TP : Modéliser un capteur de voiture

Objectifs

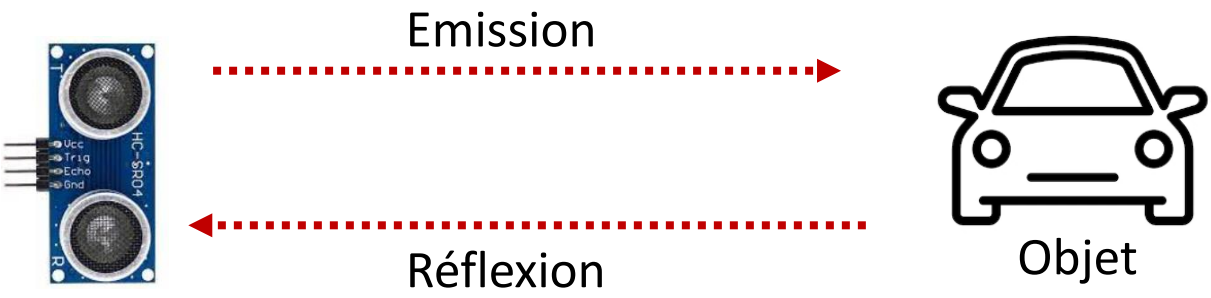
- ❑ Etape 5 : Apprendre à utiliser un capteur ultrason
- ❑ Etape 6 : Conditionner l'allumage d'une LED
- ❑ Etape 7 : Conditionner l'émission de sons par un buzzer
- ❑ Etape 8 : Bonus

Étape 5 : Capteur à ultrasons

Objectifs:

- ❑ Comprendre le fonctionnement du capteur à ultrason ;
- ❑ Connecter un capteur à ultrasons à l'Arduino ;
- ❑ Créer un capteur de distance avec le capteur à ultrasons.

Comprendre le fonctionnement du capteur à ultrason



Un capteur à ultrasons est un appareil qui utilise des ondes ultrasonores (ondes inaudibles de fréquence supérieure à 20kHz) pour mesurer la distance entre lui-même et un objet. Il est composé d'un émetteur et d'un récepteur.
Voici comment cela fonctionne de manière très simple :

(1) Emission d'ondes : L'émetteur envoie un signal ondulatoire qui se propage dans l'air en direction d'un objet.

(2) Réflexion des ondes : Lorsque les ondes atteignent l'objet, elles se réfléchissent dessus, tout comme une balle rebondit sur un mur.

(3) Retour du signal : Le récepteur "écoute" le signal sonore rebondissant et mesure le temps entre l'émission et la réception. Connaissant la vitesse de l'onde dans l'air, cela nous permet d'accéder à la distance objet/capteur. Plus l'objet est proche, plus l'intervalle de temps mesuré est court.

Pour mesurer la distance, la formule est la suivante:

$$Distance = \frac{Vitesse \cdot temps}{2}$$

Dans l'air et dans des conditions usuelles, une bonne approximation de la vitesse du son est 343 m/s.

Le temps est mesuré par le capteur entre l'instant où il émet l'onde et l'instant où il reçoit la réflexion.

La distance peut alors être estimée.

? Question pratique : pourquoi y a-t-il un "2" dans la formule de la distance ?

Combien de temps va s'écouler entre l'émission et la **réflexion** d'un objet à 10 mètres de distance ?

À quelle distance est un objet pour lequel il y a 50 ms entre les instants d'émission et de réception ?

