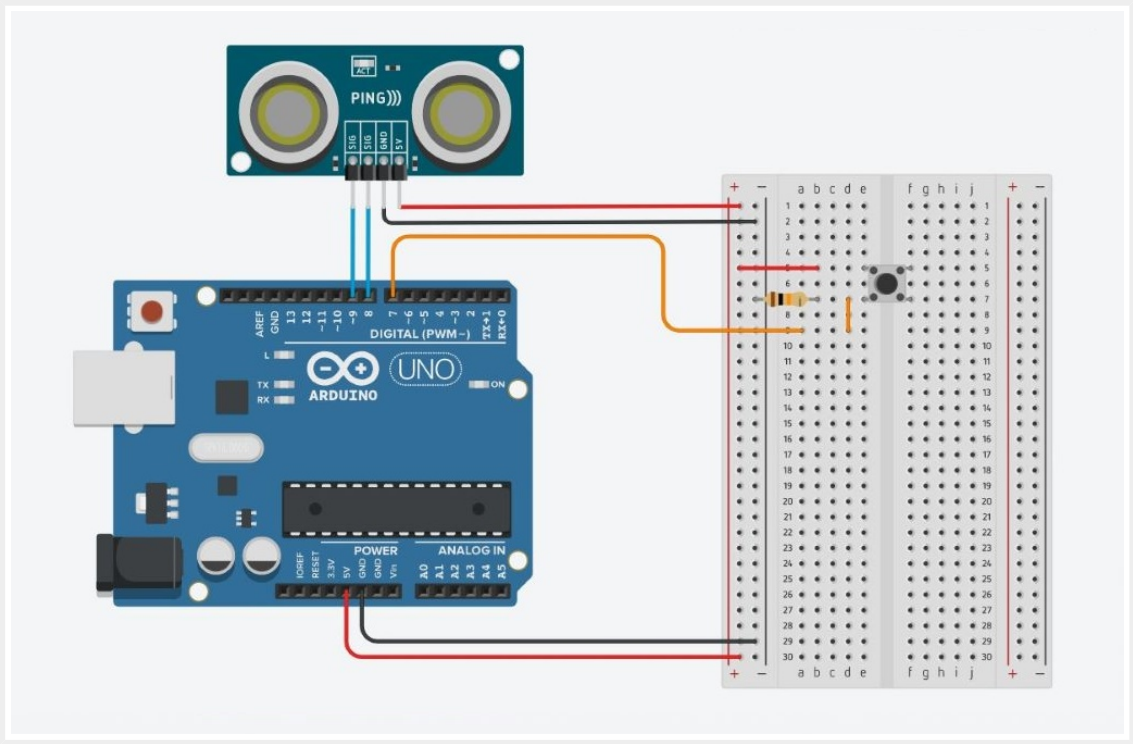
TP - Vitesse du son

(Détermination de la vitesse du son dans l'air à l'aide d'un capteur ultrasonique)



. Liste des composants :

. 1 capteur ultrasonique (HC-SR04)

