TELEMETRE ULTRASON

Esteban Chevalier-Drevon
Matyas Ricci
Groupe F

Le télémètre ultrason

1. Exercice 1 : Distance=f(durée)

Pour le télémètre ultrason utilisé, on souhaite extraire la distance de l'obstacle en fonction de la durée de l'écho.

Déterminer la relation Distance (cm) = f (temps (µs)) si la vitesse du son est de 340 m/s.

La relation distance (cm) = f (temps (µs)) si la vitesse du son est de 340 m/s est :

Distance(cm) = Temps(µs) * 0.034 /2

On divise par 2 car le temps mesuré correspond à un aller-retour, et on cherche à avoir juste un allé. La valeur 0,034 correspond à la vitesse du son en cm/µs.

2. Exercice 2 : Analyse de documentation

A l'aide de la documentation du HC-SR04 page suivante, extraire les informations suivantes :

Fréquence nominale du capteur ? 40Hz

Résolution du capteur ? 0,3cm

Distance de détection ? 2cm – 400 cm

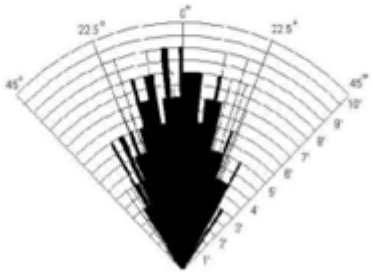
Angle de détection ? 15°

HC-SR04 - Module de détection aux ultrasons

Le capteur HC-SR04 utilise les ultrasons pour déterminer la distance d'un objet. Il offre une excellente plage de détection sans contact, avec des mesures de haute précision et stables. Son fonctionnement n'est pas influencé par la lumière du soleil ou des matériaux sombres, bien que des matériaux comme les vêtements puissent être difficiles à détecter.

Caractéristiques

- Dimensions : 45 mm x 20 mm x 15 mm
- Plage de mesure : 2 cm à 400 cm
- Résolution de la mesure : 0.3 cm
- Angle de mesure efficace : 15 °
- Largeur d'impulsion sur l'entrée de déclenchement : 10 µs (Trigger Input Pulse width)



Practical test of performance, Best in 30 degree angle

Broches de connection

- Vcc = Alimentation +5 V DC
- Trig = Entrée de déclenchement de la mesure (Trigger input)
- Echo = Sortie de mesure donnée en écho (Echo output)
- GND = Masse de l'alimentation

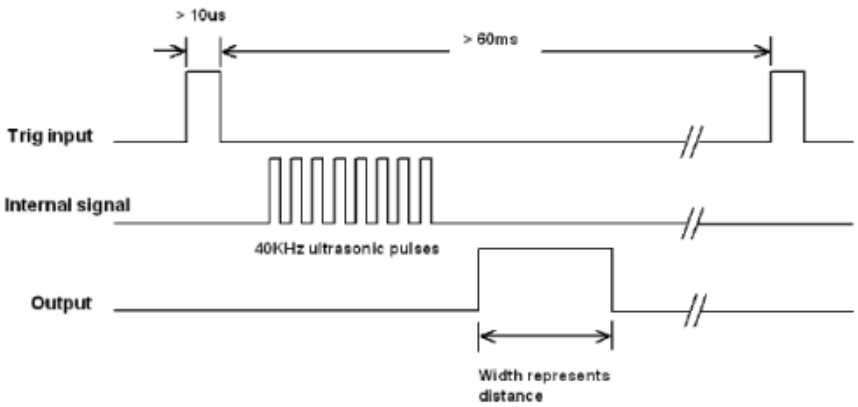
Spécifications et limites

Paramètre	Min	Type	Max	Unité
Tension d'alimentation	4.5	5.0	5.5	V
Courant de repos	1.5	2.0	2.5	mA
Courant de fonctionnement	10	15	20	mA
Fréquence des ultrasons	-	40	-	kHz

Attention : la borne GND doit être connectée en premier, avant l'alimentation sur Vcc.

Fonctionnement

Pour déclencher une mesure, il faut présenter une impulsion "high" (5 V) d'au moins 10 µs sur l'entrée "Trig". Le capteur émet alors une série de 8 impulsions ultrasoniques à 40 kHz, puis il attend le signal réfléchi. Lorsque celui-ci est détecté, il envoie un signal "high" sur la sortie "Echo", dont la durée est proportionnelle à la distance mesurée.



