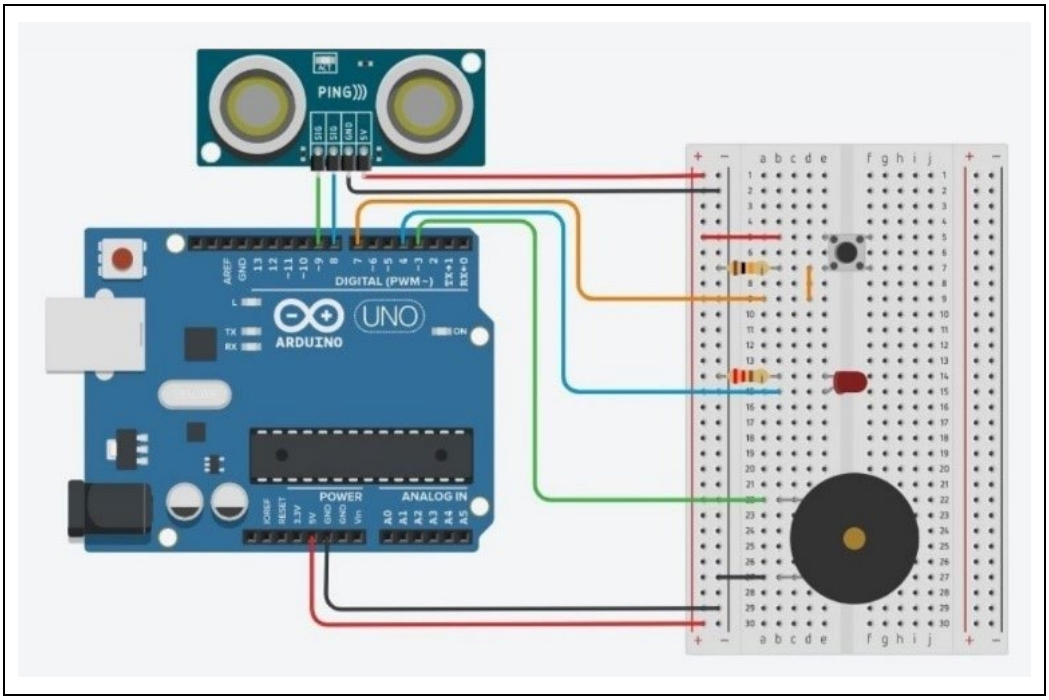
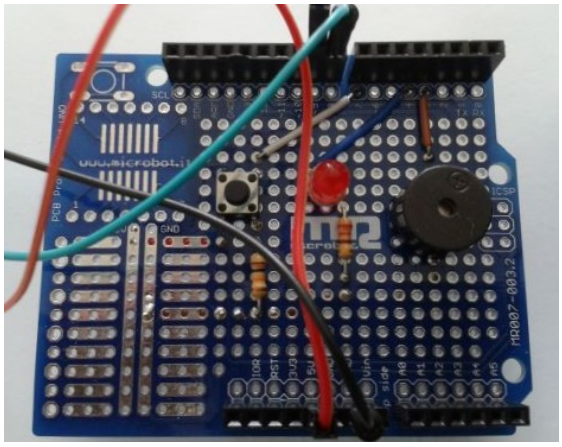
Projet 4- Ondes ultrasonores : Vitesse & Distances

Nous avons vu qu'il était possible, avec un Arduino, de produire des ondes sonores, caractérisées par leur fréquence. Nous allons maintenant, nous intéresser à la vitesse de propagation des ondes sonores.



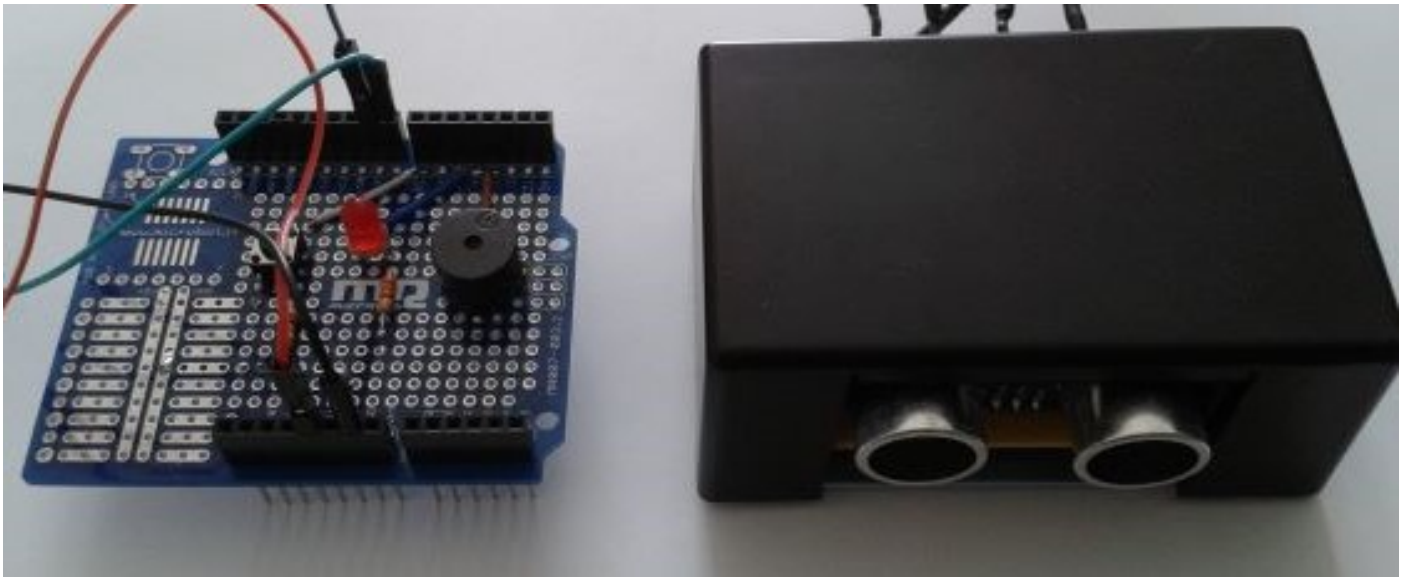
- Liste des composants :



. 1 capteur ultrasonique (par exemple, le HC-SR04)
. 1 DEL Rouge
. 1 résistance de 220 Ω
. 1 résistance de 10 kΩ
. 1 bouton poussoir
. 1 haut-parleur (ou piezzo)
. 1 plaque d'essai
. Fils de connexion

- Protocole de communication:

. Firmata Express



Le circuit sur un ”shield” pour Arduino Uno

Rappels :

Le son est une onde mécanique qui se propage dans un milieu matériel fluide (air, eau) ou solide et les ondes sonores sont caractérisées par leur fréquence.

Les sons audibles par l'Homme ont des fréquences comprises entre 20 et 20 000 Hz, les infrasons ont une fréquence inférieure à 20 Hz, et les ultrasons sont situés au-delà de 20 kHz.

La vitesse de propagation, ou célérité, du son est indépendante de sa fréquence mais dépend du milieu de propagation : plus le milieu matériel est dense plus la vitesse est grande.

Par exemple : c (air) = 340 $m.s^{-1}$; c (eau de mer) = 1 500 $m.s^{-1}$; c (acier) = 5 000 $m.s^{-1}$

La célérité du son dépend de la température, c'est-à-dire de l'agitation des particules qui constituent le milieu de propagation : plus la température est élevée plus le son se propage vite.

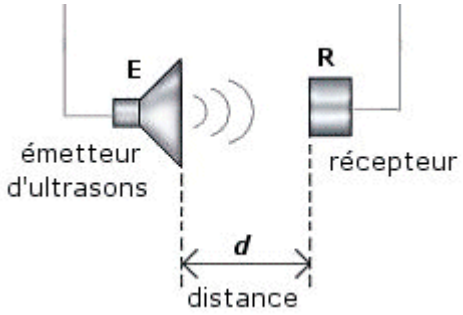
Par exemple : c (air à 0°C) = 331 $m.s^{-1}$; c (air à 15°C) = 340 $m.s^{-1}$

La relation entre la vitesse du son dans l'air en m/s et la température en kelvins est:

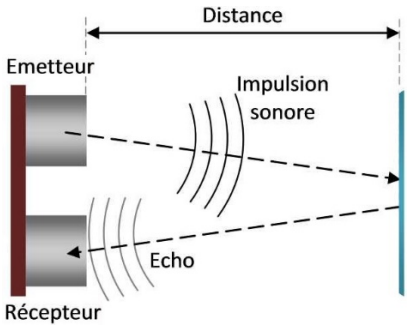
$$c_{\text{air}} = 20{,}05\sqrt{T}$$ (T en kelvins = T en °C + 273,15)

La célérité dans l’air, en $m.s^{-1}$, peut être déterminée expérimentalement en mesurant la durée de propagation Dt, en s, de l'onde sonore, entre un émetteur et un récepteur situés à une distance d, en m, grâce à la relation :

$$C_{air} = \frac{d}{Dt}$$



On peut également utiliser un ensemble émetteur - récepteur d'onde ultrasonores placé devant un obstacle. C'est le principe de la mesure par écho ou du Sonar :



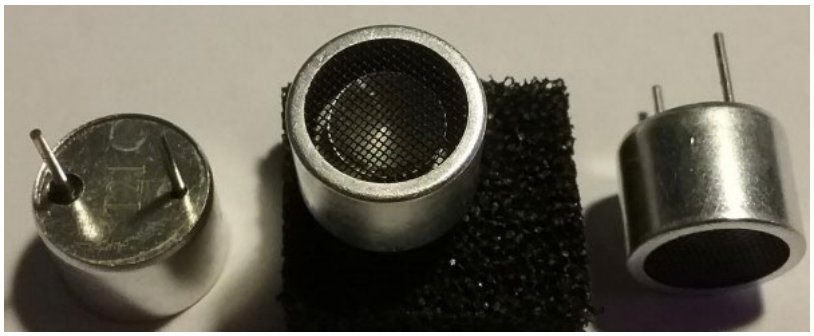
Dans ce cas, la distance parcourue par l'onde sonore, pendant la durée Dt, est 2 d, et alors :

$$C_{air} = \frac{2d}{Dt}$$

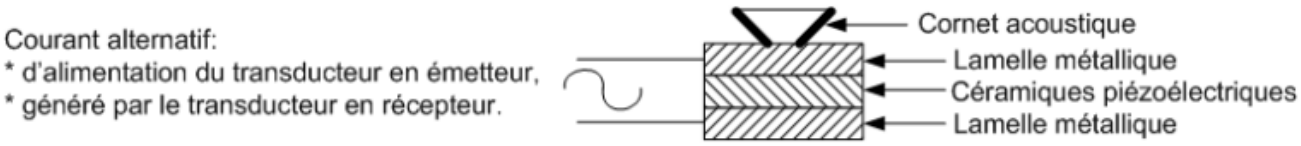
<u>Principe de fonctionnement des émetteurs et récepteurs à ultrasons</u> :

Les émetteurs et récepteurs à ultrasons sont aussi appelés transducteurs piézoélectriques, car ils convertissent une énergie électrique en énergie mécanique et réciproquement. Le principe de fonctionnement est donc identique à celui des buzzer.

Extérieurement les transducteurs d'émission (généralement, repérés par un « T » gravé) sont très semblables à ceux de réception (généralement, repérés par un « R » gravé) :



Un schéma de principe de ces transducteurs est donné ci-dessous :



. Fonctionnement d'un émetteur US :

Le transducteur est alimenté par une tension alternative à une fréquence nominale de fonctionnement (souvent 40 KHz). Cette tension, est appliquée sur les lamelles métalliques ce qui génère une déformation mécanique des céramiques qui est transformée en pression acoustique appliquée à l'air ambiant, via le cornet acoustique.

. Fonctionnement d'un récepteur US :

La pression acoustique (due à l'onde ultrasonore) reçue à travers l'air ambiant, via le cornet acoustique du récepteur US, est transformée en contrainte mécanique dans les céramiques qui génèrent des charges électriques sur les lamelles métalliques et donc une tension alternative à ses bornes.

Avec un Arduino, l'ensemble émetteur - récepteur d'onde ultrasonores ou le capteur ultrasonique le plus couramment utilisé est le HC-SR04.

Capteur ultrasonique HC-SR04

Caractéristiques

Le capteur est composé d'un émetteur d'ultrasons, d'un récepteur et du circuit de commande. Il est généralement utilisé pour mesurer des distances entre le capteur et un obstacle.

