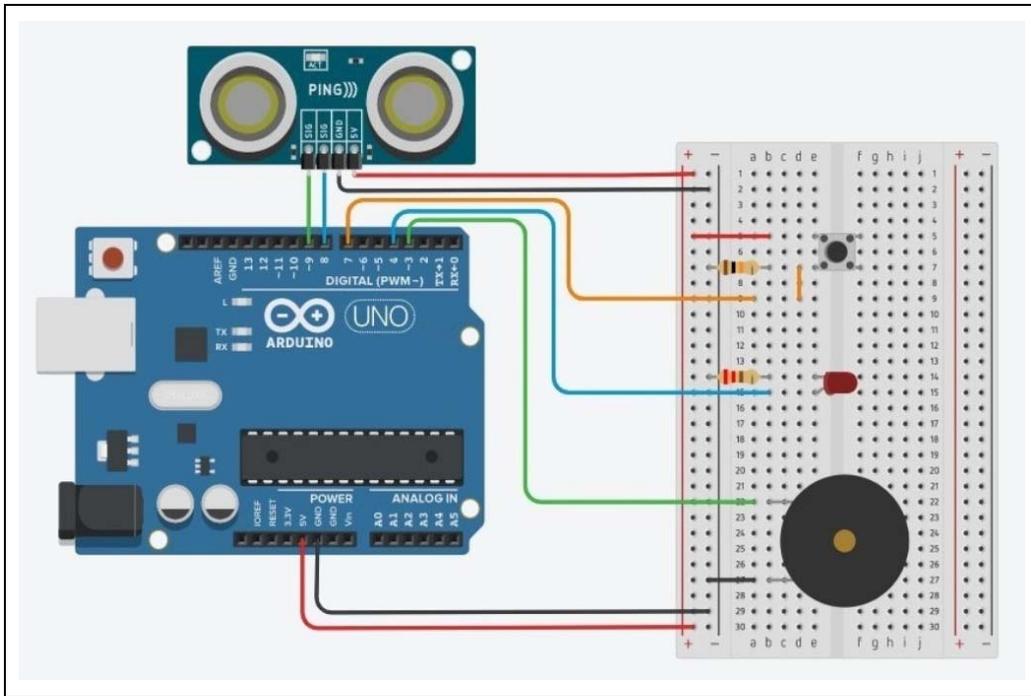


Ondes ultrasonores : Vitesse & Distances

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 à l'aide du circuit suivant :



Liste des composants :

- . 1 capteur ultrasonique (HC-SR04)
- . 1 DEL Rouge
- . 1 résistance de 220 Ω (résistance de protection de la DEL)
- . 1 résistance de 10 k Ω (résistance du bouton poussoir)
- . 1 bouton poussoir
- . 1 haut-parleur (ou piezzo)
- . 1 plaque d'essai
- . Fils de connexion

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.

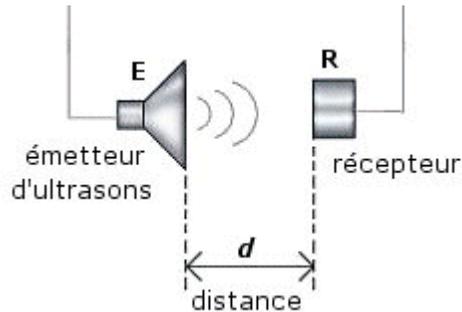
Par exemple : c (air à 0°C) = 331 m.s^{-1} ; c (air à 15°C) = 340 m.s^{-1}

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

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

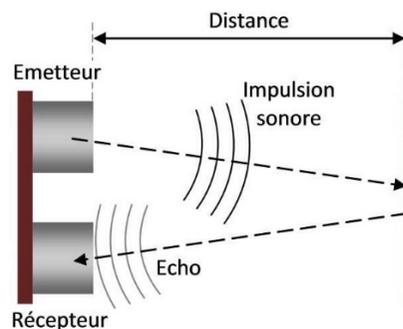
(T en kelvins = T en °C + 273,15)

