Raspberry Pi (7.3)
- télémètre à ultrason -

Télémètre à ultrason

Nom : **Prénom :** **Classe :** **Date :**	**Appréciation** :	**Note** : **/20**
Objectif : **Utilité** :		**durée** : 4h

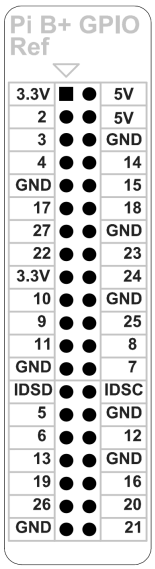
Matériel : plaque labdec – composants électroniques

Prérequis : Connexion à distance avec SSH, commande GPIO

Compétences et savoirs principalement visées :
C2-1, C2-2 , C3-2, C3-3

Travail à réaliser :
-
-
-

Schéma du système :



Le capteur à ultrason

Anglais : Ultrasonic Ranger

Le capteur à ultrason SR04 est représenté ci-contre :

Le module ultrason Grove Ultrasonic Ranger s'en distingue uniquement par le fait que la broche TRIG et la broche ECHO sont fusionnées en 1 seule broche appelée SIG

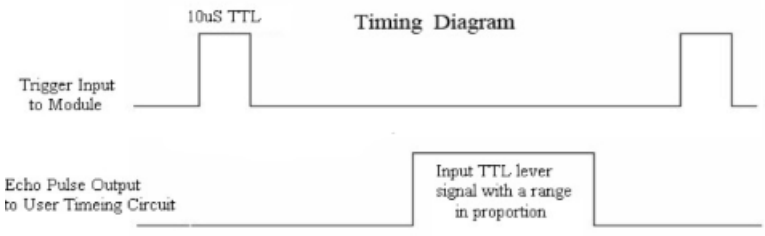
- **Question :** expliquer le principe scientifique utilisé dans la mesure de distance à l'aide d'ultrason (faire une recherche Google). Temps maximum conseillé : 20 mn

Télémètre à ultrason

Un télémètre est un appareil qui mesure la distance entre sa position actuelle et l'obstacle le plus proche ; par exemple, mesurer la distance jusqu'à un mur. La mesure se fait généralement en cm.

Dans notre cas, nous souhaitons faire des mesures en utilisant le **module ultrason HC-SR04**, connecté et **géré par le RPi**, et pouvoir lancer la mesure et en récupérer le résultat sur une **appli Web**, depuis un PC ou un **smartphone**

Principe de fonctionnement du module HC-SR04



Comme le montre le diagramme ci-dessus, il faut :

- ➔ Mettre la pin d'entrée « trig » à 1 pendant 10us (0,000 01 s)
- ➔ attendre que la pin de sortie « echo » passe à 1
- ➔ commencer à mesurer le temps
- ➔ attendre que la pin de sortie « echo » repasse à 0
- ➔ arrêter la mesure du temps
- ➔ pour avoir une mesure en cm, diviser le temps mesuré par 58

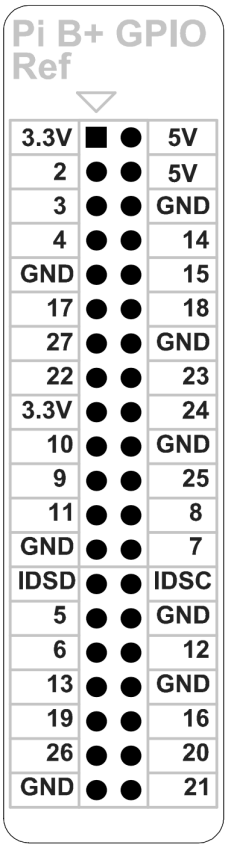
Remarque : sur le module Grove, les signaux « trig » et « echo » rassemblés sur une patte unique (« sig »)

Câblage du système

AVERTISSEMENT
Attention : en cas d'erreur de branchement, ton Raspberry Pi risque d'être **détruit** !!! Ne mets pas le circuit sous tension **avant** que le professeur l'ai vérifié.