- Dimensions : 45 mm x 20 mm x 15 mm
- Plage de mesure : 2 cm à 400 cm
- Résolution de la mesure annoncée : 0,3 cm (en pratique : 1 cm)
- Angle de mesure efficace : 15 °

Broches de connexion

- Vcc = Alimentation +5 V DC
- Trig = Entrée émetteur d'impulsion d'ultrasons (Trigger input)
- Echo = Sortie récepteur d'impulsion d'ultrasons (Echo output)
- GND = Masse 0V

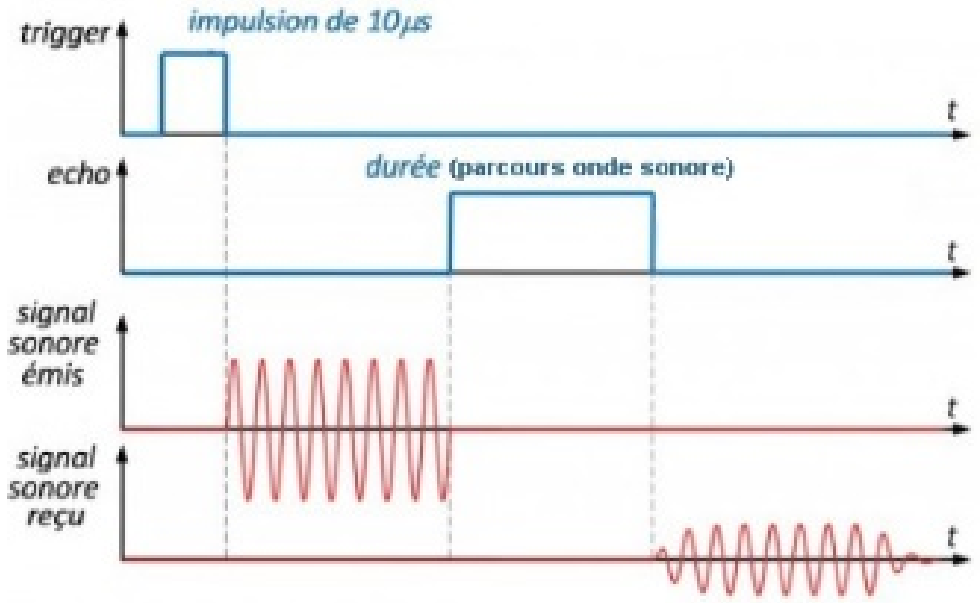
Spécifications et limites

- Tension d'alimentation : 5.0 V à ± 0.5 V
- Courant de repos : 2.0 mA à ± 0.5 mA
- Courant de fonctionnement : 15 ± 5 mA
- Fréquence des ultrasons : 40 kHz

Le principe de fonctionnement :

1. Envoyer un signal numérique à l'état haut sur l'émetteur pendant 10 µs,
2. Le capteur envoie automatiquement 8 impulsions d'ultrasons à 40 kHz,
3. A la fin des 8 impulsions, la sortie Echo du capteur passe à l'état haut,
4. Si le signal revient et est détecté par le récepteur, la sortie Echo du capteur passe à l'état bas. La durée de l'état haut du signal Echo correspond au temps entre l'émission des ultrasons et leur réception.

Le principe de fonctionnement est résumé sur le schéma suivant :



La formule couramment utilisée dans les programmes Arduino permettant de calculée la distance entre le capteur et un obstacle est :

Distance $_{\text{capteur- obstacle}}$ **(en cm) = durée propagation (en µs) / 58**

En effet, pour cela, on suppose que la vitesse des ultrasons dans l'air est de V = 340 $m.s^{-1}$, la distance parcourue, **d** (en m), par l'onde sonore pendant la durée, **Dt** (en s), est alors :

$$\mathbf{d_{parcours\ onde\ sonore} = V_{ultrasons}\ x\ Dt}$$

Soit : Distance $_{\text{capteur- obstacle}}$ (en m) = d $_{\text{parcours onde sonore}}$ /2 = $V_{ultrasons}$ x Dt / 2

Distance $_{\text{capteur- obstacle}}$ (en m) = 340 x Dt / 2

Distance $_{\text{capteur- obstacle}}$ (en cm) = 34000 x Dt (en µs) /2000000 = 17x Dt (en µs) / 1000

Distance $_{\text{capteur- obstacle}}$ **(en cm) = Dt (en µs) / 58,82**

Le capteur ultrasonique **HC-SR04** dispose de 2 broches différentes pour l'émission et la réception des ultrasons.

Il existe également des capteurs ultrasoniques à une seule broche, comme le **Grove 101020010**

Capteur ultrasonique Grove 101020010

Caractéristiques

Le capteur est composé d'un émetteur d'ultrasons, d'un récepteur et du circuit de commande. Il est généralement utilisé pour mesurer des distances entre le capteur et un obstacle.

- Dimensions : 50 mm x 25 mm x 16 mm
- Plage de mesure : 2 cm à 350 cm
- Résolution de la mesure : 1 cm
- Angle de mesure efficace : 15 °

Broches de connexion

- Vcc = Alimentation +5 V DC
- SIG = Entrée émetteur et sortie récepteur d'impulsion d'ultrasons
- GND = Masse 0V
- NC = Non connectée

Spécifications et limites

- Tension d'alimentation : 3,2 V à 5,2 V
- Courant de fonctionnement : 8 mA
- Fréquence des ultrasons : 40 kHz
- Température de fonctionnement : 10 - 60 °C

Le principe de fonctionnement :

1. Déclarer la broche de l'Arduino sur laquelle est connectée la broche **SIG** du capteur en sortie numérique et la maintenir à l'état bas pendant 2 µs,

2. Envoyer un signal numérique à l'état haut sur l'émetteur (sur la broche SIG) pendant 5 µs,

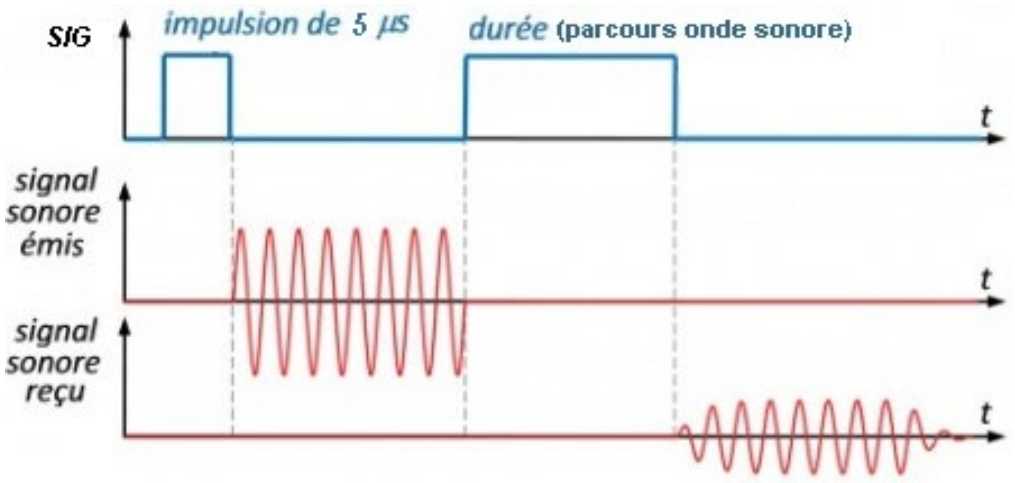
3. Le capteur envoie automatiquement des impulsions d'ultrasons à 40 kHz,

4. Déclarer la broche de l'Arduino sur laquelle est connectée la broche **SIG** du capteur en entrée numérique, afin de pouvoir recevoir le signal du récepteur d'ultrasons.

5. A la fin des impulsions, la broche **SIG** du capteur passe à l'état haut,

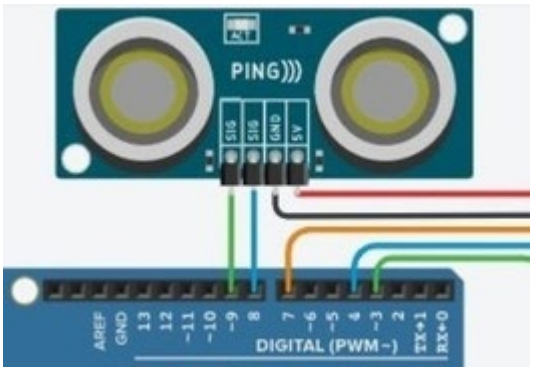
6. Si le signal revient et est détecté par le récepteur, la broche **SIG** du capteur passe à l'état bas. La durée de l'état haut du signal **SIG** correspond au temps entre l'émission des ultrasons et leur réception.

Le principe de fonctionnement est résumé sur le schéma suivant :

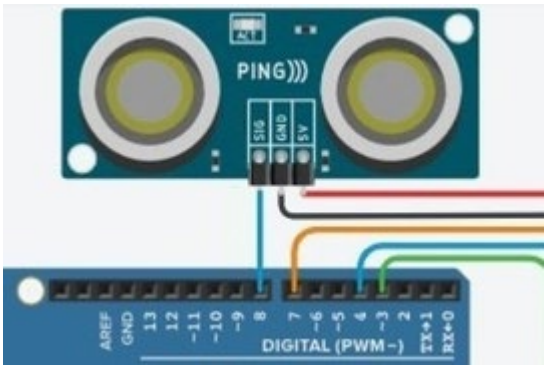


Le principe de calcul de la distance entre ce capteur et un obstacle reste le même.

Par défaut, les programmes sont conçus pour un capteur ultrasonique disposant de deux broches différentes pour l'émission et la réception des ultrasons. Pour utiliser un capteur ultrasonique à une broche commune pour l'émission et la réception, il suffit de modifier le programme et de donner la même valeur aux variables TRIGGER_PIN et ECHO_PIN.



Capteur à 2 broches différentes Trigg et Echo



Capteur à 1 broche commune pour Trigg et Echo

La prise en charge des capteurs ultrasoniques par le protocole de communication "Firmata Express"

Avec "**pymata-express**", pour utiliser un capteur ultrasonique, il faut déclarer les broches servant à l'émission et à la réception des ultrasons avec la commande :

loop.run_until_complete(board.set_pin_mode_sonar(trigger_pin, echo_pin, Get_Distance))

où :

- **"board"** est l'objet créé lors de l'appel de la méthode "**PymataExpress**" du module "**pymata-express**",
- "**trigger_pin**" est le numéro de la broche du microcontrôleur sur laquelle est connecté l'émetteur d'ultrasons,
- "**echo_pin**" est le numéro de la broche du microcontrôleur sur laquelle est connecté le récepteur d'ultrasons,
- **loop**" est la boucle des tâches "**asyncio**" déclarée ainsi **:**

  loop = asyncio.get_event_loop(),

- "**Get_Distance**" est une fonction "**asyncio**" pour récupérer la valeur de la distance en cm entre le capteur et un obstacle :

```
async def Get_Distance(data):
    global Distance
    Distance = data[1]
```

Remarque : "**data**" est une liste de 2 éléments (nombres décimaux entiers) générée lors de la déclaration des broches du capteur dont le premier élément (data[0]) est la valeur de "**trigger_pin**" et le deuxième élément (data[1]) est la distance mesurée en cm.

Dans le cas d'un capteur ultrasonique à une broche commune pour l'émission et la réception des ultrasons, il suffit de déclarer la même broche pour "**trigger_pin**" et "**echo_pin**".

On peut définir une fonction déclarant plus facilement les broches d'un capteur ultrasonique :

```
def Set_Sonar_Pins(board, trigger_pin, echo_pin):
    loop.run_until_complete(board.set_pin_mode_sonar(trigger_pin, echo_pin, Get_Distance))
```

Ainsi, dans le cas d'un capteur ultrasonique à 2 broches différentes pour l'émission (broche 8) et la réception (broche 9) des ultrasons, l'instruction de déclaration est :

Set_Sonar_Pins(board, 8, 9)

Pour obtenir la valeur de la distance en cm entre le capteur et l'obstacle, on utilise la fonction "**sonar_read**" qui fait appel à la fonction "**Get_Distance**" définie lors de la déclaration des broches du capteur :

loop.run_until_complete(board.sonar_read(trigger_pin))

où :

- **"board"** est l'objet créé lors de l'appel de la méthode "**PymataExpress**" du module "**pymata-express**",
- "**trigger_pin**" est le numéro de la broche du microcontrôleur sur laquelle est connecté l'émetteur ultrasons.

Si on définit une fonction, comme ci-dessous :

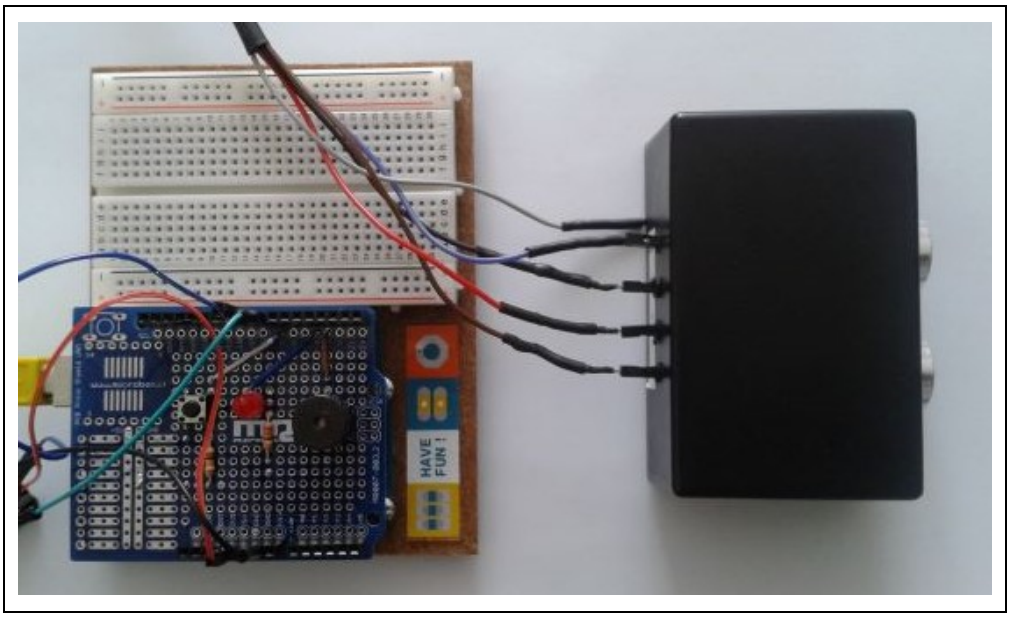
```
def Sonar_Read(board, trigger_pin):
   loop.run_until_complete(board.sonar_read(trigger_pin))
```

Pour obtenir la valeur de la distance en cm entre le capteur (connecté aux broches 8 et 9) et l'obstacle, on appelle la fonction :