3. Exercice 3 : Calcul de distance et de durée

La durée de l'impulsion ECHO du HC-SR04 est de 5 ms. Si la vitesse du son est 340 m/s, quelle est la distance de l'objet détecté ?

La distance de l'objet et Obtenu grâce à l'expression : Temps(µs) * 0.034 /2 = Distance(cm)

Ici : 5000 * 0,034 /2 = 85 m, l'objet est donc à une distance de 85cm.

Si la vitesse du son est 340 m/s, quelle sera la durée de l'impulsion ECHO si l'objet est placé à 20 cm ?

La durée de l'impulsion ECHO si l'objet est placé à 20 cm est déterminé grâce a cette expression : Distance(cm) = Temps(µs) * 0.034 /2

Temps(µs) = (Distance(cm) * 2) / 0,034
Temps(µs) = (20 * 2) / 0,034
Temps(µs) = 1176 µs

La durée de l'impulsion ECHO est donc de 1176 µs si l'objet est placé à 20 cm.

4. Programme 1 : Programme Ultrason

Ecrire un programme qui détecte et affiche la distance de l'objet sur le moniteur série.

S'il n'y a aucun obstacle détecté, quelle valeur est renvoyée par le programme ?

Si aucun obstacle est détecté, la valeur renvoyée par le programme sera de 0.

Programme :

```
#define echoPin 12
#define trigPin 11
long duration;
int distance;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
}
```

5. Programme 2 : Mesure du temps de boucle - Ultrason (TimeOut) et LCD

A l'aide de la documentation, décrire le rôle du Time Out dans l'instruction PulseIn. On estime que les obstacles à détecter sont au maximum à 1m. Quelle valeur de TimeOut doit-on utiliser ?

Le troisième paramètre (optionnel) Timeout est la durée maximale en microsecondes de l'attente d'une impulsion avant la mesure. Si aucune impulsion n'arrive avant la fin du Timeout, la fonction s'arrête et retourne 0.

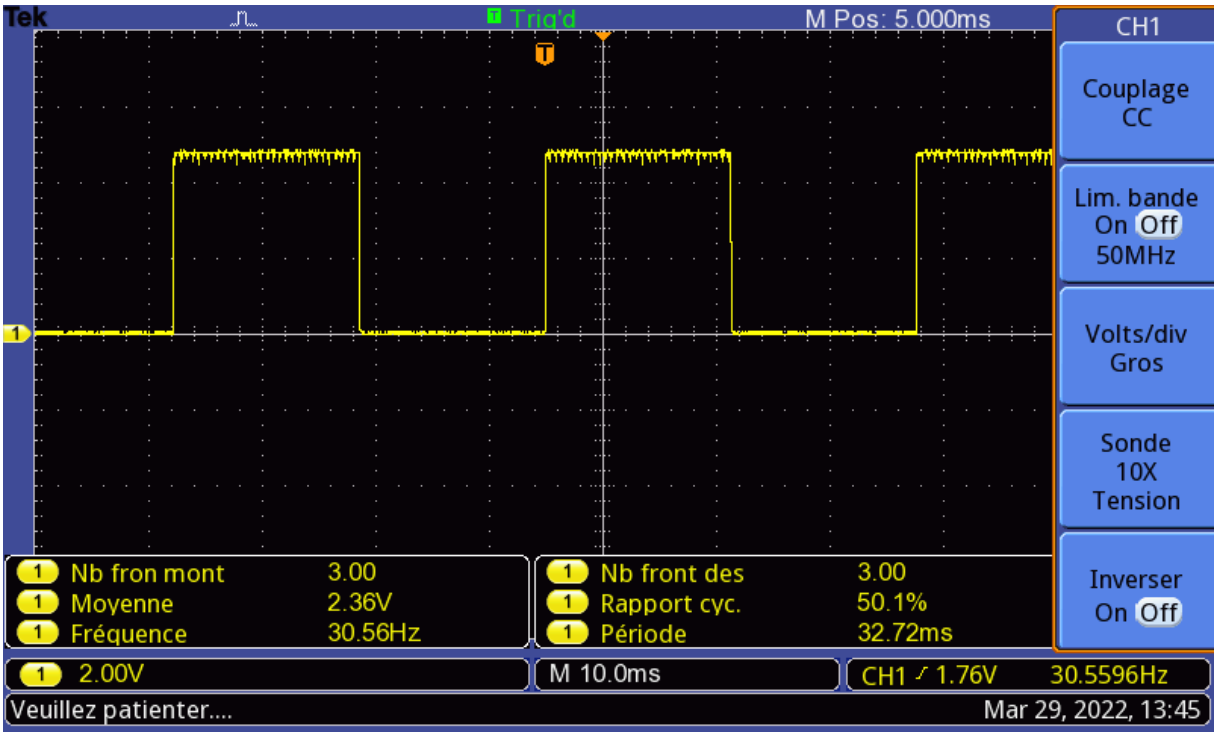
Timeout pour 1m maximum : (100*2 / 0,034) = 5883µs.