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 :



$$c_{\text{air}} = \frac{d}{Dt}$$

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 $2d$, et alors :

$$c_{\text{air}} = \frac{2d}{Dt}$$

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) :

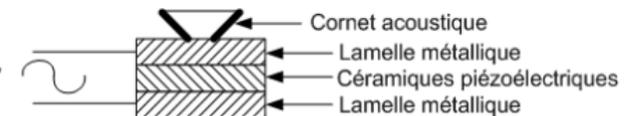


Un schéma de principe de ces transducteurs est donné ci-dessous :

Courant alternatif:

* d'alimentation du transducteur en émetteur,

* généré par le transducteur en récepteur.



. 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

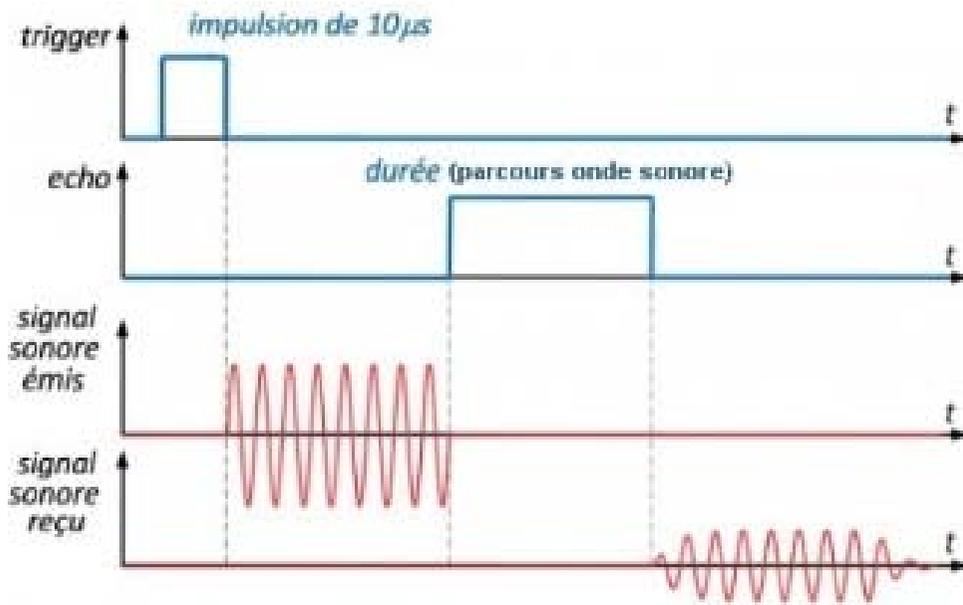
Spécifications et limites

- 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 $10 \mu\text{s}$,
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 :

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

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

$$d_{\text{parcours onde sonore}} = V_{\text{ultrasons}} \times Dt$$

Soit :

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

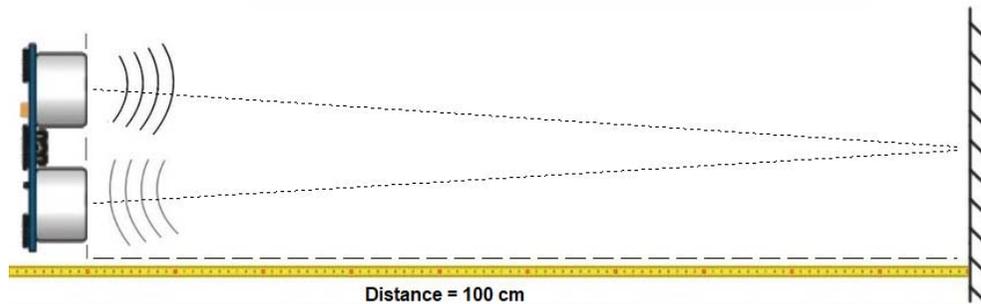
$$\text{Distance}_{\text{capteur-obstacle}} \text{ (en m)} = 340 \times Dt / 2$$

$$\begin{aligned} \text{Distance}_{\text{capteur-obstacle}} \text{ (en cm)} &= 34000 \times Dt \text{ (en } \mu\text{s)} / 2000000 \\ &= 17 \times Dt \text{ (en } \mu\text{s)} / 1000 \end{aligned}$$

