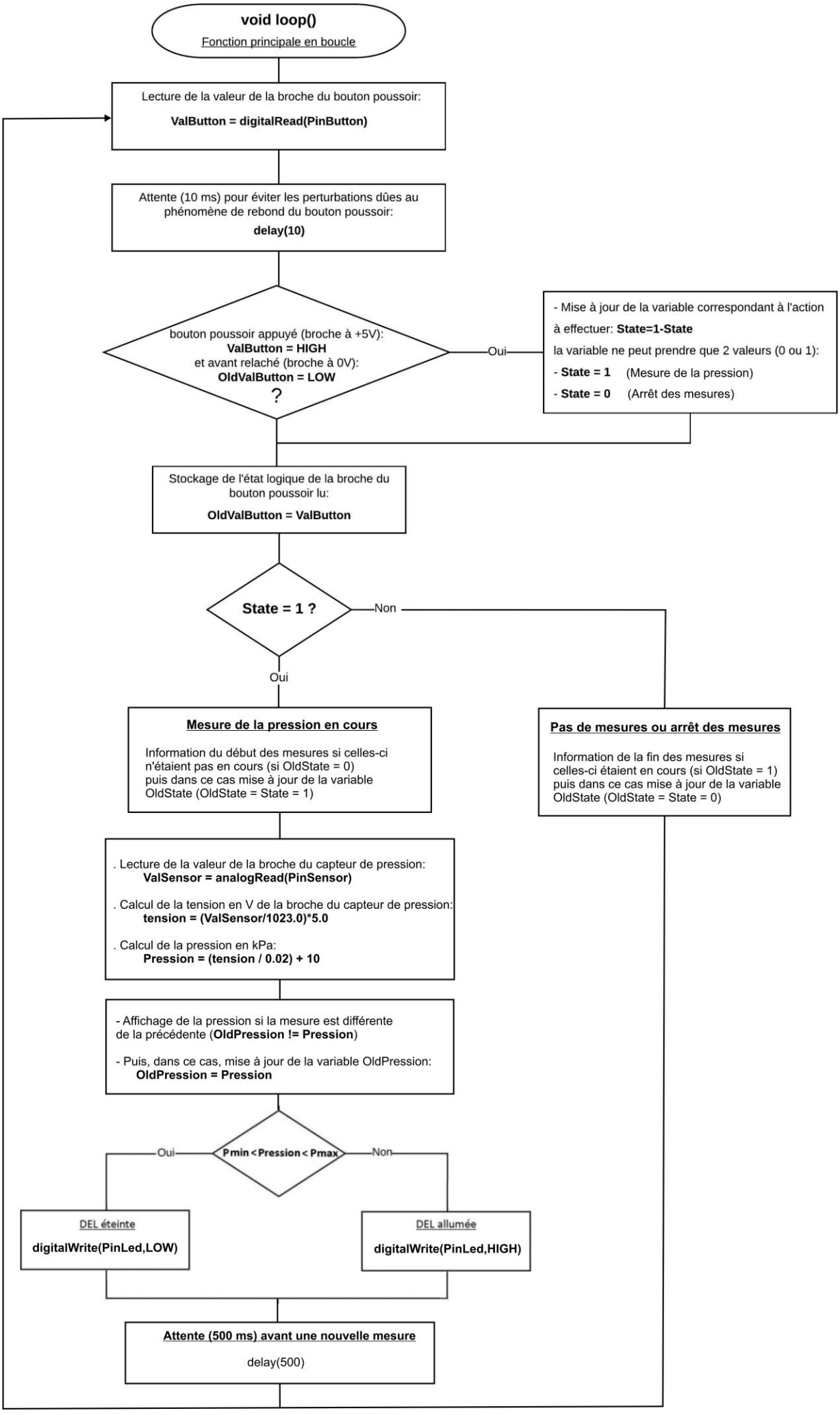
. **int ValSensor = 0** (variable nombre entier valeur broche du capteur)
. **float tension = 0.0** (variable nombre décimal calcul tension broche capteur)
. float Pression = 0.0 (variable nombre décimal calcul pression)
. float OldPression = 0.0 (variable nombre décimal ancien calcul pression)

. int ValButton = 0 (variable nombre entier valeur broche bouton poussoir)
. int OldValButton = 0 (variable nombre entier ancienne valeur broche bouton poussoir)
. int State = 0 (variable nombre entier pour action à effectuer)
. int OldState = 0 (variable nombre entier pour action effectuée précédemment)

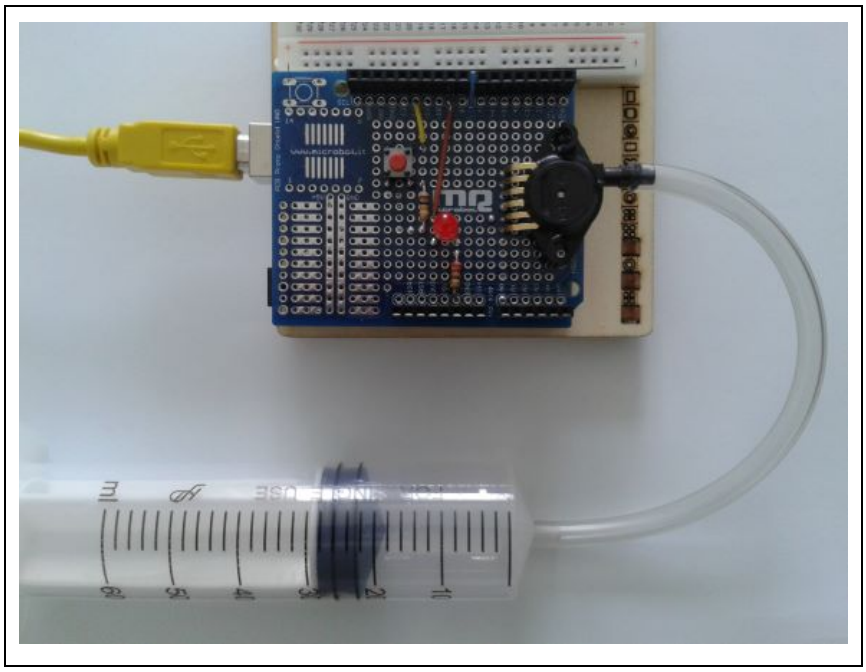
- Initialisation des entrées et sorties :

. Initialisation de la liaison série à un débit de 9600 bauds
. Initialisation de la broche du bouton poussoir en entrée
. Initialisation de la broche de la DEL en sortie

- Fonction principale en boucle :


void loop()
Fonction principale en boucle
Lecture de la valeur 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 (Mesure de la pression)
- State = 0 (Arrêt des mesures)
Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Non
Oui
Mesure de la pression en cours
Information du début des mesures si celles-ci n'étaient pas en cours (si OldState = 0) puis dans ce cas mise à jour de la variable OldState (OldState = State = 1)
Pas de mesures ou arrêt des mesures
Information de la fin des mesures si celles-ci étaient en cours (si OldState = 1) puis dans ce cas mise à jour de la variable OldState (OldState = State = 0)
. Lecture de la valeur de la broche du capteur de pression:
ValSensor = analogRead(PinSensor)
. Calcul de la tension en V de la broche du capteur de pression:
tension = (ValSensor/1023.0)*5.0
. Calcul de la pression en kPa:
Pression = (tension / 0.02) + 10
- Affichage de la pression si la mesure est différente de la précédente (OldPression != Pression)
- Puis, dans ce cas, mise à jour de la variable OldPression:
OldPression = Pression
Oui
Pmin < Pression < Pmax
Non
DEL éteinte
digitalWrite(PinLed,LOW)
DEL allumée
digitalWrite(PinLed,HIGH)
Attente (500 ms) avant une nouvelle mesure
delay(500)

- Activité 3 : Vérification de la loi de Boyle-Mariotte



. Objectif

L'objectif de cette activité est de vérifier la loi de Boyle-Mariotte à l'aide de notre capteur de pression MPX4250AP et de la seringue d'un volume utile de 60 mL.

. Énoncé de la loi de Boyle-Mariotte :

À température constante, pour une quantité de matière donnée de gaz, le produit de la pression P par le volume V de ce gaz ne varie pas :

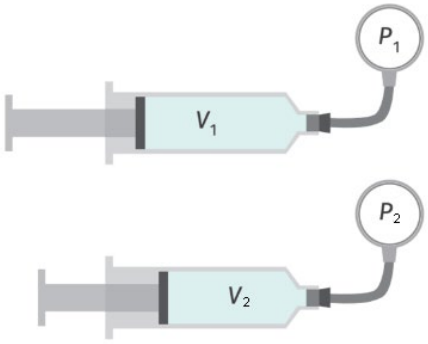
P × V = constante

Si un gaz subit une transformation qui l'amène d'un état n°1 (son volume est V1 et sa pression P1) à un état n°2 (son volume est V2 et sa pression P2) alors la loi de Boyle-Mariotte peut s'écrire:

P1.V1 = P2.V2

Cette relation est valable à condition que:

- La transformation soit à température constante,
- Les deux pressions P1 et P2 soient exprimées dans la même unité de pression qui peut être le pascal, l'hectopascal, le bar, etc…,
- Les deux volumes soient exprimés dans la même unité de volume qui peut être le litre, le mètre cube, le centimètre cube, etc.



. Descriptif de l'activité

En déplaçant le piston, initialement placé sur la graduation 30 mL, on fait varier le volume de l'air enfermé dans le corps de la seringue reliée au capteur de pression.

Après avoir appuyé sur le bouton poussoir, le volume lu (entre 10 et 60 mL) sur le corps de la seringue est saisi au clavier dans la fenêtre Python Shell ou le moniteur série.
La pression est ensuite mesurée par le capteur qui délivre une tension électrique proportionnelle à la pression. Une moyenne est réalisée sur dix mesures et le résultat de la pression moyenne (p en kPa) pour le volume (V en mL) est affiché dans la fenêtre Python Shell ou le moniteur série.

Les mesures pour le volume V sont arrêtées en appuyant sur le bouton poussoir.
Une nouvelle mesure de la pression pour un volume différent est réalisable en appuyant de nouveau sur le bouton poussoir.

Il est donc possible d'acquérir des couples de données (p, V) afin de vérifier la loi de Boyle-Mariotte.

Remarque :

La DEL rouge est allumée si la pression est inférieure ou supérieure aux seuils de pression correspondant à un volume de 10 ou de 60 mL.