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

. 1 bouton poussoir

. 1 plaque d'essai

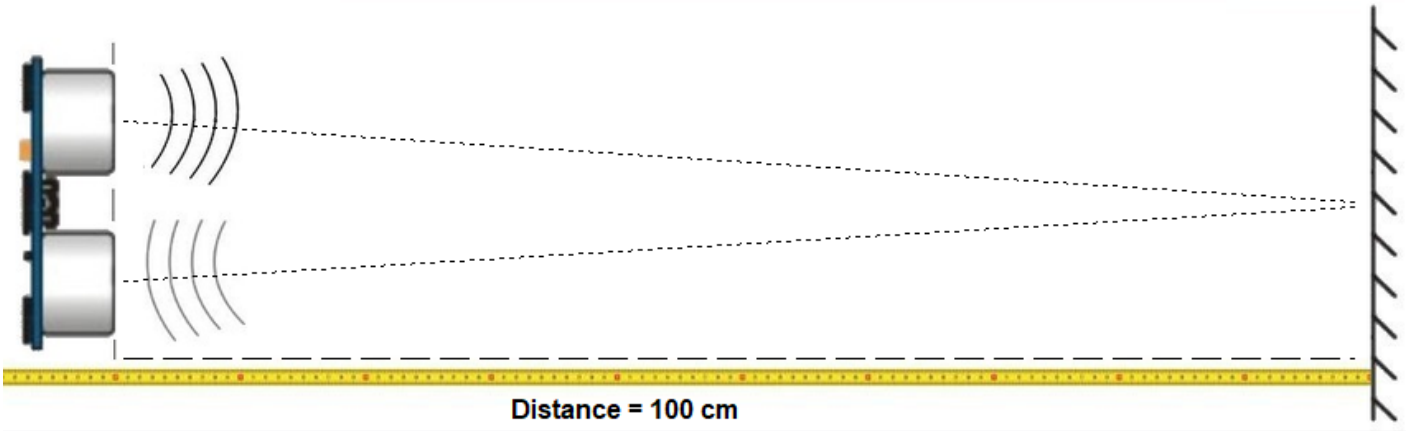
. Fils de connexion

. Objectif

Dans cette activité, nous allons déterminer expérimentalement la vitesse de propagation des ondes sonores en mesurant, à l'aide d'un capteur à ultrasons, 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.

Les mesures des durées de propagation Dt de l'onde sonore entre l'émetteur et le récepteur commencent après un appui sur le bouton poussoir et sont arrêtées en appuyant de nouveau sur celui-ci.

Il est donc possible d'acquérir des couples de données (Dt, d) afin de déterminer la vitesse du son dans l'air par le tracé de la caractéristique Dt = f(d).



. Rappels sur le son

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 20000 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) = 5000 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.

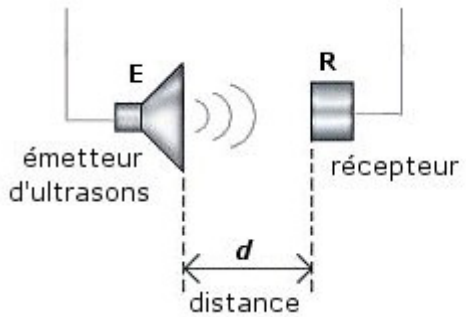
Par exemple : c (air à 0°C) = 331 $m.s^{-1}$; c (air à 15°C) = 340 $m.s^{-1}$

La relation entre la vitesse du son dans l'air en $m.s^{-1}$ et la température en kelvins est :

$$c_{air} = 20{,}05\sqrt{T}$$

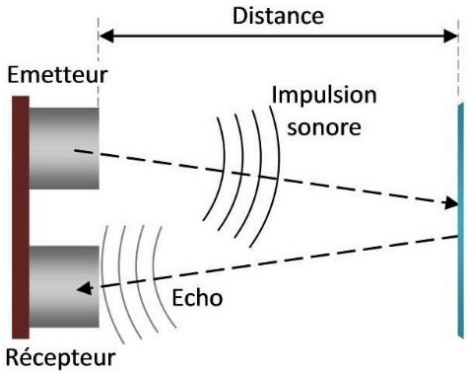
(T en kelvins = T en °C + 273,15)

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 :



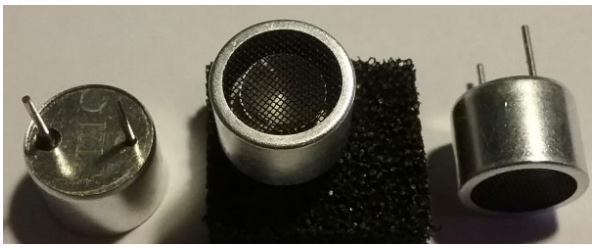
Dans ce cas, la distance parcourue par l'onde sonore, pendant la durée **Dt**, est **2 d**, et alors :

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

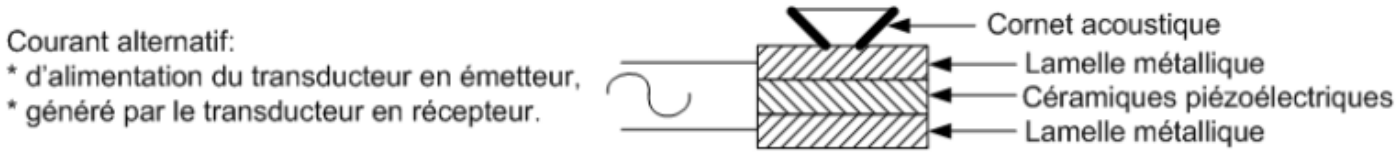
. Principe de fonctionnement des émetteurs et récepteurs à ultrasons

