

Beste de savoir

Réaliser un télémètre à ultrasons

12 août 2019

Table des matières

1.	Rappel sur les ultrasons	1
1.1.	Une onde sonore c'est quoi ?	2
1.2.	Principe de la mesure	2
2.	Mise en œuvre du télémètre	3
2.1.	Le HC-SR04	3
2.2.	Exploitions le capteur avec Arduino	4
3.	Ajout de l'écran et montage final	6
3.1.	Faire fonctionner l'écran seul	7
3.2.	L'écran avec le télémètre	7
4.	Aller plus loin	9

Lorsque l'on veut mesurer des distances, on utilise en général une règle ou un ruban mètre. Mais comment faire pour avoir une distance numérique en temps réel ?

Une réponse : utiliser la technologie des ultrasons pour réaliser un **télémètre**.

Bien que peu précis à l'échelle des centimètres, ces derniers sont relativement fiables à l'échelle de la dizaine de centimètres. Les ultrasons étant des ondes sonores se propageant sous la forme d'un cône (ils sont peu directifs), ils font de très bons détecteurs d'obstacles.

Nous allons donc nous lancer dans la réalisation d'un petit télémètre à ultrasons via une Arduino et avec un affichage intégré sur un écran LCD (parce que la voie série, c'est un peu pénible pour se promener). Voici les prérequis pour suivre ce tutoriel :

- savoir se servir d'une Arduino ;
- savoir utiliser un écran LCD alphanumérique avec l'Arduino.

À la fin de ce tutoriel, vous serez en mesure de :

- utiliser un capteur HC-SR04 ;
- comprendre et utiliser une information de type "temps de vol" ;
- utiliser la fonction `pulseIn()` pour mesurer une durée d'état sur une broche.

1. Rappel sur les ultrasons

Faisons un peu de sciences et rappelons ce que sont des ultrasons.

Un ultrason est une onde sonore à haute fréquence. Par haute fréquence j'entends toutes les fréquences sonores inaudibles pour l'oreille humaine, soit celles au-delà de 20 kHz. Elles sont l'opposé des *infrasons* qui sont les ondes sonores dont la fréquence est inférieure à la plus faible audible pour l'Homme et qui est de 20 Hz.

Bon, c'est bien beau mais cette histoire de fréquence ça ne nous en dit pas beaucoup plus !

1. Rappel sur les ultrasons

1.1. Une onde sonore c'est quoi?

Une onde sonore est un phénomène physique de compression et décompression. Lorsqu'une vibration est produite (par n'importe quel objet qui vibre), l'air subit alors une onde de choc qui se traduit en mouvement des atomes. Il y a alors ce phénomène de compression et décompression (des "trous" dans l'air) que des récepteurs dans nos oreilles convertissent en bruit.

Plus les compressions sont proches et plus la fréquence est élevée. On parle alors de son aigu. Au contraire, plus les compressions sont éloignées et plus la fréquence est faible, on parle d'un son grave. Une fréquence s'exprime en Hertz et traduit la répétition d'un motif d'un phénomène durant une seconde. Par exemple si je cligne des yeux trois fois par seconde, on peut dire que je cligne des yeux à 3 Hz.

Dans le cas des ultrasons, les compressions/décompressions sont très courtes. En effet, le motif se répète plus de 20 000 fois par seconde, donc à plus de 20 kHz. En général, en électronique on utilise un *transducteur piézo* pour générer cela. C'est une sorte de petit buzzer capable de vibrer très vite. Très souvent, les télémètres à ultrasons vibrent à une fréquence de 40 kHz.

Une dernière caractéristique des ondes sonores est leur capacité à être réfléchiée par les obstacles. En effet, les ondes sonores ont tendance à "rebondir" sur les obstacles. On entend alors l'onde de départ et un peu plus tard la même avec un retard et une plus faible intensité. C'est exactement le même phénomène qu'un écho dans une pièce vide ou en montagne. L'onde sonore se déplace, rebondit sur les murs lisses et revient à votre oreille avec un retard entre le moment où vous avez parlé et celui où vous l'entendez (et une puissance sonore plus faible). En général, dans le domaine de l'acoustique et de la musique, on cherche à supprimer cette caractéristique en recouvrant les murs de matériaux spéciaux. Cependant, dans le cas d'une mesure de distance, on va exploiter cet effet.

