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

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 .

Sommaire

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

Pour mesurer des distances, il faut un capteur de distance. 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.

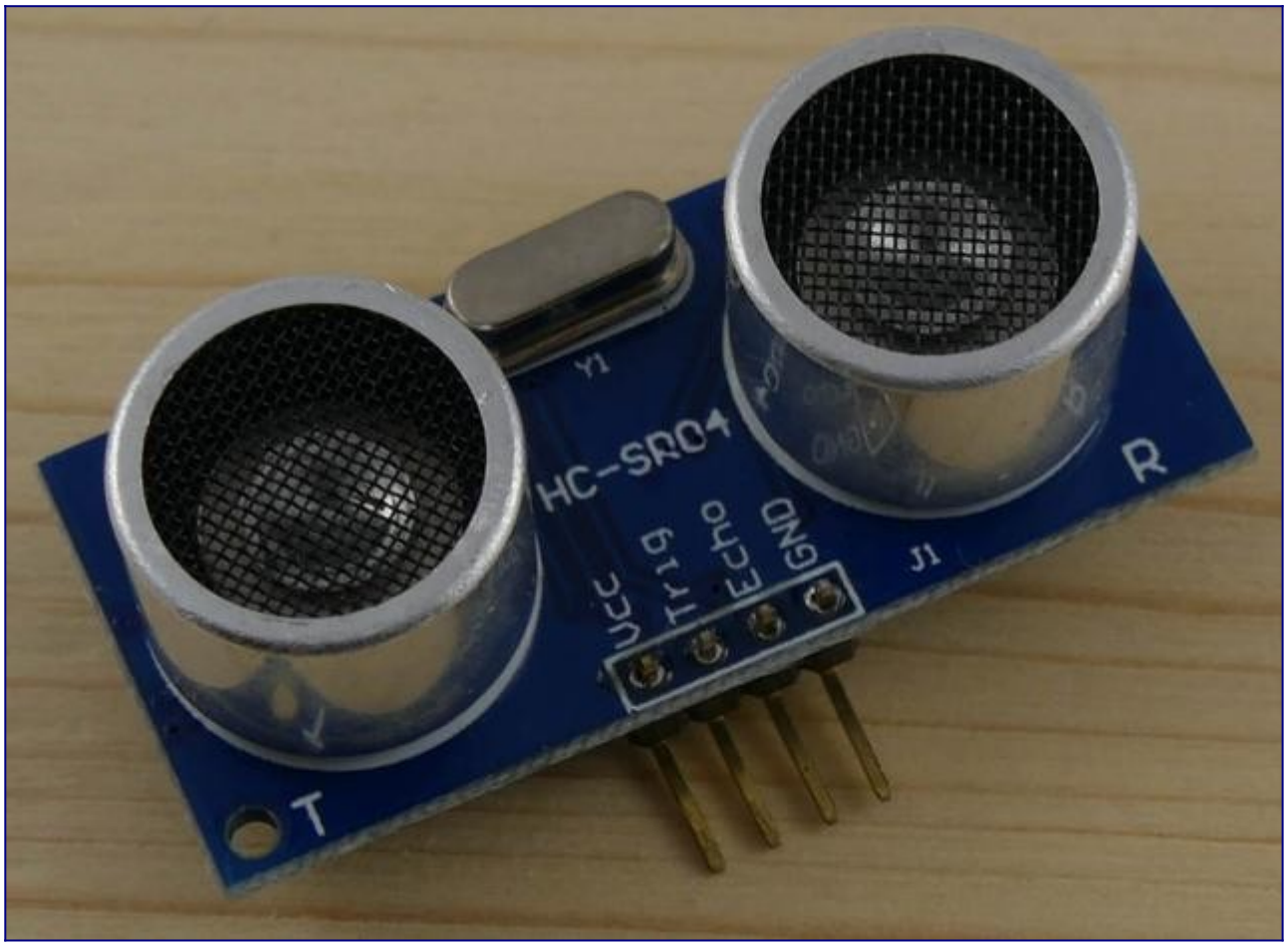
- Les capteurs infrarouges ont l'avantage d'être bon marché, relativement précis et disponibles à peu près partout. Malheureusement, ils sont assez complexes à mettre en oeuvre du fait de leurs non-linéarités. Il faut appliquer une formule complexe pour obtenir une mesure utilisable. De plus, ils sont 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.