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

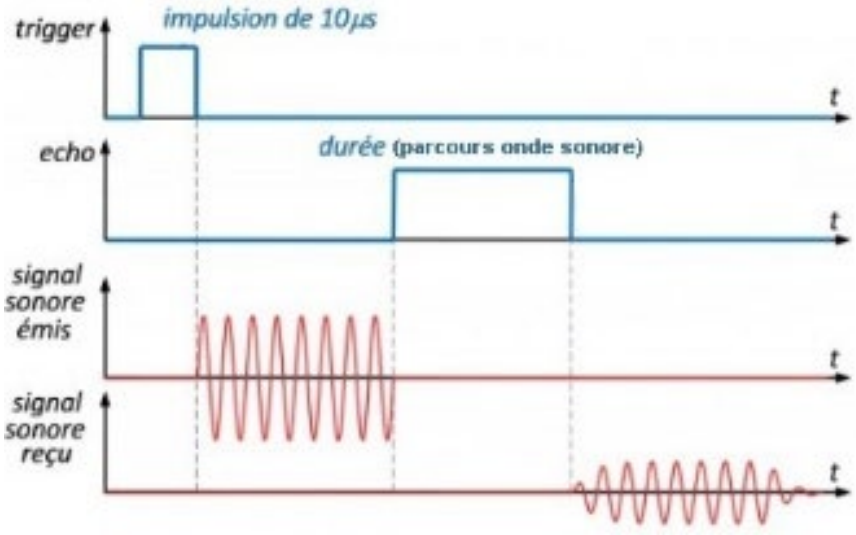
Spécifications et limites

- 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

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 calculer la distance entre le capteur et un obstacle est :

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

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 :

$$\text{Distance}_{\text{capteur- obstacle}}\ \text{(en m)} = d_{\text{parcours onde sonore}}/2 = V_{\text{ultrasons}} \times Dt / 2$$

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

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

$$= 17x\ Dt\ \text{(en µs)} / 1000$$

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

. Le programme

Voici le code de l'activité:

TP_Vitesse_Son

```
// 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;

// Initialisation des entrées et sorties

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

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

Serial.println("Appuyez sur le bouton poussoir pour mesurer la duree de propagation de l'onde sonore.");
Serial.println();
}

// 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)
  {
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("");
   Serial.println("Mesure de la duree de propagation de l'onde sonore en cours.");
   Serial.println("");
   Serial.println("d (cm) ; Dt (microS):");
   OldState=1;
  }

  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN, LOW);
  Dt = pulseIn(ECHO_PIN, HIGH, MEASURE_TIMEOUT);

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

. Fonction “pulseIn()” :

. Description

Lit la durée d’une impulsion (soit niveau HAUT, soit niveau BAS) appliquée sur une broche configurée en entrée.

Par exemple, si le paramètre valeur est HAUT, l’instruction pulseIn() attend que la broche passe à HAUT, commence alors le chronométrage, attend que la broche repasse au niveau BAS et stoppe alors le chronométrage. L’instruction renvoie la durée de l’impulsion en microsecondes. L’instruction s’arrête et renvoie 0 si aucune impulsion n’est survenue dans un temps spécifié.

. Syntaxe

pulseIn(broche, valeur)

pulseIn(broche, valeur, delai_sortie)

. Paramètres

broche: le numéro de la broche sur laquelle vous voulez lire la durée de l’impulsion. (type int)

valeur: le type d’impulsion à « lire » : soit **HIGH** (niveau HAUT) ou **LOW** (niveau BAS). (type int)

delai_sortie (optionnel): le nombre de microsecondes à attendre pour début de l’impulsion. La valeur par défaut est 1 seconde. (type unsigned long)

. Valeur renvoyée :

La durée de l’impulsion (en µs) ou 0 si aucune impulsion n’a démarrer avant le délai de sortie (type unsigned long)

Déroulement du programme :

– 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: **long Dt = 0**
- Variable correspondant à la valeur précédente de la durée de parcours de l'onde ultrasonore: **long DtMesure = 0**
- Variable correspondant à la distance entre le capteur et l'obstacle: **int Distance = 0**

– Initialisation des entrées et sorties :

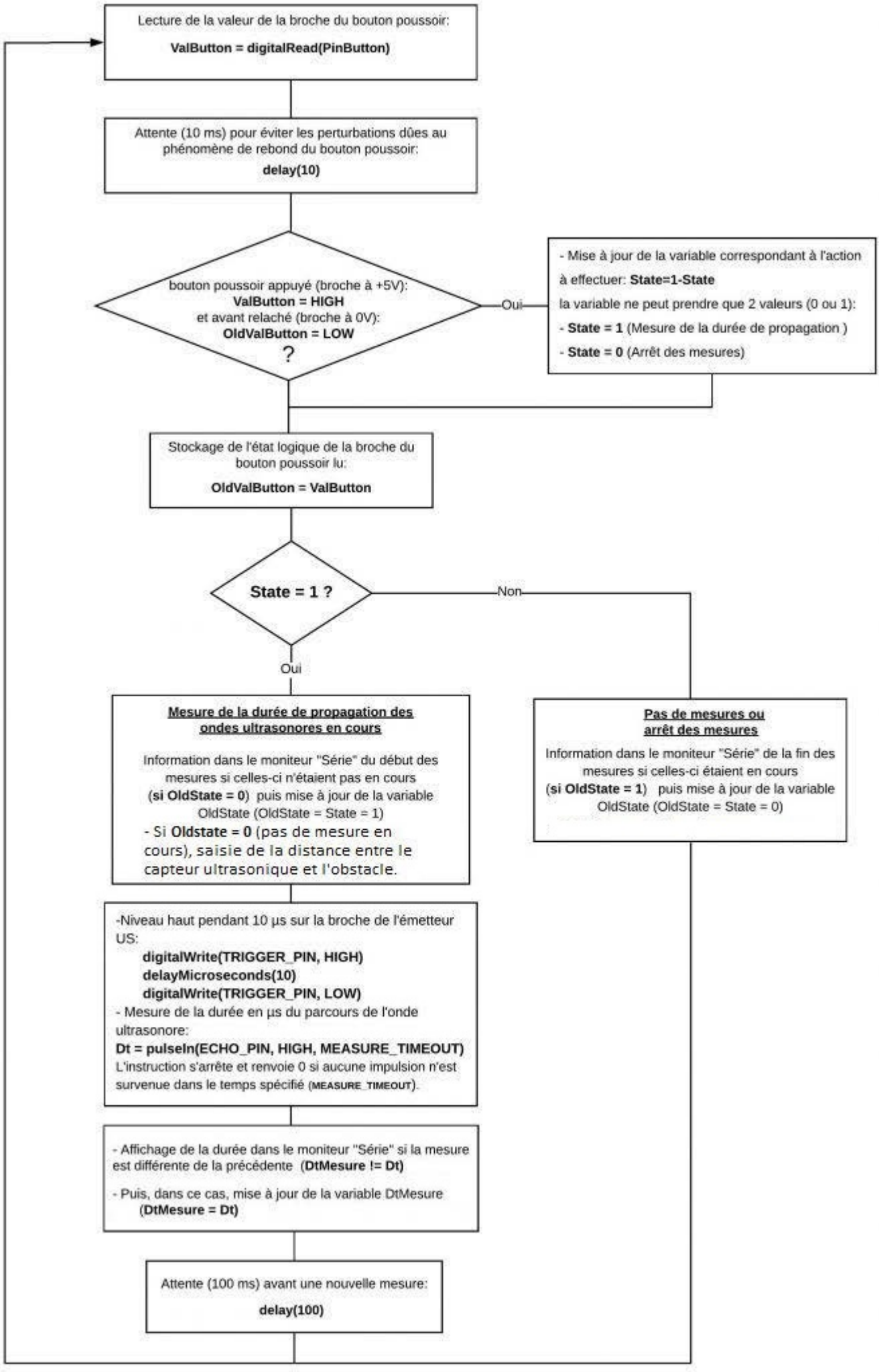
- le débit de communication en nombre de caractères par seconde pour la communication série est fixé à 9600 bauds: **Serial.begin(9600)**

- 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)**

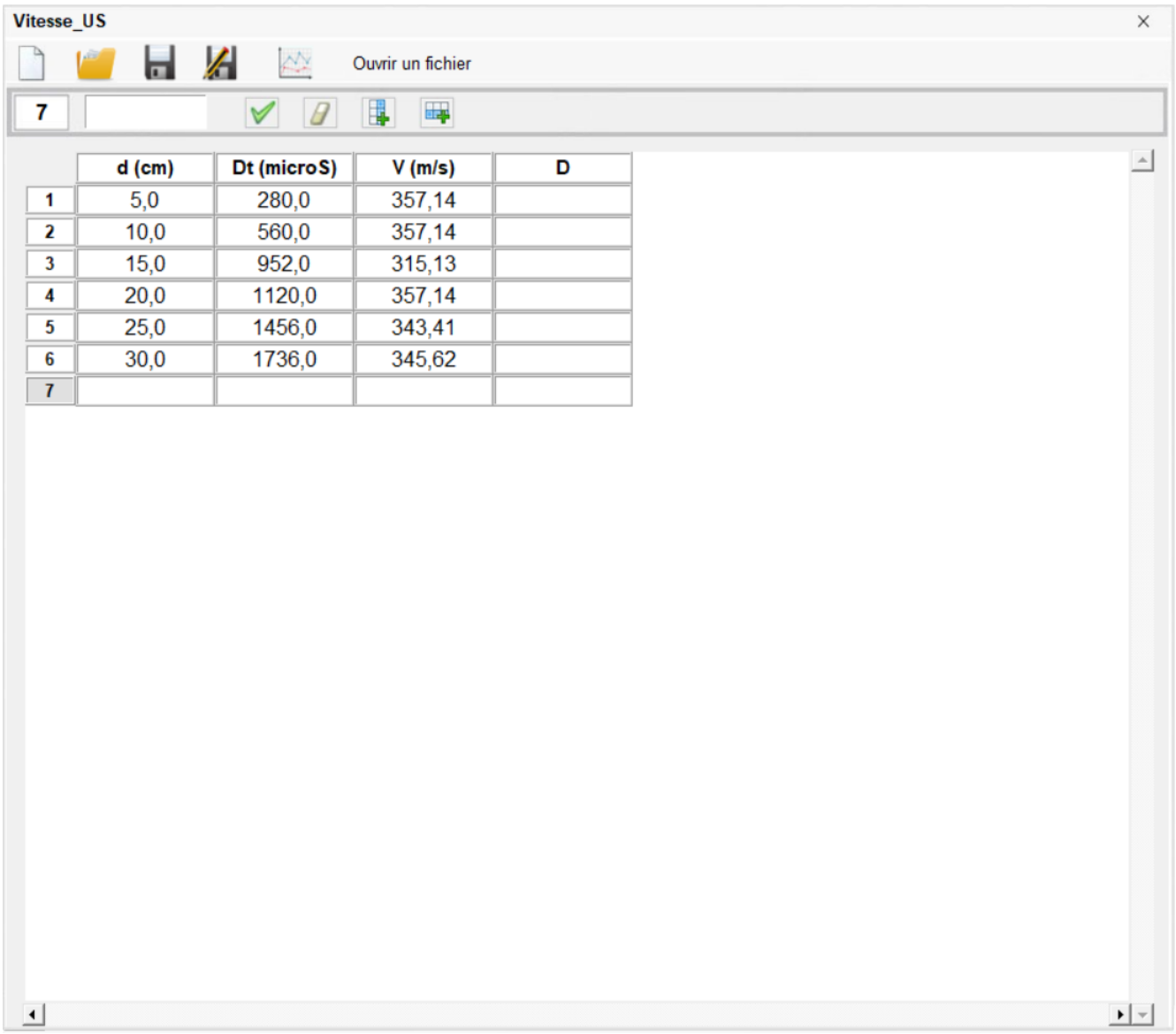