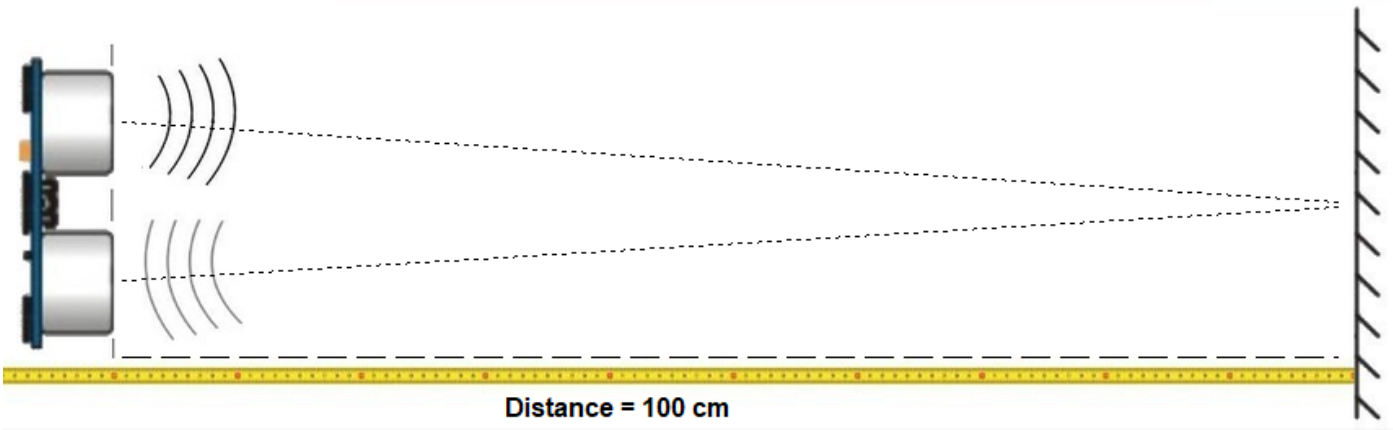
Sonar_Read(board,8)

La variable globale "**Distance**" est alors mise à jour via la fonction "**Get_Distance**", celle-ci étant appelée lors de l'appel de la fonction " **Sonar_Read** ".

- Activité 1 : Détermination de la vitesse du son dans l'air



Dans cette activité, nous allons déterminer expérimentalement la vitesse de propagation des ondes sonores en mesurant, à l'aide d'un capteur à ultrasons (par exemple, le HC-SR04), la durée de propagation, **Dt**, de l'onde sonore entre l'émetteur et le récepteur situés à une distance, **d**, connue d'un obstacle.



Le programme ci-dessous est adapté pour un capteur ultrasonique à deux broches différentes pour l'émission et la réception des ultrasons.

Dans le cas d'un capteur ultrasonique à une broche commune pour l'émission et la réception des ultrasons, il suffit de déclarer la même broche pour "trigger_pin" (émetteur US) et "echo_pin" (récepteur US).

. Programme en Python (Projet4\Activity1\PY\Activity1.py)

```
# Importations des librairies et définition des fonctions

from PymataExpressDef import *
from ConnectToArduino import *
import time


async def Get_Dt(data):
    global Dt
    Dt = data[1]*56

def Set_Sonar_Pins(board, trigger_pin, echo_pin):
    loop.run_until_complete(board.set_pin_mode_sonar(trigger_pin, echo_pin, Get_Dt))

def Sonar_Read(board, trigger_pin):
    loop.run_until_complete(board.sonar_read(trigger_pin))


# Déclaration des constantes et variables

TRIGGER_PIN = 8
ECHO_PIN = 9
PinButton = 7

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

Distance = 0
Vitesse = 0
Dt = 0
DtMesure = 0
Distancechoisie = False


# Connexion à l'Arduino

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)


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

Set_Sonar_Pins(board, TRIGGER_PIN, ECHO_PIN)
Set_DigitalInput_Pin(board, PinButton)

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


# Boucle principale du programme

print("Appuyez sur le bouton poussoir pour mesurer la durée de propagation de l'onde sonore.")

while True:
    try:
        ValButton = Digital_Read(board, PinButton)
        time.sleep(0.01)
        if ValButton == 1 and OldValButton==0:
            State = 1 - State
        OldValButton = ValButton
        if State==1:
            if OldState == 0:
                while Distancechoisie==False:
                    Distance = input("\nVeuillez entrer la distance en cm entre le capteur et l'obstacle:")
                    try:
                        Distance = int(Distance)

                        assert Distance >= 3 and Distance <= 200
                        Distancechoisie = True

                    except AssertionError:
                        print("La distance choisie n'est pas entre 3 et 200 cm!")
                        Distancechoisie = False

                    except:
                        print("vous n'avez pas saisi une distance entre 3 et 200 cm!")
                        Distancechoisie = False
```

```
                OldState=1
                print("\nMesure de la durée de propagation de l'onde sonore en cours.")
                print("\nDistance (cm) ; Dt (microS) ; Vitesse onde sonore (m/s) :\n")

            Sonar_Read(board,TRIGGER_PIN)
            if DtMesure != Dt:
                Vitesse=int(2*int(Distance) / Dt * 10000)
                print(Distance, " ; ", Dt , " ; ", Vitesse)
                DtMesure = Dt
            time.sleep(.01)
        else:
            if OldState == 1:
                print("\nFin des mesures.")
                OldState = 0
                Distancechoisie = False

    except KeyboardInterrupt:
        Arduino_Exit(board)
        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 Express”**,

. Le module "**PymataExpressDef.Py**" regroupant toutes les fonctions utiles à l’utilisation de " **Pymata-express** " (fonction de déclaration des entrées et sorties, de lectures, d’écritures…) ,

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

. La fonction **”Get_Dt”** pour obtenir la durée de propagation dans l’air de l’onde sonore,

. La fonction **”Set_Sonar_Pins”** pour déclarer les broches du capteur ultrasonique,

. La fonction **”Sonar_Read”** qui appelle la fonction **”Get_Dt”.**

- Déclaration des constantes et variables :

. **TRIGGER_PIN = 8** (cst correspondant au n° de la broche sur laquelle l’émetteur US est connecté)

. **ECHO_PIN = 9** (cst correspondant au n° de la broche sur laquelle le récepteur US est connecté)

. **PinButton = 7** (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)

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

. **Distance = 0** (variable correspondant à la distance en cm entre le capteur et l’obstacle)

. **Vitesse = 0** (variable correspondant à la vitesse des ondes ultrasonores dans l'air en m/s)

. **Dt = 0** (variable correspondant à la durée de propagation des ondes ultrasonores en µs entre le capteur et l'obstacle)

. **DtMesure = 0** (variable correspondant à la durée de propagation des ondes ultrasonores en µs précédente)

. **Distancechoisie = False** (variable booléenne indiquant si la distance entre le capteur et l'obstacle a été saisie)

. **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 :

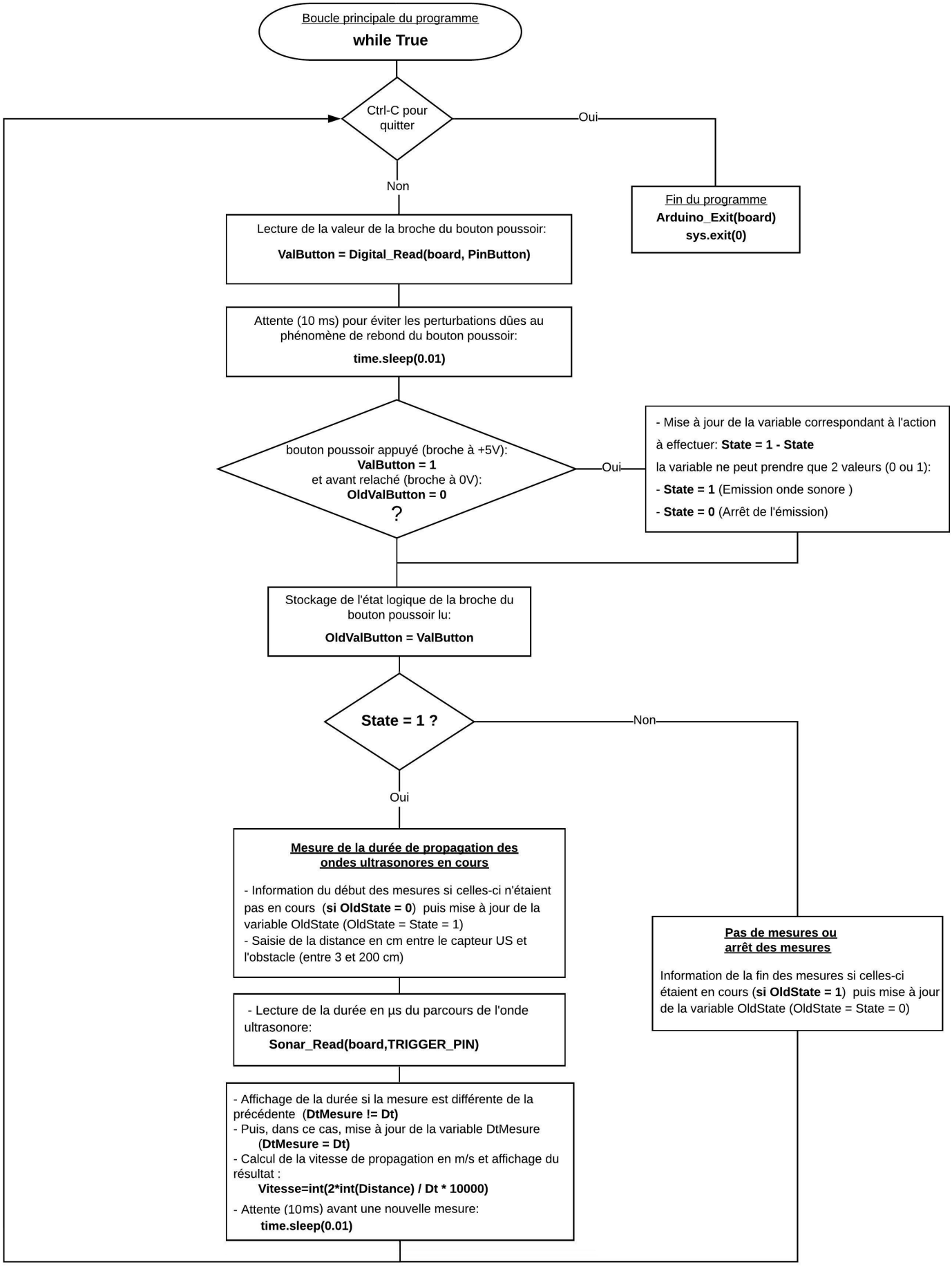
- Déclaration des broches du capteur ultrasonique :

Set_Sonar_Pins(board, TRIGGER_PIN, ECHO_PIN)

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

Set_DigitalInput_Pin(board, PinButton)

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



Résultats dans la fenêtre Python Shell :

```
Opening COM3 ...
Waiting 4 seconds for the Arduino To Reset.

Arduino found and connected to COM3

Retrieving Arduino Firmware ID...
Arduino Firmware ID: 2.5 FirmataExpress.ino
Auto-discovery complete. Found 20 Digital Pins and 6 Analog Pins


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

Appuyez sur le bouton poussoir pour mesurer la durée de propagation de l'onde sonore.

Veuillez entrer la distance en cm entre le capteur et l'obstacle:30

Mesure de la durée de propagation de l'onde sonore en cours.

Distance (cm) ; Dt (microS) ; Vitesse onde sonore (m/s) :

30  ;  1736  ;  345
30  ;  1792  ;  334
30  ;  1736  ;  345
30  ;  1792  ;  334
30  ;  1736  ;  345
30  ;  1792  ;  334

Fin des mesures.
>>>
```

. Programme en langage Arduino (Projet4\Activity1\INO\Activity1.ino)

Activity1

```
// Déclaration des constantes et variables

int TRIGGER_PIN = 8;
int ECHO_PIN = 9;
const int PinButton = 7;

const unsigned long MEASURE_TIMEOUT = 25000UL;

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

long DtMesure = 0;
long Dt = 0.0;
int Distance = 0;
float Vitesse = 0.0;


// Initialisation des entrées et sorties

void setup()
{
Serial.begin(9600);
pinMode(PinButton,INPUT);
pinMode(TRIGGER_PIN, OUTPUT);
digitalWrite(TRIGGER_PIN, LOW);
if (ECHO_PIN != TRIGGER_PIN){
pinMode(ECHO_PIN, INPUT);}

while(Distance<3 || Distance>200)
  {
  int Val=0;
  char tampon[10]="";
  Serial.println("Veuillez entrer la distance en cm entre le capteur et l'obstacle (valeur entre 3 et 200):");
  while(!Val)
    {
    delay(200);
    Val=Serial.available();
    }
  for (int i=0; i < Val; i++)
    {
    tampon[i]=Serial.read();
    delay(15);
    }
  Distance = atoi(tampon);
  }
Serial.print("Distance entre le capteur et l'obstacle = ");    Serial.print(Distance);    Serial.println(" cm");
Serial.println("Appuyez sur le bouton poussoir pour mesurer la duree de propagation de l'onde sonore.");
}