1.2. Principe de la mesure

Comme dit précédemment, on va tirer parti du fait que l'onde sonore rebondit sur les obstacles et revient souvent vers l'expéditeur. On va aussi exploiter une autre chose connue, sa vitesse!

En effet, la vitesse de déplacement d'une onde sonore dans l'air est connue depuis longtemps. Elle est d'environ 340 mètres par seconde à 25 degrés Celsius (plutôt lent comparé à la lumière et ses 300 000 km/s). À partir de là, si on sait quand l'onde est partie et quand on la reçoit de nouveau (après le rebond), on est en mesure de calculer un *temps de vol* de l'onde. On a alors une durée, une vitesse, et on peut en déduire une distance!

Comme l'onde fait un aller-retour (le voyage depuis l'émission de l'onde, le rebond, puis le retour sur le récepteur), il faudra diviser le temps de vol par deux pour ne considérer qu'un trajet (l'aller ou le retour). Le calcul sera alors simple. Une vitesse s'exprime par une distance divisée par un temps $v = d/t$ donc la distance sera la vitesse multipliée par le temps $d = v \times t$.

Passons un peu à la pratique pour mieux comprendre!



FIGURE 1. – Un capteur à ultrasons

2. Mise en œuvre du télémètre

Le télémètre que nous allons utiliser est assez simple. Son nom est HC-SR04 et existe en différentes variations. Découvrons-le et voyons comment le faire fonctionner avec notre Arduino.

2.1. Le HC-SR04

2.1.1. Présentation du composant

Le HC-SR04 est ce que l'on appelle communément un "Télémètre à ultrasons". Il est trouvable relativement facilement sur de nombreux sites de fournisseurs de composants électroniques. Les sites livrant depuis la Chine proposent des prix inférieurs à 5 euros en général.

Ce composant possède plusieurs petites choses. Tout d'abord, sur la face avant on peut voir l'émetteur US et son récepteur. Ce sont des petites cellules piézo-électriques qui vont soit vibrer lorsqu'une tension est appliquée (émetteur), soit au contraire produire une tension lorsque une vibration est reçue (récepteur).

Sur la face arrière on trouve plusieurs petits circuits permettant la génération du signal et le traitement de ce dernier. Ainsi, un composant va générer une onde de 40 kHz lors d'un "top départ" et la partie restante s'occupera de la mise en forme de la réception (amplification et filtrage) et de mettre en forme cela proprement sur une broche de sortie. Parlons d'ailleurs des broches. On en trouve 4.

Les premières sont comme toujours VCC et GND qui vont accueillir l'alimentation (respectivement 5V et masse). On trouve ensuite la broche "echo" sur laquelle sera présent le signal de sortie. Enfin, une broche nommée "Trig". Cela signifie "Trigger" soit "déclencheur" ou "gâchette". En mettant cette broche à l'état haut pendant 10µs vous allez déclencher le *ping* pour la mesure. Un "ping" représente le lancement d'une onde ultrason. Pour reprendre l'exemple de l'écho dans la pièce vide, le ping correspondrait au moment où vous émettez un son en parlant.

2. Mise en œuvre du télémètre

Le signal de sortie est assez simple à exploiter. Il est initialement à 0, puis passe à 1 lorsque le *ping* est envoyé. Il repasse ensuite à 0 quand l'écho est revenu au récepteur OU s'il n'y a pas de retour durant les 30ms après l'envoi (l'onde est alors considérée perdue).

2.1.2. Branchement

Les branchements sont eux-même assez simples. Il suffira de relier 5V et GND à leurs broches respectives sur Arduino et mettre "Trig" et "Echo" sur des I/O numériques (8 et 9 par exemple). Pas la peine d'un schéma pour cela !

2.2. Exploitions le capteur avec Arduino

Passons maintenant à la pratique avec Arduino. Le but ici sera d'être capable de faire une mesure de distance puis de l'afficher en millimètres dans la console de la voie série.

