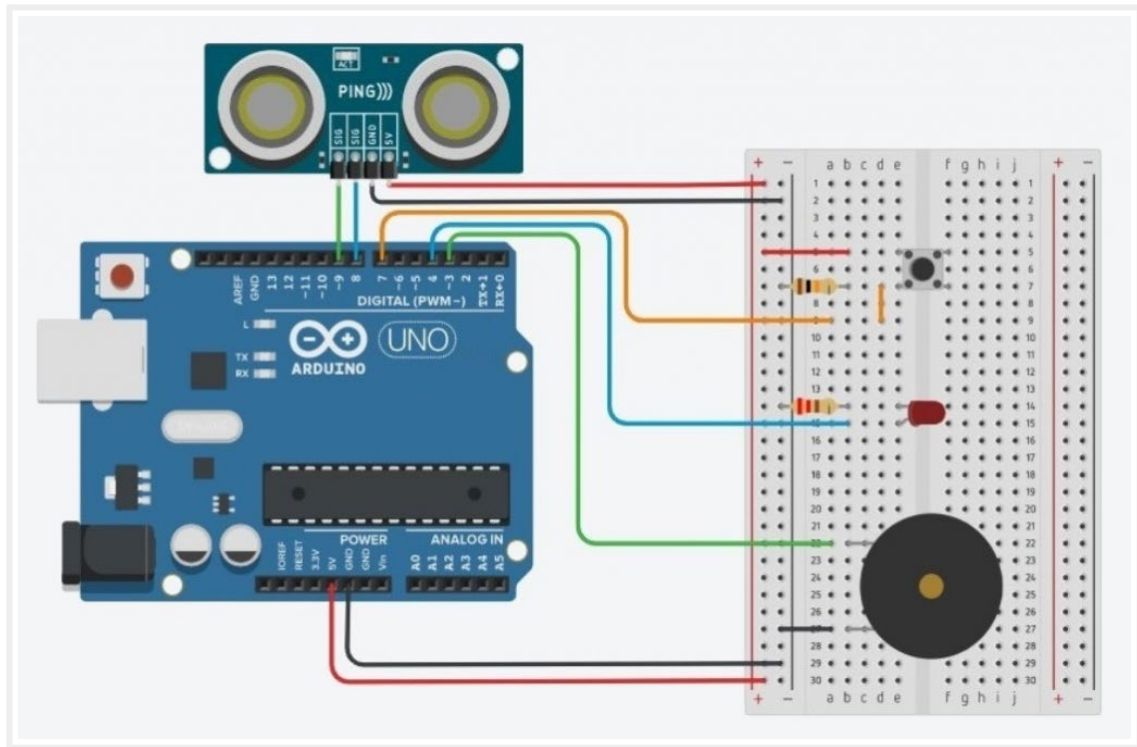


Détecteur d'obstacles

(Réalisation d'un détecteur d'obstacles à l'aide d'un capteur ultrasonique)



. 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 circuit du bouton poussoir)
- . 1 bouton poussoir
- . 1 haut-parleur (ou piezo)
- . 1 plaque d'essai
- . Fils de connexion

. Objectif

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, avec la DEL et le buzzer de notre circuit d'étude quand le capteur est situé en dessous d'une distance **d** fixée.

. Le programme

Dans le [code de l'activité](#), 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.

