

Mesurer une distance avec un capteur à ultrason HC-SR04 et une carte Arduino / Genuino

Haaaa les tests en extérieur



par [skywodd](#) | | Licence (voir pied de page)

Catégories : [Tutoriels Arduino](#) | Mots clefs : [Arduino](#) [Genuino](#) [Capteur](#) [Distance](#) [Sonar](#) [Ultrason](#)

Cet article n'a pas été mis à jour depuis un certain temps, son contenu n'est peut être plus d'actualité.

Dans ce tutoriel, nous allons apprendre ensemble à utiliser un capteur de distance à ultrason de référence HC-SR04 avec une carte Arduino / Genuino. En bonus, nous testerons la précision des mesures dans diverses situations.

Sommaire

- [Comment mesurer une distance](#)
- [Le capteur HC-SR04](#)
- [Principe de fonctionnement du capteur](#)
- [Le montage](#)
- [Le code](#)
- [Bonus : La précision du capteur](#)
 - [Test en intérieur avec un petit obstacle](#)
 - [Test en extérieur dans des conditions idéales](#)
 - [Résultats des tests](#)
- [Conclusion](#)

Bonjour à toutes et à tous !

Parfois quand on réalise un projet, on a besoin de mesurer des distances, détecter des obstacles, etc.

En robotique par exemple, il est très classique d'avoir un capteur de distance sur l'avant du robot pour éviter de se prendre un mur en pleine face.

Comment mesurer une distance

À l'anglaise : avec votre pouce, votre pied, ou n'importe quoi en fait. God save the queen.

Pour mesurer des distances, il faut un capteur de distance (s'en blague !). Il existe sur le marché un grand nombre de capteurs de distance : infrarouge (réflectif), laser (par temps de parcours ou par calcul d'angle), physique (règles optiques absolues ou incrémentielles), ou ultrason.

- Les capteurs infrarouges ont l'avantage d'être bon marché, relativement précis et disponibles à peu près partout. Malheureusement, ils sont assez complexes à mettre en oeuvre du fait de leurs non-linéarités. Il faut appliquer une formule complexe pour obtenir une mesure utilisable. De plus, ils sont très sensibles à la lumière ambiante et au coefficient de réflexion lumineuse de la surface en face du capteur.
- Les (vrais) capteurs de distance laser sont extrêmement précis, mais aussi extrêmement chers. Un capteur de distance laser (par mesure de temps de parcours) coûte facilement plus de 200€, mais fait des mesures à plus de 30 mètres sans problème pour certains modèles. C'est donc au final une question de budget / utilisation.