// 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 duree de propagation de l'onde sonore en cours.");
   Serial.println("");
   Serial.println("Dt (microS) ; Vitesse onde sonore (m/s):");
   OldState=1;
  }

  if (ECHO_PIN != TRIGGER_PIN){
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN, LOW);
  Dt = pulseIn(ECHO_PIN, HIGH, MEASURE_TIMEOUT);}
  else {
  pinMode(TRIGGER_PIN, OUTPUT);
  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(5);
  digitalWrite(TRIGGER_PIN,LOW);
  pinMode(ECHO_PIN,INPUT);
  Dt = pulseIn(ECHO_PIN,HIGH);}

  if (DtMesure != Dt)
  {
  Vitesse = 2*float(Distance)/Dt*10000;
  Serial.print(Dt); Serial.print(" ; ");  Serial.println(int(Vitesse));
  DtMesure = Dt ;
  }
  Serial.flush();
  delay(100);
}
else
{
  if (OldState == 1){
  Serial.println("Fin des mesures (Appuyez sur le bouton Reset pour modifier la distance).");
  OldState = 0;}
}
}
```

Déroulement du programme :

Activité 1

Déclaration des constantes et variables

- N° de la broche correspondant au bouton poussoir: **const int PinButton = 7**
- N° de la broche correspondant à l'émetteur US : **int TRIGGER_PIN = 8**
- N° de la broche correspondant au récepteur US : **int ECHO_PIN = 9**
- Constante pour définir la durée maximale des mesures : **const unsigned long MEASURE_TIMEOUT = 25000UL**
- Variable pour stocker la valeur de la broche du bouton poussoir: **int ValButton = 0**
- Variable pour stocker l'ancienne valeur de la broche du bouton poussoir: **int OldValButton = 0**
- Variable correspondant à l'action à effectuer: **int State = 0**
- Variable pour stocker l'ancienne valeur de la variable correspondant à l'action à effectuer: **int OldState = 0**
- Variable correspondant à la durée de parcours de l'onde ultrasonore: l**ong Dt = 0**
- Variable correspondant à la valeur précédente de la durée de parcours de l'onde ultrasonore: l**ong DtMesure = 0**
- Variable correspondant à la distance entre le capteur et l'obstacle: **int Distance = 0**
- Variable correspondant à la vitesse de propagation des ondes ultrasonores dans l'air : **float Vitesse = 0.0**

void setup()

initialisation des entrées et sorties

- le débit de communication en nombre de caractères par seconde pour la communication série est fixé à 9600 bauds: **Serial.begin(9600)**
- La broche du bouton poussoir est initialisée comme une entrée digitale. Des données seront donc envoyées depuis cette broche vers le microcontrôleur: **pinMode (PinButton, INPUT)**
- La broche de l'émetteur US est initialisée comme une sortie digitale. Des données seront donc envoyées depuis le microcontrôleur vers cette broche: **pinMode(TRIGGER_PIN, OUTPUT)** et initialisée à un niveau bas (0 V): **digitalWrite(TRIGGER_PIN, LOW)**
- Si le capteur US dispose de 2 broches différentes pour l'émission et la réception du signal (**if (ECHO_PIN != TRIGGER_PIN**), la broche du récepteur est initialisée comme une entrée digitale: **pinMode(ECHO_PIN, INPUT)**
- Entrée de la distance entre le capteur ultrasonique et l'obstacle dans le moniteur "Série"

void loop()

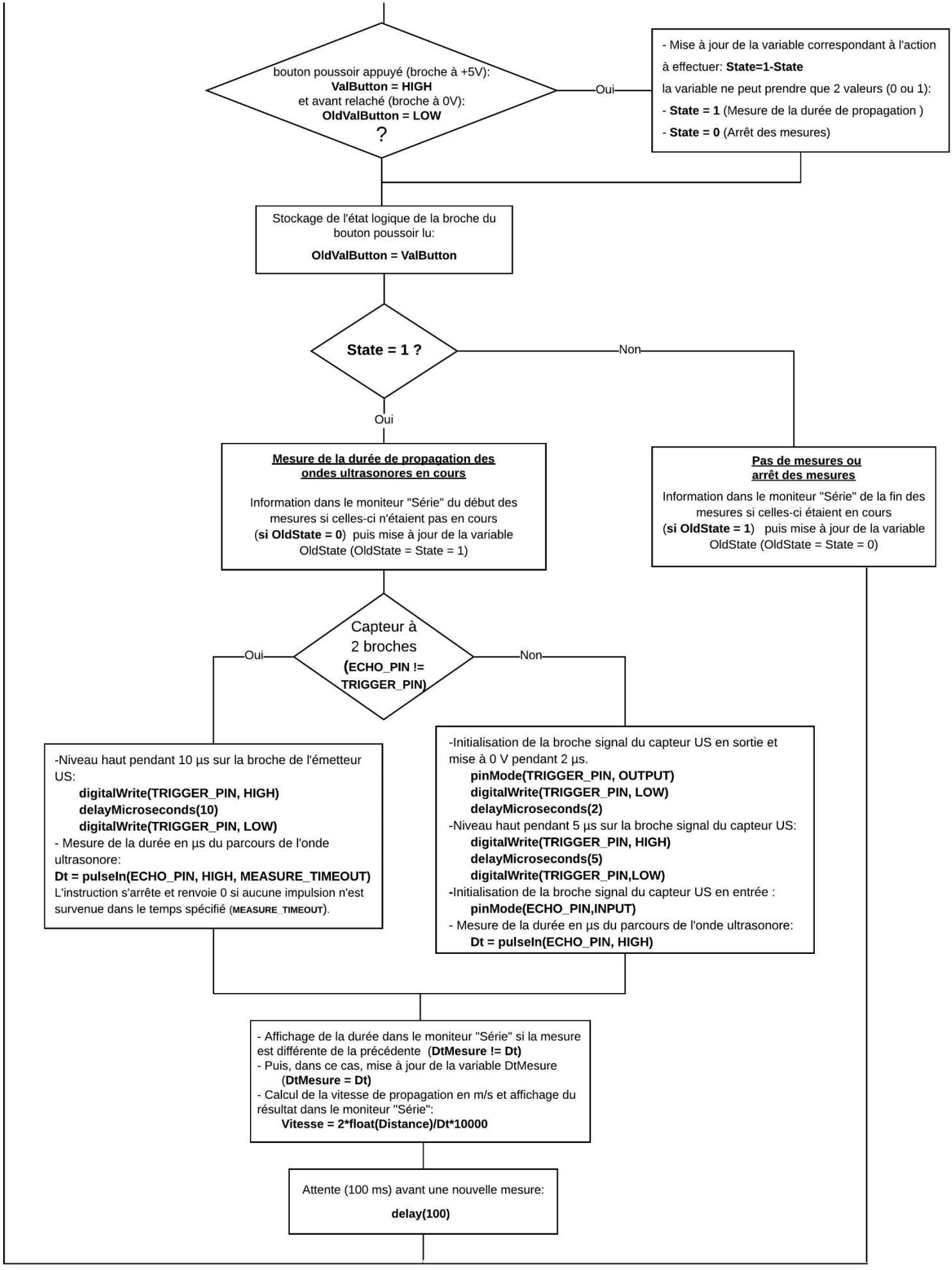
Fonction principale en boucle

Lecture de la valeur de la broche du bouton poussoir:

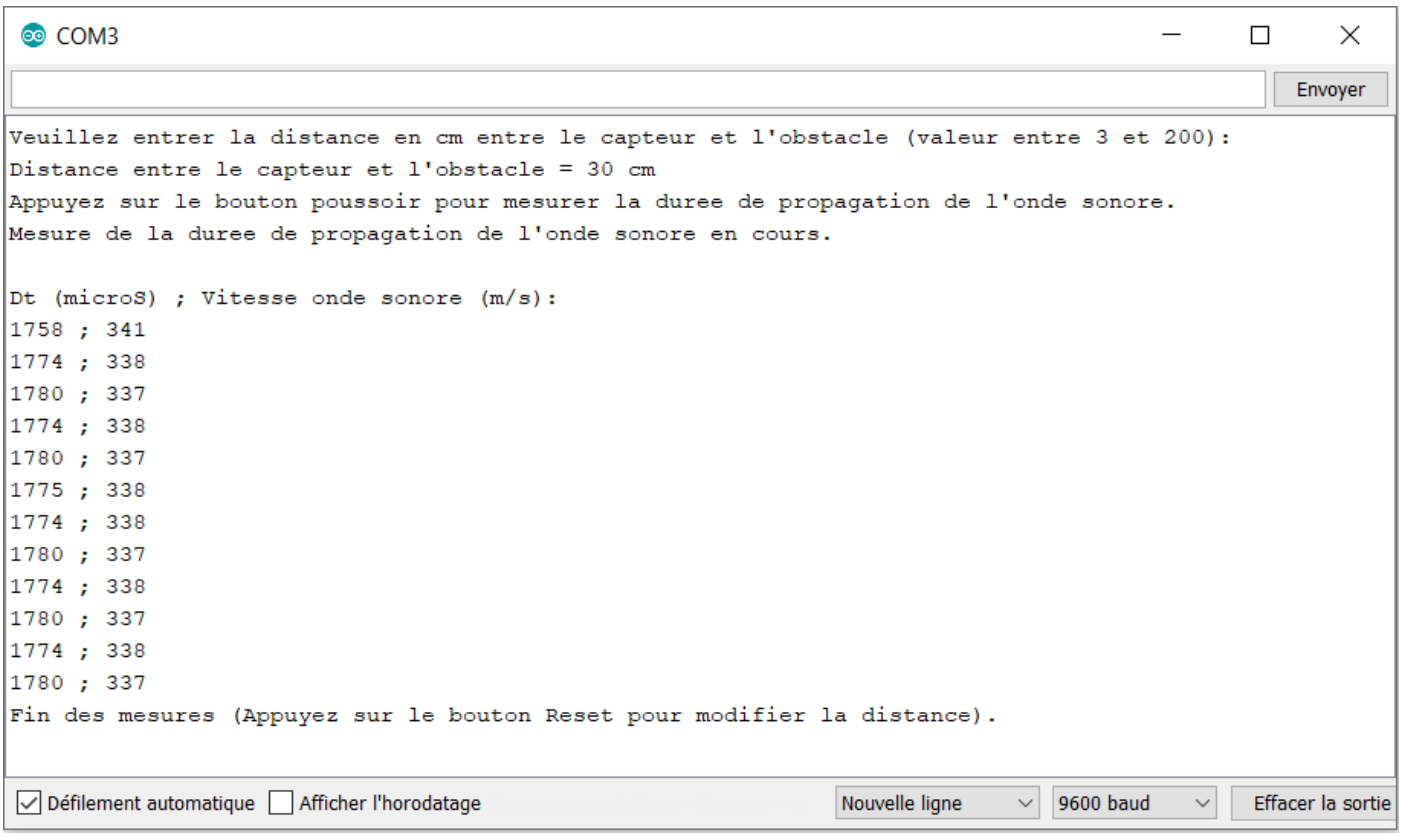
ValButton = digitalRead(PinButton)

Attente (10 ms) pour éviter les perturbations dûes au phénomène de rebond du bouton poussoir:

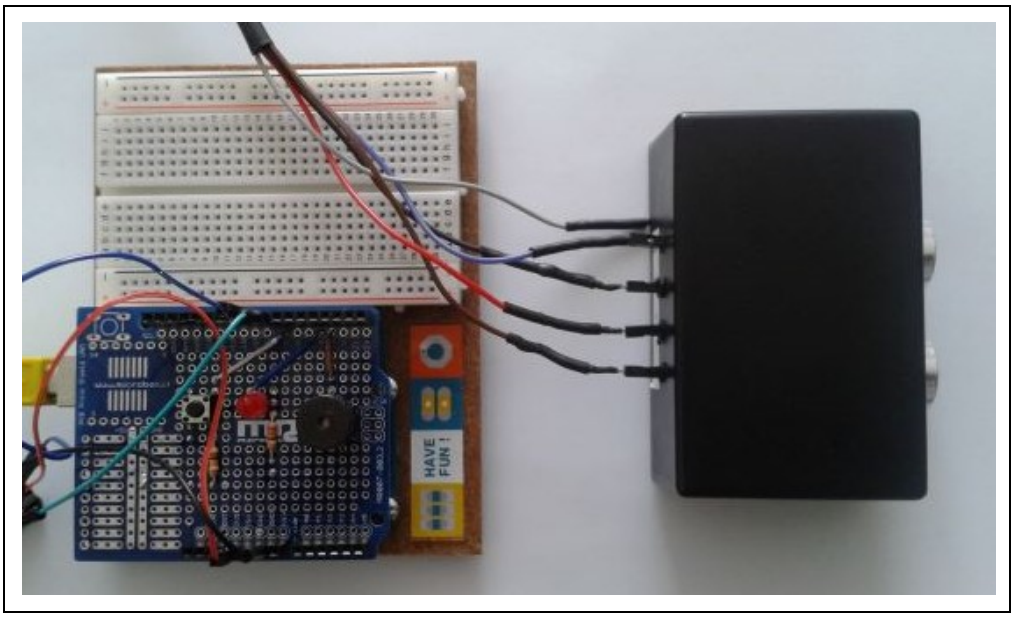
delay(10)


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 durée de propagation)
- 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 durée de propagation des ondes ultrasonores en cours
Information dans le moniteur "Série" du début des mesures si celles-ci n'étaient pas en cours (si OldState = 0) puis mise à jour de la variable OldState (OldState = State = 1)
Pas de mesures ou arrêt des mesures
Information dans le moniteur "Série" de la fin des mesures si celles-ci étaient en cours (si OldState = 1) puis mise à jour de la variable OldState (OldState = State = 0)
Capteur à 2 broches (ECHO_PIN != TRIGGER_PIN)
Oui
Non
-Niveau haut pendant 10 µs sur la broche de l'émetteur US:
digitalWrite(TRIGGER_PIN, HIGH)
delayMicroseconds(10)
digitalWrite(TRIGGER_PIN, LOW)
- Mesure de la durée en µs du parcours de l'onde ultrasonore:
Dt = pulseIn(ECHO_PIN, HIGH, MEASURE_TIMEOUT)
L'instruction s'arrête et renvoie 0 si aucune impulsion n'est survenue dans le temps spécifié (MEASURE_TIMEOUT).
-Initialisation de la broche signal du capteur US en sortie et mise à 0 V pendant 2 µs.
pinMode(TRIGGER_PIN, OUTPUT)
digitalWrite(TRIGGER_PIN, LOW)
delayMicroseconds(2)
-Niveau haut pendant 5 µs sur la broche signal du capteur US:
digitalWrite(TRIGGER_PIN, HIGH)
delayMicroseconds(5)
digitalWrite(TRIGGER_PIN,LOW)
-Initialisation de la broche signal du capteur US en entrée :
pinMode(ECHO_PIN,INPUT)
- Mesure de la durée en µs du parcours de l'onde ultrasonore:
Dt = pulseIn(ECHO_PIN, HIGH)
- Affichage de la durée dans le moniteur "Série" si la mesure est différente de la précédente (DtMesure != Dt)
- Puis, dans ce cas, mise à jour de la variable DtMesure (DtMesure = Dt)
- Calcul de la vitesse de propagation en m/s et affichage du résultat dans le moniteur "Série":
Vitesse = 2*float(Distance)/Dt*10000
Attente (100 ms) avant une nouvelle mesure:
delay(100)

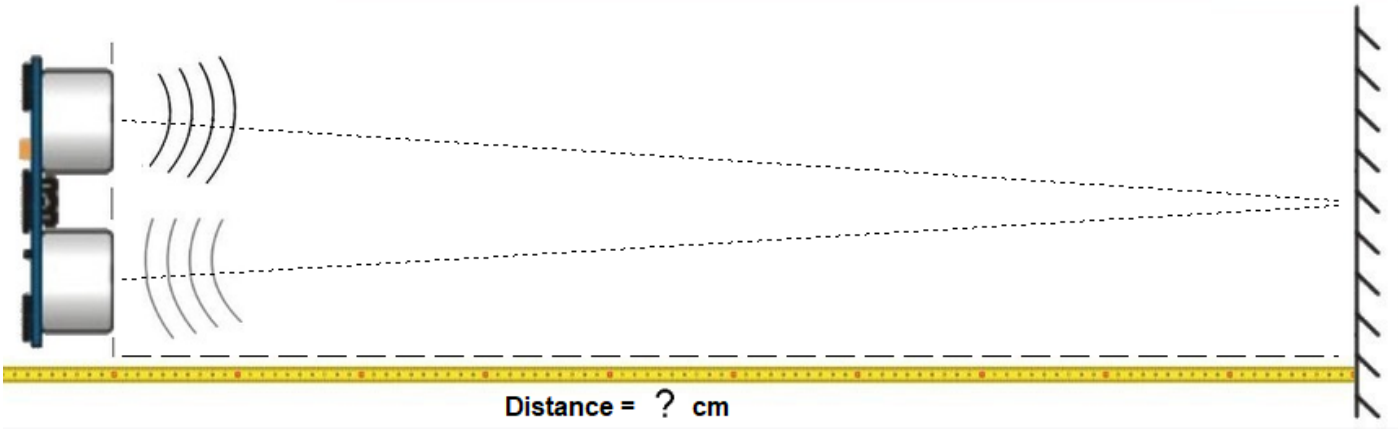
Résultats dans le moniteur série :



- Activité 2 : Mesure de distances



Maintenant que nous avons déterminer la célérité des ondes sonores dans l'air, dans cette activité, nous allons utiliser le capteur à ultrasons pour mesurer des distances, selon le même principe que l'activité précédente, en mesurant la durée de propagation, **Dt**, de l'onde sonore entre l'émetteur et le récepteur situés à une distance, **d**, inconnue d'un obstacle. C'est le principe du Sonar.



Dans les codes en langage Arduino ou en Python, la température de l'air est une variable qu'il est possible de modifier. La vitesse théorique du son dans l'air en fonction de la température indiquée est alors calculée et utilisée pour déterminer la distance entre le capteur et l'obstacle.

Comme pour l'activité précédente, le programme ci-dessous est adapté pour un capteur ultrasonique à deux broches différentes pour l'émission et la réception des ultrasons.

Dans le cas d'un capteur ultrasonique à une broche commune pour l'émission et la réception des ultrasons, il suffit de déclarer la même broche pour "trigger_pin" (émetteur US) et "echo_pin" (récepteur US).

. Programme en Python (Projet4\Activity2\PY\Activity2.py)