  . 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, 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 , 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).

Principe de fonctionnement du capteur

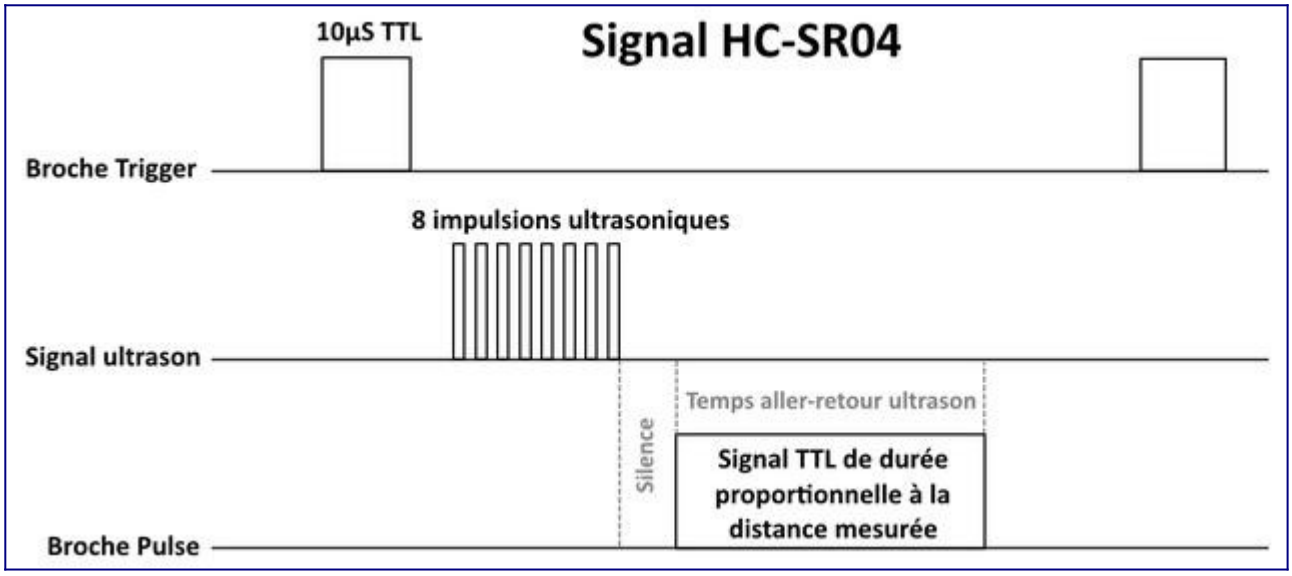


Illustration du signal TRIGGER et ECHO

Le principe de fonctionnement du capteur est entièrement basé sur la vitesse du son.

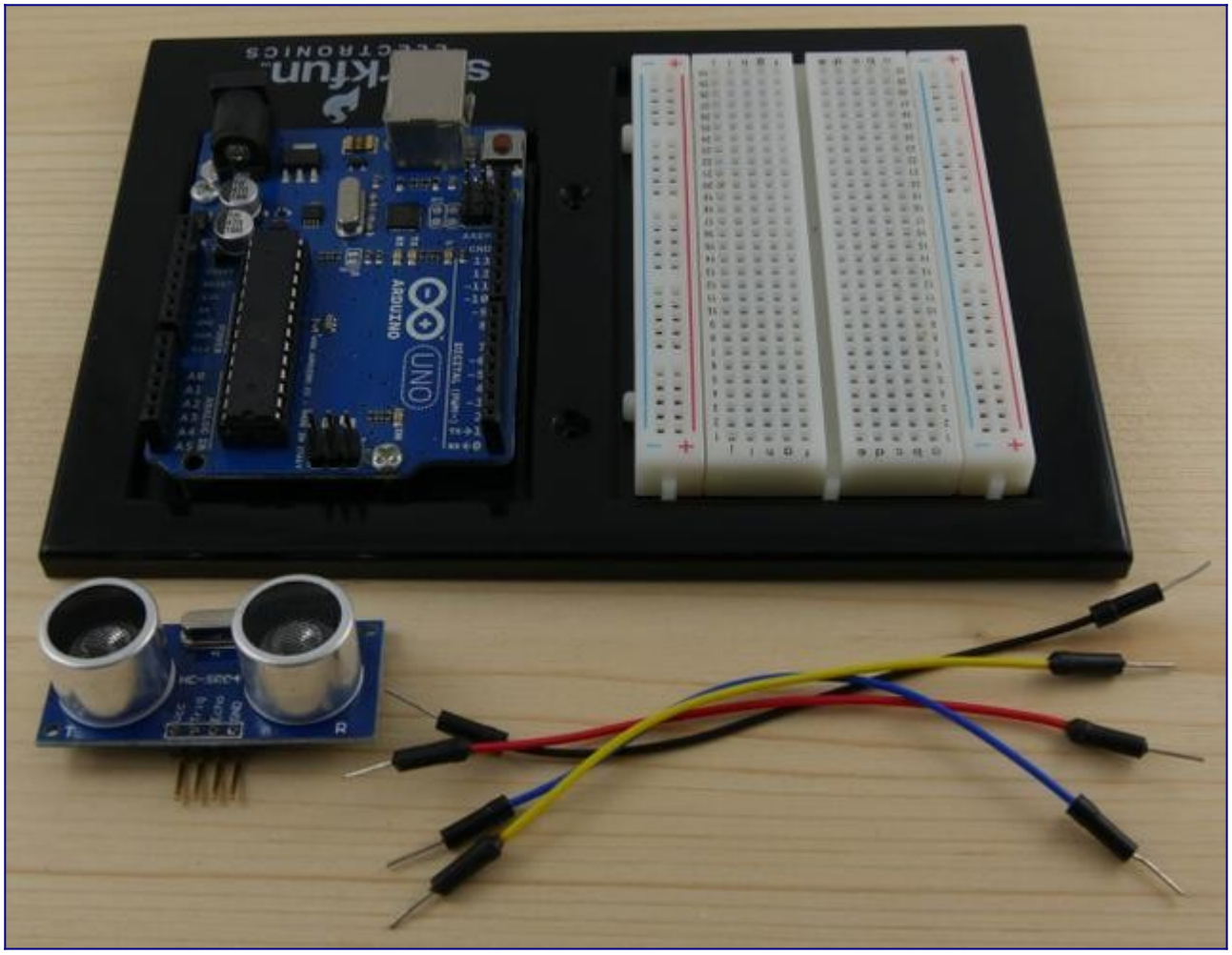
Voilà comment se déroule une prise de mesure :

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 biiiiiiiip).
3. Les ultrasons se propagent dans l'air jusqu'à toucher un obstacle et retourne 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.

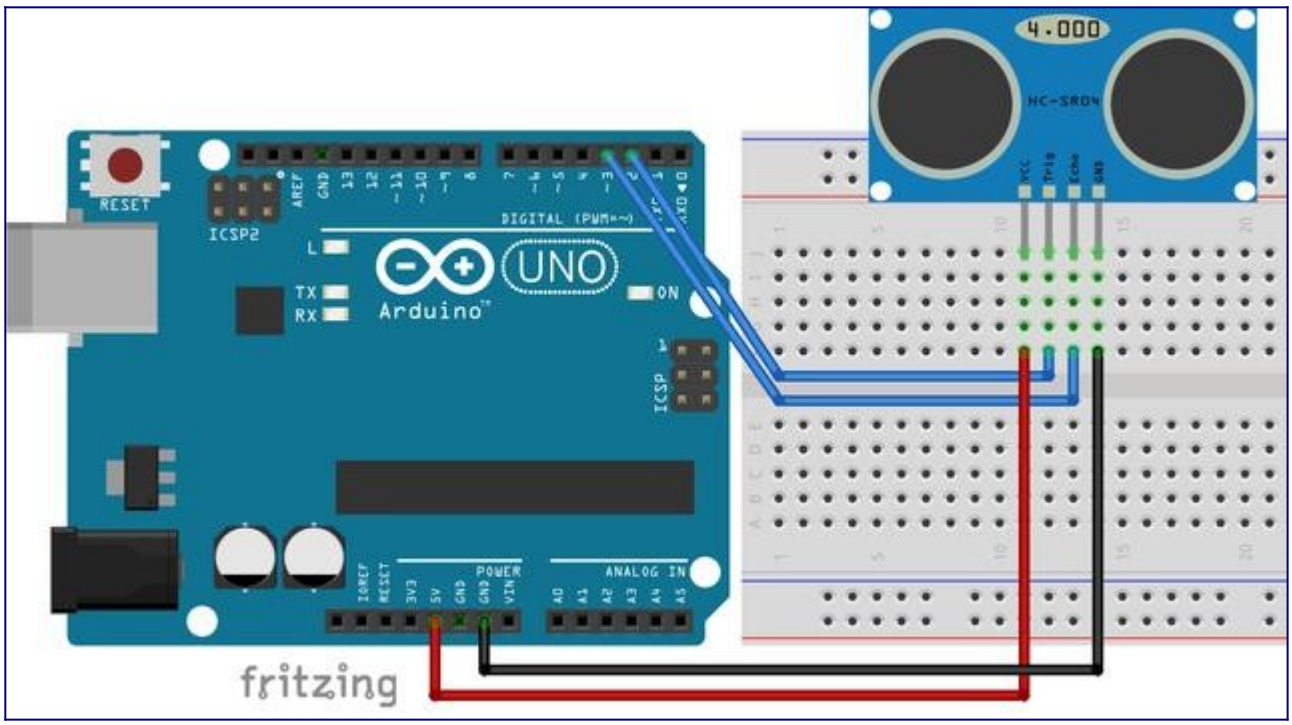
Le montage



Matériel nécessaire

Pour réaliser ce premier montage, il va nous falloir :

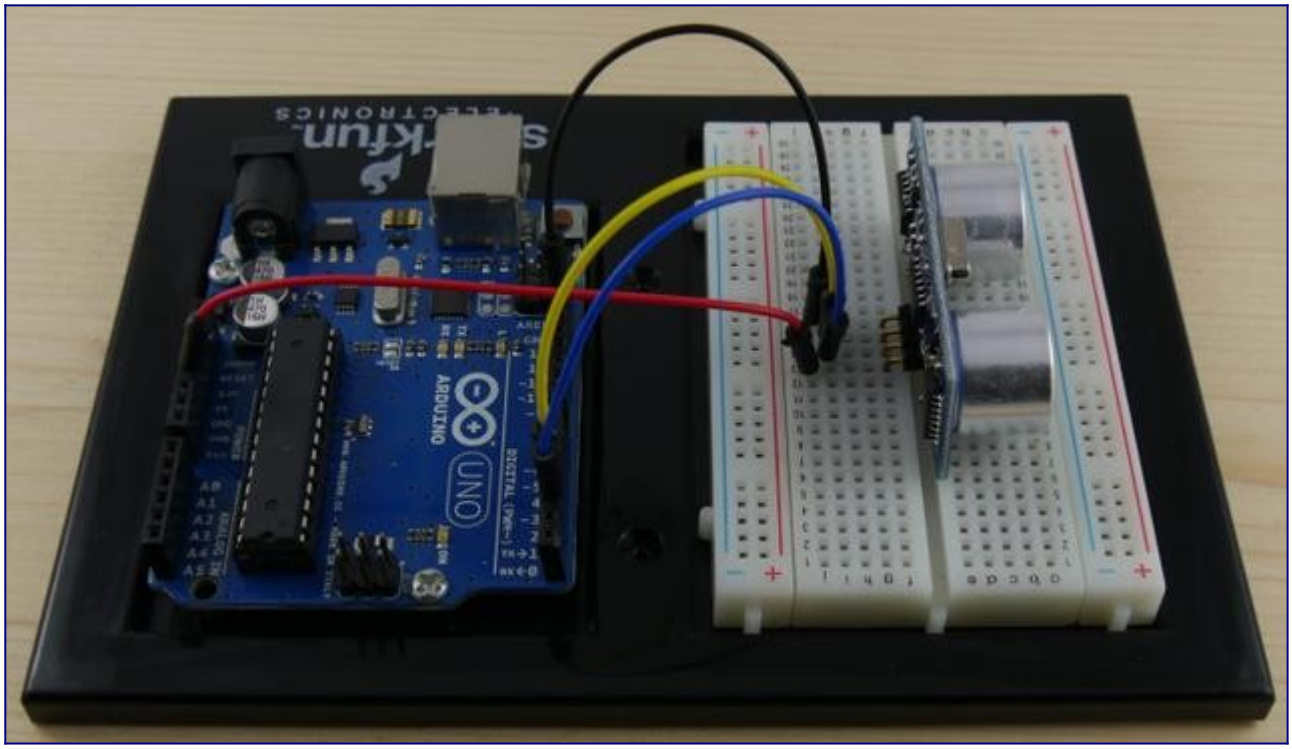
- Une carte Arduino UNO (et son câble USB),
- Un capteur HC-SR04,
- Une plaque d'essai et des fils pour câbler notre montage.



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

```
/* Constantes pour les broches */
const byte TRIGGER_PIN = 2; // Broche TRIGGER
const byte ECHO_PIN = 3;    // Broche ECHO

/* 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;
```

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. On achoisi d'utiliser une timeout de 25 millisecondes (4 mètres aller-retour à 340m/s).

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

```
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);
}
```

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**.

```
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);
}
```

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 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 :

```
/*
 * Code d'exemple pour un capteur à ultrasons HC-SR04.
 */

/* Constantes pour les broches */
const byte TRIGGER_PIN = 2; // Broche TRIGGER
const byte ECHO_PIN = 3;    // Broche ECHO

/* 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);
}
```

La précision du capteur

Essais pour voir si le capteur donne des mesures précises ou non.

Test en intérieur avec un petit obstacle

Banc de test en intérieur

Commençons les tests avec une configuration 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.