On cherche à mesurer le temps pris pour réaliser les instructions dédiées à l'écran LCD (lcd.setCursor et lcd.print).

On vous demande d'écrire un programme qui écrit la distance sur l'écran LCD et qui fait changer d'état une sortie Tout Ou Rien (TOR) à chaque passage dans la boucle (exemple : In1=!In1;). On mesurera le temps de boucle avec un oscilloscope branché sur la sortie associée à In1.

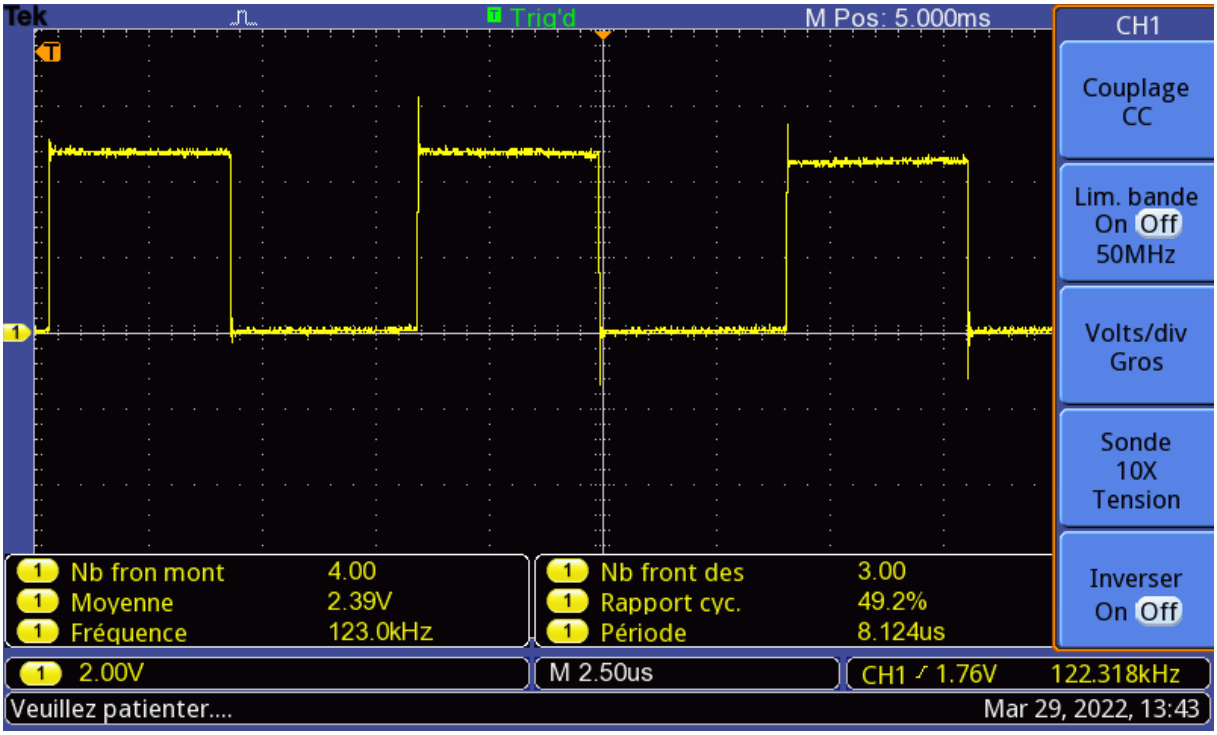
Temps de boucle avec lcd.print : Période de 32,72ms

Donc 32,72 / 2 = 16,36 ms



Temps de boucle sans lcd.print : Période de 8,214 µs

8,124 / 2 = 4,62 µs



Programme :

```
1   #include <Wire.h>
2   #include <LiquidCrystal_I2C.h>
3   LiquidCrystal_I2C lcd(0x27, 16, 2);
4
5   #define IN1 2
6   bool state = 0;
7
8
9   void setup() {
10      lcd.init();
11      lcd.backlight();
12      pinMode(IN1, OUTPUT);
13
14  }
15
16
17  void loop() {
18      lcd.print("Hello World");
19      state = !state;
20      digitalWrite(IN1, state);
21  }
```

Pour la mesure sans le lcd.print, on retire la ligne 18 du programme ci-dessus.