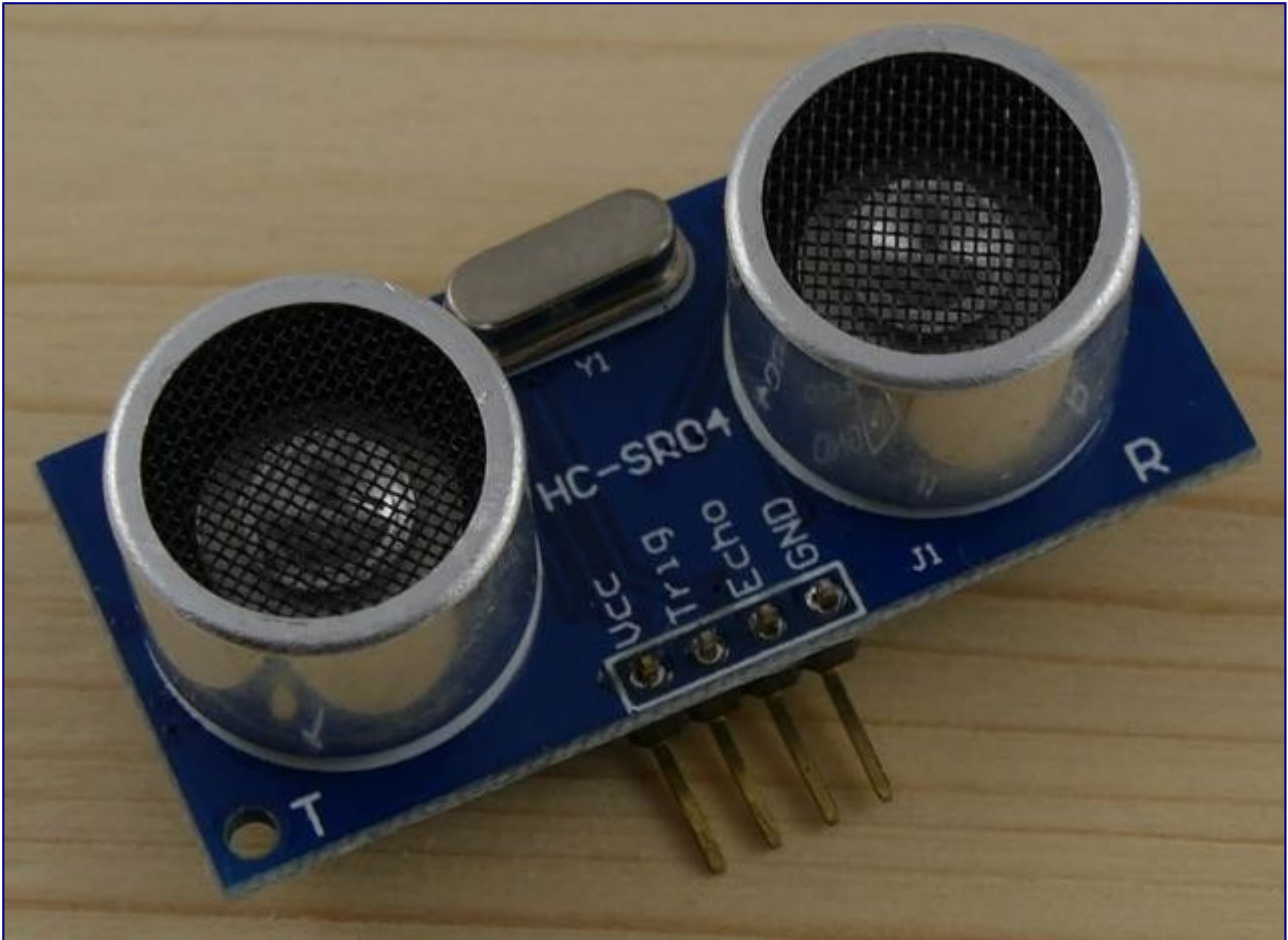
PS Il existe des (faux) capteurs de distance laser fonctionnant par triangulation. Au lieu de mesurer le temps d'aller-retour d'un faisceau laser, ces modules calculent l'angle entre le point du laser et le capteur. Ces modules sont moins chers, mais aussi beaucoup moins précis.

- Les capteurs physiques, le plus souvent un duo comportant une règle graduée et un capteur optique, sont à la fois bon marché et très précis. Mais ils sont très limités en distance mesurable et se retrouvent donc généralement dans des imprimantes.

Reste les capteurs ultrasons, et ça tombe bien, c'est le sujet de cet article.

Un capteur de distance à ultrason utilise le même principe qu'un capteur laser, mais en utilisant des ondes sonores (inaudible) au lieu d'un faisceau de lumière. Ils sont bien moins chers qu'un capteur laser, mais aussi bien moins précis. Cependant, contrairement aux capteurs à infrarouge, la lumière ambiante et l'opacité de la surface en face du capteur ne jouent pas sur la mesure.

Le capteur HC-SR04



Capteur à ultrason HC-SR04

Le capteur qui nous intéresse dans ce tutoriel est un capteur à ultrason made in chinois, bien connu des amateurs de robotique et d'Arduino : le HC-SR04 (aussi disponible sous d'autres références en fonction du vendeur).

Le [capteur HC-SR04](#) est un capteur à ultrason low cost. Ce capteur fonctionne avec une tension d'alimentation de 5 volts, dispose d'un angle de mesure de 15° environ et permet de faire des mesures de distance entre 2 centimètres et 4 mètres avec une précision de 3mm (en théorie, dans la pratique ce n'est pas tout à fait exact).

N.B. Il existe des capteurs à ultrason bien plus haut de gamme (donc précis). Un capteur à ultrason monocapsule de qualité coûte entre 20€ et 30€. Un capteur HC-SR04 revient seulement à 3€, frais de port inclus en import de Chine. Ce capteur est donc clairement "low cost", mais pas mauvais pour autant comme on le verra plus tard.