```
# Importations des librairies et définition des fonctions

from PymataExpressDef import *
from ConnectToArduino import *
import time
from math import *

async def Get_Dt(data):
    global Dt
    Dt = data[1]*56

def Set_Sonar_Pins(board, trigger_pin, echo_pin):
    loop.run_until_complete(board.set_pin_mode_sonar(trigger_pin, echo_pin, Get_Dt))

def Sonar_Read(board, trigger_pin):
    loop.run_until_complete(board.sonar_read(trigger_pin))


# Déclaration des constantes et variables

TRIGGER_PIN = 8
ECHO_PIN = 9
PinButton = 7

Distance = 0
DistanceMesure = 0
Dt = 0
Temp = 20
VitTheoUS = 20.05*(sqrt(Temp+273.15))

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


# Connexion à l'Arduino

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)


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


Set_Sonar_Pins(board, TRIGGER_PIN, ECHO_PIN)
Set_DigitalInput_Pin(board, PinButton)

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

```
# Boucle principale du programme

print("Appuyez sur le bouton poussoir pour mesurer la distance entre le capteur et l'obstacle.\n")

while True:
    try:
        ValButton = Digital_Read(board, PinButton)
        time.sleep(0.01)
        if ValButton == 1 and OldValButton==0:
            State = 1 - State
        OldValButton = ValButton
        if State == 1:
            if OldState == 0:
                print("Mesure de la distance en cours.\n")
                print("Distance (cm):")
                OldState=1

            Sonar_Read(board,TRIGGER_PIN)
            Distance = int(Dt/2 * VitTheoUS/10000)
            if DistanceMesure != Distance:
                print(Distance)
                DistanceMesure = Distance
            time.sleep(.01)
        else:
            if OldState == 1:
                print("\nFin des mesures.\n")
                OldState = 0
    except KeyboardInterrupt:
        Arduino_Exit(board)
        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 Express”**,

. Le module "**PymataExpressDef.Py**" regroupant toutes les fonctions utiles à l’utilisation de " **Pymata-express** " (fonction de déclaration des entrées et sorties, de lectures, d’écritures…) ,

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

. La bibliothèque **"math"** pour le calcul de la vitesse théorique du son dans l’air,

. La fonction **”Get_Dt”** pour obtenir la durée de propagation dans l’air de l’onde sonore,

. La fonction **”Set_Sonar_Pins”** pour déclarer les broches du capteur ultrasonique,

. La fonction **”Sonar_Read”** qui appelle la fonction **”Get_Dt”.**

- Déclaration des constantes et variables :

. **TRIGGER_PIN = 8** (cst correspondant au n° de la broche sur laquelle l'émetteur US est connecté)

. **ECHO_PIN = 9** (cst correspondant au n° de la broche sur laquelle le récepteur US est connecté)

. **PinButton = 7** (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)

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

. **Distance = 0** (variable correspondant à la distance en cm entre le capteur et l'obstacle)

. **DistanceMesure = 0** (variable correspondant à la distance en cm entre le capteur et l'obstacle mesurée précédemment)

. **Dt = 0** (variable correspondant à la durée de propagation des ondes ultrasonores en µs entre le capteur et l'obstacle)

. **Temp = 20** (variable correspondant à la température de l'air en °C)

. **VitTheoUS = 20.05*(sqrt(Temp+273.15))** (constante correspondant à la vitesse théorique du son dans l'air en m/s)

. **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 :

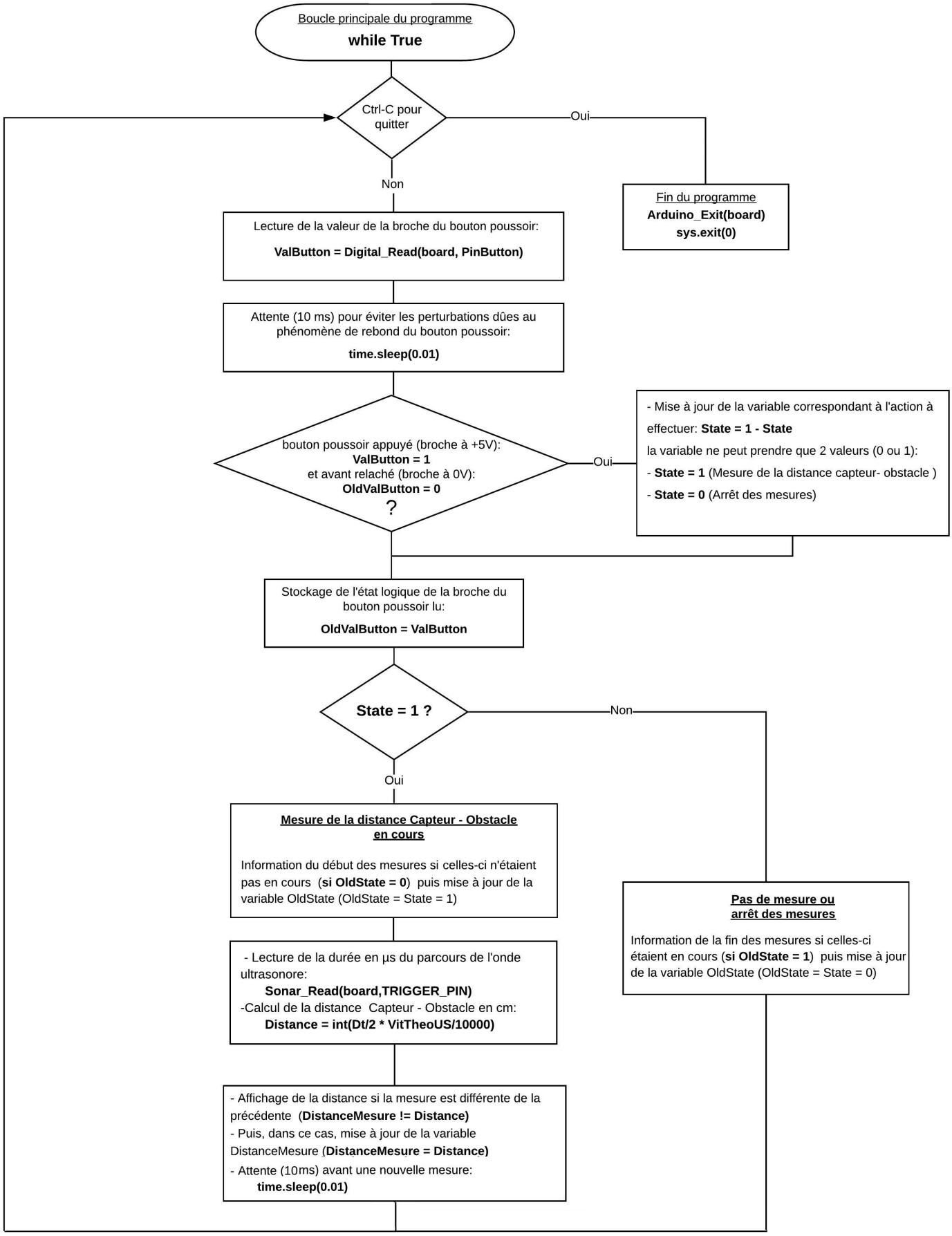
- Déclaration des broches du capteur ultrasonique :

Set_Sonar_Pins(board, TRIGGER_PIN, ECHO_PIN)

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

Set_DigitalInput_Pin(board, PinButton)

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



Résultats dans la fenêtre Python Shell :

```
Pymata Express Version 1.4
Copyright (c) 2018-2019 Alan Yorinks All rights reserved.

Opening COM3 ...
Waiting 4 seconds for the Arduino To Reset.

Arduino found and connected to COM3

Retrieving Arduino Firmware ID...
Arduino Firmware ID: 2.5 FirmataExpress.ino
Auto-discovery complete. Found 20 Digital Pins and 6 Analog Pins


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

Appuyez sur le bouton poussoir pour mesurer la distance entre le capteur et l'obstacle.

Mesure de la distance en cours.