Connecter un capteur à ultrasons avec l'Arduino
Créer un capteur de distance avec le capteur à ultrasons

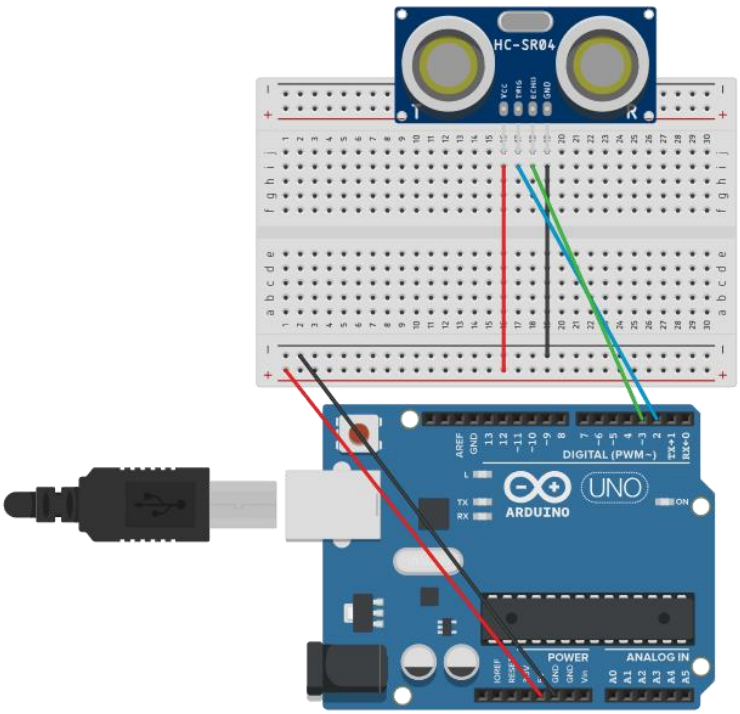
A – Entrée alimentation (5V)
B – TRIGGER (port numérique)
C – ECHO (port numérique)
D – Masse (0V)

Réponses:

- Car il y a un aller et un retour
- $\frac{10}{343} \approx 0{,}029$
- 8,575 m

Montage

On connecte les GND (pour Ground = Masse) à la même ligne (-).
On choisit aussi les ports de la carte reliés au trigger et à l'echo (ici, 3 et 2 respectivement), et on n'oublie pas d'alimenter le composant !



Pour utiliser un capteur à ultrasons, trois parties du code sont nécessaires :

(1) Déclaration: Le trigger et l'echo sont connectés à des ports, donc on va changer le nom des broches pour des noms plus intuitifs.

```
#define  port_trigger  2
#define  port_echo 3
```

(2) Mode: Le trigger est une sortie et l'echo une entrée, donc on va les définir.

```
pinMode(port_trigger, OUTPUT)
pinMode(port_echo, INPUT)
```

(3) Utilisation : Pour détecter la distance, il est nécessaire d'envoyer un signal au déclencheur (trigger) et de vérifier la réception d'un signal dans l'écho.

(3.1) Pour envoyer une onde, il **faut changer l'état du déclencheur (trigger) de LOW à HIGH pendant 10 microsecondes.**

(3.2) Pour vérifier le temps entre l'émission et la réception du signal, on peut utiliser la fonction **pulseIn(port_echo, HIGH).**

? Exercice pratique: compléter le code ci-dessous avec les informations manquantes pour vérifier la distance d'un objet.

```
1.  #define port_trigger 2
2.  #define port_echo 3
3.
4.  /* Vitesse du son en cm/us */
5.  float vitesse_son = 343.0 / 10000;
6.
7.  void setup() {
8.    pinMode(port_trigger, OUTPUT);
9.    digitalWrite(port_trigger,LOW);
10.   pinMode(port_echo,INPUT);          //le capteur est une entrée
11.   Serial.begin(9600);
12.  }
13.
14. void loop() {
15.   digitalWrite(port_trigger,HIGH);
16.   delayMicroseconds(10);
17.   digitalWrite(port_trigger,LOW);
18.   long measure = pulseIn(port_echo,HIGH);
19.   float distance_cm = measure / 2.0 * vitesse_son;
20.   Serial.println(distance_cm);
21.   delay(500);
22.  }
```