- La broche de l'émetteur US est initialisée comme une sortie digitale. Des données seront donc envoyées depuis le microcontrôleur vers cette broche: **pinMode(TRIGGER_PIN, OUTPUT)** et initialisée à un niveau bas (0 V): **digitalWrite(TRIGGER_PIN, LOW)**

- La broche du récepteur est initialisée comme une entrée digitale: **pinMode(ECHO_PIN, INPUT)**

– Fonction principale en boucle :

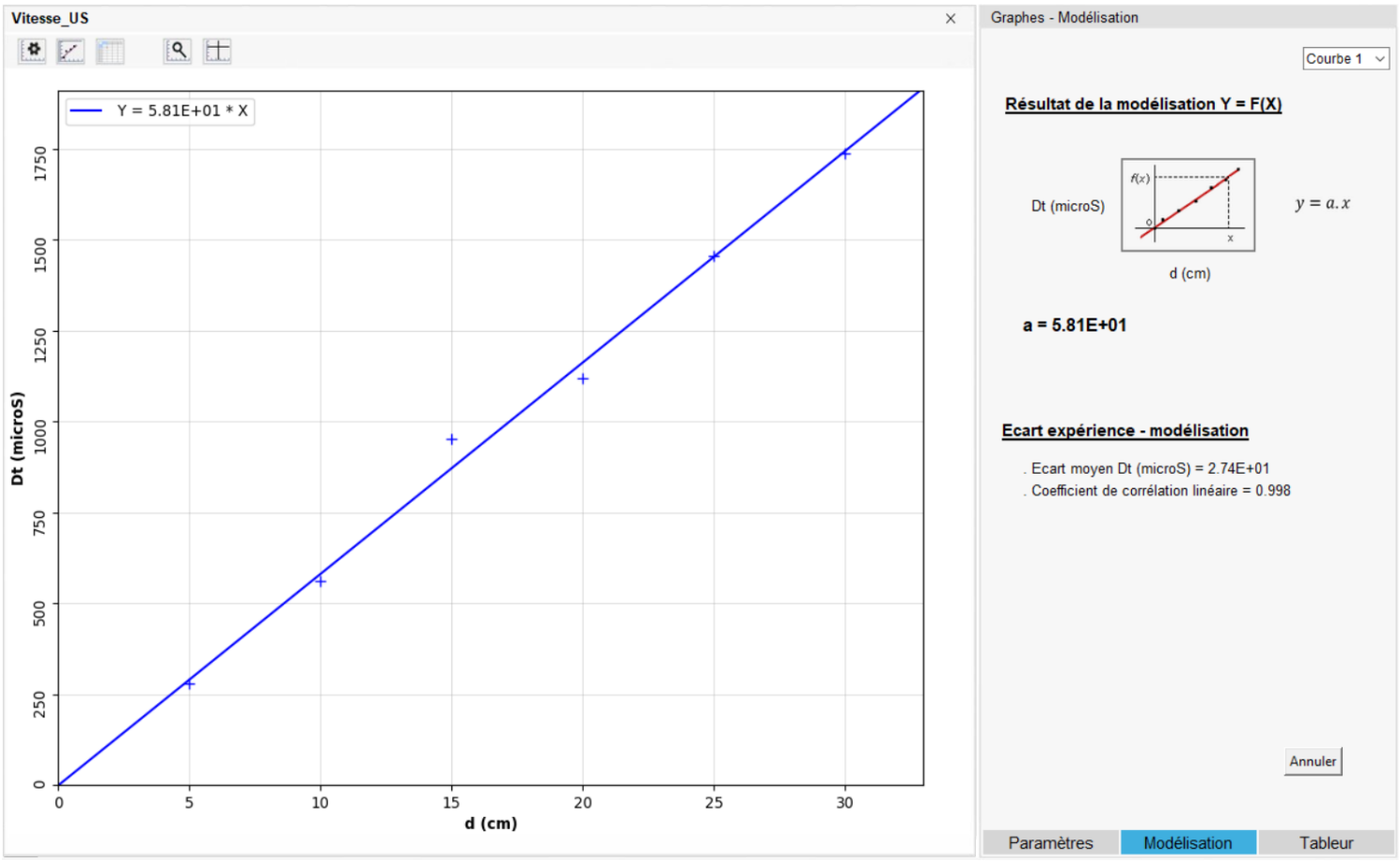

Lecture de la valeur de la broche du bouton poussoir:
ValButton = digitalRead(PinButton)
Attente (10 ms) pour éviter les perturbations dûes au phénomène de rebond du bouton poussoir:
delay(10)
bouton poussoir appuyé (broche à +5V):
ValButton = HIGH
et avant relaché (broche à 0V):
OldValButton = LOW
?
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)
- Si Oldstate = 0 (pas de mesure en cours), saisie de la distance entre le capteur ultrasonique et l'obstacle.
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)
-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).
- 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)
Attente (100 ms) avant une nouvelle mesure:
delay(100)

Résultats:



	d (cm)	Dt (microS)	V (m/s)	D
1	5,0	280,0	357,14	
2	10,0	560,0	357,14	
3	15,0	952,0	315,13	
4	20,0	1120,0	357,14	
5	25,0	1456,0	343,41	
6	30,0	1736,0	345,62	
7				

Modélisation de Dt = f(d) :



On en déduit la vitesse de propagation de l'onde ultrasonore:

v = 2*d / Dt = 2/a

v = 2/58,1 = 0,03442 cm/microS = 344,2 m/s