Principe de fonctionnement du capteur

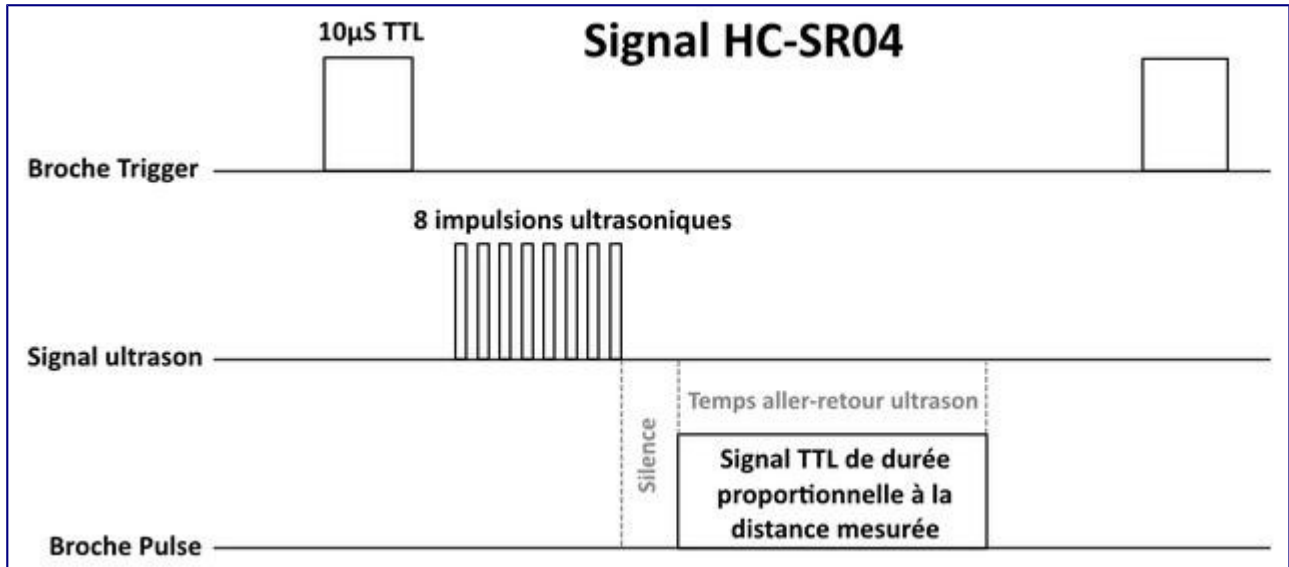


Illustration du signal TRIGGER et ECHO

Le principe de fonctionnement du capteur est entièrement basé sur [la vitesse du son](#).