. Le programme

Voici le code de l'activité en Python et en langage Arduino :

. Programme en Python (**"Projet6\Activity3\PY\Activity3.py"**)

```
# Importations des librairies et définition de fonctions

from PyFirmataDef import *
from ConnectToArduino import *
import time


def InputVal(msg, valmin, valmax, unit):
    InputValid = False
    while InputValid==False:
        val = input(msg+"(valeur entre {0} et {1} {2}) :" .format(valmin, valmax, unit))
        try:
            val = int(val)
            assert val >= valmin and val <= valmax

        except AssertionError:
            print("La valeur saisie n'est pas entre {0} et {1} {2} !".format(valmin, valmax, unit))

        except:
            print("vous n'avez pas saisi une valeur entre {0} et {1} {2} !".format(valmin, valmax, unit))

        else:
            InputValid = True

    return val


# Déclaration des constantes et variables

PinSensor = 0
PinButton = 12
PinLed = 9
PMax = 240
PMin = 50

ValSensor = 0
tension = 0
Pression = 0
OldPression = 0
Vol = 0

ValButton = 0
OldValButton = 0
State = 0
OldState = 0


# Connexion à l'Arduino

print("\nConnexion à l'Arduino en cours...")

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)

InputPinSensor = AnalogInput(board, PinSensor)
InputPinButton = DigitalInput(board, PinButton)

ArduinoIterateur = Iterateur(board)
time.sleep(0.5)

print("\nConnexion à l'Arduino établie - Appuyez sur Ctrl-C pour quitter\n")
print("\nAppuyez sur le bouton poussoir pour commencer les mesures.\n");


# Boucle principale du programme

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

        OldValButton = ValButton;
```

```
        if State==1:
            if OldState == 0:
                Vol = InputVal("Veuillez saisir le volume V de la seringue en mL ",10,60,"mL")
                print("\nMesure de la pression en cours.\n")
                print("Volume (mL) ; Pression (en kPa):")
                OldState=1

            Pression = 0
            for i in range(10):
                ValSensor = InputPinSensor.read()
                tension = ValSensor*5.0
                Pression = Pression +(tension / 0.02) + 10
            Pression = Pression / 10

            if OldPression != Pression:
                print(Vol, " ; ", round(Pression,1))
                OldPression = Pression

            if Pression>PMax or Pression<PMin:
                DigitalWrite(board,PinLed,1)
            else:
                DigitalWrite(board,PinLed,0)

            time.sleep(0.5)

        else:
            if OldState == 1:
                print("\nFin des mesures.\n")
                DigitalWrite(board,PinLed,0)
                OldState = 0

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

Résultats dans la fenêtre Python Shell :

```
Connexion à l'Arduino en cours...

Connexion à l'Arduino établie - Appuyez sur Ctrl-C pour quitter


Appuyez sur le bouton poussoir pour commencer les mesures.

Veuillez saisir le volume V de la seringue en mL (valeur entre 10 et 60 mL) :30

Mesure de la pression en cours.

Volume (mL) ; Pression (en kPa):
30  ;  103.1
30  ;  102.9
30  ;  103.1
30  ;  103.3
30  ;  103.1

Fin des mesures.

Veuillez saisir le volume V de la seringue en mL (valeur entre 10 et 60 mL) :40

Mesure de la pression en cours.

Volume (mL) ; Pression (en kPa):
40  ;  80.1
40  ;  80.4
40  ;  80.1
40  ;  80.4
40  ;  80.1
40  ;  82.1
40  ;  80.6
40  ;  80.4

Fin des mesures.
```

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 Satandard"**,

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

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

. La fonction **"InputVal"** pour la saisie du volume de la seringue en mL.

- Déclaration des constantes et variables :

. **PinSensor = 0** (cst correspondant au n° de la broche A0 sur laquelle le capteur de pression est connecté)

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

. **PinLed = 9** (cst correspondant au n° de la broche sur laquelle la DEL est connectée)

. **PMax = 240** (Pression maximale admissible en kPa)

. **PMin = 50** (Pression minimale admissible en kPa)

. **ValSensor = 0** (variable pour stocker la valeur de la broche du capteur de pression)

. **tension = 0** (variable correspondant à la valeur de la tension en V de la broche du capteur de pression)

. **Pression = 0** (variable correspondant à la pression en kPa calculée à partir de la valeur de la broche du capteur)

. **OldPression = 0** (variable correspondant à la pression en kPa calculée précédemment)

. **Vol=0** (variable correspondant au volume de la seringue en mL)

. **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)

. **OldState = 0** (variable pour stocker la valeur précédente de la variable **state**)

. **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 :

- Déclaration de la broche du capteur de pression en entrée analogique :

  InputPinSensor = AnalogInput(board, PinSensor)

- Déclaration de la broche du bouton poussoir en entrée numérique :

  InputPinButton = DigitalInput(board, PinButton)

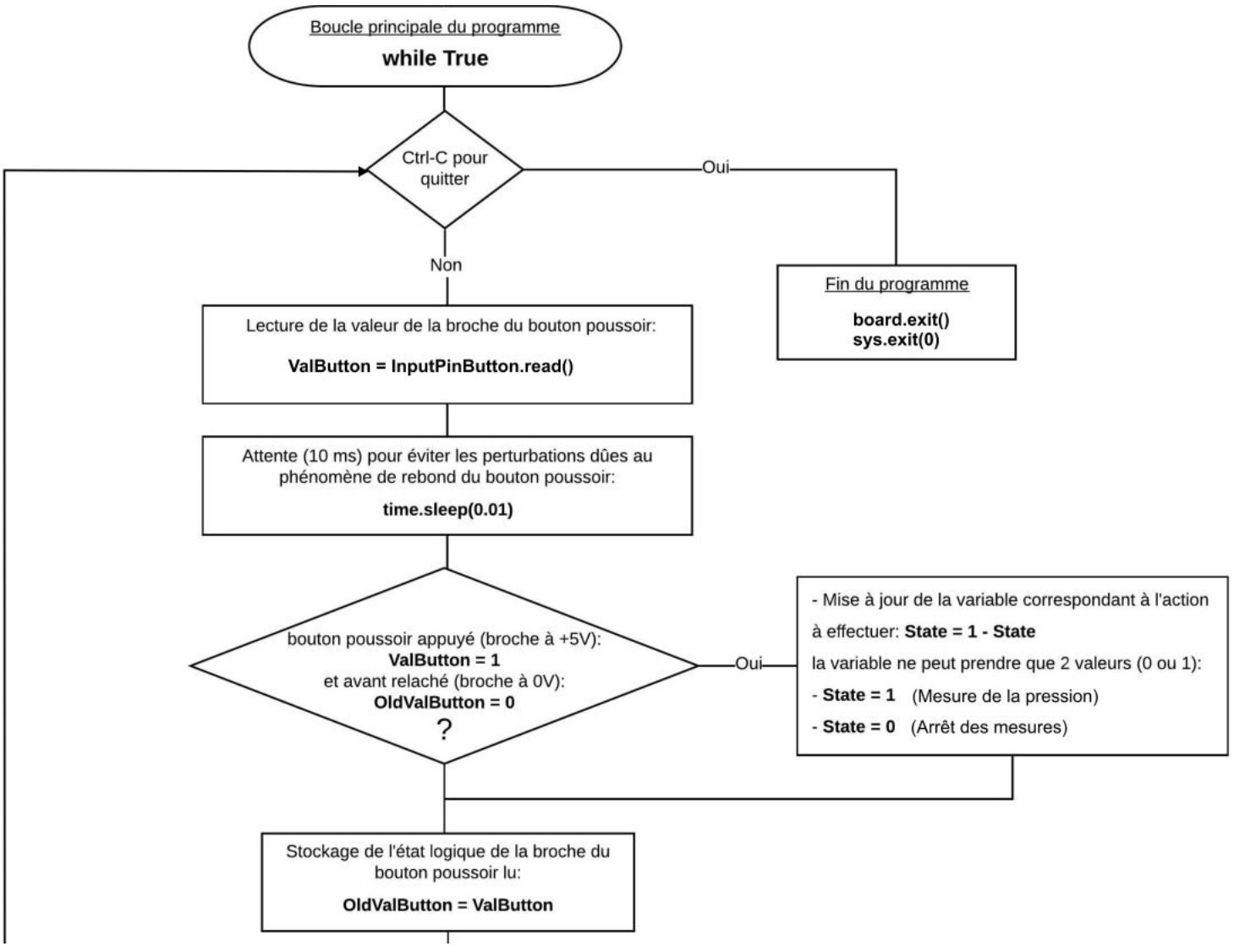
- Lancement de l'itérateur :

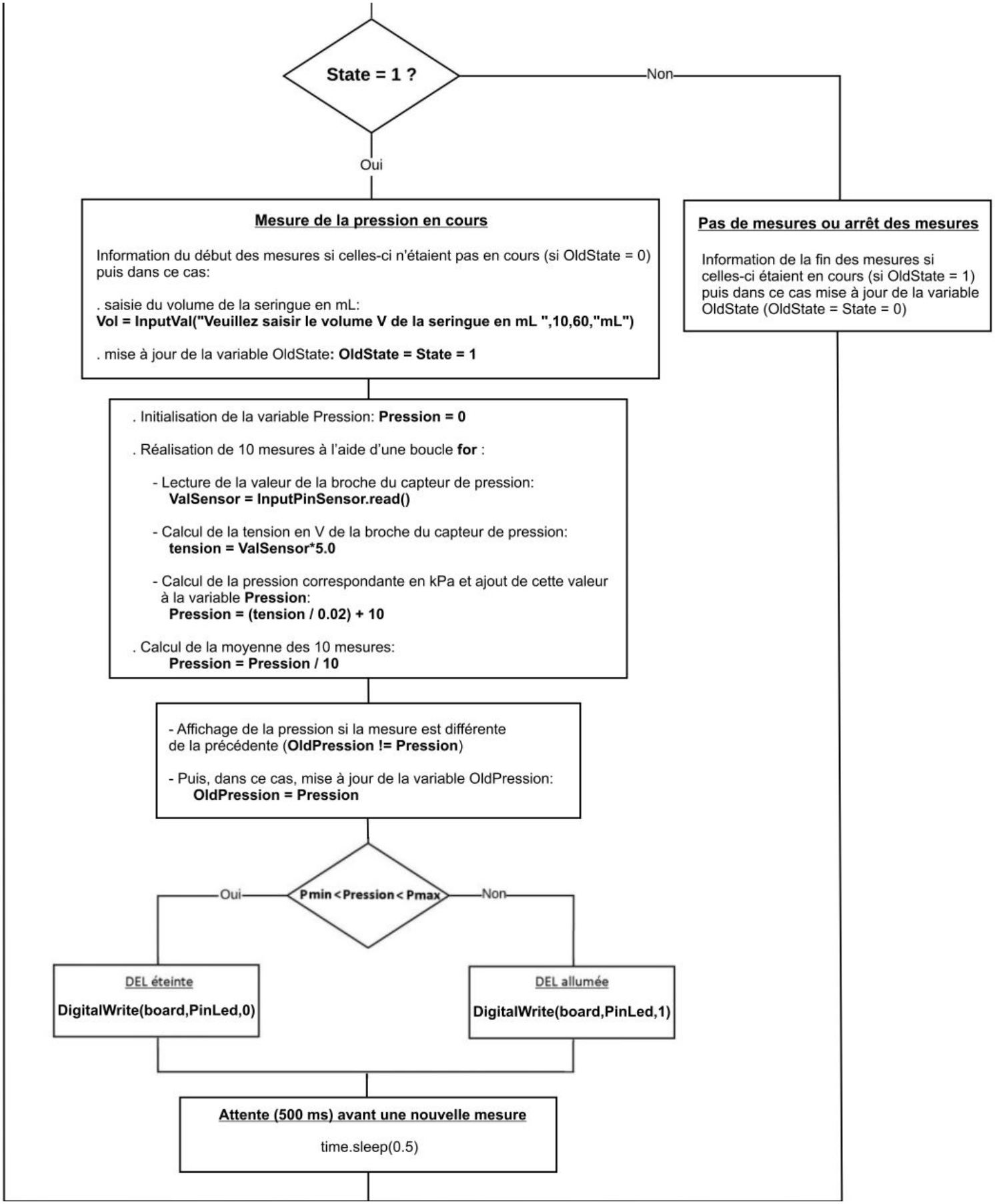
  ArduinoIterateur = Iterateur(board)

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

  time.sleep(0.5)

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




State = 1 ?
Non
Oui
Mesure de la pression en cours
Information du début des mesures si celles-ci n'étaient pas en cours (si OldState = 0) puis dans ce cas:
. saisie du volume de la seringue en mL:
Vol = InputVal("Veuillez saisir le volume V de la seringue en mL ",10,60,"mL")
. mise à jour de la variable OldState: OldState = State = 1
Pas de mesures ou arrêt des mesures
Information de la fin des mesures si celles-ci étaient en cours (si OldState = 1) puis dans ce cas mise à jour de la variable OldState (OldState = State = 0)
. Initialisation de la variable Pression: Pression = 0
. Réalisation de 10 mesures à l'aide d'une boucle for :
- Lecture de la valeur de la broche du capteur de pression:
ValSensor = InputPinSensor.read()
- Calcul de la tension en V de la broche du capteur de pression:
tension = ValSensor*5.0
- Calcul de la pression correspondante en kPa et ajout de cette valeur à la variable Pression:
Pression = (tension / 0.02) + 10
. Calcul de la moyenne des 10 mesures:
Pression = Pression / 10
- Affichage de la pression si la mesure est différente de la précédente (OldPression != Pression)
- Puis, dans ce cas, mise à jour de la variable OldPression:
OldPression = Pression
Pmin < Pression < Pmax
Oui
Non
DEL éteinte
DigitalWrite(board,PinLed,0)
DEL allumée
DigitalWrite(board,PinLed,1)
Attente (500 ms) avant une nouvelle mesure
time.sleep(0.5)

. Programme en langage Arduino (”Projet6\Activity3\INO\Activity3.ino”)

Activity3