(1) Déclaration

(2) Modes

(3) Utilisation

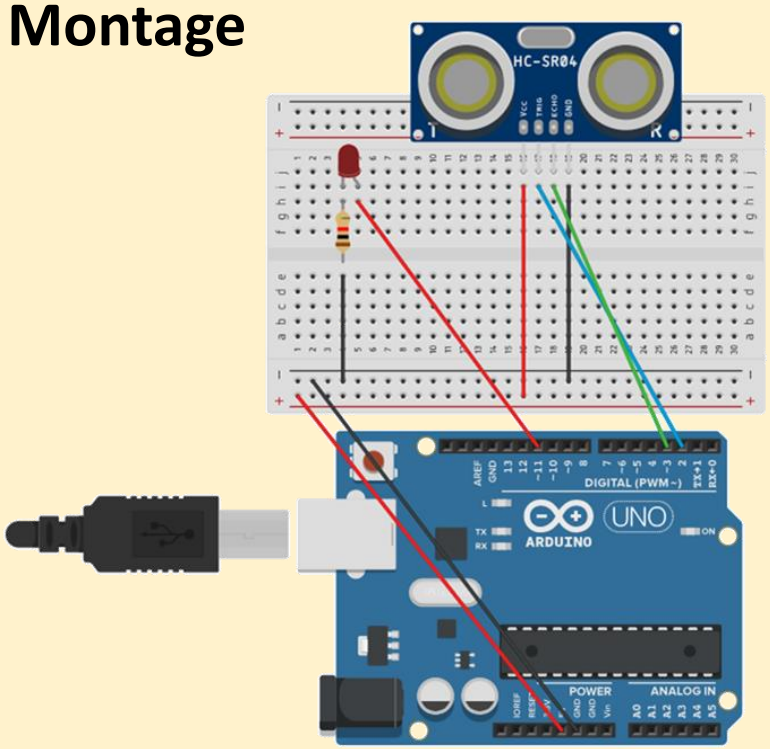
Étape 6 : Lier une LED au capteur

Objectifs:

- ❑ Conditionner l'allumage de la LED selon les valeurs recueillies par le capteur
- ❑ Changer la fréquence d'allumage de la LED

Conditionner l'allumage de la LED selon les valeurs recueillies par le capteur

Montage



Par rapport au montage précédent, on rajoute une LED connectée sur le port 11 et à la masse. On n'oublie pas la résistance pour ne pas endommager la LED.

? Exercice pratique: compléter le code ci-dessous avec les informations manquantes pour vérifier la distance d'un objet.

```
1.  #define port_trigger 2
2.  #define port_echo 3
3.  const int LED = 11;
4.
5.  /* Vitesse du son en cm/us */
6.  float vitesse_son = 343.0 / 10000;
7.
8.  void setup() {
9.    pinMode(LED, OUTPUT);
10.   pinMode(port_trigger, OUTPUT);
11.   digitalWrite(port_trigger,LOW);
12.   pinMode(port_echo,INPUT);
13.   Serial.begin(9600);
14.  }
15.
16. void loop() {
17.   digitalWrite(port_trigger,HIGH);
18.   delayMicroseconds(10);
19.   digitalWrite(port_trigger,LOW);
20.   long measure = pulseIn(port_echo,HIGH);
21.   float distance_cm = measure / 2.0 * vitesse_son;
22.   Serial.println(distance_cm);
23.
24.
25.   if (distance_cm < 20) {
26.     digitalWrite(LED, HIGH);
27.   }
28.   else {
29.     digitalWrite(LED,LOW);
30.  }
31.   delay(500);
32.  }
```

Code identique à l'étape 1

Si la distance est inférieure à 20 cm, on allume la LED

Simuler un capteur de voiture

Dans un **radar de recul pour voiture**, l'alarme sonne **de plus en plus vite** pour alerter le conducteur de sa **proximité** avec l'obstacle. On souhaite donc faire clignoter la LED de plus en plus rapidement lorsque la distance diminue.

? Exercice pratique : Réfléchir à une solution pour adapter ce fonctionnement à notre circuit, en s'inspirant de ce qui précède.