- Avant de réaliser le câblage physiquement, **dessine les connexions** (relie par des traits) des éléments ci-dessous en t'aidant du document «ANNEXE – Détecteur d'obstacle à ultrason»

  Note : il y a **3** connexions à réaliser



- Une fois que le schéma est complet, réalise ces connexions à l'aide des câbles fournis, puis fait valider par les professeur

Programme de gestion du module ultrason

Principe de fonctionnement des module HC-SR04 et Grove Ultrasonic

Voir page 2.

Choix du langage de programmation

Il n'est pas possible de réaliser ce programme avec un langage de scripting (type bash ou python), car ceux-ci ne sont pas assez rapide pour générer ou capter des impulsions de l'ordre de quelques microsecondes, comme c'est le cas ici (voir l'encadré)

Nous utiliserons donc un langage compilé rapide : le langage C ; tu trouveras le code source du logiciel à la page suivante.

- Le programme ultrason, dont le code source est disponible sur la page suivante, doit être lancé avec un paramètre : **./ultrason n°_pin_SIG**

➔ *En analysant le code, découvre et explique à quoi correspond ce paramètre :*

(indice : la réponse se trouve dans la partie surlignée en jaune)

- **Analyse le code source ultrason.c** pour essayer de **découvrir** quelles parties effectuent les fonctions indiquées ;

➔ *pour cela, complètes le tableau :*

Fonction	Recopie ici le code source correspondant
Mettre la pin d'entrée «SIG» à 1 pendant 10us	
attendre que la pin de sortie «SIG» passe à 1	
commencer à mesurer le temps	
attendre que la pin de sortie «SIG» repasse à 0	
arrêter la mesure du temps	
pour avoir une mesure en cm, diviser le temps mesuré par 58	

Listing du code source ultrason.c

```
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <wiringPi.h>

// la fonction  divRoundClosest permet de faire une division en arrondissant à l'unité
unsigned int divRoundClosest(const int n, const int d)

{
  return ((n + d/2)/d);
}
// la fonction  help
void help(void)
{
        printf("Syntaxe : ultrason n°_pin_SIG \n");
}
// la programme principal
int main (int argc, char *argv[])
{
   int start, stop , dist, diff, SIG;

 if (wiringPiSetup () == -1)
   {
     fprintf (stdout, "Erreur gpio : %s\n", strerror (errno)) ;
     return 1 ;
   }

  piHiPri (99);

  if (argc>1)
 {
        SIG=atoi(argv[1]);
        if ( SIG > 29 || SIG<0 ) {help(); exit (1);}

 } else {exit (1);}

pinMode(SIG, OUTPUT);

//10us pulse on trig
digitalWrite(SIG, 1);
delayMicroseconds(10);
digitalWrite(SIG, 0);

// SIG en mode INPUT pour capter la reponse
pinMode(SIG, INPUT);

//wait for ECHO being 1
while (digitalRead(SIG) == 0)    {}
start = micros();

//wait for ECHO being 0
while (digitalRead(SIG) == 1)    {}
stop = micros();
diff = stop-start;

//calcul de la distance en cm
dist = divRoundClosest(diff,58);
 if (dist > 400 )   {  dist = -1;   }
printf("%d",dist);
return dist;
}
```

- Crée nouveau fichier nommé **ultrason.c** et recopie le code source ci-dessus dans ce fichier (utiliser le logiciel notepad++)
- Vérifie que le codage du fichier ultrason.c est bien : **UTF-8** ; préciser ci-dessous le codage du fichier :
- Vérifie que les caractères de fin de ligne sont bien au **format UNIX (LF)** ; préciser ci-dessous le caractères de fin de ligne du fichier ultrason.c :

La compilation d'un programme en C

Compiler un programme, c'est convertir le code source en langage machine.

Le code source est écrit en langage C, lisible par un être humain ; le langage machine est le seul que comprend le microprocesseur ; le code compilé est parfois appelé « binaire ».