```
// Déclaration des constantes et variables

const int PinSensor = 0;
const int PinButton = 12;
const int PinLed = 9;
const int PMax = 240;
const int PMin = 50;

int ValSensor = 0;
float tension = 0.0;
float Pression = 0.0;
float OldPression = 0.0;
int Vol = 0;

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


// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  pinMode(PinButton,INPUT);
  pinMode(PinLed,OUTPUT);
  Serial.println("Appuyez sur le bouton poussoir pour commencer les mesures.");
}


// Fonction principale en boucle

void loop() {

ValButton = digitalRead(PinButton);
delay(10);

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

OldValButton = ValButton;
```

```
if (State==1)
{
  if (OldState == 0)
  {
    Serial.print("Veuillez saisir le volume V de la seringue en mL ");
    Serial.println("(valeur entre 10 et 60 mL).");
    Vol = 0;
    while(Vol<10 || Vol>60)
    {
    Vol=Serial.parseInt();
    }
    Serial.println("Mesure de la pression en cours.");
    Serial.println("");
    Serial.println ("Volume (mL);Pression (kPa):");
    OldState=1;
  }
  Pression = 0;
  for(int i = 0 ; i < 10 ; i++){
    ValSensor = analogRead(PinSensor);
    tension = (ValSensor/1023.0)*5.0;
    Pression = Pression + (tension / 0.02 + 10);
  }
  Pression = Pression / 10;
  if (OldPression != Pression)
  {
    Serial.print(Vol);
    Serial.print(";");
    Serial.println(Pression,1);
    OldPression = Pression;
  }