2.2.1. Setup

On va comme toujours commencer par le setup. Pour une fois, pas de bibliothèque externe à rajouter, tout sera fait "à la main". Comme vu plus tôt, nous allons utiliser deux broches que je vais très justement nommer "USTrig" et "USEcho", pour le déclencheur (une sortie numérique) et le retour d'informations (une entrée numérique). Je vais ensuite préparer la voie série pour afficher les mesures.

```
1  const int USTrig = 8; // Déclencheur sur la broche 8
2  const int USEcho = 9; // Réception sur la broche 9
3
4  void setup() {
5      pinMode(USTrig, OUTPUT);
6      pinMode(USEcho, INPUT);
7
8      digitalWrite(USTrig, LOW);
9
10     Serial.begin(9600);
11 }
```

Et c'est tout pour le setup !

2.2.2. Loop

Les choses faciles sont réglées, passons aux choses amusantes et faisons des mesures !

Comme je le disais dans la présentation du composant, il y a une suite d'actions à faire pour mesurer la distance. Schématiquement cela se traduirait par la liste suivante :

1. Un état haut de 10 microsecondes est mis sur la broche "Trig"

2. Mise en œuvre du télémètre

2. On remet à l'état bas la broche Trig
3. On lit la durée d'état haut sur la broche "Echo"
4. On divise cette durée par deux pour n'avoir qu'un trajet
5. On calcule la distance avec la formule $d = v \times t$
6. On affiche la distance

Toutes ces étapes sont en fait assez simples, à part peut-être la mesure de la durée. Heureusement, une fonction nommée `pulseIn()` [↗](#) est là pour ça. Cette dernière, qui est **bloquante**, se chargera de mesurer combien de temps une broche reste dans un état (HIGH or LOW). Elle prend en paramètre le numéro de la broche et l'état à observer.

Voici alors un exemple de programme que l'on obtient :

```
1 #define VITESSE 340 //vitesse du son 340 m/s
2
3 // setup()
4
5 void loop()
6 {
7     // 1. Un état haut de 10 microsecondes est mis sur la broche
8     //    "Trig"
9     digitalWrite(USTrig, HIGH);
10    delayMicroseconds(10); //on attend 10 µs
11    // 2. On remet à l'état bas la broche Trig
12    digitalWrite(USTrig, LOW);
13
14    // 3. On lit la durée d'état haut sur la broche "Echo"
15    unsigned long duree = pulseIn(USEcho, HIGH);
16
17    if(duree > 30000)
18    {
19        // si la durée est supérieure à 30ms, l'onde est perdue
20        Serial.println("Onde perdue, mesure échouée !");
21    }
22    else
23    {
24        // 4. On divise cette durée par deux pour n'avoir qu'un
25        //    trajet
26        duree = duree/2;
27
28        // 5. On calcule la distance avec la formule d=v*t
29        float temps = duree/1000000.0; //on met en secondes
30        float distance = temps*VITESSE; //on multiplie par la
31        //    vitesse, d=t*v
32
33        // 6. On affiche la distance
34        Serial.print("Duree = ");
35        Serial.println(temps); //affiche le temps de vol d'un trajet
36        //    en secondes
```

3. Ajout de l'écran et montage final

```
33     Serial.print("Distance = ");
34     Serial.println(distance); //affiche la distance mesurée (en
        mètres)
35 }
36
37 // petite pause
38 delay(250);
39 }
```

Si vous voulez obtenir la distance en millimètres, il suffira de multiplier par 1000. Mais soyons malins, nous pouvons aussi optimiser en évitant une opération. Pour cela c'est simple, la ligne calculant le temps en seconde passe de :

```
1 float temps = duree/1000000.0;
```

à

```
1 float temps = duree/1000.0;
```

puisque multiplier la distance par 1000 dans notre situation revient exactement à avoir un temps divisé par le même facteur (pour avoir l'information en centimètres, il aurait fallu utiliser un facteur 100 plutôt que 1000). Je vous laisse poser les maths si vous voulez me croire sur parole .

i

Si les mesures ne sont pas ultra-précises, essayez de modifier la constante de vitesse. Les ondes sont sensibles à la température ce qui influe sur leur vitesse. Nous verrons cela plus tard .