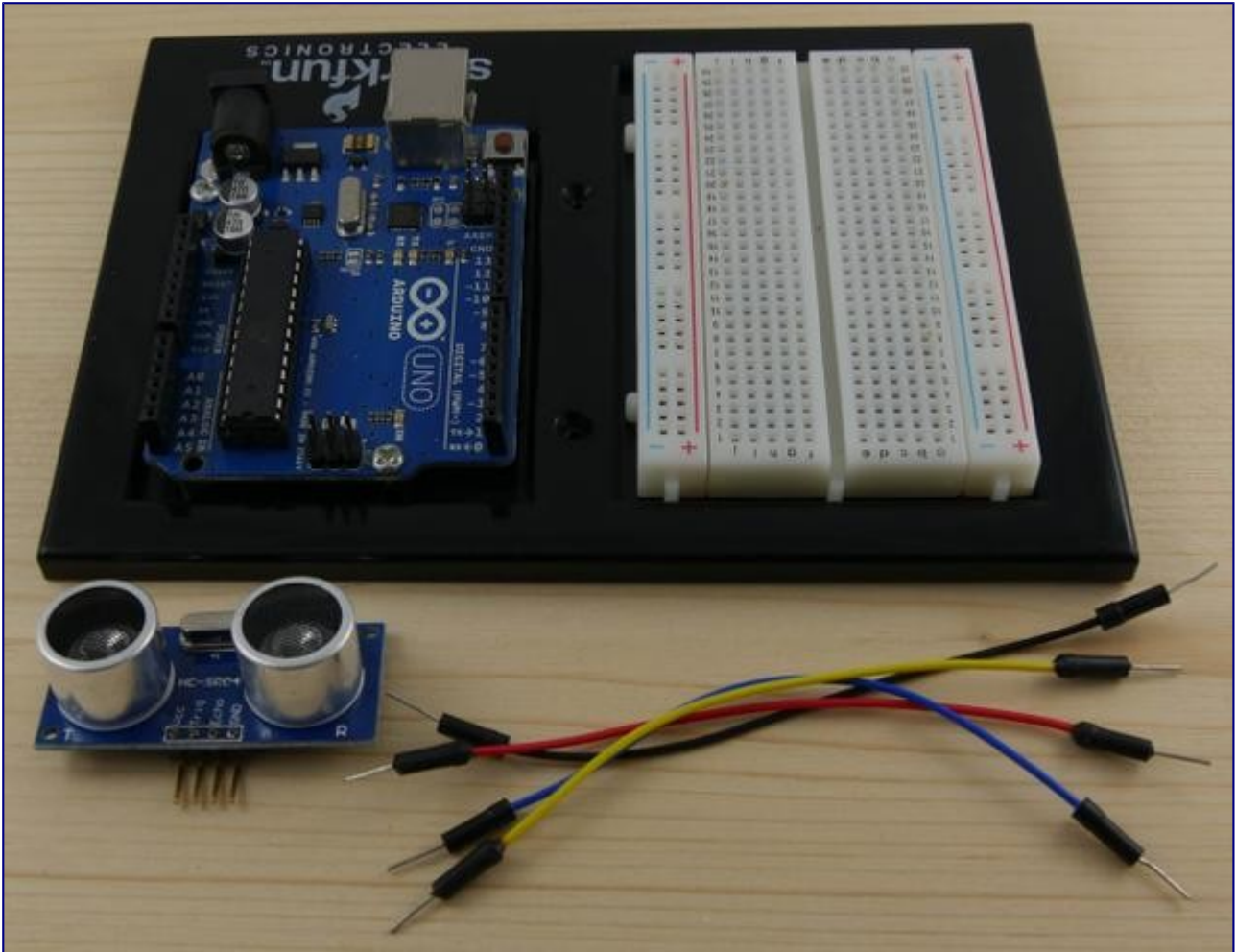
Voilà comment se déroule une prise de mesure :

1. On envoie une impulsion HIGH de 10µs sur la broche TRIGGER du capteur.
2. Le capteur envoie alors une série de 8 impulsions ultrasoniques à 40KHz (inaudible pour l'être humain, c'est quand plus agréable qu'un biiiiiiip).
3. Les ultrasons se propagent dans l'air jusqu'à toucher un obstacle et retournent dans l'autre sens vers le capteur.
4. Le capteur détecte l'écho et clôture la prise de mesure.

Le signal sur la broche ECHO du capteur reste à HIGH durant les étapes 3 et 4, ce qui permet de mesurer la durée de l'aller-retour des ultrasons et donc de déterminer la distance.

N.B. Il y a toujours un silence de durée fixe après l'émission des ultrasons pour éviter de recevoir prématurément un écho en provenance directement du capteur.