Distance (cm):
29
30

Fin des mesures.
```

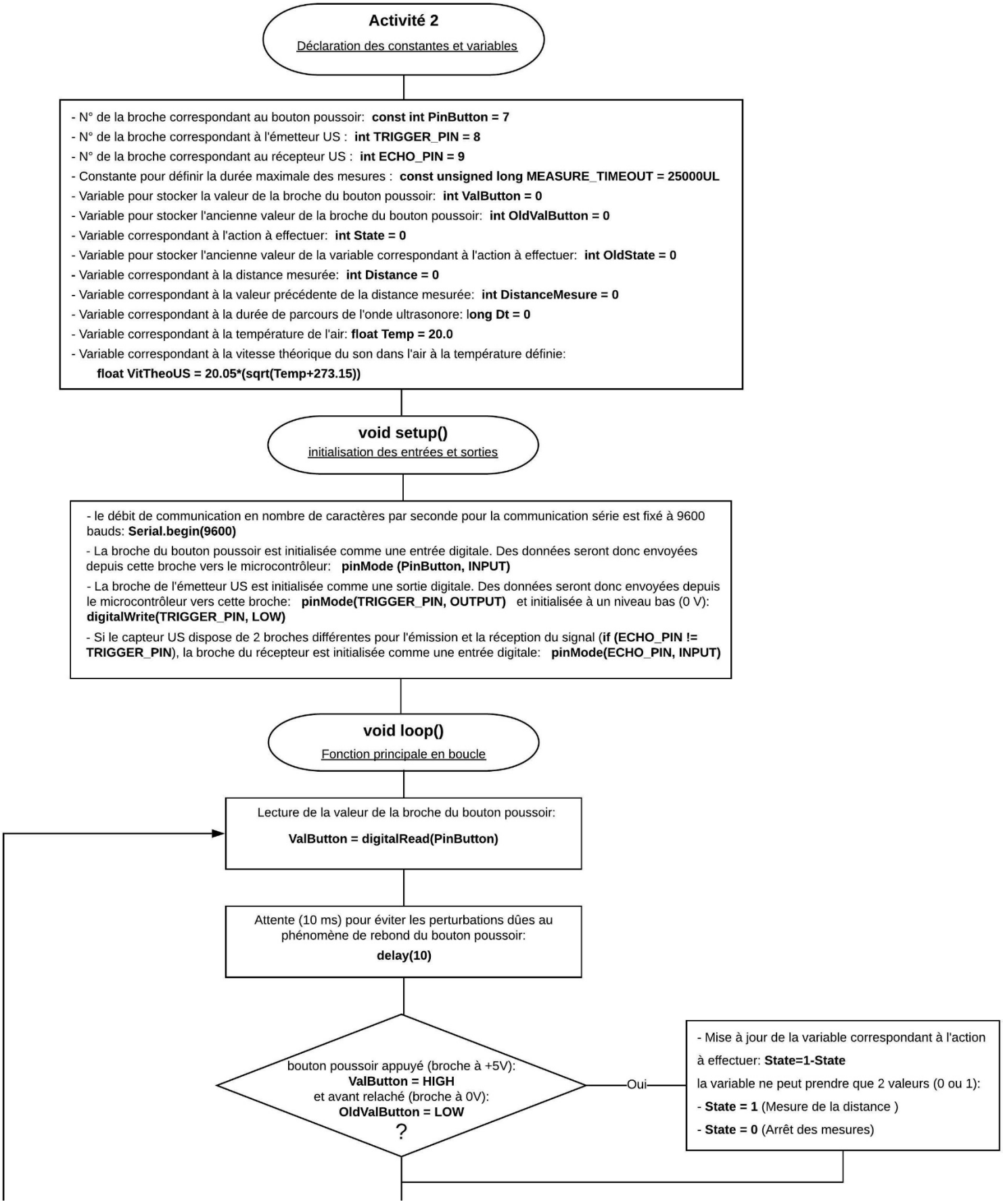
. Programme en langage Arduino (Projet4\Activity2\INO\Activity2.ino)

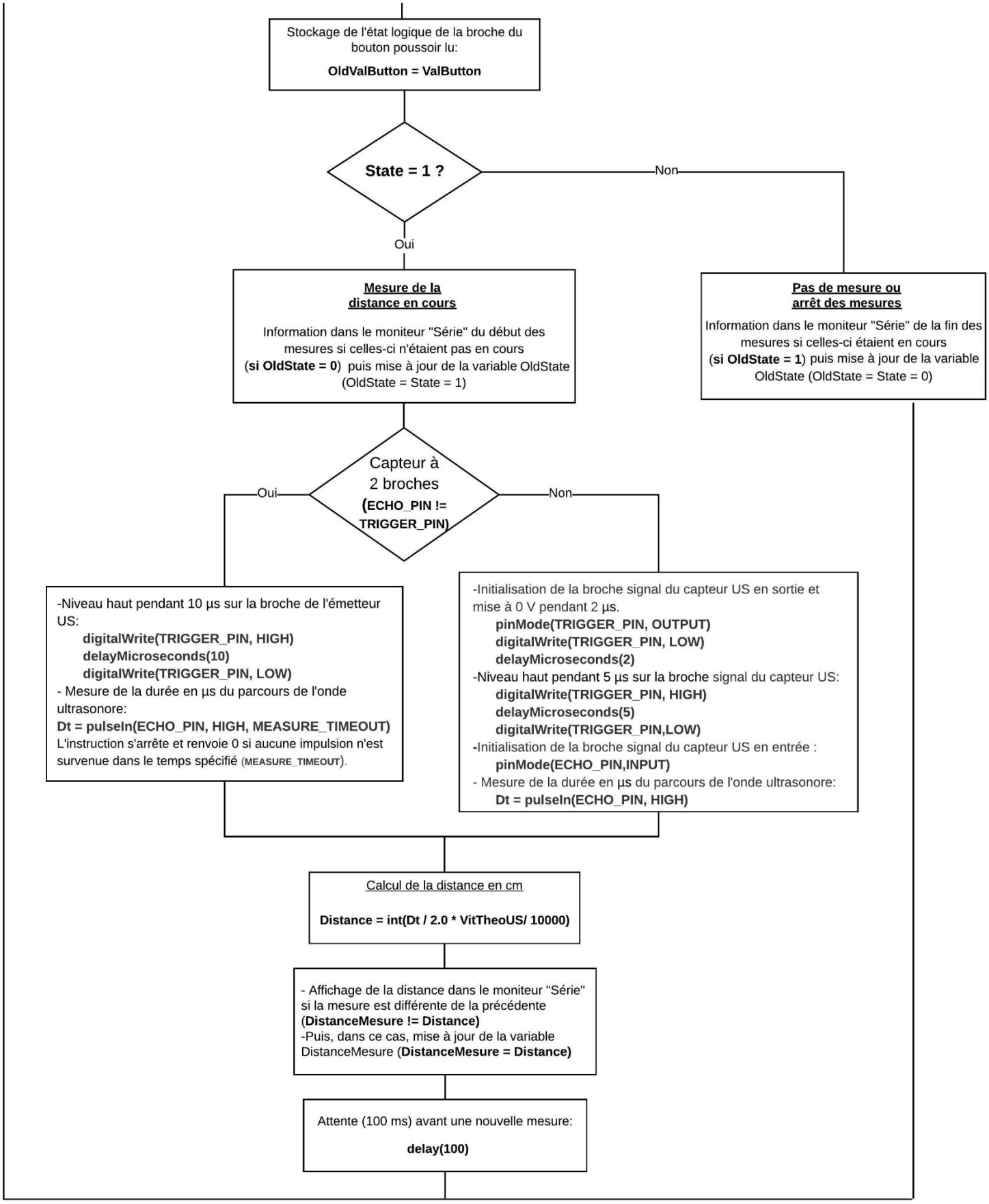
Activity2

```
// Déclaration des constantes et variables

int TRIGGER_PIN = 8;
int ECHO_PIN = 9;
const int PinButton = 7;

const unsigned long MEASURE_TIMEOUT = 25000UL;

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

int Distance = 0;
int DistanceMesure = 0;
long Dt = 0;

float Temp = 20.0;
float VitTheoUS = 20.05*(sqrt(Temp+273.15));


// Initialisation des entrées et sorties

void setup()
{
Serial.begin(9600);
pinMode(PinButton,INPUT);
pinMode(TRIGGER_PIN, OUTPUT);
digitalWrite(TRIGGER_PIN, LOW);
if (ECHO_PIN != TRIGGER_PIN){
pinMode(ECHO_PIN, INPUT);}
Serial.println("Appuyez sur le bouton poussoir pour mesurer la distance entre le capteur et l'obstacle.");
}


// 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 distance en cours.");
   Serial.println("");
   Serial.println ("Distance (cm) :");
   OldState=1;
  }
```

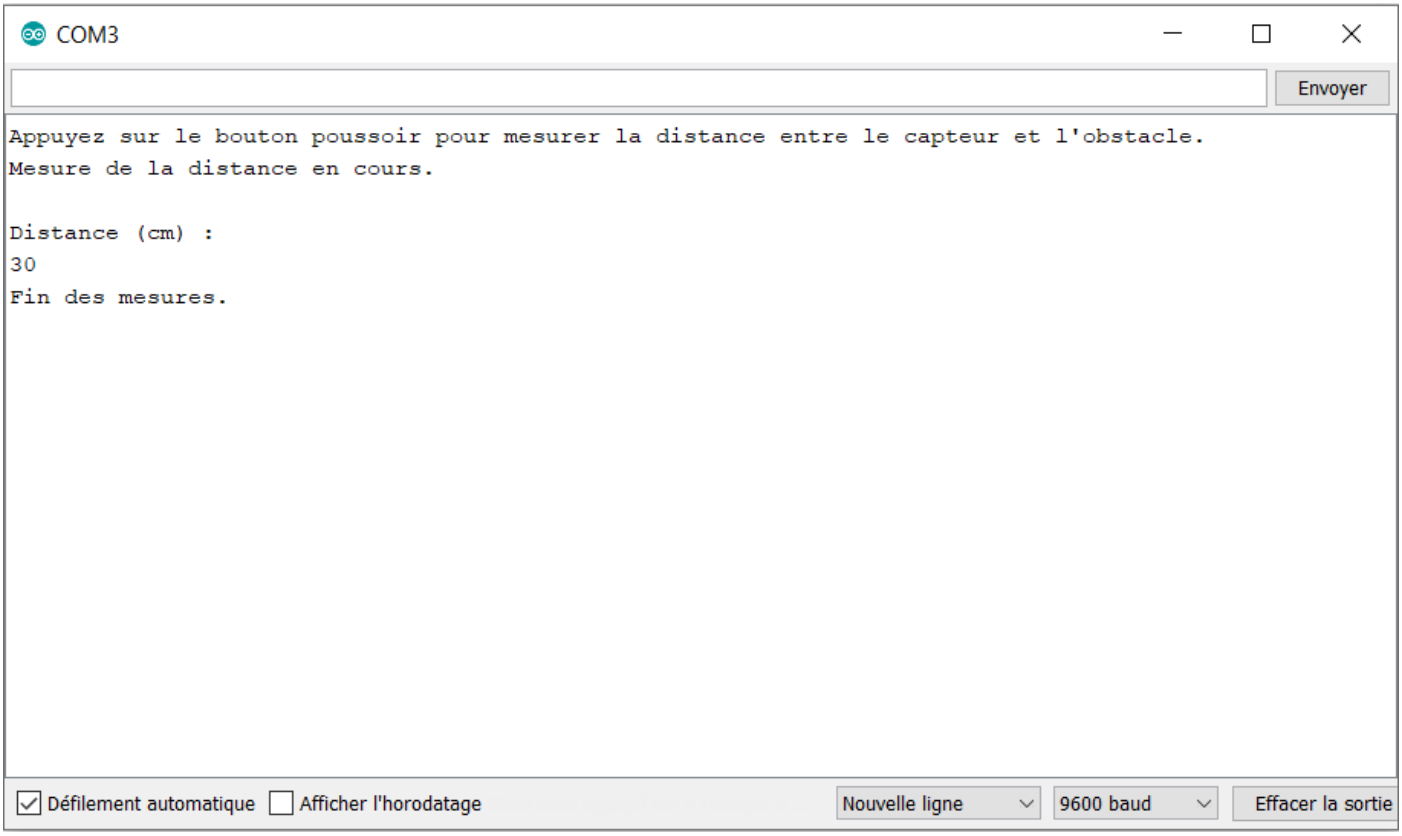
```
  if (ECHO_PIN != TRIGGER_PIN){
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN, LOW);
  Dt = pulseIn(ECHO_PIN, HIGH, MEASURE_TIMEOUT);}

  else {
  pinMode(TRIGGER_PIN, OUTPUT);
  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(5);
  digitalWrite(TRIGGER_PIN,LOW);
  pinMode(ECHO_PIN,INPUT);
  Dt = pulseIn(ECHO_PIN,HIGH);}

  Distance = int(Dt / 2.0 * VitTheoUS/ 10000);