3. Ajout de l'écran et montage final

Tout va bien ? Les échos reviennent ? Alors place à la portabilité ! Dans cette dernière étape nous allons rajouter l'écran à notre nouvel outil.

Pour rajouter l'écran, commençons par vérifier le câblage nécessaire. L'écran nécessite 4 (ou 8 selon vos choix) fils de données (+ 2 de contrôle) et une alimentation. Cette dernière sera prise sur les broches 5V et GND de l'Arduino et les fils de données seront sur les broches 2 à 5. Les fils de contrôle EN et R/W seront sur les broches 6 et 7.

3. Ajout de l'écran et montage final

3.1. Faire fonctionner l'écran seul

En science, lorsque l'on veut tester quelque chose on l'isole du reste. Ici c'est pareil! Nous avons pu constater que notre télémètre fonctionne bien en le faisant marcher seul, nous allons faire de même en utilisant l'écran seul. Pour cela, nous allons simplement afficher une ligne de caractères.

On commence logiquement par le setup et ce qui va avec, notamment la bibliothèque `LiquidCrystal` et un objet du même nom.

```
1 #include "LiquidCrystal.h"
2
3 // VÉRIFIEZ AVEC VOS PROPRES BROCHES
4 LiquidCrystal lcd(6,7,2,3,4,5); //liaison 4 bits de données
5
6 void setup() {
7     lcd.begin(2, 16); // Initialisation de l'écran (2 lignes 16
8     // caractères pour ma part)
9 }
```

Une fois cela fait, on va afficher une simple ligne dans notre programme principal :

```
1 void loop() {
2     lcd.clear();
3     lcd.home(); // Replace le curseur en haut à gauche
4     lcd.print("-+ Eskimetre +-");
5     delay(10000); //pause de 10 secondes inutile, juste pour éviter
6     // de rafraîchir sans arrêt
7 }
```

Si le message s'affiche correctement, alors tout va bien et vous pouvez continuer!

3.2. L'écran avec le télémètre

Nous y voilà! Le moment tant attendu où nous allons afficher les mesures sur l'écran! Pour cela, commencez par rajouter sur votre montage le télémètre (broche 8 et 9) et fusionner les setup pour n'en faire qu'un sans la voie série. Vous devriez obtenir quelque chose comme ça :

```
1 #include "LiquidCrystal.h"
2 #define VITESSE 340
3
4 // VÉRIFIEZ AVEC VOS PROPRES BROCHES
5 LiquidCrystal lcd(6,7,2,3,4,5); //liaison 4 bits de données
6 const int USTrig = 8; // Déclencheur sur la broche 8
```

3. Ajout de l'écran et montage final

```
7  const int USEcho = 9; // Réception sur la broche 9
8
9  void setup() {
10     lcd.begin(2, 16); // Initialisation de l'écran (2 lignes 16
        caractères pour ma part)
11
12     pinMode(USTrig, OUTPUT);
13     pinMode(USEcho, INPUT);
14
15     lcd.print("-+ Eskimetre +-");
16 }
```

Il faut ensuite fusionner le tout pour faire un affichage sur le LCD. Je vous propose d'exploiter les deux lignes pour afficher :

- En haut : la distance en millimètres (`int`)
- En bas : le temps en millisecondes (`int`)

```
1  void loop() {
2     // 1. Un état haut de 10 microsecondes est mis sur la broche
        "Trig"
3     digitalWrite(USTrig, HIGH);
4     delayMicroseconds(10); //on attend 10 µs
5     // 2. On remet à l'état bas la broche Trig
6     digitalWrite(USTrig, LOW);
7
8     // 3. On lit la durée d'état haut sur la broche "Echo"
9     unsigned long duree = pulseIn(USEcho, HIGH);
10
11    if(duree > 30000)
12    {
13        //si la durée est supérieure à 30ms, l'onde est perdue
14        lcd.clear();
15        lcd.home(); // Replace le curseur en haut à gauche
16        lcd.print("Onde perdue :(");
17        lcd.setCursor(0,1);
18        lcd.print("Mesure echouee !");
19    }
20    else
21    {
22        // 4. On divise cette durée par deux pour n'avoir qu'un
            trajet
23        duree = duree/2;
24
25        // 5. On calcul la distance avec la formule d=v*t
26        unsigned int distance = duree*(VITESSE/1000.0); //on
            multiplie par la vitesse, d=t*v
27    }
```