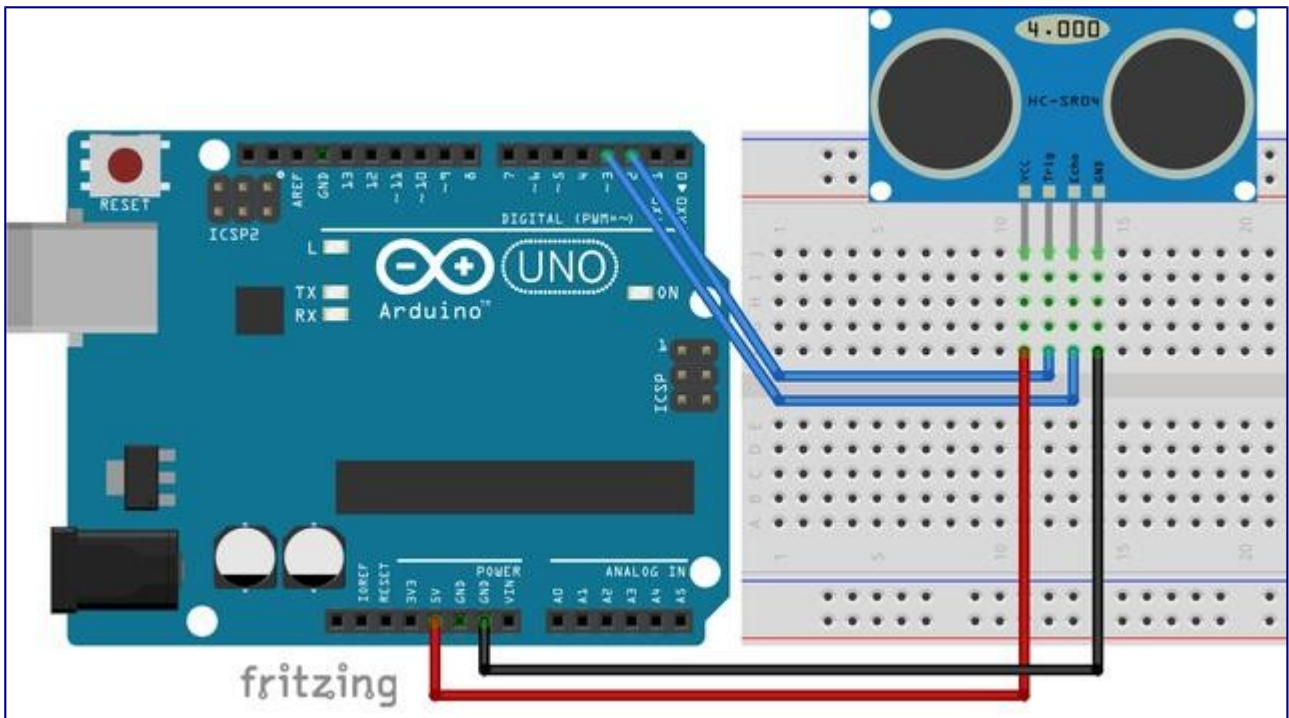
Le montage



Matériel nécessaire

Pour réaliser ce premier montage, il va nous falloir :

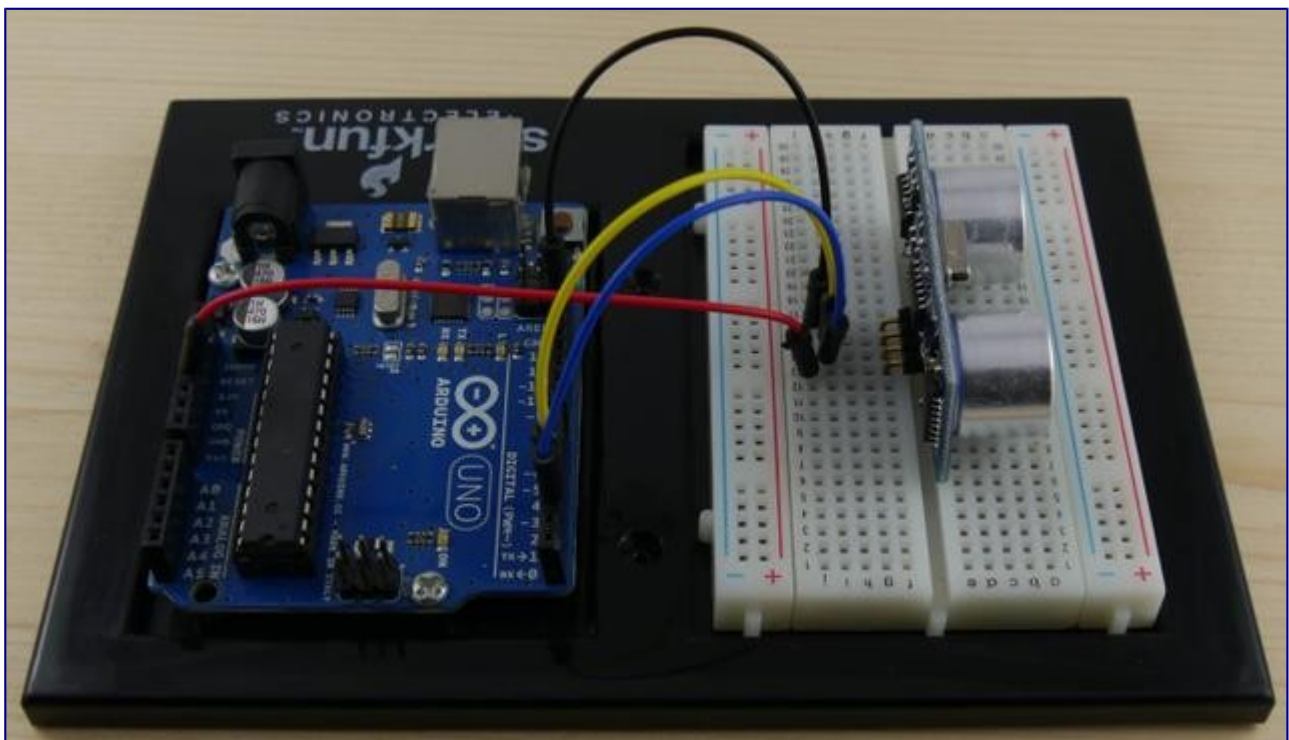
- Une carte Arduino UNO (et son câble USB),
- Un capteur HC-SR04,
- Une plaque d'essai et des fils pour câbler notre montage.