Detecteur_obstacles

```
// 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 :

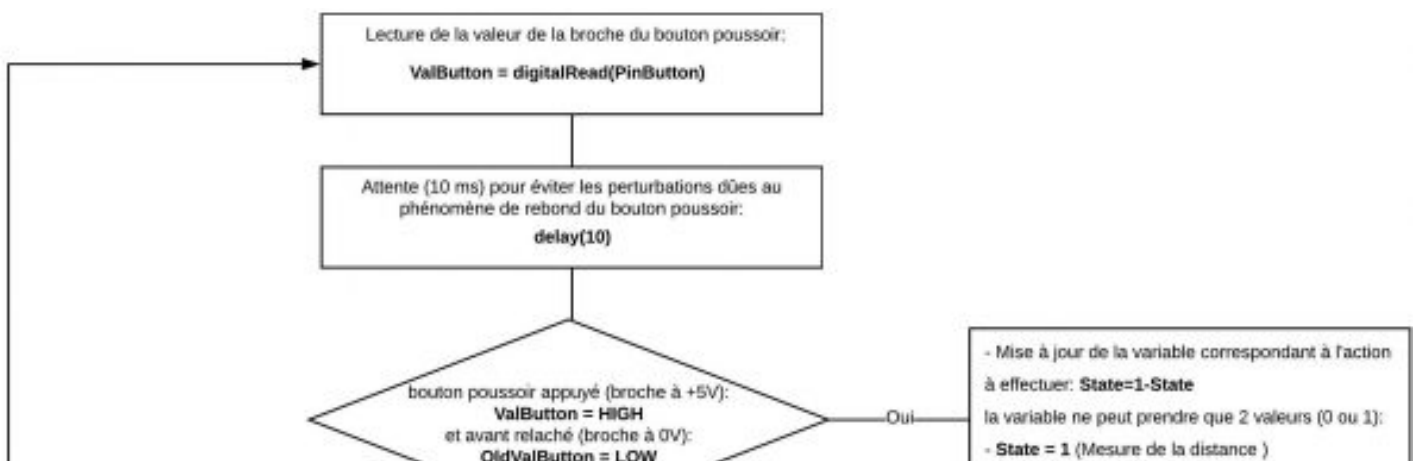
– Déclaration des constantes et variables :

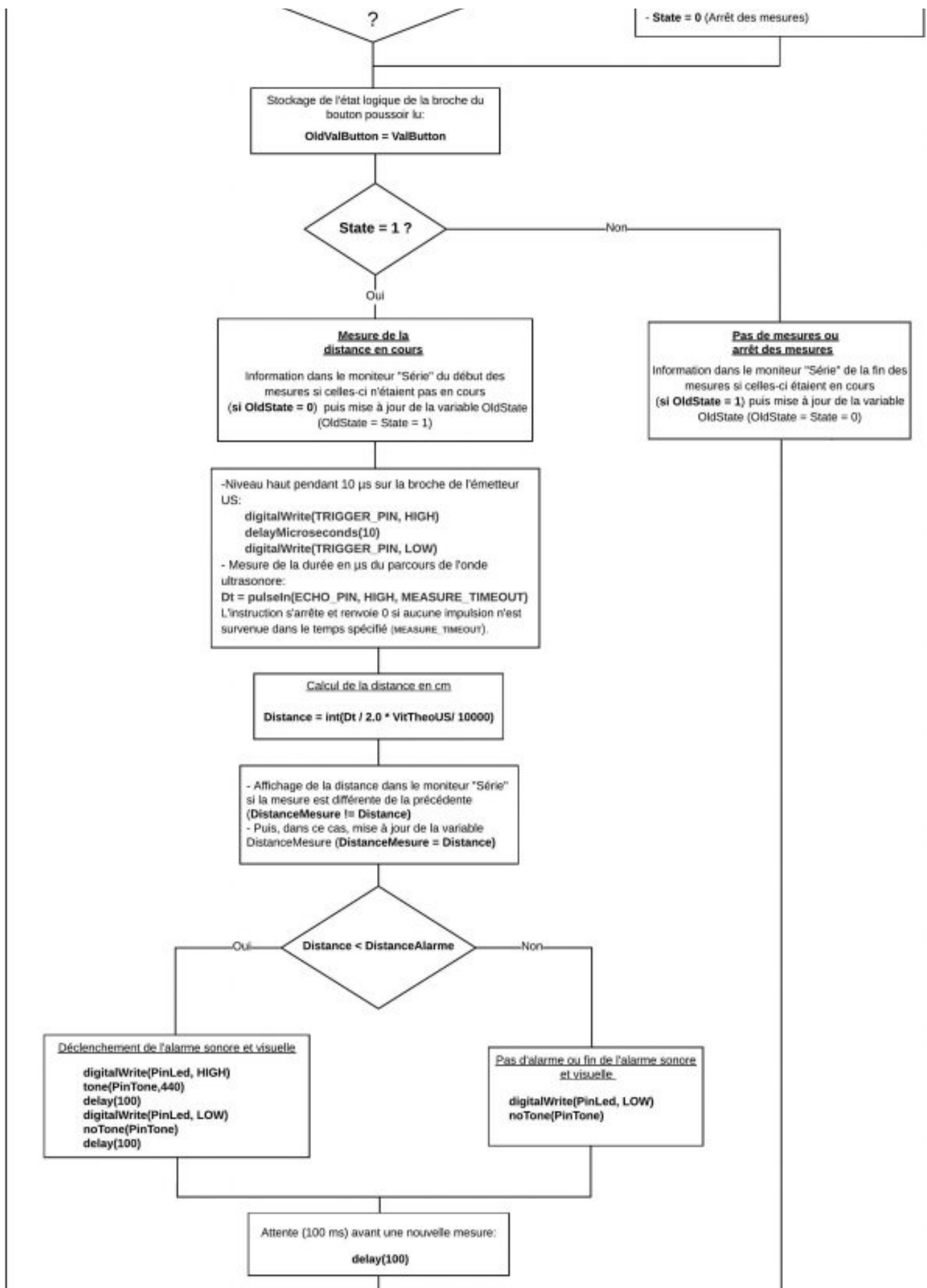
- N° de la broche correspondant au bouton poussoir: **const int PinButton = 7**
- N° de la broche correspondant au buzzer: **const int PinTone = 3**
- N° de la broche correspondant à la DEL: **const int PinLed = 4**
- N° de la broche correspondant à l'émetteur US : **int TRIGGER_PIN = 8**
- N° de la 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 :





Résultats dans le moniteur série:

