Capteurs - Distance - Ultrasons

[Mise à jour le 3/5/2024]

- **Lectures connexes**
  - Bien choisir un capteur de proximité
  - Comment choisir le meilleur capteur à ultrason
  - Vidéo - Exemples d'applications

1. Généralités

L'ultrason est une onde mécanique et élastique, qui se propage au travers de supports fluides, solides, gazeux ou liquides. La gamme de fréquences des ultrasons se situe entre **16 000 et 10 000 000 Hertz**.

Le nom vient du fait que leur fréquence est trop élevée pour être audible pour l'oreille humaine (le son est trop aigu : la gamme de fréquences audibles par l'homme se situe entre **20 et 20 000 Hertz**. Ces seuils sont cependant variables avec l'âge), de la même façon que les infrasons désignent les sons dont la fréquence est trop faible pour être perceptible par l'oreille humaine. Lorsque la fréquence est audible pour l'oreille humaine, on parle tout simplement de son.

Les ultrasons sont utilisés dans l'industrie ainsi que dans le domaine médical. Wikipédia

- ***Ressources***
  - Rudiments d'acoustique : Quelques définitions

> **Principe de fonctionnement des capteurs HC-SR0x**
> Voir l'article sur le “Carnet du maker” **ici**

2. Capteurs numériques

2.1 HC-SR04 (GPIO)

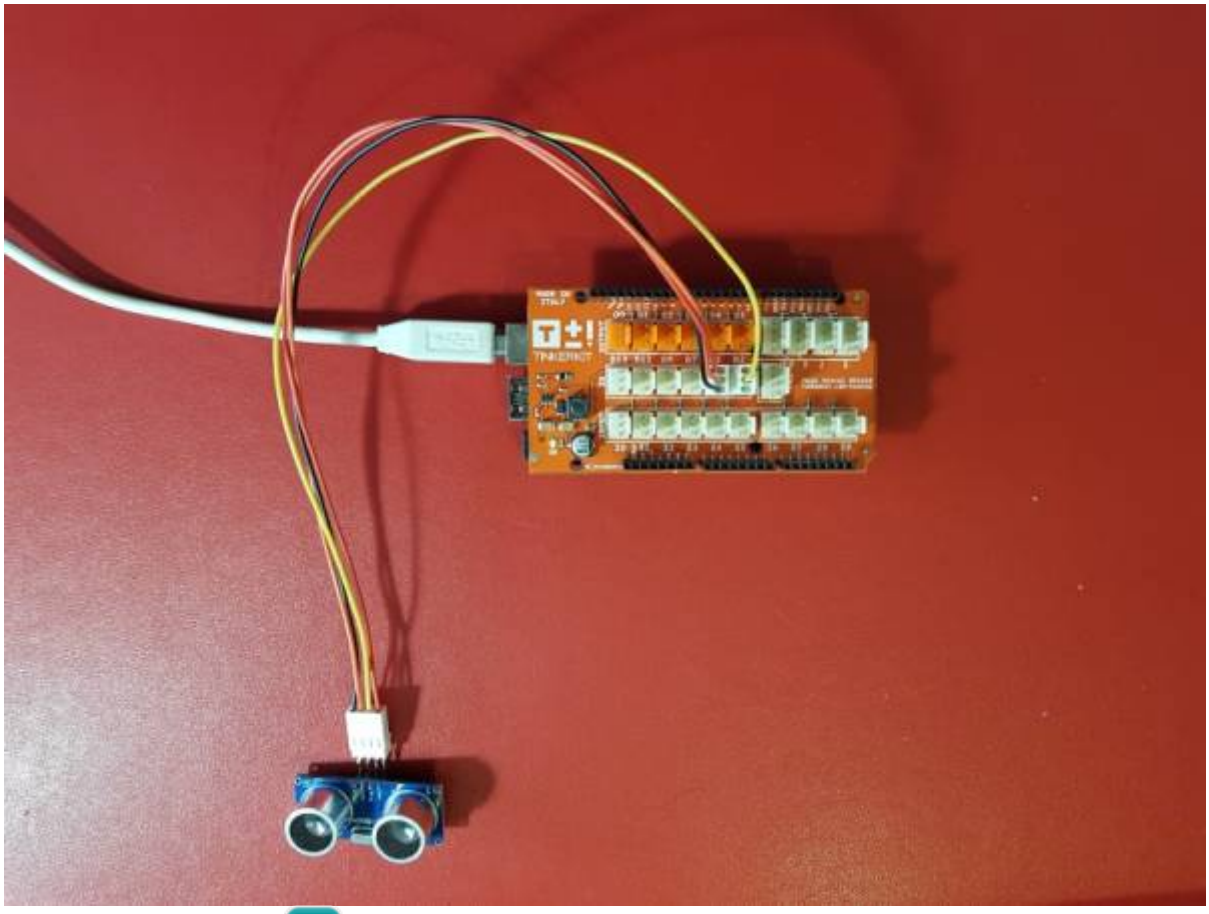
2.1.1 Présentation

- ***Source*** : pdf

> Ce module permet d'évaluer les distances entre un objet mobile et les obstacles rencontrés. Il suffit d'envoyer une impulsion de 10 µs en entrée et le capteur renvoie une largeur d'impulsion proportionnelle à la distance.

- ***Distributeur*** : Gotronic
- ***Caractéristiques***
  - Alimentation: 5 Vcc
  - Consommation: 15 mA
  - Fréquence: 40 kHz
  - Portée: de 6...10 cm à 4 m
  - Déclenchement: impulsion TTL positive de 10µs
  - Signal écho: impulsion positive TTL proportionnelle à la distance.
  - Calcul: distance (cm) = impulsion (µs) / 58
  - Dimensions: 45 x 21 x 18 mm

- ***Documentation***
  - Fichier Acrobat Reader à télécharger ici
- ***Programmation d'une carte Arduino Mega 2560***
  - *Bibliothèques à installer dans l'IDE* : aucune
  - *Connexion à un shield* Tinkerkit Mega v2.



- *Un premier exemple*

HCSR04.cpp

```
/*
 * Code d'exemple pour un capteur à ultrasons HC-SR04.
 * Carte Arduino Mega 2560
 */

/* Constantes pour les broches */
const byte TRIGGER_PIN = 2; // Broche TRIGGER
const byte ECHO_PIN = 4;    // 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);
}
```

Télécharger le projet PlatformIO pour VSCode.

2.2 HC-SR05 (GPIO)

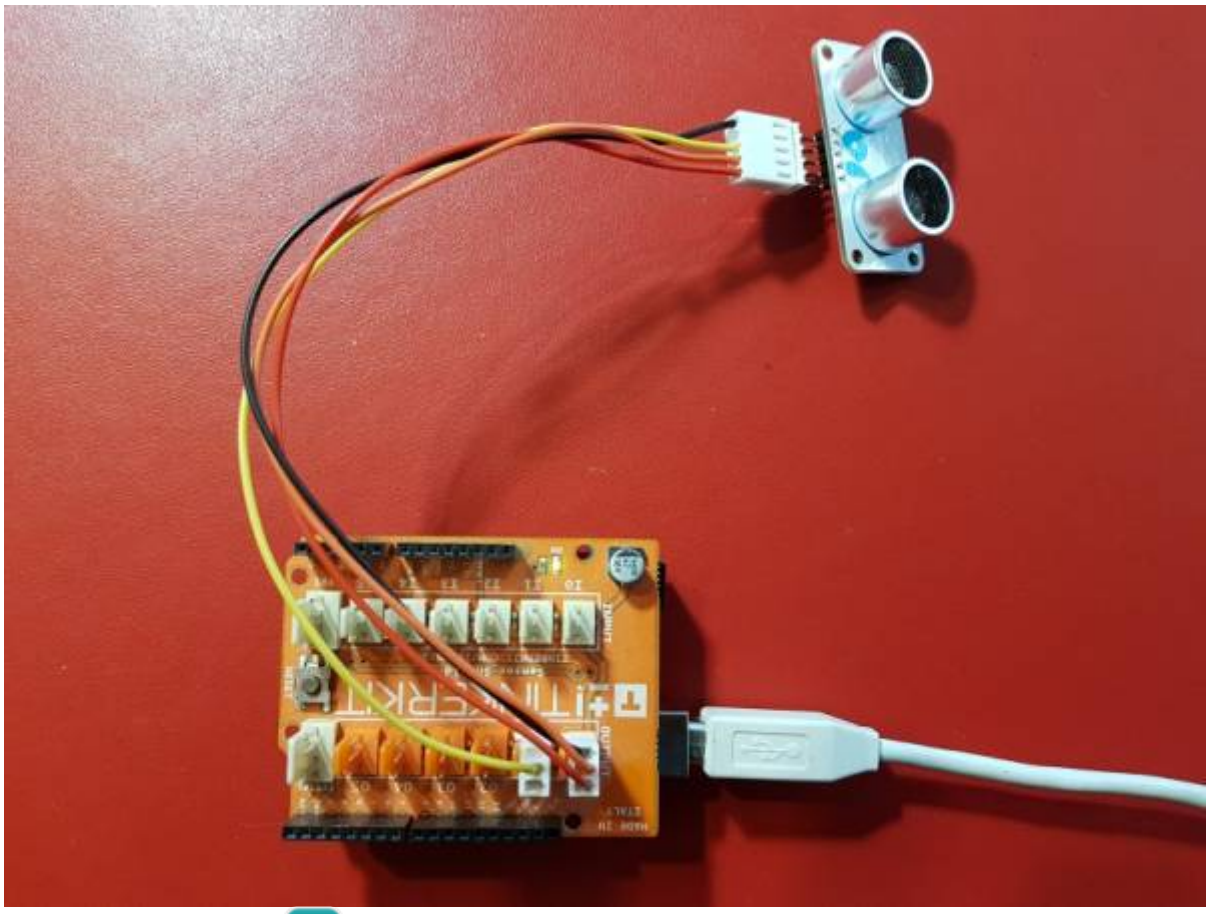
2.2.1 Présentation

- ***Source*** : pdf

> Ce module est basé sur un capteur à ultrasons HC-SR05 et permet d'évaluer les distances entre un objet mobile et les obstacles rencontrés.

- ***Distributeur*** : Gotronic
- ***Caractéristiques***
  - Alimentation: 4,5 à 5,5 Vcc
  - Consommation:
    - mini: 10 mA
    - maxi: 40 mA
  - Fréquence: 40 kHz
  - Portée: de 2 cm à 4,5 m
  - Déclenchement: impulsion TTL positive de 10µs
  - Signal écho: impulsion positive TTL proportionnelle à la distance.

- ***Documentation***
  - Manuel d'utilisation du capteur à ultrasons VMA306 à télécharger ici
- ***Programmation d'une carte Arduino Uno R3***
  - *Bibliothèques à installer dans l'IDE* : aucune
  - *Connexion à un shield* Tinkerkit v2.



- *Un premier exemple*

HCSR05.cpp

```
////////////////////////////////////////////
// Programme test pour capteur HC-SR05      //
// Go Tronic 2017                           //
////////////////////////////////////////////
#define trigPin 10 // Tinkerkit O1
#define echoPin 11 // Tinkerkit O0

long duration, distance;

void setup()
{
    Serial.begin(9600);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    Serial.println("== Debut du programme ==");
}
void loop()
{
    // Envoie de l'onde
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
```

```
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    // Réception de l'écho
    duration = pulseIn(echoPin, HIGH);

    // Calcul de la distance
    distance = (duration / 2) / 29.1;
    if (distance >= 400 || distance <= 0)
    {
        Serial.println("Hors plage");
    }
    else
    {
        Serial.print("distance = ");
        Serial.print(distance);
        Serial.println(" cm");
    }
    delay(500); // délai entre deux mesures
}
```

Télécharger le projet PlatformIO pour VSCode.

2.3 HC-SR04P (GPIO - I2C - UART)

2.3.1 Présentation

Ce transducteur ultrason à base de HC-SR04P délivre un signal proportionnel à la distance qui le sépare d'un obstacle. Le signal pourra être récupérer directement via des **GPIO**, une interface **UART** ou **I2C**. Ce qui le rend compatible avec la plupart des microcontrôleurs (Arduino® ou compatible, Raspberry Pi, etc...).

- ***Distributeur*** : Lextronic
- ***Caractéristiques***
  - Capteurs: HC-SR04
  - Tension d'alimentation: 3 Vcc à 5,5 Vcc
  - Interface: GPIO / UART / I2C (**SLA** = **0x57**)
  - Niveau logique: 3,3 Vcc / 5Vcc
  - Consommation: 15 mA (< 3mA au repos)
  - Angle de mesure: < 15°
  - Portée de détection: 2 à 450 cm
  - Précision: 0,3 cm
  - Connecteur mâle 4 broches coudé pré-soudé (pas 2,54 mm): Vcc/Trig/Echo/GND
  - Dimensions: 45,5 x 20,3 x 15,5 mm
- ***Configuration*** de M1 et M2 sur le CI (1 : R=10k à ajouter)
  M1 M2
  0 0 : GPIO
  1 0 : I2C
  1 1 : 1-wire
  0 1 : UART

- ***Documentation***
  - A télécharger **ici**

- ***Chronogrammes***
  - Relevé des signaux du bus I2C. A télécharger **ici**.

2.3.2 Exemples de code

- Arduino UNO (GPIO)
- Arduino UNO (I2C)
- Arduino UNO (UART)
- Rpi Pico (µPython)

- ***Ressources*** : pinMode() | pulseIn() | serial
- ***Exemple*** *utilisant les* ***GPIO*** *pour tester le capteur*

*.cpp

```
// Arduino MKR sur Arduino MKR Connector
// Connexion sur le connecteur D5 D6B
const int echo = 6;
const int trigger = 5;

float distance, duration;

void setup() {
  Serial.begin(9600);
  pinMode(echo, INPUT);
  pinMode(trigger, OUTPUT);
}

void loop() {
  digitalWrite(trigger, HIGH);  // Mesure de la distance en utilisant
une impulsion de 10us
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);
  duration = pulseIn(echo, HIGH);
  distance = duration * 340 / 2 / 10000;
  Serial.println("distance: " + String(distance) + " cm");  //
Affichage dans la console
  delay(3000);  // Attente entre 2 mesures
}
```

- ***Ressource*** : Wire
- ***Exemple*** *utilisant l'**I2C** pour tester le capteur*

*.cpp

```
// Arduino MKR sur Arduino MKR Connector
// Connexion sur le connecteur TWI
#include <Wire.h>

int     address = 0x57;     // i2c address of SEN-US01
float   distance;
long int     bytes[3];

void setup(){
    Serial.begin(9600);
    Wire.begin();
}

void loop(){
    Wire.beginTransmission(0x57);   // Start measurement
    Wire.write(0x01);
    Wire.endTransmission();
```

```
    delay(200); // Wait 200 ms

    Wire.requestFrom(0x57,3);       // Read 3 bytes from sensor
    int i = 0;
    while(Wire.available()){
        bytes[i++] = Wire.read();
    }

    distance = ((bytes[0] << 16) + (bytes[1] << 8) + bytes[2]) / 10000;
// Calculate distance based on received bytes

    if (false){        // Checking whether measured value is within the
permissible distance
        Serial.println("distance is outside of the measuring range");
// If not, an error message is output
    }
    else{
        Serial.println("distance: " + String(distance) + " cm");    //
The calculated distance is output to the console
    }
    delay(3000);    // Pause between the individual measurements
}
```

- ***Ressource*** : SoftwareSerial Library
- ***Exemple*** *utilisant l'****UART*** *pour tester le capteur*

*.cpp

```
// Arduino MKR sur Arduino MKR Connector
// Connexion sur le connecteur SERIAL

#include <SoftwareSerial.h>

SoftwareSerial ser(A4,A5); // initialize a serial connection (software)
float   distance;
long int     bytes[3];

void setup(){
    Serial.begin(9600);     // start serial connection to computer
    ser.begin(9600);        // start serial connection to sensor
}

void loop(){
    ser.flush();    // clear communication
    ser.write(0xA0);    // start measurement
    delay(200); // Wait 200 ms
```

```
    for (int i = 0; i < 3; i++) bytes[i] = ser.read();  // Read 3 bytes from sensor

    distance = ((bytes[0] << 16) + (bytes[1] << 8) + bytes[2]) / 10000; // Calculate distance based on received bytes

    if (false){        // Checking whether measured value is within the permissible distance
        Serial.println("distance is outside of the measuring range"); // If not, an error message is output
    }
    else{
        Serial.println("distance: " + String(distance) + " cm");    // The calculated distance is output to the console
    }
    delay(3000);    // Pause between the individual measurements
}
```

A venir