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

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, 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

Voici le code de l'activité ("Activity1.ino" dans le dossier "Codes/Ondes Ultrasonores") :

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);
  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("");
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;
        }
    }
}

```

```

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

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).");
        Serial.println("");
        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 μ s) ou 0 si aucune impulsion n'a démarré avant le délai de sortie. (type unsigned long)

Déroulement du programme :

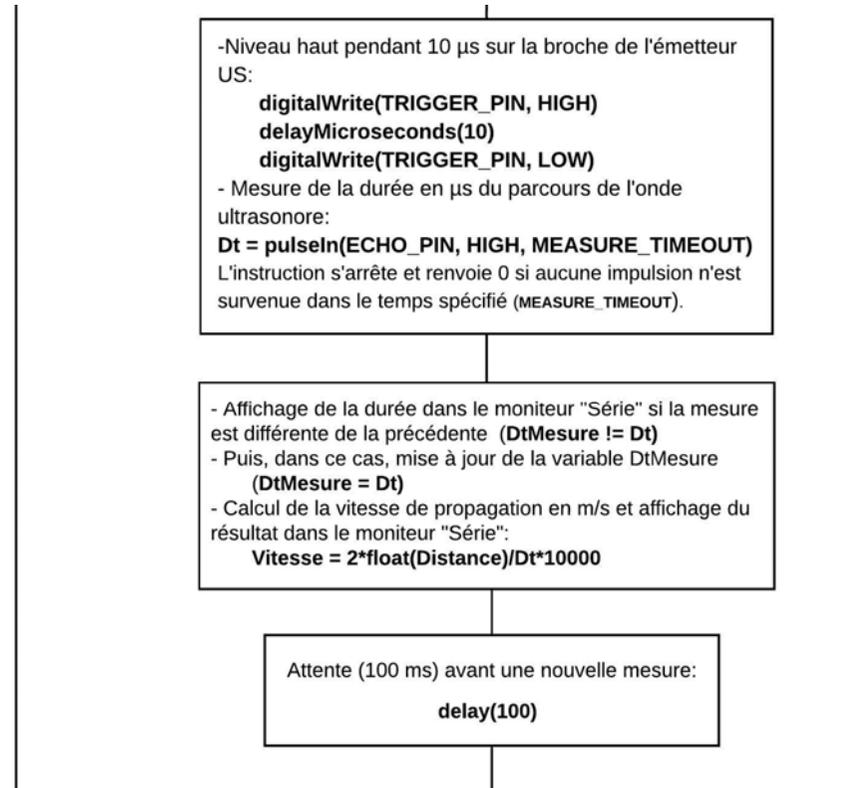
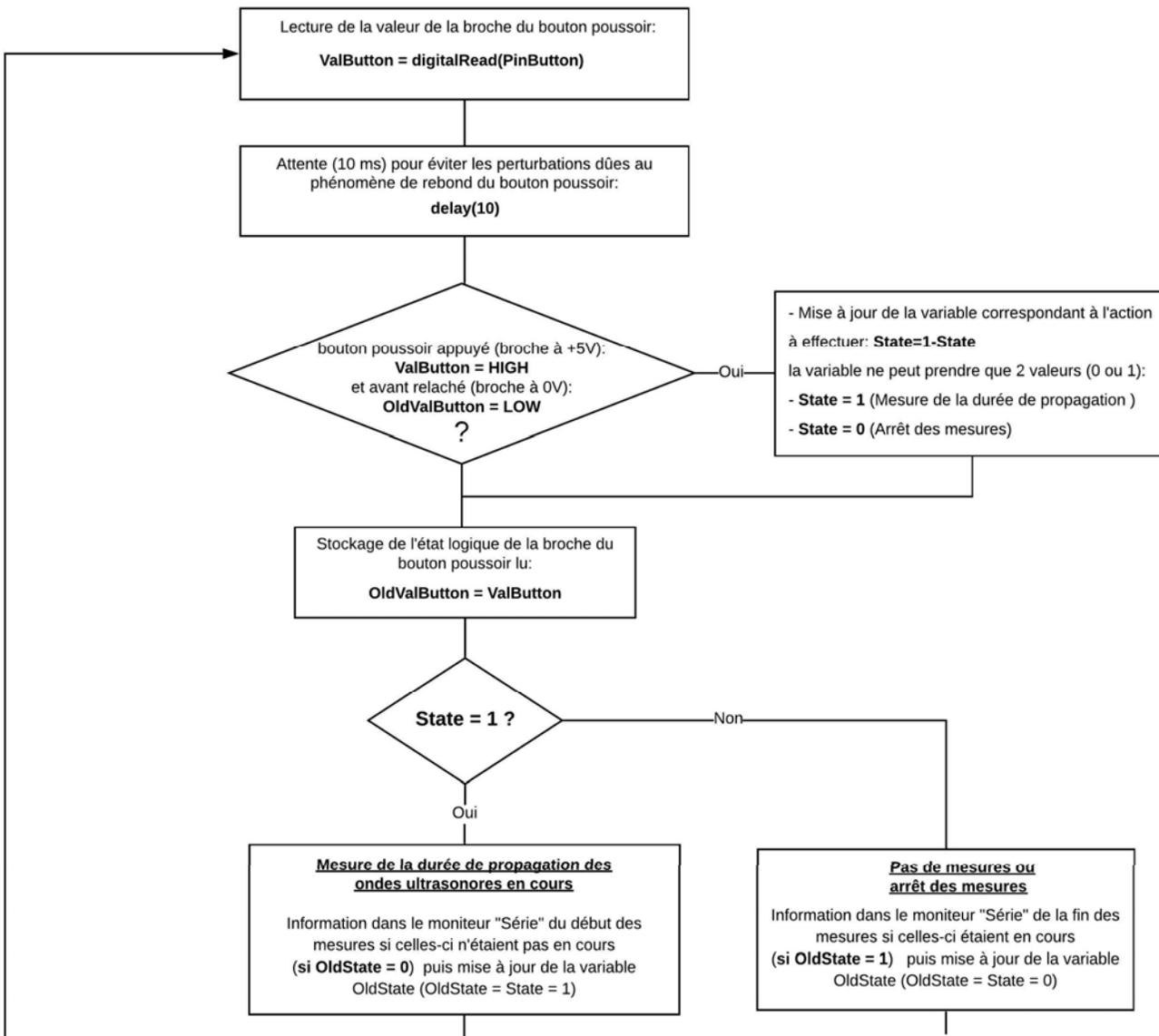
- Déclaration des constantes et variables :

```
- 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
- Variable correspondant à la vitesse de propagation des ondes ultrasonores dans l'air : float Vitesse = 0.0
```

- 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)
- Entrée de la distance entre le capteur ultrasonique et l'obstacle dans le moniteur "Série"
```

- Fonction principale en boucle :



Activité 2 : Mesure de distances

Maintenant que nous avons déterminé 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.

Le programme

Dans le code ("Activity2.ino" dans le dossier "Codes/Ondes Ultrasonores"), 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.