La compilation nécessite l'utilisation d'un logiciel appelé « compilateur ». Pour le langage C, le compilateur le plus répandu est **gcc**

- **Compile** le code source du logiciel avec la commande :

```
gcc -Wall -o ultrason ultrason.c -lwiringPi -lm
```

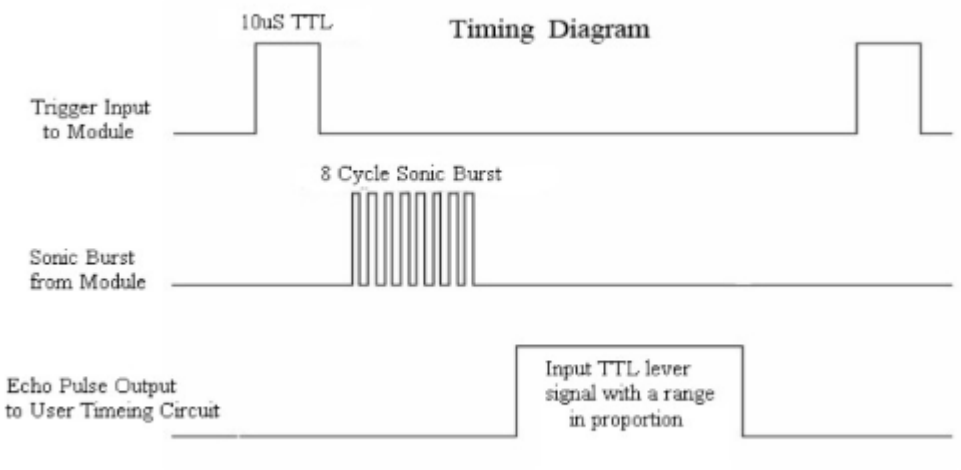
- Vérifie que le fichier binaire **ultrason** a bien été créé, sinon recommence l'opération ; s'il y a des messages d'erreur ; ceux-ci doivent t'aider à corriger le code source

Test du programme « ultrason »

- teste le programme en lançant la commande « ./ultrason 0 », tout en plaçant ton autre main à 10cm environ du capteur. Quelle est la valeur renvoyée par le programme ?
- Fais quelques tests supplémentaires en modifiant la distance entre ta main et le capteur. La valeur renvoyée par le programme varie-t-elle bien en fonction de cette distance ?
- Utilise une règle graduée pour mesurer la distance entre le capteur et l'obstacle (tu peux utiliser une feuille de papier ou un autre objet de ton choix) ; puis fais une série d'expériences de façon à remplir le tableau suivant :

Distance mesurée par la règle (cm)	30	25	20	15	10	8	6	4	2	0
Distance mesurée par le capteur (cm)										

ANNEXE : Détecteur d'obstacle à ultrason SR04



Connexion au RPi :

- la patte Vcc est relié au +5V (fil rouge)
- la patte GND est reliée à GND (fil noir)

ANNEXE : Détecteur d'obstacle à GROVE Ultrasonic Ranger

- la patte Vcc est relié au +5V (fil rouge)
- la patte GND est reliée à GND (fil noir)
- la patte SIG est reliée au GPIO BCM=17, wPi=0 (fil jaune)

Listing du code source ultrason.c

```
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <wiringPi.h>

// la fonction  divRoundClosest permet de faire une division en arrondissant à l'unité
unsigned int divRoundClosest(const int n, const int d)

{
  return ((n + d/2)/d);
}
// la fonction  help
void help(void)
{
            printf("Syntaxe : ultrason n°_pin_SIG \n");
}
// la programme principal
int main (int argc, char *argv[])
{
    int start, stop , dist, diff, SIG;

 if (wiringPiSetup () == -1)
   {
     fprintf (stdout, "Erreur gpio : %s\n", strerror (errno)) ;
     return 1 ;
   }

  piHiPri (99);

   if (argc>1)
  {
            SIG=atoi(argv[1]);
            if ( SIG > 29 || SIG<0 ) {help(); exit (1);}

  } else {exit (1);}

pinMode(SIG, OUTPUT);

//10us pulse on trig
digitalWrite(SIG, 1);
delayMicroseconds(10);
digitalWrite(SIG, 0);

// SIG en mode INPUT pour capter la reponse
pinMode(SIG, INPUT);

//wait for ECHO being 1
while (digitalRead(SIG) == 0)    {}
start = micros();

//wait for ECHO being 0
while (digitalRead(SIG) == 1)    {}
stop = micros();
diff = stop-start;

//calcul de la distance en cm
dist = divRoundClosest(diff,58);
 if (dist > 400 )   {  dist = -1;   }
printf("%d",dist);
return dist;
}
```