Réponse:
Voir suite. Les deux solutions possibles étant soit de mettre plusieurs seuils, soit de mettre une fréquence de clignotement inversement proportionnel à la distance.

En pratique, de nombreuses solutions sont envisageables. On pourrait très bien adapter ce qui a été fait en début d'étape. On a utilisé un **seuil** (une valeur limite au-delà de laquelle le comportement du code change). On pourrait donc utiliser plusieurs seuils. Cependant, à des fins de concision dans notre code, on choisira une autre solution : la **fréquence** d'allumage de la LED sera **inversement proportionnelle** à la distance à l'obstacle.

? Question pratique : Que signifie « fréquence » ici ?

Si la distance augmente, comment varie la fréquence d'allumage de la LED ?

Ici, on ne modifiera que le void loop.
Pour influer sur la fréquence, on changera le *delay.*

```
1. #define port_trigger 2
2. #define port_echo 3
3. const int LED = 11;
4.
5. /* Vitesse du son en cm/us */
6. float vitesse_son = 343.0 / 10000;
7.
8. void setup() {
9.   pinMode(LED, OUTPUT);
10.  pinMode(port_trigger, OUTPUT);
11.  digitalWrite(port_trigger,LOW);
12.  pinMode(port_echo,INPUT);
13.  Serial.begin(9600);
14. }
15.
16.void loop() {
17.  digitalWrite(port_trigger,HIGH);
18.  delayMicroseconds(10);
19.  digitalWrite(port_trigger,LOW);
20.  long measure = pulseIn(port_echo,HIGH);
21.  float distance_cm = measure / 2.0 * vitesse_son;
22.  Serial.println(distance_cm);
23.  if (distance_cm > 50) {
24.   digitalWrite(LED, LOW);
25.   delay(500);
26.  }
27.  else {
28.   digitalWrite(LED,HIGH);
29.   delay(distance_cm*20);
30.   digitalWrite(LED,LOW);
31.   delay(distance_cm*20);
32.  }
33. }
```

Ici, le 20 est plutôt arbitraire mais permet de bien voir le changement de fréquence

Réponses:
Il s’agit de la fréquence du signal correspondant à l’état de la LED

Si la distance augmente, la LED s’allume de moins en moins vite.

Étape 7 : De la lumière au son

Objectifs:

- ❑ Générer un son d'une certaine fréquence
- ❑ Réaliser un capteur de voiture avec du son !