  if (Pression>PMax or Pression<PMin)
  {
    digitalWrite(PinLed,HIGH);
  }
  else {
    digitalWrite(PinLed,LOW);
  }
  delay(500);
  }
  else
  {
    if (OldState == 1){
    Serial.println("Fin des mesures.");
    Serial.println("");
    digitalWrite(PinLed,LOW);
    OldState = 0;}
  }
}
```

Déroulement du programme :

- Déclaration des constantes et variables :

. **const int PinSensor = 0** (broche du capteur de pression)
. const int PinButton = 12 (Broche du bouton poussoir)
. **const int PinLed = 9** (broche de la DEL)
. const int PMax = 200 (constante nombre entier valeur pression max en kPa)
. **const int PMin = 55** (constante nombre entier valeur pression min en kPa)

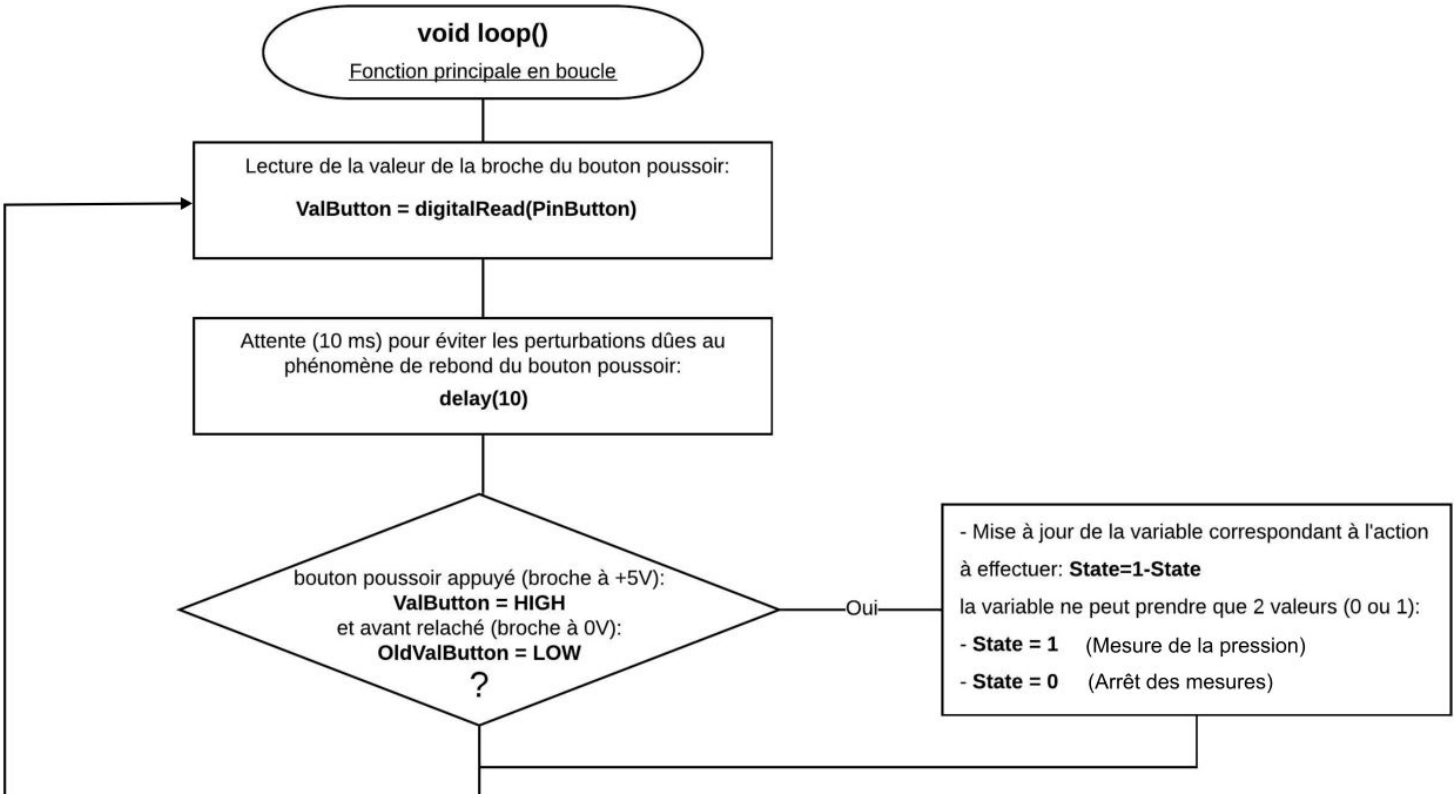
. **int ValSensor = 0** (variable nombre entier valeur broche du capteur)
. **float tension = 0.0** (variable nombre décimal calcul tension broche capteur)
. float Pression = 0.0 (variable nombre décimal calcul pression)
. float OldPression = 0.0 (variable nombre décimal ancien calcul pression)
. int Vol = 0 (variable nombre entier valeur du volume en mL)

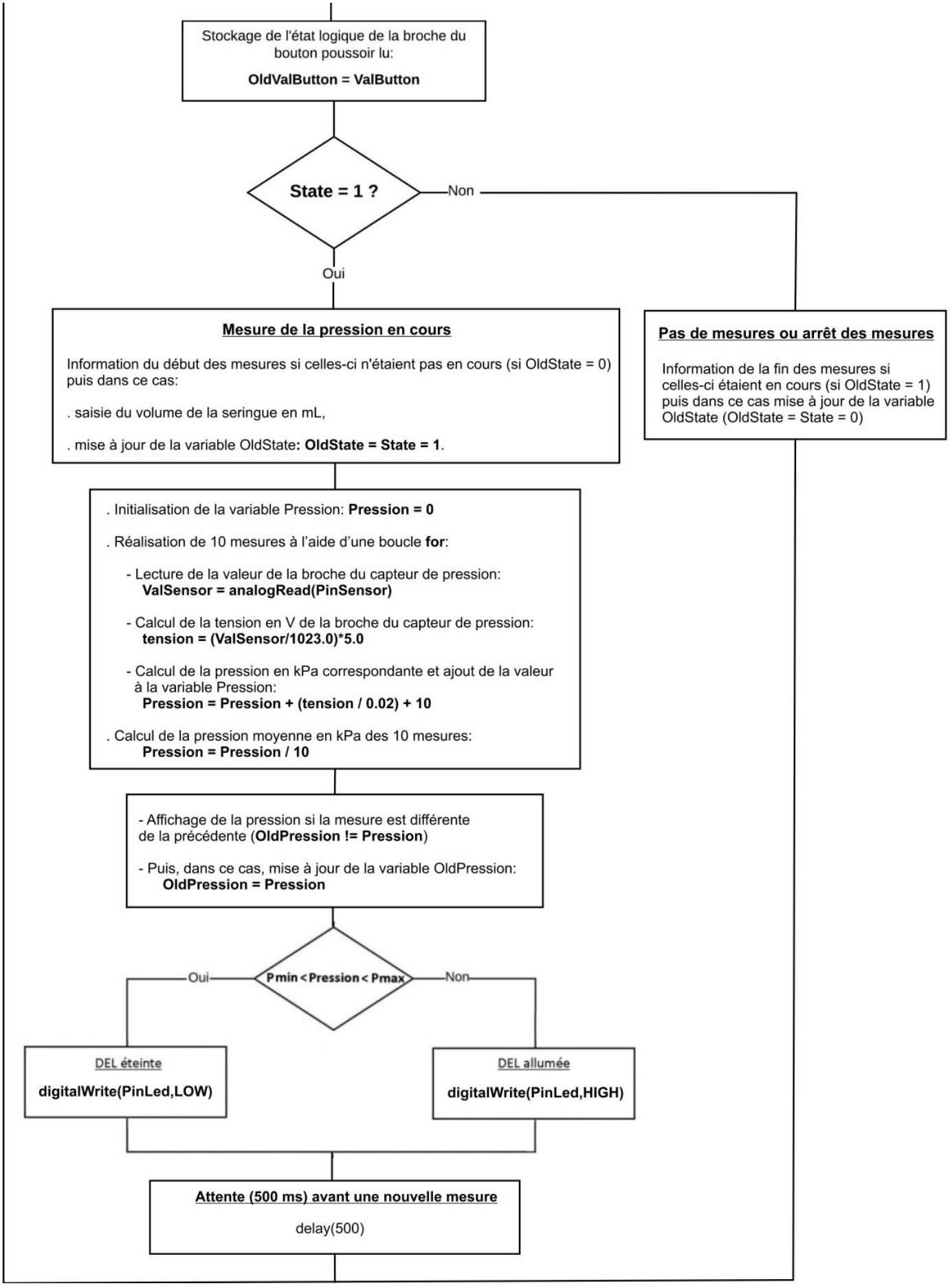
. int ValButton = 0 (variable nombre entier valeur broche bouton poussoir)
. int OldValButton = 0 (variable nombre entier ancienne valeur broche bouton poussoir)
. int State = 0 (variable nombre entier pour action à effectuer)
. int OldState = 0 (variable nombre entier pour action effectuée précédemment)

- Initialisation des entrées et sorties :

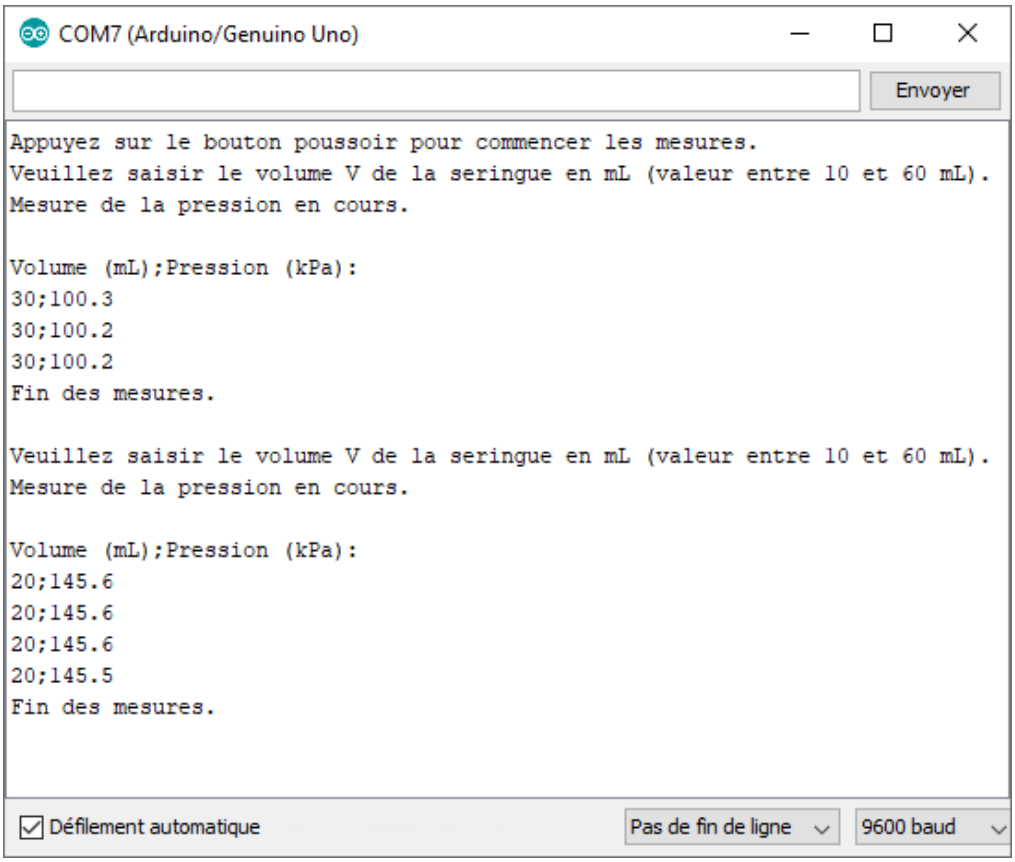
. Initialisation de la liaison série à un débit de 9600 bauds
. Initialisation de la broche du bouton poussoir en entrée
. Initialisation de la broche de la DEL en sortie

- Fonction principale en boucle :



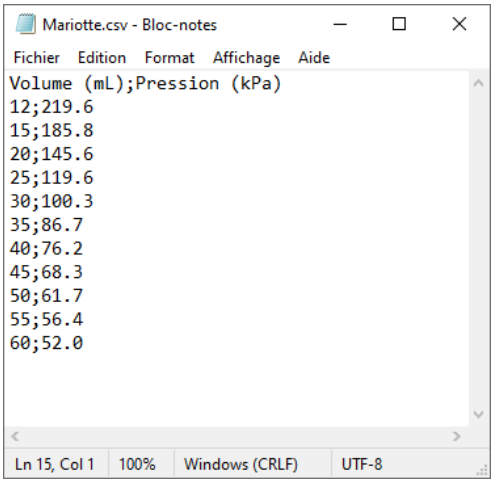

Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Non
Oui
Mesure de la pression en cours
Information du début des mesures si celles-ci n'étaient pas en cours (si OldState = 0) puis dans ce cas:
. saisie du volume de la seringue en mL,
. mise à jour de la variable OldState: OldState = State = 1.
Pas de mesures ou arrêt des mesures
Information de la fin des mesures si celles-ci étaient en cours (si OldState = 1) puis dans ce cas mise à jour de la variable OldState (OldState = State = 0)
. Initialisation de la variable Pression: Pression = 0
. Réalisation de 10 mesures à l'aide d'une boucle for:
- Lecture de la valeur de la broche du capteur de pression: ValSensor = analogRead(PinSensor)
- Calcul de la tension en V de la broche du capteur de pression: tension = (ValSensor/1023.0)*5.0
- Calcul de la pression en kPa correspondante et ajout de la valeur à la variable Pression: Pression = Pression + (tension / 0.02) + 10
. Calcul de la pression moyenne en kPa des 10 mesures: Pression = Pression / 10
- Affichage de la pression si la mesure est différente de la précédente (OldPression != Pression)
- Puis, dans ce cas, mise à jour de la variable OldPression: OldPression = Pression
Pmin < Pression < Pmax
Oui
Non
DEL éteinte
digitalWrite(PinLed,LOW)
DEL allumée
digitalWrite(PinLed,HIGH)
Attente (500 ms) avant une nouvelle mesure
delay(500)

. Résultats dans le moniteur série :

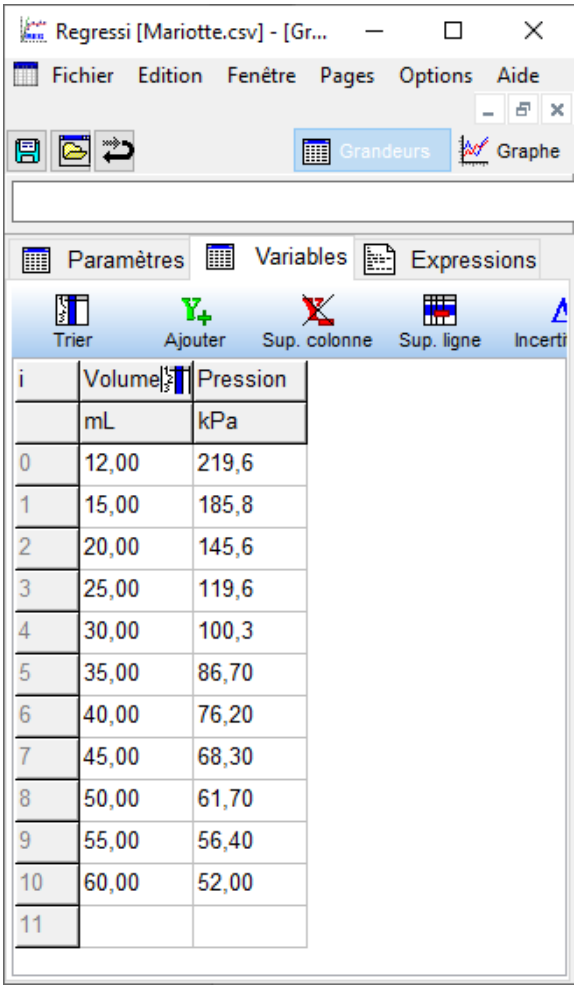


. Exploitation des résultats

Pour exploiter les mesures, on peut dans un premier temps, copier/coller les résultats affichés dans la fenêtre Python Shell ou le moniteur série dans un fichier **"csv"** créé avec le bloc-notes de Windows, comme ci-dessous :

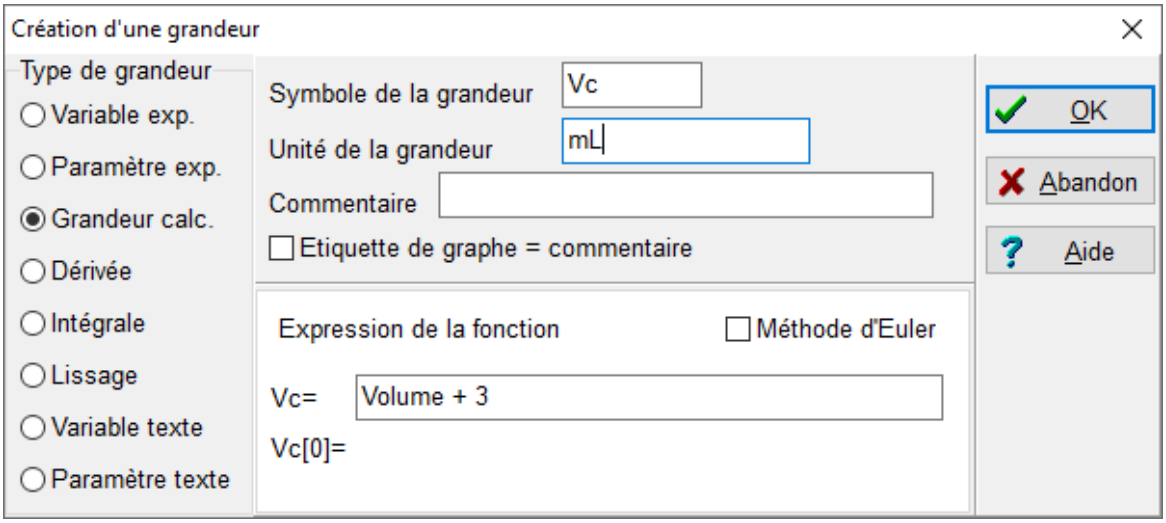


Puis ouvrir le fichier avec un tableur comme Regressi :



i	Volume	Pression
	mL	kPa
0	12,00	219,6
1	15,00	185,8
2	20,00	145,6
3	25,00	119,6
4	30,00	100,3
5	35,00	86,70
6	40,00	76,20
7	45,00	68,30
8	50,00	61,70
9	55,00	56,40
10	60,00	52,00
11		

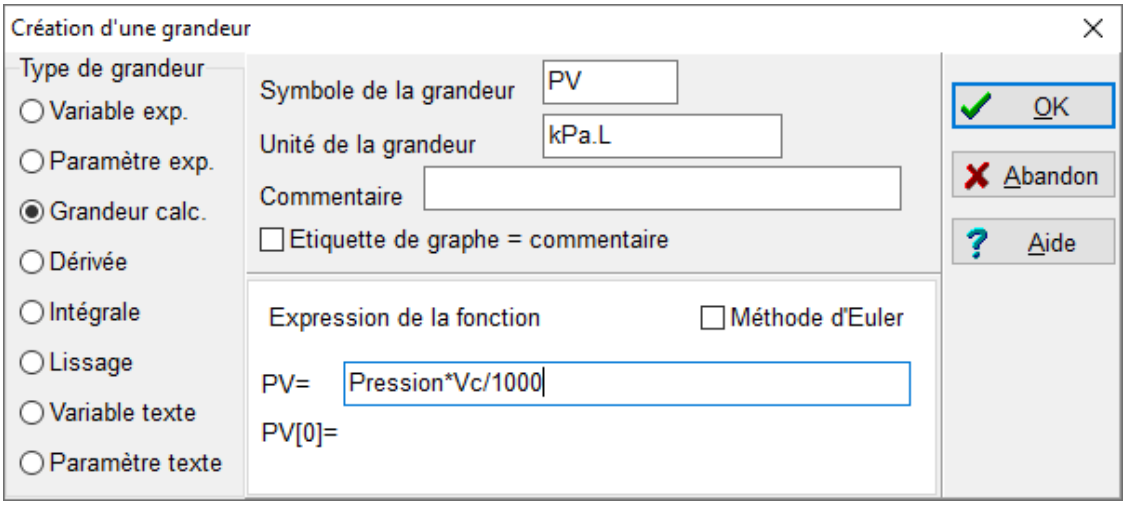
Pour les calculs, il ne faut pas oublier d'ajouter le volume d'air contenu dans le tuyau, qui relie la seringue et le capteur, au volume d'air de la seringue.
Pour cela, on va créer une grandeur, appelée **Vc** en mL, pour volume corrigé :

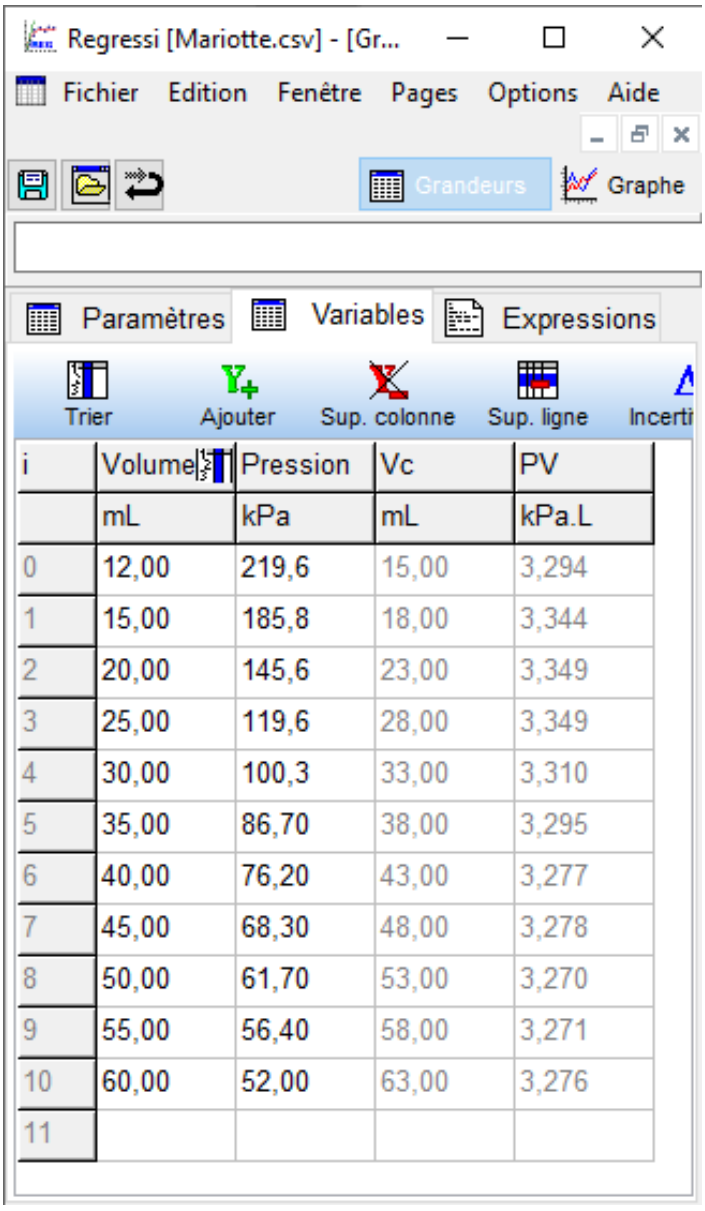


Le tuyau a un diamètre de 5 mm pour une longueur de 15 cm :

$\mathbf{V_{air\ tuyau} \cong 3\ mL}$

Puis on crée une grandeur, appelée **PV**, de façon à déterminer la valeur de la constante de la loi de Boyle-Mariotte (en kPa.L) :

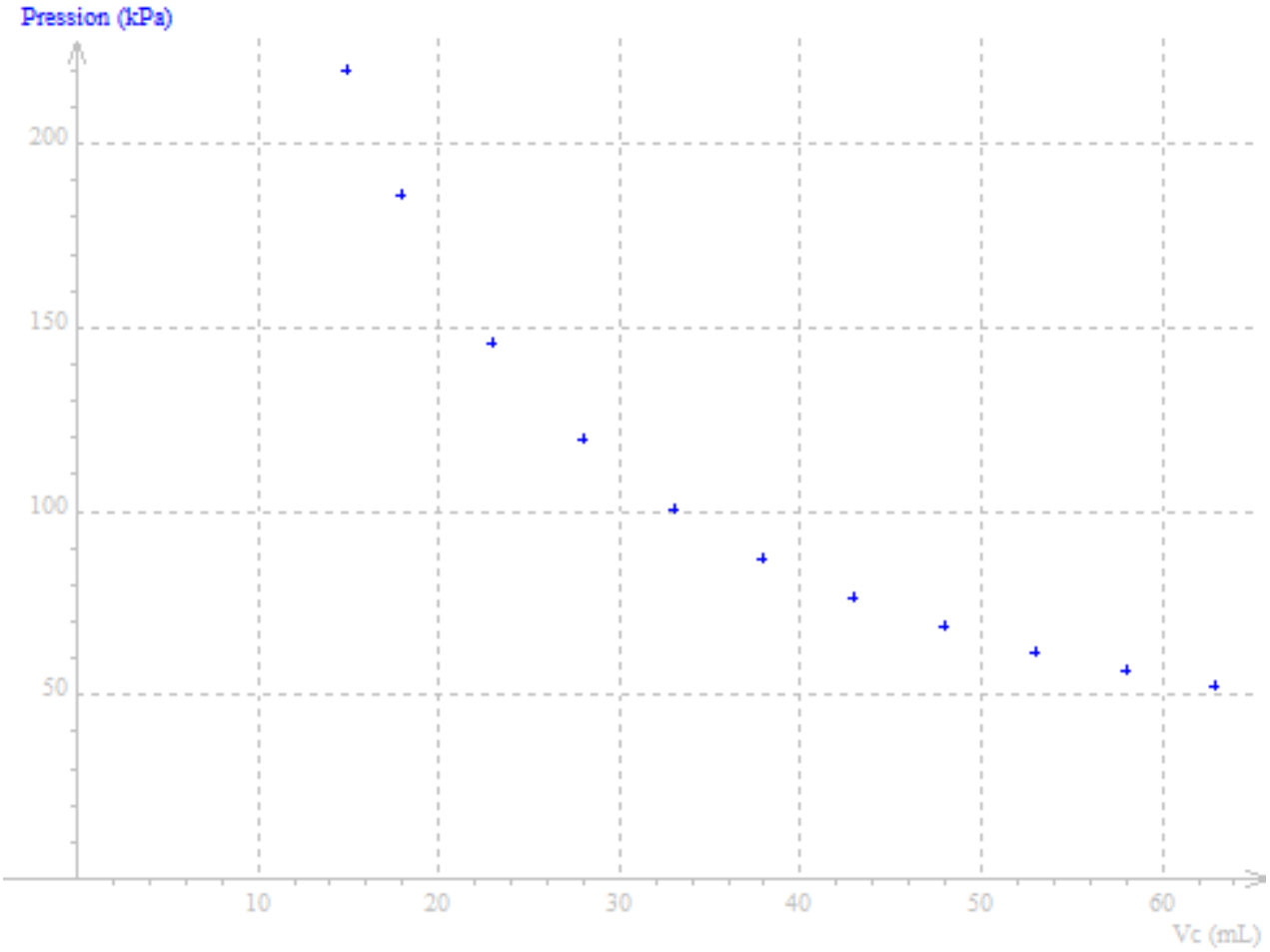




i	Volume	Pression	Vc	PV
	mL	kPa	mL	kPa.L
0	12,00	219,6	15,00	3,294
1	15,00	185,8	18,00	3,344
2	20,00	145,6	23,00	3,349
3	25,00	119,6	28,00	3,349
4	30,00	100,3	33,00	3,310
5	35,00	86,70	38,00	3,295
6	40,00	76,20	43,00	3,277
7	45,00	68,30	48,00	3,278
8	50,00	61,70	53,00	3,270
9	55,00	56,40	58,00	3,271
10	60,00	52,00	63,00	3,276
11				

La constante de la loi de Boyle-Mariotte est d'environ **3,3** kPa.L

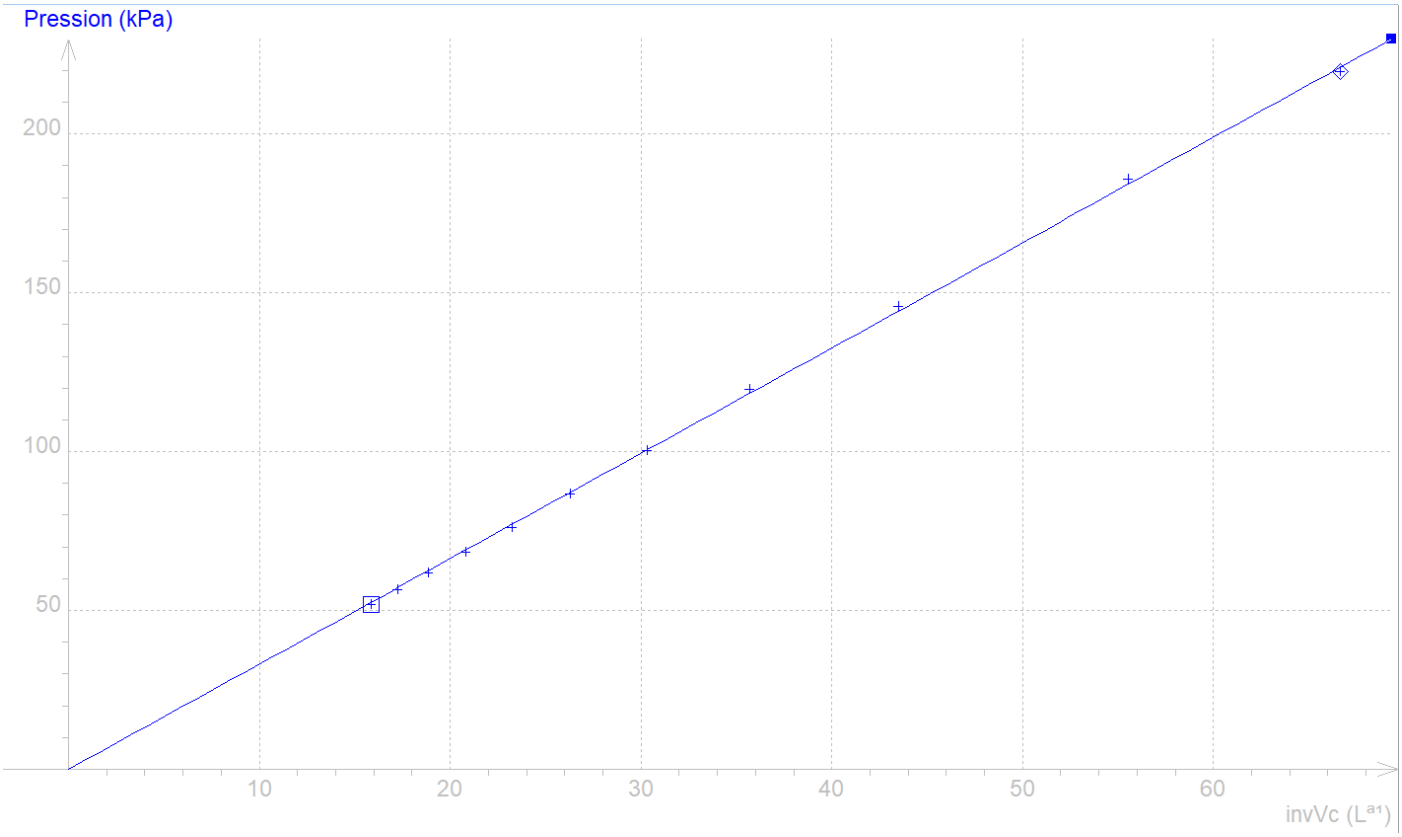
On peut tracer le graphe représentant la pression en fonction du volume corrigé :



Mais comme d'après la loi de Boyle-Mariotte :

P = k / V (où k est une constante en kPa.L),

Il est préférable de tracer le graphe représentant **P = f (1/Vc)**.



La représentation graphique de **P = f(1/Vc)** peut être modélisée par une fonction linéaire :

Expression du modèle	Résultats de la modélisation
Pression(invVc)=a*invVc	Ecart expérience-modèle 0,86 % sur Pression(invVc) Ecart quad. Pression=1,077 kPa a=3,32 ±0,02 J

On retrouve bien la valeur de la constante de la loi de Boyle-Mariotte d'environ **3,3** kPa.L approximée précédemment.

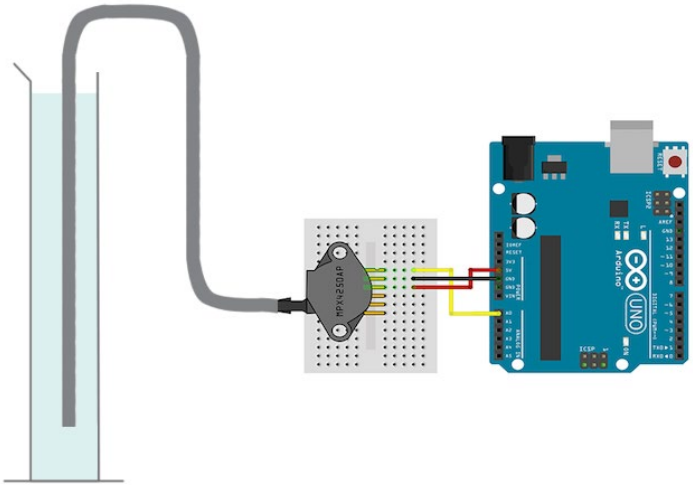