5. Test d'intégration 1

AFFICHER LA VALEUR DES TELEMETRES ULTRASON SUR L'ECRAN LCD :

Distance D : xx

Distance G : xx

Programme :

```
#define echoPinG 12
#define trigPinG 11

#define echoPinD A3
#define trigPinD A0

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

long duration;
int distance;
bool state = 0;

void setup() {
  pinMode(trigPinG, OUTPUT);
  pinMode(echoPinG, INPUT);
  pinMode(trigPinD, OUTPUT);
  pinMode(echoPinD, INPUT);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Distance D:");
  lcd.setCursor(0,1);
  lcd.print("Distance G:");
  Serial.begin(9600);
}

void loop() {
  static unsigned short DistanceD, DistanceG, lastD, lastG, durationG, durationD;
  digitalWrite(trigPinG, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPinG, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPinG, LOW);

```

```
36
37    durationG = pulseIn(echoPinG, HIGH, 5883);
38
39    digitalWrite(trigPinD, LOW);
40    delayMicroseconds(2);
41    digitalWrite(trigPinD, HIGH);
42    delayMicroseconds(10);
43    digitalWrite(trigPinD, LOW);
44
45
46    durationD = pulseIn(echoPinD, HIGH, 5883);
47
48    DistanceG = durationG * 0.034 / 2;
49    DistanceD = durationD * 0.034 / 2;
50
51    if (lastD != DistanceD) {
52      lastD = DistanceD;
53      lcd.setCursor(11, 0);
54      if (DistanceD < 10) {
55        lcd.print((String)DistanceD + ' ');
56      } else {
57      lcd.print(DistanceD);
58      }
59    }
60
61    if (lastG != DistanceG) {
62      lastG = DistanceG;
63      lcd.setCursor(11, 1);
64      if (DistanceG < 10) {
65        lcd.print((String) DistanceG + ' ');
66      } else {
67        lcd.print(DistanceG);
68      }
69    }
70  }
```

6. Test d’intégration 2

FAIRE AVANCER LE ROBOT DANS UN PARCOURS AVEC OBSTACLES DEFINI PAR L'ENSEIGNANT

Le Test d’intégration a été validé par l’enseignant nous avons pu observer un inconvénient majeur des capteurs ultrasons. En effet, avec un certain angle l’onde rebondit sur la surface et l’obstacle n’est pas détecter. Nous allons donc passer aux capteurs infrarouges.

Programme :

```
    } else { // mode Automatique
        digitalWrite(trigPinF, LOW);
        delayMicroseconds(2);
        digitalWrite(trigPinF, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPinF, LOW);

        durationF = pulseIn(echoPinF, HIGH, 5883);

        DistanceFSon = durationF * 0.034 / 2;

        DistanceD = (pow(analogRead(IrD), - 1.25) * 49570);
        DistanceG = (pow(analogRead(IrG), - 1.25) * 49570);

        if (DistanceD >= 80) DistanceD = 70;
        if (DistanceG >= 80) DistanceG = 70;


        static unsigned short AverageG = 0;
        static unsigned short AverageD = 0;
        static unsigned short AverageF = 0;

    DistanceD = map(DistanceD, 0, 70, 20, 35);
    DistanceG = map(DistanceG, 0, 70, 20, 35);


    if (DistanceD > DistanceG) {
        DistanceD = floor(DistanceD * 1.20);
        cmd_motor(1, 1, DistanceG);
        cmd_motor(0, 1, DistanceD);

    } else if (DistanceD < DistanceG) {
        DistanceG = floor(DistanceG * 1.20);
        cmd_motor(1, 1, DistanceG);
        cmd_motor(0, 1, DistanceD);

    } else {
        cmd_motor(1, 1, 30);
        cmd_motor(0, 1, 30);
    }
}}
```