Vue prototypage du montage

Le montage est d'une simplicité déconcertante :

- L'alimentation 5V de la carte Arduino va sur la broche VCC du capteur.
- La broche GND de la carte Arduino va sur la broche GND du capteur.
- La broche D2 de la carte Arduino va sur la broche TRIGGER du capteur.
- La broche D3 de la carte Arduino va sur la broche ECHO du capteur.



Le montage fini

Vous pouvez choisir d'utiliser d'autres broches que D2 et D3 si vous le souhaitez. Il suffira de mettre à jour les numéros de broches dans le code du chapitre suivant.

N.B. La plaque d'essai est ici totalement optionnelle. Si vous avez des fils mâles / femelles, vous pouvez directement câbler le capteur à la carte Arduino.

Le code

Au final, le plus compliqué dans ce tutoriel, c'est le code 😁

```
1      /* Constantes pour les broches */
2      const byte TRIGGER_PIN = 2; // Broche TRIGGER
3
4      /* Constantes pour le timeout */
5      const unsigned long MEASURE_TIMEOUT = 25000UL; // 25ms = ~8m à
6      340m/s
7
8      /* Vitesse du son dans l'air en mm/us */
9      const float SOUND_SPEED = 340.0 / 1000;
```

On commence le code avec quatre constantes : deux constantes pour les broches TRIGGER et ECHO du capteur, une constante qui servira de timeout pour la prise de mesure et une constante pour définir la vitesse du son.

Le timeout correspond au temps nécessaire avant de considérer qu'il n'y a pas d'obstacle, donc pas de mesure possible. J'ai choisi d'utiliser une timeout de 25 millisecondes (4 mètres aller-retour à 340m/s).

N.B. Vous remarquerez que j'ai déclaré la vitesse du son en millimètres par microseconde. Cela est nécessaire, car la mesure du temps se fait en microsecondes et je souhaite avoir un résultat en millimètres en sortie du calcul.

```
1      void setup() {
2
3          /* Initialise le port série */
4          Serial.begin(115200);
5
6          /* Initialise les broches */
7          pinMode(TRIGGER_PIN, OUTPUT);
8          digitalWrite(TRIGGER_PIN, LOW); // La broche TRIGGER doit être
9          à LOW au repos
10         pinMode(ECHO_PIN, INPUT);
11     }
```

La fonction `setup()` initialise le port série, met la broche TRIGGER du capteur en sortie et à LOW, et met la broche ECHO du capteur en entrée. Rien de bien palpitant.

```

1 void loop() {
2     /* 1. Lance une mesure de distance en envoyant une impulsion
3     HIGH de 10µs sur la broche TRIGGER */
4     digitalWrite(TRIGGER_PIN, HIGH);
5     delayMicroseconds(10);
6     digitalWrite(TRIGGER_PIN, LOW);
7
8     /* 2. Mesure le temps entre l'envoi de l'impulsion ultrasonique
9     et son écho (si il existe) */
10    long measure = pulseIn(ECHO_PIN, HIGH, MEASURE_TIMEOUT);
11
12    /* 3. Calcul la distance à partir du temps mesuré */
13    float distance_mm = measure / 2.0 * SOUND_SPEED;
14
15    /* Affiche les résultats en mm, cm et m */
16    Serial.print(F("Distance: "));
17    Serial.print(distance_mm);
18    Serial.print(F("mm ("));
19    Serial.print(distance_mm / 10.0, 2);
20    Serial.print(F("cm, "));
21    Serial.print(distance_mm / 1000.0, 2);
22    Serial.println(F("m"));
23
24    /* Délai d'attente pour éviter d'afficher trop de résultats à
25    la seconde */
26    delay(500);
27 }

```

La fonction `loop()` s'occupe de la mesure et de l'affichage.