```
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 DistanceMeasure = 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);
  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;
  }

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

  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 :

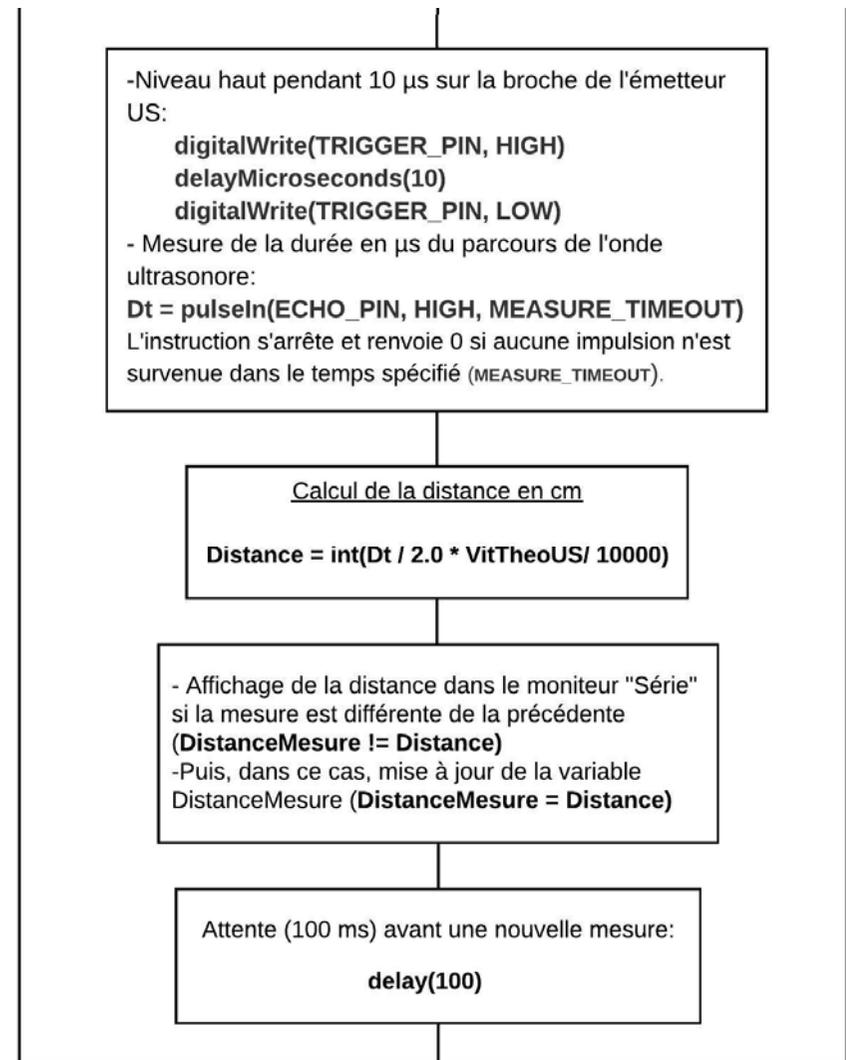
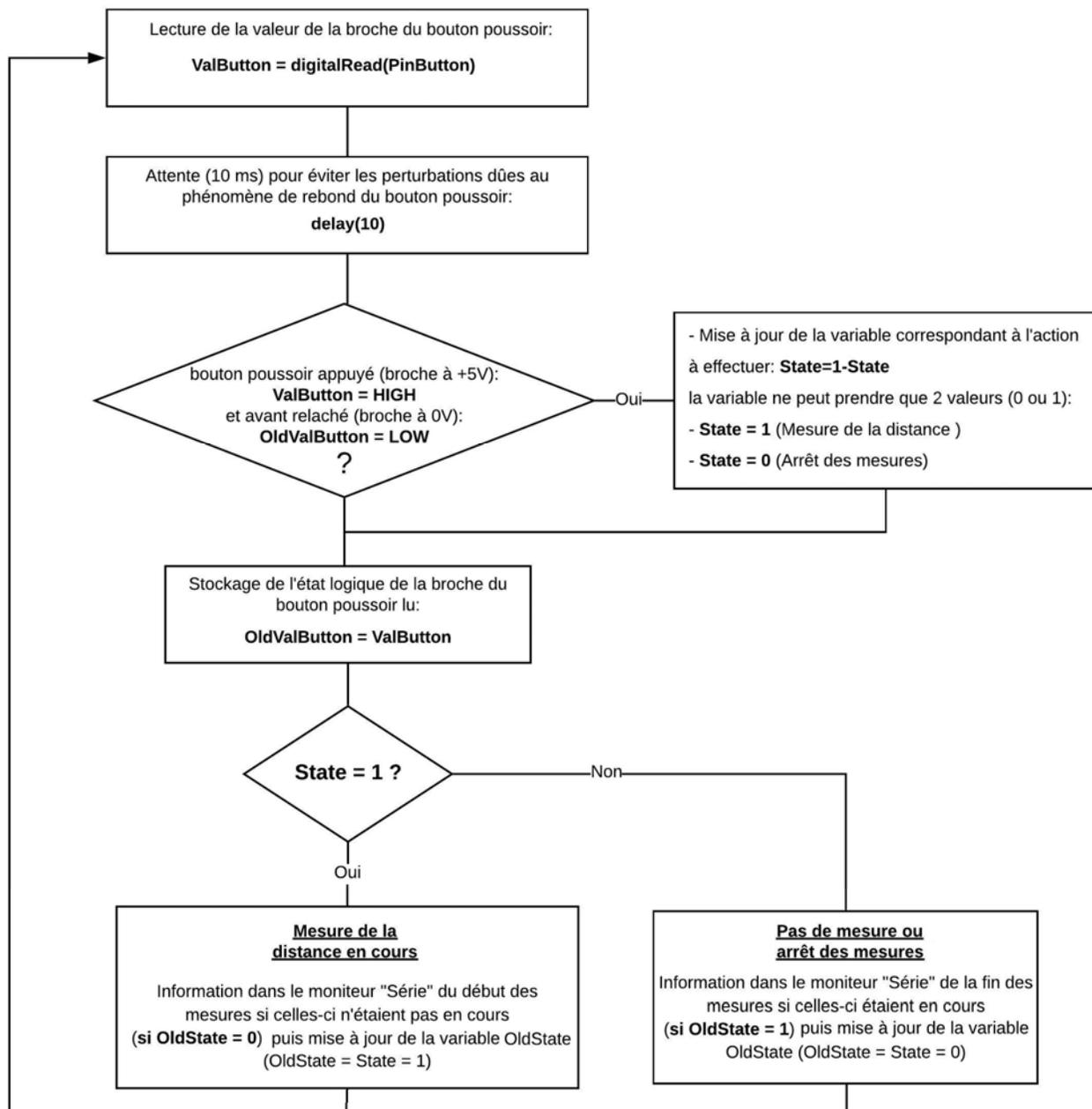
- Déclaration des constantes et variables :

- 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 distance mesurée: **int Distance = 0**
- Variable correspondant à la valeur précédente de la distance mesurée: **int DistanceMesure = 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))

- 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 :



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'obstacle qui déclenchera une alarme lumineuse et sonore quand le capteur est situé en dessous d'une distance, d , fixée.

. Le programme

Dans le code ("**Activity3.ino**" dans le dossier "**Codes/Ondes Ultrasonores**"), 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.

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

```
Activity3
// Déclaration des constantes et variables

const int PinButton = 7;
const int PinTone = 3;
const int PinLed = 4;
int TRIGGER_PIN = 8;
int ECHO_PIN = 9;

const unsigned long MEASURE_TIMEOUT = 25000UL;

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

int Distance = 0;
int DistanceMeasure = 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);
  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;
  }

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

  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);
  }
}

```

```

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

```

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 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))

- 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 la DEL est initialisée comme une sortie digitale. Des données seront donc envoyées depuis le microcontrôleur vers cette broche: **pinMode(PinLed, OUTPUT)**
- 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 :