Générer un son d'une certaine fréquence

Figure 2

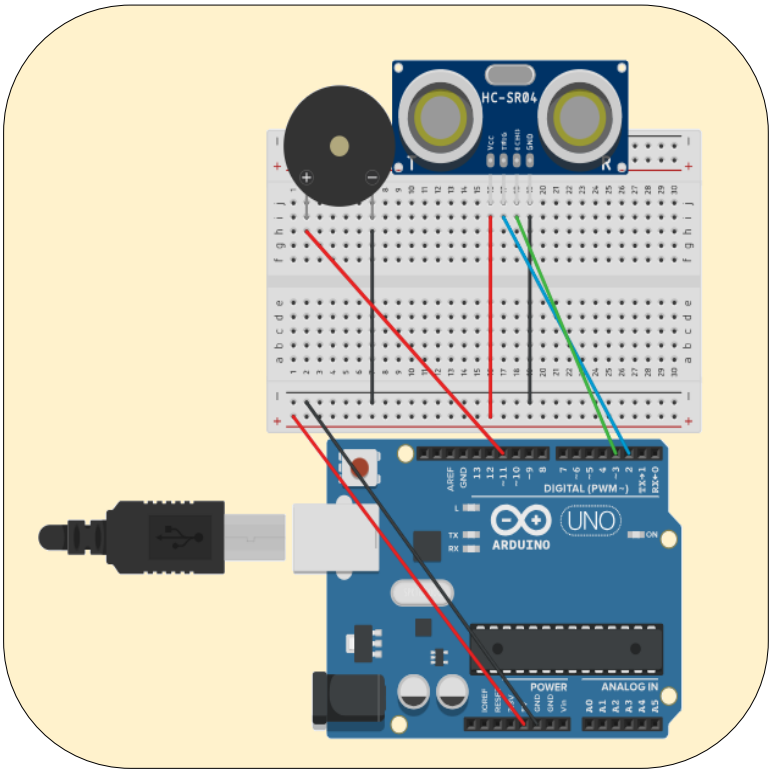


Figure 3

Le buzzer du kit Arduino ressemble à celui de la Figure 2. Pour utiliser le buzzer correctement, on va connecter la patte (-) à la masse, et la patte (+) à la patte de la carte qui commandera le buzzer (11 sur Figure 3).

Il y a plusieurs fonctions importantes à retenir pour l'utilisation du buzzer.

D'abord, dans le void setup, on définit la patte utilisée :

```
pinMode(PIN,OUTPUT);
```

Puis, pour générer un son :

```
tone(PIN,f);
```

Où `PIN` est la patte utilisée (ici 11) et `f` la fréquence du son généré.

Enfin, pour que le buzzer n'émette plus de son :

```
noTone(PIN)
```

? Exercice pratique : Compléter le code pour générer un son à une fréquence de 100 Hz pendant 500ms peu importe le résultat de la distance.

```
1. #define port_trigger 2
2. #define port_echo 3
3. float vitesse_son = 343.0 / 10000;
4. void setup() {
5.   pinMode(11, OUTPUT);
6.   pinMode(port_trigger, OUTPUT);
7.   digitalWrite(port_trigger,LOW);
8.   pinMode(port_echo,INPUT);
9.   Serial.begin(9600);
10.}
11.void loop() {
12.  digitalWrite(port_trigger,HIGH);
13.  delayMicroseconds(10);
14.  digitalWrite(port_trigger,LOW);
15.  long measure = pulseIn(port_echo,HIGH);
16.  float distance_cm = measure / 2.0 * vitesse_son;
17.  Serial.println(distance_cm);
18.
19.  tone(11,100);
20.  delay(500);
21.  noTone(11);
22.  delay(500);
23. }
```

Réaliser un capteur de voiture avec du son !

Passons maintenant au capteur de voiture avec avertisseur sonore, comme sur les vraies voitures !

Avec ce que vous venez d'apprendre sur le son, on adapte le code de la LED pour le buzzer

Le début du code ne change pas :

```
1. #define port_trigger 2
2. #define port_echo 3
3. /* Vitesse du son en cm/us */
4. float vitesse_son = 343.0 / 10000;
5.
6. void setup() {
7.   pinMode(11, OUTPUT);
8.   pinMode(port_trigger, OUTPUT);
9.   digitalWrite(port_trigger,LOW);
10.  pinMode(port_echo,INPUT);
11.  Serial.begin(9600);
12. }
13.
14. void loop() {
15.  digitalWrite(port_trigger,HIGH);
16.  delayMicroseconds(10);
17.  digitalWrite(port_trigger,LOW);
18.  long measure = pulseIn(port_echo,HIGH);
19.  float distance_cm = measure / 2.0 * vitesse_son;
20.  Serial.println(distance_cm);
```

Suite et fin du code : Les choses changent

```
21.   if (distance_cm > 50) {
22.      noTone(11);
23.      delay(500);
24.    }
25.
26.    else if (distance_cm<4) {
27.      tone(11,440);
28.      delay(400);
29.    }
30.    else {
31.      tone(11,440);
32.      delay(distance_cm*20);
33.      noTone(11);
34.      delay(distance_cm*20);
35.      tone(11,440);
36.      delay(distance_cm*20);
37.      noTone(11);
38.      delay(distance_cm*20);
39.    }
40. }
```