Elle génère d'abord l'impulsion HIGH de 10µs qui déclenche la prise de mesure. Elle mesure ensuite le temps nécessaire a pour un aller-retour du signal ultrason avec la fonction [pulseIn\(\)](#). Pour finir, elle calcule la distance avant de l'afficher sur le port série.

N.B. La fonction `pulseIn()` retourne 0 si le temps de timeout est atteint. Il est donc possible de gérer l'absence d'obstacle si vous le souhaitez avec un `if (measure == 0) { ... }` par exemple.

PS La valeur retournée par `pulseIn()` doit être divisée par deux avant de faire le calcul de distance. Un aller-retour est égal à deux fois la distance mesurée.

Le code complet avec commentaires :

```

1     /*
2     * Code d'exemple pour un capteur à ultrasons HC-SR04.
3     */
4
5     /* Constantes pour les broches */
6     const byte TRIGGER_PIN = 2; // Broche TRIGGER
7     const byte ECHO_PIN = 3; // Broche ECHO

```



```

8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52

```

```

/* Constantes pour le timeout */
const unsigned long MEASURE_TIMEOUT = 25000UL; // 25ms = ~8m à
340m/s

/* Vitesse du son dans l'air en mm/us */
const float SOUND_SPEED = 340.0 / 1000;

/** Fonction setup() */
void setup() {
    /* Initialise le port série */
    Serial.begin(115200);

    /* Initialise les broches */
    pinMode(TRIGGER_PIN, OUTPUT);
    digitalWrite(TRIGGER_PIN, LOW); // La broche TRIGGER doit être
à LOW au repos
    pinMode(ECHO_PIN, INPUT);
}

/** Fonction loop() */
void loop() {
    /* 1. Lance une mesure de distance en envoyant une impulsion
HIGH de 10µs sur la broche TRIGGER */
    digitalWrite(TRIGGER_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIGGER_PIN, LOW);

    /* 2. Mesure le temps entre l'envoi de l'impulsion ultrasonique
et son écho (si il existe) */
    long measure = pulseIn(ECHO_PIN, HIGH, MEASURE_TIMEOUT);

    /* 3. Calcul la distance à partir du temps mesuré */
    float distance_mm = measure / 2.0 * SOUND_SPEED;

    /* Affiche les résultats en mm, cm et m */
    Serial.print(F("Distance: "));
    Serial.print(distance_mm);
    Serial.print(F("mm ("));
    Serial.print(distance_mm / 10.0, 2);
    Serial.print(F("cm, "));
    Serial.print(distance_mm / 1000.0, 2);
    Serial.println(F("m"));

    /* Délai d'attente pour éviter d'afficher trop de résultats à
la seconde */
    delay(500);
}

```

L'extrait de code ci-dessus est disponible en téléchargement sur [cette page](#) (le lien de téléchargement en .zip contient le projet Arduino prêt à l'emploi).

Bonus : La précision du capteur

Pour terminer ce tutoriel en beauté, je vous propose un petit chapitre bonus très scientifique (ou pas). L'idée est de voir si ce capteur à 3€ donne des mesures précises, ou non.

Test en intérieur avec un petit obstacle



Banc de test en intérieur

Commençons les tests avec une configuration (malheureusement, voir encadré plus bas) très classique : un montage en intérieur, à ras du sol ou sur une table, avec un obstacle de petite taille (ici un simple morceau de bois).

C'est la pire situation possible pour ce type de capteurs à ultrason. La surface en dessous du capteur génère des échos intempestifs. L'obstacle est bien trop petit pour donner des échos "propres" et de multiples obstacles se trouvent sur les côtés de la zone de mesure.