Quelques conseils d'utilisations

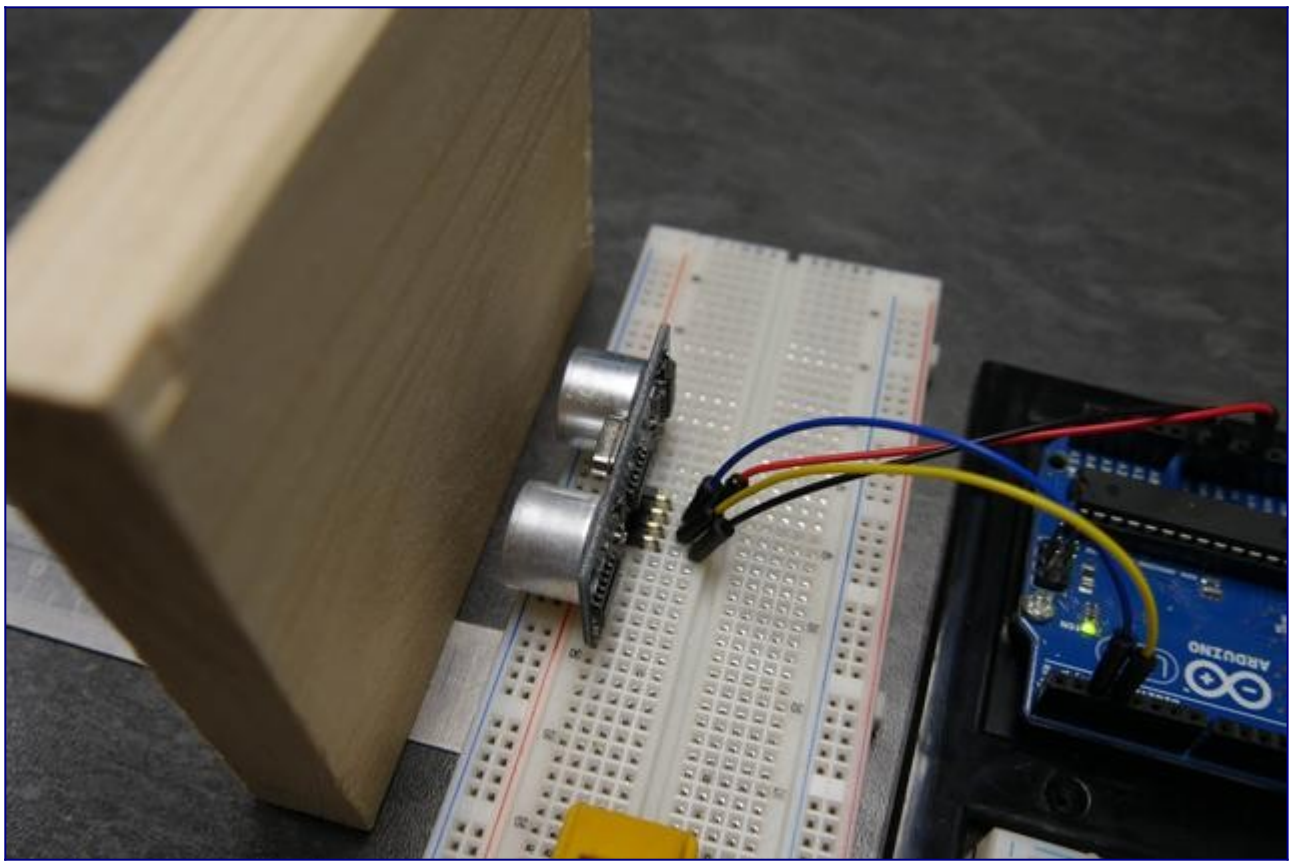
Ces capteurs à ultrason ont besoin d'une zone dégagée, avec une surface dure et lisse d'au moins 50cm² en face du capteur pour donner des résultats corrects.

Inutile d'essayer de mesurer une distance par rapport à un rideau ou une surface absorbant le son, ça ne donnera rien. De même, utiliser ce genre de capteur à ras le sol ou sur une table, voir pire, ce n'est définitivement pas une bonne façon d'avoir des mesures correctes.

Voici les résultats de mes mesures sur une plage limitée de 10 millimètres à 50 centimètres :

Distance mm	Mesure	Erreur mm	Erreur %
10	10,20	0,20	2,00%
20	18,70	1,30	6,50%
30	23,46	6,54	21,80%

40	32,64	7,36	18,40%
50	42,50	7,50	15,00%
60	52,02	7,98	13,30%
70	66,30	3,70	5,29%
80	76,40	3,60	4,50%
90	85,34	4,66	5,18%
100	95,54	4,46	4,46%
150	144,50	5,50	3,67%
200	188,70	11,30	5,65%
250	237,66	12,34	4,94%
300	286,62	13,38	4,46%
350	334,56	15,44	4,41%
400	384,86	15,14	3,79%
450	433,50	16,50	3,67%
500	482,46	17,54	3,51%