Remarques :

- Les résultats obtenus sont compatibles avec la loi de Boyle-Mariotte
- Les incertitudes dans les mesures sont dues à :
  . la précision du capteur,
  . la valeur de la tension de référence de l'Arduino qui n'est pas strictement égale à 5,0 V,
  . la précision de la mesure du volume d'air dans la seringue et le tuyau de connexion.

- Activité 4 : Principe fondamental de la statique des fluides

. Objectif

L'objectif de cette activité est de déterminer la profondeur d'immersion dans une colonne d'eau d'un tuyau relié à un capteur de pression MPX4250AP en appliquant le principe fondamental de la statique des fluides au montage suivant :



. Énoncé du principe fondamental de la statique des fluides

La statique des fluides constitue l'étude des fluides au repos.
Les fluides se déforment sous l'effet de forces très faibles, un fluide n'a pas de forme propre.

On distingue les liquides et les gaz :

- Le liquide prend la forme du récipient qui le contient, mais il est incompressible (ρ varie peu avec P et T).
- Le gaz occupe tout le volume mis à sa disposition et il est compressible (ρ varie beaucoup avec P et T)

La pression est la même en tout point d'un même plan horizontal d'un fluide homogène au repos.

La différence de pression (en Pa) entre deux points A et B (Cf. schéma ci-dessous) d'un fluide homogène au repos est égale à :

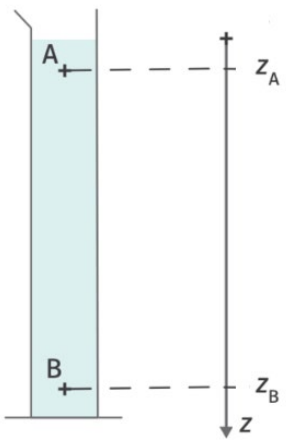
$$P_B - P_A = \rho\, g\, (Z_B - Z_A)$$

Où : . B est en-dessous de A,

. ρ = masse volumique du fluide en $kg.m^{-3}$,

. g = champ de pesanteur ($g = 9{,}81\ m.s^{-2}$),

. $Z_B - Z_A$ = distance entre les plans horizontaux passant par A et B en m.



. Descriptif de l'activité

Après avoir positionné le tuyau au niveau du point A, constituant l'origine du repère (Z_A=0), et mesuré la pression P_A de ce point, on déplacera l'extrémité du tuyau (Point B) dans la colonne d'eau. Le microcontrôleur mesurera en continu la pression du point B et calculera la distance (en m) entre les 2 points :

$$Z_B = \frac{(P_B - P_A)}{\rho g}$$

Avec ρ_{eau} = 1000 kg/m^3

On a donc :

$$Z_B = \frac{1000\ \Delta P(en\ kPa)}{9{,}81\ x\ 1000} = \frac{\Delta P(en\ kPa)}{9{,}81}$$

Remarques :

La sensibilité du capteur de pression MPX4250AP est de 20 mv/kPa.
La résolution par défaut du convertisseur analogique/numérique de l'Arduino étant de l'ordre de 5 mV, il est possible de mesurer des différences de pression de 0,25 kPa, soit une différence de profondeur d'immersion (Δz) d'environ 0,025 m (2,5 cm).

La précision de la mesure de Δz (Z_B - Z_A) sera donc de ± 2,5 cm.

. Le programme

Voici le code de l'activité en Python et en langage Arduino :

. Programme en Python ("Projet6\Activity4\PY\Activity4.py")