  if (DistanceMesure != Distance)
  {
  Serial.println(Distance);
  DistanceMesure = Distance;
  }
  Serial.flush();
  delay(100);
}
else
{
  if (OldState == 1){
  Serial.println("Fin des mesures.");
  OldState = 0;}
}
}
```

Déroulement du programme :

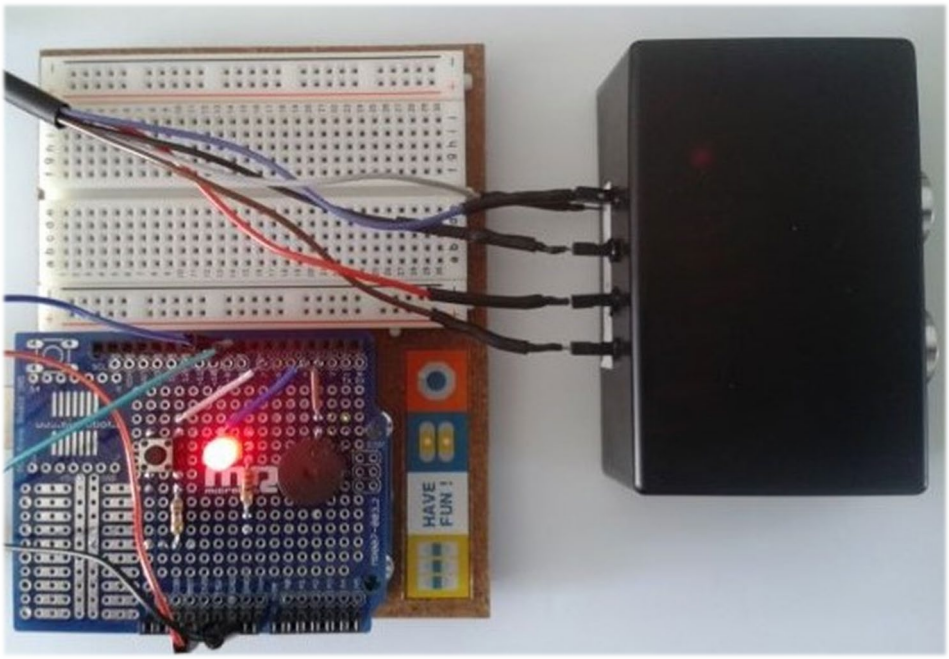



Stockage de l'état logique de la broche du bouton poussoir lu:
OldValButton = ValButton
State = 1 ?
Non
Oui
Mesure de la distance en cours
Information dans le moniteur "Série" du début des mesures si celles-ci n'étaient pas en cours (si OldState = 0) puis mise à jour de la variable OldState (OldState = State = 1)
Pas de mesure ou arrêt des mesures
Information dans le moniteur "Série" de la fin des mesures si celles-ci étaient en cours (si OldState = 1) puis mise à jour de la variable OldState (OldState = State = 0)
Capteur à 2 broches (ECHO_PIN != TRIGGER_PIN)
Oui
Non
-Niveau haut pendant 10 µs sur la broche de l'émetteur US:
digitalWrite(TRIGGER_PIN, HIGH)
delayMicroseconds(10)
digitalWrite(TRIGGER_PIN, LOW)
- Mesure de la durée en µs du parcours de l'onde ultrasonore:
Dt = pulseIn(ECHO_PIN, HIGH, MEASURE_TIMEOUT)
L'instruction s'arrête et renvoie 0 si aucune impulsion n'est survenue dans le temps spécifié (MEASURE_TIMEOUT).
-Initialisation de la broche signal du capteur US en sortie et mise à 0 V pendant 2 µs.
pinMode(TRIGGER_PIN, OUTPUT)
digitalWrite(TRIGGER_PIN, LOW)
delayMicroseconds(2)
-Niveau haut pendant 5 µs sur la broche signal du capteur US:
digitalWrite(TRIGGER_PIN, HIGH)
delayMicroseconds(5)
digitalWrite(TRIGGER_PIN,LOW)
-Initialisation de la broche signal du capteur US en entrée :
pinMode(ECHO_PIN,INPUT)
- Mesure de la durée en µs du parcours de l'onde ultrasonore:
Dt = pulseIn(ECHO_PIN, HIGH)
Calcul de la distance en cm
Distance = int(Dt / 2.0 * VitTheoUS/ 10000)
- Affichage de la distance dans le moniteur "Série" si la mesure est différente de la précédente (DistanceMesure != Distance)
-Puis, dans ce cas, mise à jour de la variable DistanceMesure (DistanceMesure = Distance)
Attente (100 ms) avant une nouvelle mesure:
delay(100)

Résultats dans le moniteur série :



- Activité 3 : Détecteur d'obstacles



Comme il est possible de mesurer une distance avec un Arduino et un capteur à ultrasons, nous allons dans cette activité, réaliser un détecteur d'obstacles qui déclenchera une alarme lumineuse et sonore quand le capteur est situé en dessous d'une distance, **d**, fixée.

Dans les codes en langage Arduino ou en Python, la température de l'air est une variable qu'il est possible de modifier. La vitesse théorique du son dans l'air en fonction de la température indiquée est alors calculée et utilisée pour déterminée la distance entre le capteur et l'obstacle.

On pourra également modifier la distance minimale de déclenchement de l'alarme.

Comme pour l'activité précédente, le programme ci-dessous est adapté pour un capteur ultrasonique à deux broches différentes pour l'émission et la réception des ultrasons.

Dans le cas d'un capteur ultrasonique à une broche commune pour l'émission et la réception des ultrasons, il suffit de déclarer la même broche pour "trigger_pin" (émetteur US) et "echo_pin" (récepteur US).

. Programme en Python (**Projet4\Activity3\PY\Activity3.py**)

```
# Importations des librairies et définition des fonctions

from PymataExpressDef import *
from ConnectToArduino import *
import time
from math import *

async def Get_Dt(data):
    global Dt
    Dt = data[1]*56

def Set_Sonar_Pins(board, trigger_pin, echo_pin):
    loop.run_until_complete(board.set_pin_mode_sonar(trigger_pin, echo_pin, Get_Dt))

def Sonar_Read(board, trigger_pin):
    loop.run_until_complete(board.sonar_read(trigger_pin))


# Déclaration des constantes et variables

TRIGGER_PIN = 8
ECHO_PIN = 9
PinButton = 7
PinTone = 3
PinLed = 4

Distance=0
DistanceMesure = 0
DistanceAlarme = 10
Dt=0
Temp = 20
VitTheoUS = 20.05*(sqrt(Temp+273.15))

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


# Connexion à l'Arduino

PortComArduino = SelectPortCOM()
board = OpenPortCom(PortComArduino)


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

Set_Sonar_Pins(board, TRIGGER_PIN, ECHO_PIN)
Set_DigitalInput_Pin(board, PinButton)
Set_Tone_Pin(board, PinTone)
Set_DigitalOutput_Pin(board, PinLed)

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


# Boucle principale du programme

print("Appuyez sur le bouton poussoir pour mesurer la distance entre le capteur et l'obstacle.\n")

while True:
    try:
        ValButton = Digital_Read(board, PinButton)
        time.sleep(0.01)
        if ValButton == 1 and OldValButton==0:
            State = 1 - State
        OldValButton = ValButton
```

```
        if State==1:
            if OldState == 0:
                print("Mesure de la distance en cours.\n")
                print("Distance (cm):")
                OldState=1
            Sonar_Read(board,TRIGGER_PIN)
            Distance= int(Dt/2 * VitTheoUS/10000)
            if DistanceMesure != Distance:
                print(Distance)
                DistanceMesure = Distance
            if Distance < DistanceAlarme:
                Tone(board, PinTone, 440, 0)
                Digital_Write(board, PinLed, 1)
                time.sleep(0.1)
                No_Tone(board, PinTone)
                Digital_Write(board, PinLed, 0)
                time.sleep(0.1)
            else:
                No_Tone(board, PinTone)
                Digital_Write(board, PinLed, 0)
            time.sleep(.01)
        else:
            if OldState == 1:
                print("\nFin des mesures.\n")
                OldState = 0
            No_Tone(board, PinTone)
            Digital_Write(board, PinLed, 0)

    except KeyboardInterrupt:
        No_Tone(board, PinTone)
        Digital_Write(board, PinLed, 0)
        Arduino_Exit(board)
        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 Express”**,

. Le module "**PymataExpressDef.Py**" regroupant toutes les fonctions utiles à l’utilisation de " **Pymata-express** " (fonction de déclaration des entrées et sorties, de lectures, d’écritures…) ,

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

. La bibliothèque **"math"** pour le calcul de la vitesse théorique du son dans l’air,

. La fonction **”Get_Dt”** pour obtenir la durée de propagation dans l’air de l’onde sonore,

. La fonction **”Set_Sonar_Pins”** pour déclarer les broches du capteur ultrasonique,

. La fonction **”Sonar_Read”** qui appelle la fonction **”Get_Dt”.**

- Déclaration des constantes et variables :

. **TRIGGER_PIN = 8** (cst correspondant au n° de la broche sur laquelle l’émetteur US est connecté)

. **ECHO_PIN = 9** (cst correspondant au n° de la broche sur laquelle le récepteur US est connecté)

. **PinButton = 7** (cst correspondant au n° de la broche sur laquelle le bouton poussoir est connecté)

. **PinTone = 3** (cst correspondant au n° de la broche sur laquelle le buzzer est connecté)

. **PinLed = 4** (cst correspondant au n° de la broche sur laquelle la DEL est connectée)

. **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**)

. **Distance = 0** (variable correspondant à la distance en cm entre le capteur et l'obstacle)

. **DistanceMesure = 0** (variable correspondant à la distance en cm entre le capteur et l'obstacle mesurée précédemment)

. **DistanceAlarme = 10** (constante correspondant à la distance en cm en dessous de laquelle l'alarme est déclenchée)

. **Dt = 0** (variable correspondant à la durée de propagation des ondes ultrasonores en µs entre le capteur et l'obstacle)

. **Temp = 20** (variable correspondant à la température de l'air en °C)

. **VitTheoUS = 20.05*(sqrt(Temp+273.15))** (constante correspondant à la vitesse théorique du son dans l'air en m/s)

. **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 des broches du capteur ultrasonique :

Set_Sonar_Pins(board, TRIGGER_PIN, ECHO_PIN)

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

Set_DigitalInput_Pin(board, PinButton)

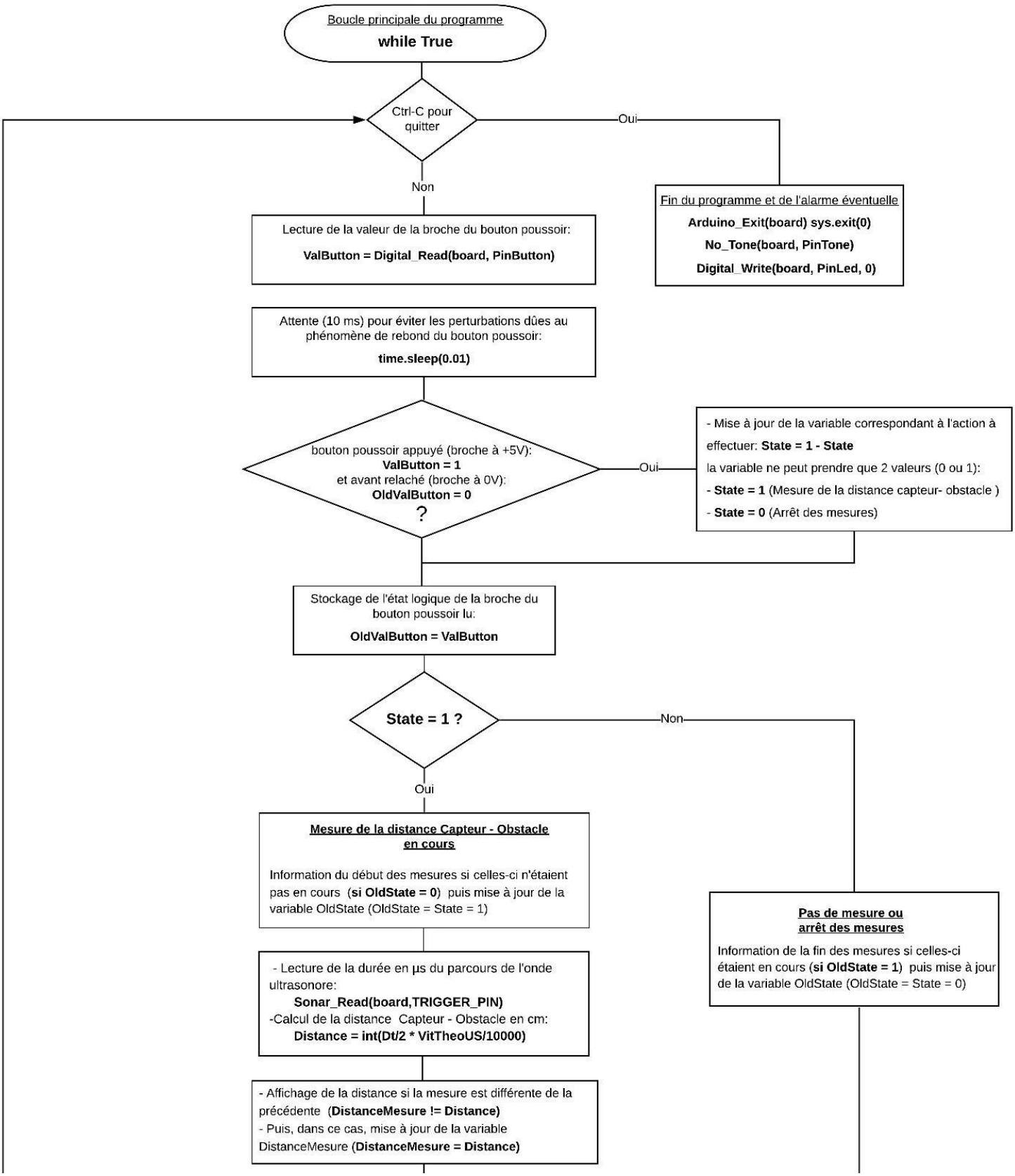
- Déclaration de la broche du buzzer en mode "Tone" :