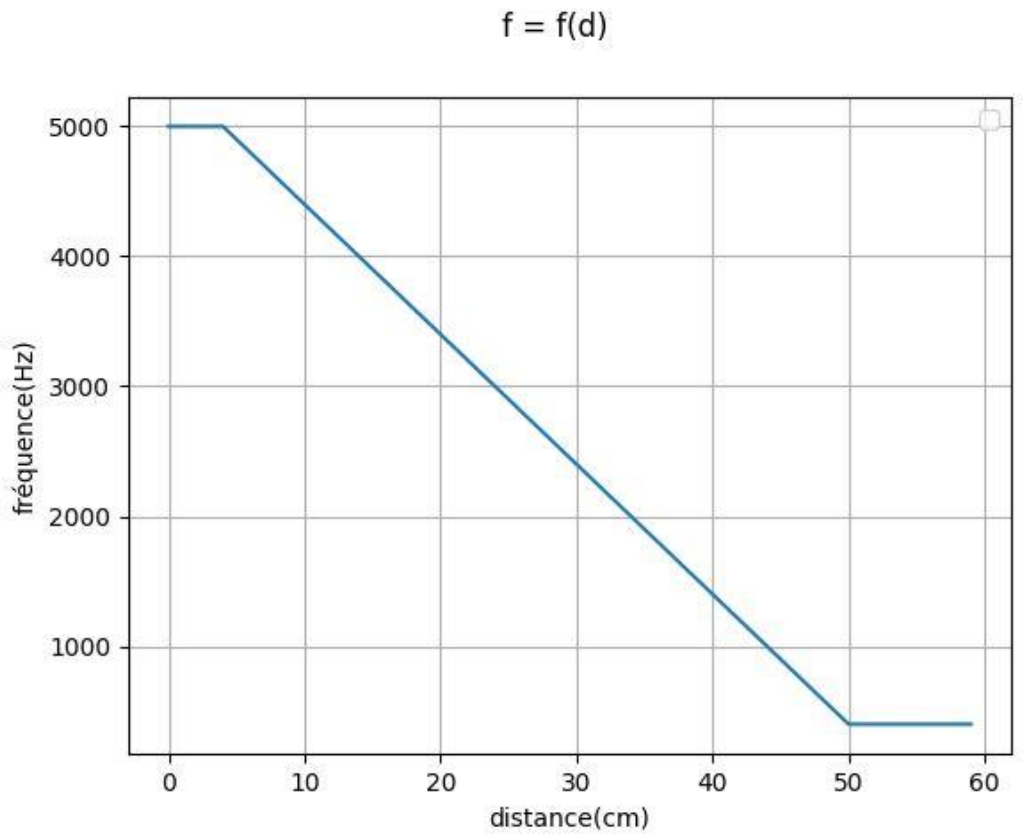
? Exercice pratique : Compléter le code. Faire appel à un encadrant. Admirer votre capteur de voiture fonctionnel.

Étape 8 : Bonus

? Exercice pratique : En pratique, en plus de sonner de plus en plus rapidement, le son émis est de plus en plus aigu lorsque la distance est faible. Complétez le code suivant (la première partie reste identique) pour que le son soit plus aigu à mesure que la distance diminue. On commencera par à un son à 400 Hz lorsque la distance vaut 50cm et on finira par un son à 5000 Hz lorsque la distance est inférieure à 4 cm.

Il s'agit d'une fonction affine par morceau qui est :

- Constante égale à 5000 entre 0 et 4
- D'équation : y = 5400 – 100x entre 4 et 50
- Constante égale à 400 à partir de 50



```
21.   if (distance_cm > 50) {
22.      noTone(11);
23.      delay(500);
24.    }
25.
26.    else if (distance_cm<4) {
27.      tone(11,5000);
28.      delay(400);
29.    }
30.    else {
31.      tone(11,20000/distance_cm);
32.      delay(distance_cm*20);
33.      noTone(11);
34.      delay(distance_cm*20);
35.      tone(11,20000/distance_cm);
36.      delay(distance_cm*20);
37.      noTone(11);
38.      delay(distance_cm*20);
39.    }
40. }
```