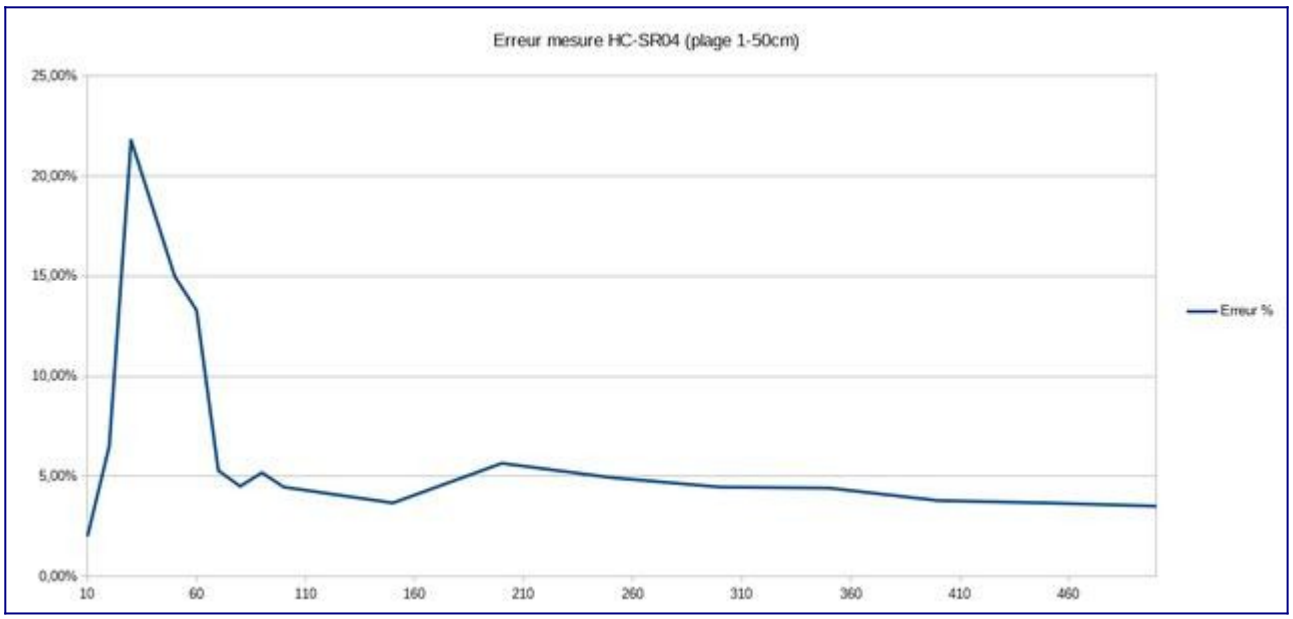


Distance minimum possible

Pour commencer, même avec toute la bonne volonté du monde, on n'a pas pu descendre en dessous de 10mm, ce qui est déjà pas mal et bien plus que les deux centimètres minimum de la fiche constructeur. Ce sera donc le minimum de ce test.

PS On a pu remarquer qu'en dessous d'un centimètre, le capteur faisait n'importe quoi. on suppose donc que le constructeur donne un minimum de deux centimètres pour garder un peu de marge.

Graphique de linéarité du capteur sur la plage 1-50cm



Graphique d'erreurs du capteur sur la plage 1-50cm

La courbe orange est la théorie. La courbe bleue est la pratique.

On remarque de suite que la théorie et la pratique ne sont pas tout à fait identiques. Il y a une légère dérive avec un taux d'erreur de ~5% en moyenne au-dessus de 7 centimètres.

D'un point de vue précision, en dessous de dix centimètres, c'est du grand n'importe quoi. Si on veut

=-=-=-=-=-=