```
# Importations des librairies et définition de fonctions

from PyFirmataDef import *
from ConnectToArduino import *
import time


# Déclaration des constantes et variables

PinSensor = 0
PinButton = 12

ValSensor = 0
tension = 0
Pression = 0
PRef = 0
Dz=0
OldDz = 0

ValButton = 0
OldValButton = 0
State = 0
OldState = 0
BtnAppui = False


# Connexion à l'Arduino

print("\nConnexion à l'Arduino en cours...")

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)

InputPinSensor = AnalogInput(board, PinSensor)
InputPinButton = DigitalInput(board, PinButton)

ArduinoIterateur = Iterateur(board)
time.sleep(0.5)

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

print("Placez le tuyau à la position de référence et appuyez sur le bouton poussoir.\n")

while BtnAppui==False:
    ValButton = InputPinButton.read()
    if ValButton == 1 and OldValButton == 0:
       BtnAppui=True
    time.sleep(0.2)

OldValButton = ValButton
ValSensor = InputPinSensor.read()
tension = ValSensor*5.0
PRef = (tension / 0.02) + 10

print("\nDéplacez le tuyau et appuyez sur le bouton poussoir pour commencer les mesures.\n")


# Boucle principale du programme

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

        OldValButton = ValButton;
```

```
        if State==1:
            if OldState == 0:
               print("\nMesure de la pression en cours.\n")
               print("Pression (en kPa) ; Profondeur (cm):")
               OldState=1

            ValSensor = InputPinSensor.read()
            tension = ValSensor*5.0
            Pression = (tension / 0.02) + 10
            Dz = 100*(Pression-PRef)/9.81

            if (abs(OldDz - Dz)>2):
                print(round(Pression,1), " ; ",round(Dz,0))
                OldDz = Dz

            time.sleep(0.5)

        else:
            if OldState == 1:
                print("\nFin des mesures.\n")
                OldState = 0

    except KeyboardInterrupt:
        board.exit()
        sys.exit(0)
```

. <u>Résultats dans la fenêtre Python Shell</u> :

```
Connexion à l'Arduino en cours...

Connexion à l'Arduino établie - Appuyez sur Ctrl-C pour quitter

Placez le tuyau à la position de référence et appuyez sur le bouton poussoir.


Déplacez le tuyau et appuyez sur le bouton poussoir pour commencer les mesures.


Mesure de la pression en cours.

Pression (en kPa) ; Profondeur (cm):
103.1  ;  -3.0
103.3  ;  0.0
103.1  ;  -3.0
103.3  ;  0.0
103.1  ;  -3.0
103.3  ;  0.0
104.8  ;  15.0
106.3  ;  30.0
106.5  ;  32.0
106.3  ;  30.0
106.0  ;  28.0
106.3  ;  30.0
106.0  ;  28.0
106.3  ;  30.0
105.1  ;  18.0
103.8  ;  5.0
104.1  ;  7.0

Fin des mesures.

>>>
```

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 Satandard"**,

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

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

- Déclaration des constantes et variables :

. **PinSensor = 0** (cst correspondant au n° de la broche A0 sur laquelle le capteur de pression est connecté)

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

. **ValSensor = 0** (variable pour stocker la valeur de la broche du capteur de pression)

. **tension = 0** (variable correspondant à la valeur de la tension en V de la broche du capteur de pression)

. **Pression = 0** (variable correspondant à la pression en kPa calculée à partir de la valeur de la broche du capteur)

. **PRef = 0** (variable correspondant à la pression en kPa du point de référence)

. **Dz=0** (variable correspondant à la profondeur d'immersion du tuyau en cm)

. **OldDz = 0** (variable correspondant à la profondeur d'immersion du tuyau en cm précédente)

. **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)

. **OldState = 0** (variable pour stocker la valeur précédente de la variable **State**)

. **BtnAppui = False** (variable booléenne permettant de savoir si le bouton poussoir a été appuyé)

. **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 :

- Déclaration de la broche du capteur de pression en entrée analogique :

InputPinSensor = AnalogInput(board, PinSensor)

- Déclaration de la broche du bouton poussoir en entrée numérique :

InputPinButton = DigitalInput(board, PinButton)

- Lancement de l'itérateur :

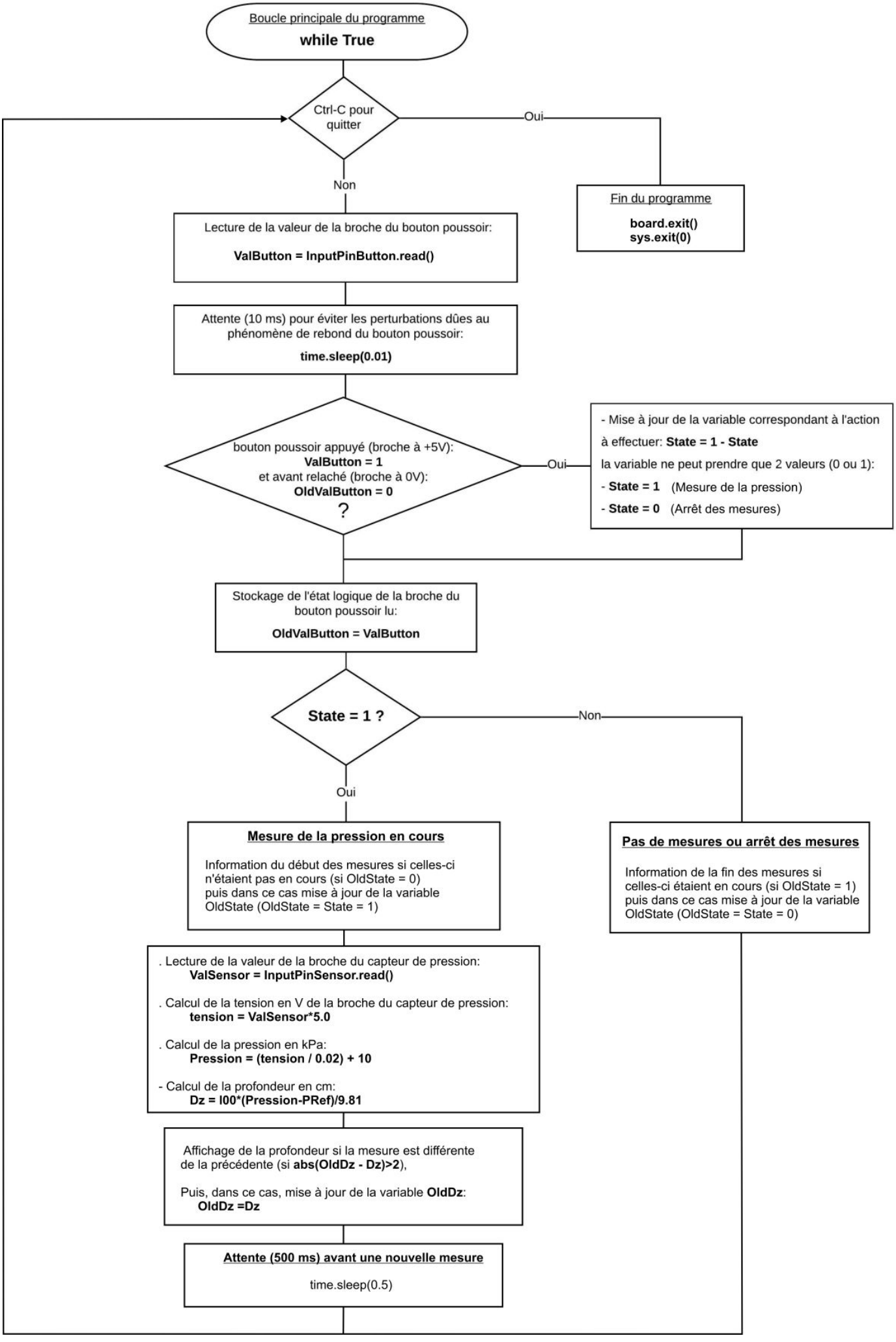
ArduinoIterateur = Iterateur(board)

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

time.sleep(0.5)

- Mesure de la pression à la position de référence.

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


Boucle principale du programme
while True
Ctrl-C pour quitter
Oui
Non
Fin du programme
board.exit()
sys.exit(0)
Lecture de la valeur de la broche du bouton poussoir:
ValButton = InputPinButton.read()
Attente (10 ms) pour éviter les perturbations dûes au phénomène de rebond du bouton poussoir:
time.sleep(0.01)
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 prendre que 2 valeurs (0 ou 1):
- State = 1 (Mesure de la pression)
- State = 0 (Arrêt des mesures)
Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Non
Oui
Mesure de la pression en cours
Information du début des mesures si celles-ci n'étaient pas en cours (si OldState = 0) puis dans ce cas mise à jour de la variable OldState (OldState = State = 1)
Pas de mesures ou arrêt des mesures
Information de la fin des mesures si celles-ci étaient en cours (si OldState = 1) puis dans ce cas mise à jour de la variable OldState (OldState = State = 0)
. Lecture de la valeur de la broche du capteur de pression:
ValSensor = InputPinSensor.read()
. Calcul de la tension en V de la broche du capteur de pression:
tension = ValSensor*5.0
. Calcul de la pression en kPa:
Pression = (tension / 0.02) + 10
- Calcul de la profondeur en cm:
Dz = l00*(Pression-PRef)/9.81
Affichage de la profondeur si la mesure est différente de la précédente (si abs(OldDz - Dz)>2),
Puis, dans ce cas, mise à jour de la variable OldDz:
OldDz =Dz
Attente (500 ms) avant une nouvelle mesure
time.sleep(0.5)

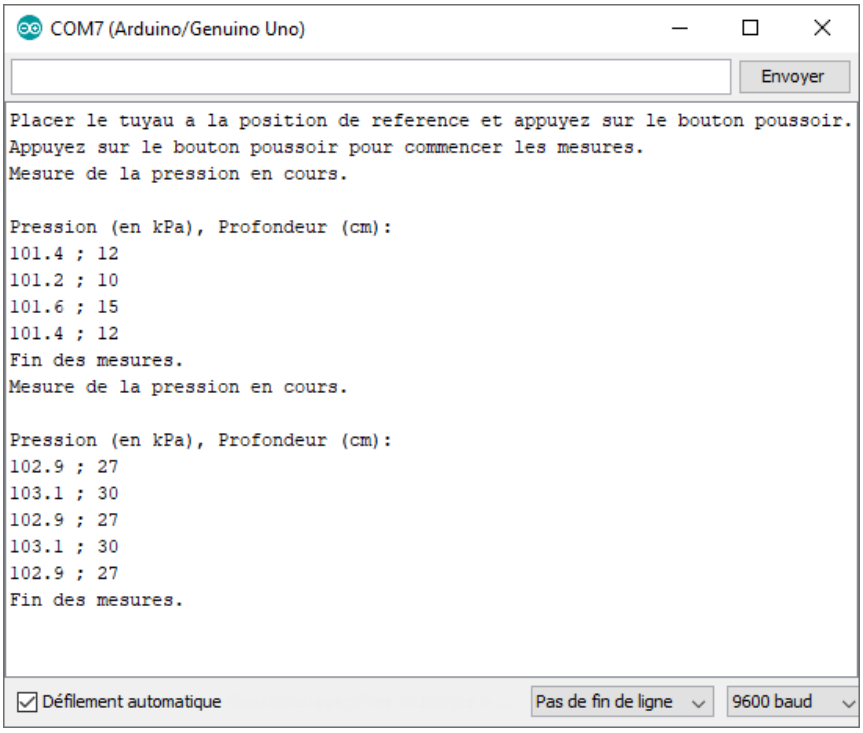
. Programme en langage Arduino (”Projet6\Activity4\INO\Activity4.ino”)

Activity4