4. Aller plus loin

```
28     // 6. On affiche !
29     lcd.clear();
30     lcd.home(); // Replace le curseur en haut à gauche
31     char message[16] = "";
32     sprintf(message, " Dist : %4d mm", distance);
33     lcd.print(message);
34     sprintf(message, "Temps : %4d us", duree);
35     lcd.setCursor(0,1);
36     lcd.print(message);
37 }
38 delay(250); // Petite pause
39 }
```

Il est pas beau ce télémètre ? Voici ce que ça donne chez moi :

ÉLÉMENT EXTERNE (VIDEO) —

Consultez cet élément à l'adresse <https://www.youtube.com/embed/n-fqDFQj6Bo?feature=oembed>.

Mon télémètre

Et voici une version sur simulateur <https://www.tinkercad.com/embed/3NbcDdiiH1v> ↗

!(<https://www.tinkercad.com/embed/3NbcDdiiH1v>)

4. Aller plus loin

Comme je l'ai brièvement mentionné plus tôt, la température a un impact sur la vitesse des ondes (ainsi que la pression et d'autres choses). Une amélioration de notre télémètre serait donc d'avoir une correction de la vitesse via une mesure de la température !

Pour débiter, prenons connaissance des données. [Wikipédia nous dit](#) ↗ que la vitesse de propagation du son dans l'air suit à peu près le tableau suivant :

4. Aller plus loin

θ en °C	c en m·s ⁻¹	ρ en kg·m ⁻³	Z en N·s·m ⁻³
- 10	325,4	1,341	436,5
- 5	328,5	1,316	432,4
0	331,5	1,293	428,3
+ 5	334,5	1,269	424,5
+ 10	337,5	1,247	420,7
+ 15	340,5	1,225	417,0
+ 20	343,4	1,204	413,5
+ 25	346,3	1,184	410,0
+ 30	349,2	1,164	406,6

FIGURE 4. – Vitesse du son en fonction de la température

Si l'on fait un graphique avec ce tableau, on peut déduire les valeurs de la relation $V = f(t)$ (vitesse en fonction de la température) qui est presque linéaire et donc peut être modélisée par une fonction affine. En faisant le calcul de la variation via les deux points les plus extrêmes du bout de droite, on obtient un coefficient de 0.6 m/s par degrés Celsius ($\frac{325.4-349.2}{-10-30}$). On observe aussi une ordonnée à l'origine (soit à 0 degré) de 331.5 mètres par seconde.

Tout cela nous donne la relation : $V = at + b$ (avec V la vitesse en mètres par seconde et t la température en degrés Celsius). a vaudra ainsi 0.6 et b vaudra 331.5.

Je ne donnerai pas le détail du montage ni le code ici cependant. Je vous le laisse en tant qu'évolution/exercice si vous le souhaitez, en rajoutant un capteur de température au montage (un LM35 par exemple). Voici cependant la logique de code qu'il faudrait suivre pour implémenter cela :

```
1 void loop() {
2     // On fait la mesure de température
3
4     // On stocke la température en degrés Celsius dans "degres".
5     float degres;
6
7     // puis on ajuste la vitesse en fonction de la température
8     float vitesse = 0.6*degres + 331.5;
9
10    // et enfin, on fait tout le reste comme avant (mesure +
        affichage)
```

4. *Aller plus loin*

```
11     // La seule différence sera que l'on utilisera la variable
12     //     "vitesse" ...
13     // ... au lieu de la constante VITESSE
13 }
```

Vous avez maintenant un outil de mesure complètement fonctionnel. Il suffit de fabriquer un boîtier et de rajouter une batterie ou quelques piles pour avoir un module totalement autonome ! On pourrait aussi pousser le vice en utilisant une Arduino mini ou micro pour réduire encore un peu plus la taille du produit final !