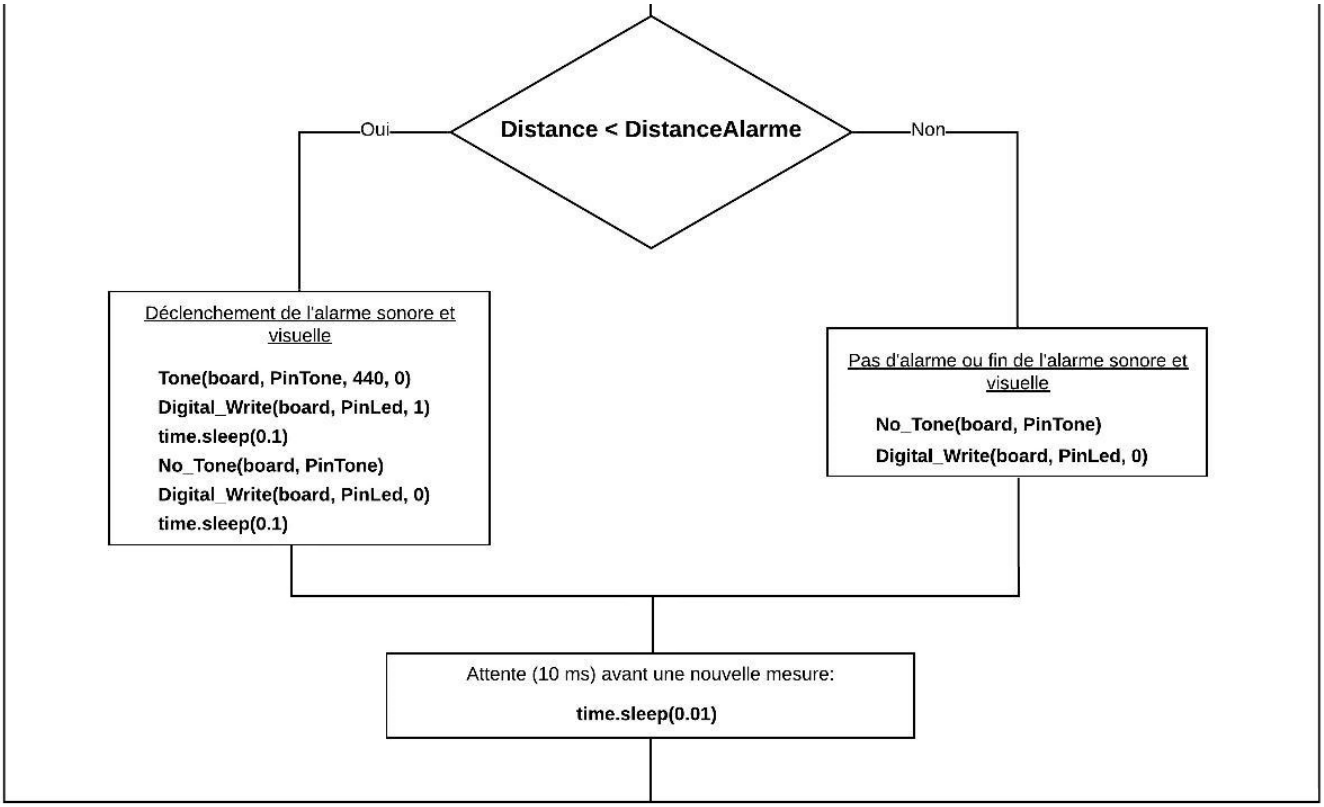
Set_Tone_Pin(board, PinTone)

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

Set_DigitalOutput_Pin(board, PinLED)

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





Résultats dans la fenêtre Python Shell :

```
Connexion à l'arduino établie - Appuyez sur Ctrl-C pour quitter.

Appuyez sur le bouton poussoir pour mesurer la distance entre le capteur et l'ob
stacle.

Mesure de la distance en cours.

Distance (cm):
14
15
14
13
12
11
10
9
7


Fin des mesures.
```

. Programme en langage Arduino (Projet4\Activity3\INO\Activity3.ino)

Activity3

```
// Déclaration des constantes et variables

const int PinButton = 7;
const int PinTone = 3;
const int PinLed = 4;

const unsigned long MEASURE_TIMEOUT = 25000UL;

int TRIGGER_PIN = 8;
int ECHO_PIN = 8;

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

int Distance = 0;
int DistanceMesure = 0;
int DistanceAlarme = 10;
long Dt = 0.0;

float Temp = 20.0;
float VitTheoUS = 20.05*(sqrt(Temp+273.15));


// Initialisation des entrées et sorties

void setup()
{
Serial.begin(9600);

pinMode(PinButton,INPUT);
pinMode(PinLed,OUTPUT);
pinMode(TRIGGER_PIN, OUTPUT);
digitalWrite(TRIGGER_PIN, LOW);

if (ECHO_PIN != TRIGGER_PIN){
pinMode(ECHO_PIN, INPUT);}

Serial.println("Appuyez sur le bouton poussoir pour mesurer la distance entre le capteur et l'obstacle.");
}


// 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 distance en cours.");
   Serial.println("");
   Serial.println ("Distance (cm) :");
   OldState=1;
  }

  if (ECHO_PIN != TRIGGER_PIN){
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN, LOW);
  Dt = pulseIn(ECHO_PIN, HIGH, MEASURE_TIMEOUT);}
  else {
  pinMode(TRIGGER_PIN, OUTPUT);
  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(5);
  digitalWrite(TRIGGER_PIN,LOW);
  pinMode(TRIGGER_PIN,INPUT);
  Dt = pulseIn(TRIGGER_PIN,HIGH);}

  Distance = int(Dt / 2.0 * VitTheoUS/ 10000);

  if (DistanceMesure != Distance)
  {
  Serial.println(Distance);
  DistanceMesure = Distance;
  }
  if (Distance < DistanceAlarme)
   {
      digitalWrite(PinLed, HIGH);
      tone(PinTone,440);
      delay(100);
      digitalWrite(PinLed, LOW);
      noTone(PinTone);
      delay(100);
   }
   else
   {
      digitalWrite(PinLed, LOW);
      noTone(PinTone);
   }
  Serial.flush();
  delay(100);
}
else
{
  if (OldState == 1){
  Serial.println("Fin des mesures.");
  OldState = 0;}
}
}
```

Déroulement du programme :

Activité 3

Déclaration des constantes et variables

- N° de la broche correspondant au bouton poussoir: **const int PinButton = 7**
- N° de la broche correspondant au buzzer: **const int PinTone = 3**
- N° de la broche correspondant à la DEL: **const int PinLed = 4**
- N° de la broche correspondant à l'émetteur US : **int TRIGGER_PIN = 8**
- N° de la broche correspondant au récepteur US : **int ECHO_PIN = 9**
- Constante pour définir la durée maximale des mesures : **const unsigned long MEASURE_TIMEOUT = 25000UL**
- Variable pour stocker la valeur de la broche du bouton poussoir: **int ValButton = 0**
- Variable pour stocker l'ancienne valeur de la broche du bouton poussoir: **int OldValButton = 0**
- Variable correspondant à l'action à effectuer: **int State = 0**
- Variable pour stocker l'ancienne valeur de la variable correspondant à l'action à effectuer: **int OldState = 0**
- Variable correspondant à la distance mesurée: **int Distance = 0**
- Variable correspondant à la valeur précédente de la distance mesurée: **int DistanceMesure = 0**
- Variable correspondant à la distance minimale pour le déclenchement de l'alarme: **int DistanceAlarme = 0**
- Variable correspondant à la durée de parcours de l'onde ultrasonore: **long Dt = 0**
- Variable correspondant à la température de l'air: **float Temp = 20.0**
- Variable correspondant à la vitesse théorique du son dans l'air à la température définie:
  float VitTheoUS = 20.05*(sqrt(Temp+273.15))

void setup()

initialisation des entrées et sorties

- le débit de communication en nombre de caractères par seconde pour la communication série est fixé à 9600 bauds: **Serial.begin(9600)**
- La broche du bouton poussoir est initialisée comme une entrée digitale. Des données seront donc envoyées depuis cette broche vers le microcontrôleur: **pinMode (PinButton, INPUT)**
- La broche de l'émetteur US est initialisée comme une sortie digitale. Des données seront donc envoyées depuis le microcontrôleur vers cette broche: **pinMode(TRIGGER_PIN, OUTPUT)** et initialisée à un niveau bas (0 V): **digitalWrite(TRIGGER_PIN, LOW)**
- Si le capteur US dispose de 2 broches différentes pour l'émission et la réception du signal (**if (ECHO_PIN != TRIGGER_PIN**), la broche du récepteur est initialisée comme une entrée digitale: **pinMode(ECHO_PIN, INPUT)**

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

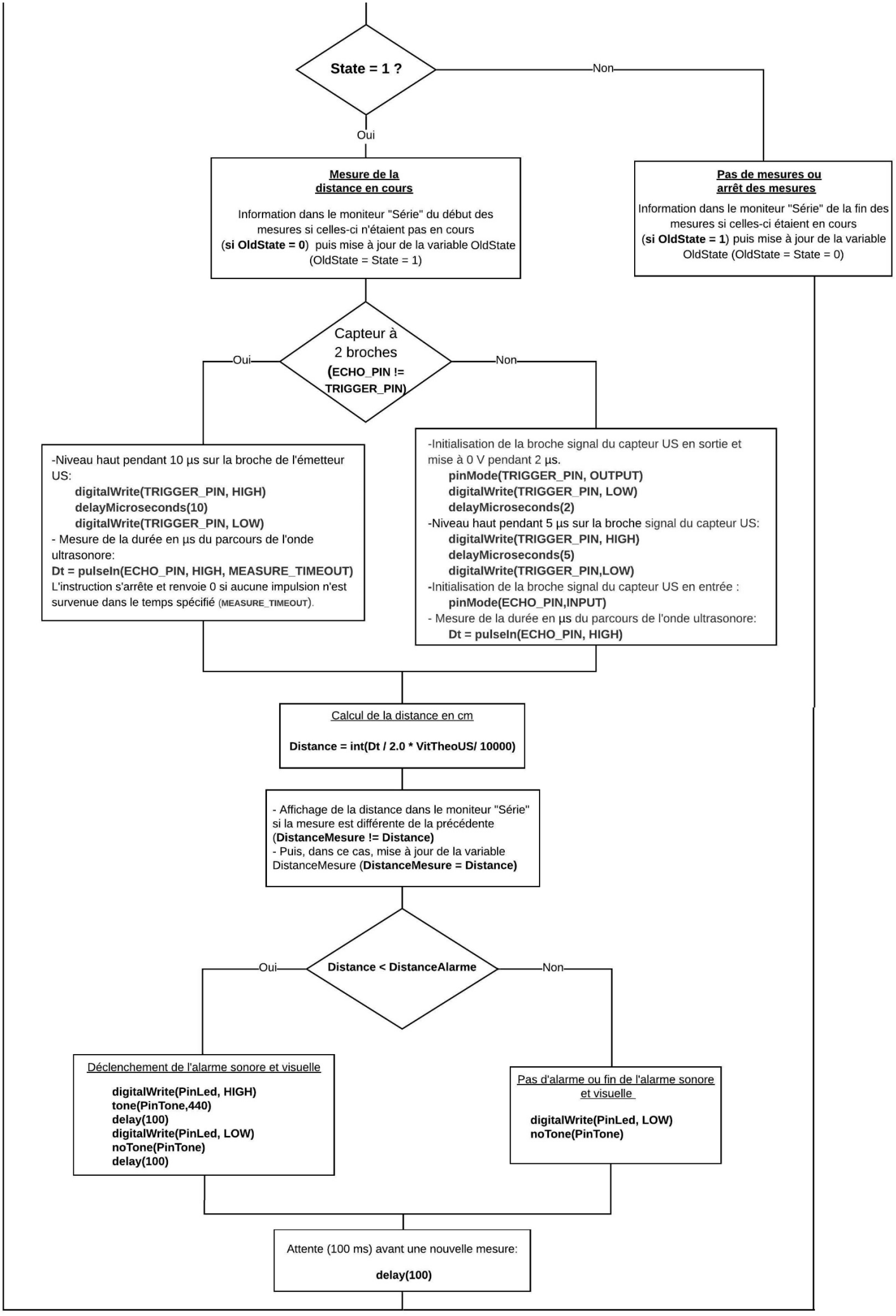
- Mise à jour de la variable correspondant à l'action à effectuer: **State=1-State**

la variable ne peut prendre que 2 valeurs (0 ou 1):

- **State = 1** (Mesure de la distance)
- **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 distance en cours
Information dans le moniteur "Série" du début des mesures si celles-ci n'étaient pas en cours (si OldState = 0) puis mise à jour de la variable OldState (OldState = State = 1)
Pas de mesures ou arrêt des mesures
Information dans le moniteur "Série" de la fin des mesures si celles-ci étaient en cours (si OldState = 1) puis mise à jour de la variable OldState (OldState = State = 0)
Capteur à 2 broches (ECHO_PIN != TRIGGER_PIN)
Oui
Non
-Niveau haut pendant 10 µs sur la broche de l'émetteur US:
digitalWrite(TRIGGER_PIN, HIGH)
delayMicroseconds(10)
digitalWrite(TRIGGER_PIN, LOW)
- Mesure de la durée en µs du parcours de l'onde ultrasonore:
Dt = pulseIn(ECHO_PIN, HIGH, MEASURE_TIMEOUT)
L'instruction s'arrête et renvoie 0 si aucune impulsion n'est survenue dans le temps spécifié (MEASURE_TIMEOUT).
-Initialisation de la broche signal du capteur US en sortie et mise à 0 V pendant 2 µs.
pinMode(TRIGGER_PIN, OUTPUT)
digitalWrite(TRIGGER_PIN, LOW)
delayMicroseconds(2)
-Niveau haut pendant 5 µs sur la broche signal du capteur US:
digitalWrite(TRIGGER_PIN, HIGH)
delayMicroseconds(5)
digitalWrite(TRIGGER_PIN,LOW)
-Initialisation de la broche signal du capteur US en entrée :
pinMode(ECHO_PIN,INPUT)
- Mesure de la durée en µs du parcours de l'onde ultrasonore:
Dt = pulseIn(ECHO_PIN, HIGH)
Calcul de la distance en cm
Distance = int(Dt / 2.0 * VitTheoUS/ 10000)
- Affichage de la distance dans le moniteur "Série" si la mesure est différente de la précédente (DistanceMesure != Distance)
- Puis, dans ce cas, mise à jour de la variable DistanceMesure (DistanceMesure = Distance)
Distance < DistanceAlarme
Oui
Non
Déclenchement de l'alarme sonore et visuelle
digitalWrite(PinLed, HIGH)
tone(PinTone,440)
delay(100)
digitalWrite(PinLed, LOW)
noTone(PinTone)
delay(100)
Pas d'alarme ou fin de l'alarme sonore et visuelle
digitalWrite(PinLed, LOW)
noTone(PinTone)
Attente (100 ms) avant une nouvelle mesure:
delay(100)

Résultats dans le moniteur série :