```
// Déclaration des constantes et variables

const int PinSensor=0;
const int PinButton= 12;

int ValSensor = 0;
float tension = 0.0;
float Pression = 0.0;
float PRef = 0.0;
float Dz=0.0;
float OldDz = 0.0;

int ValButton = 0;
int OldValButton = 0;
int State = 0;
int OldState = 0;
boolean BtnAppui = false;


// Initialisation des entrées et sorties

void setup() {
  Serial.begin(9600);
  pinMode(PinButton,INPUT);
  Serial.print("Placer le tuyau a la position de reference ");
  Serial.println("et appuyez sur le bouton poussoir.");
  while (BtnAppui==false){
    ValButton = digitalRead(PinButton);
    if ((ValButton == HIGH)&&(OldValButton == LOW)){
       BtnAppui=true;
    delay (200);
    }
  }
  OldValButton = ValButton;
  ValSensor = analogRead(PinSensor);
  tension = (ValSensor/1023.0)*5.0;
  PRef = (tension / 0.02) + 10;
  Serial.println("Appuyez sur le bouton poussoir pour commencer les mesures.");
}

// Fonction principale en boucle

void loop() {
ValButton = digitalRead(PinButton);
delay(10);

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

```
if (State==1)
  {
  if (OldState == 0)
  {
   Serial.println("Mesure de la pression en cours.");
   Serial.println("");
   Serial.println ("Pression (en kPa), Profondeur (cm):");
   OldState=1;
  }
  ValSensor = analogRead(PinSensor);
  tension = (ValSensor/1023.0)*5.0;
  Pression = (tension / 0.02) + 10;
  Dz = 100*(Pression-PRef)/9.81;

  if (abs(OldDz - Dz)>2)
  {
    Serial.print(Pression,1);
    Serial.print(" ; ");
    Serial.println(Dz,0);
    OldDz = Dz;
  }
  delay(500);
  }
  else
  {
    if (OldState == 1){
    Serial.println("Fin des mesures.");
    OldState = 0;}
  }
}
```

Résultats dans le moniteur série :



Déroulement du programme :

- Déclaration des constantes et variables :

. **const int PinSensor = 0** (broche du capteur de pression)
. const int PinButton = 12 (Broche du bouton poussoir)

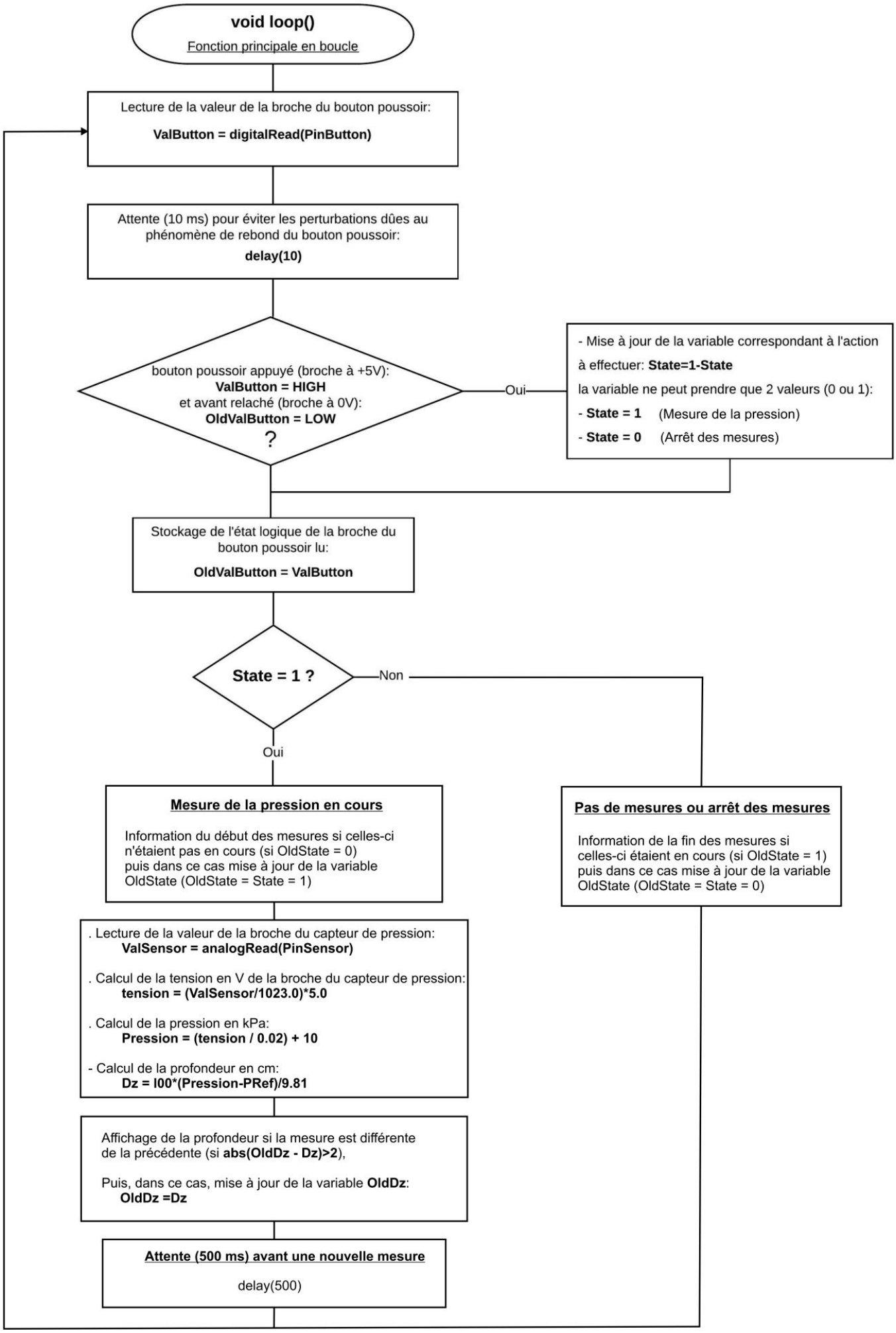
. **int ValSensor = 0** (variable nombre entier valeur broche du capteur)
. **float tension = 0.0** (variable nombre décimal calcul tension broche capteur)
. float Pression = 0.0 (variable nombre décimal calcul pression)
. float Pref = 0.0 (variable nombre décimal pression point référence)
. float Dz=0.0 (variable nombre décimal profondeur en cm)
. float OldDz = 0.0 (variable nombre décimal ancienne valeur profondeur en cm)

. int ValButton = 0 (variable nombre entier valeur broche bouton poussoir)
. int OldValButton = 0 (variable nbr entier ancienne valeur broche btn poussoir)
. int State = 0 (variable nombre entier pour action à effectuer)
. int OldState = 0 (variable nombre entier pour action effectuée précédemment)
. boolean BtnAppui = false (variable booléenne indiquant appui sur btn poussoir)

- Initialisation des entrées et sorties :

. Initialisation de la liaison série à un débit de 9600 bauds
. Initialisation de la broche du bouton poussoir en entrée
. Mesure de la pression au point de référence (Z_A=0)

- Fonction principale en boucle :


void loop()
Fonction principale en boucle
Lecture de la valeur 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 (Mesure de la pression)
- State = 0 (Arrêt des mesures)
Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Non
Oui
Mesure de la pression en cours
Information du début des mesures si celles-ci n'étaient pas en cours (si OldState = 0) puis dans ce cas mise à jour de la variable OldState (OldState = State = 1)
Pas de mesures ou arrêt des mesures
Information de la fin des mesures si celles-ci étaient en cours (si OldState = 1) puis dans ce cas mise à jour de la variable OldState (OldState = State = 0)
. Lecture de la valeur de la broche du capteur de pression:
ValSensor = analogRead(PinSensor)
. Calcul de la tension en V de la broche du capteur de pression:
tension = (ValSensor/1023.0)*5.0
. Calcul de la pression en kPa:
Pression = (tension / 0.02) + 10
- Calcul de la profondeur en cm:
Dz = I00*(Pression-PRef)/9.81
Affichage de la profondeur si la mesure est différente de la précédente (si abs(OldDz - Dz)>2),
Puis, dans ce cas, mise à jour de la variable OldDz:
OldDz =Dz
Attente (500 ms) avant une nouvelle mesure
delay(500)