Programmation de la maquette Arduino au Collège :

Utilisation du module ultrason.

1) Présentation du module ultrason HC-SR04.

Un module ultrason est un ensemble constitué d'un **émetteur** ultrason, d'un **récepteur** ultrason et de toute l'électronique (quartz, temporisateur, circuit de commande, amplificateur, etc...) permettant d'envoyer par l'émetteur une salve ultrasonore et de mesurer le temps mis par cette salve pour atteindre le récepteur. Dans notre cas l'émetteur est situé à côté du récepteur, ce qui nous indique que **ce module est destiné à mesurer une distance avec un obstacle** (sur lequel va « rebondir » la salve ultrason) et que donc, le trajet fait un allé (entre l'émetteur et l'obstacle) et retour (entre l'obstacle et le récepteur).

- Il faudra donc **prendre en compte cet allé-retour** dans nos calculs
- Voir un site de vente de ce module : http://www.gotronic.fr/art-module-de-detection-us-hc-sr04-20912.htm

2) Questions génériques

2.1) Donner la définition d'un ultrason.

2.2) Quelles sont (en général) les fréquences audibles par l'être humain ? Pourquoi les ultrasons ne sont-ils pas audibles par l'humain ?

2.3) Quelle est la fréquence de ce module et pourquoi considère t'on qu'il est ultrasonore ?

2.4) Quelle est sa portée (ou plage de mesure) ?

2.5) Quel est le calcul approché (donné par la documentation) permettant de déterminer la distance de l'obstacle en fonction du temps mis par l'onde pour faire le trajet ?

RQ : Nous ferons le véritable calcul à la fin de cette séquence.

2.6) Quel est le prix de ce module ?

2.7) Quel est le nombre de fils nécessaire à l'utilisation de ce module ?

3) Exercice n°1 : Implanter un programme de test dans la maquette et vérifier son fonctionnement.

L'ensemble des actions à faire avec la maquette se fait grâce au logiciel « Arduino » situé sur votre bureau.

1 **Lancez le logiciel** « Arduino » à partir de votre bureau.

2 **Ouvrez le fichier :** « Sequence4.ino » situé dans le sous-dossier « Sequence4 » de votre dossier « Technologie/3eme ».

3 Repérez-vous dans le programme :

=> **Lisez la totalité des commentaires.**

=> Repérez les deux sous-programmes « setup » et « loop ».

4 **Connectez la maquette sur un port USB de l'ordinateur.** Au bout de quelques secondes la maquette est reconnue et le pilote activé.

5 **Paramétrez le port USB de programmation.**

6 Faites le « téléversement ».

7 **Testez le fonctionnement du programme** en plaçant un obstacle à une vingtaine de centimètres en face du module et en regardant l'afficheur.

Que réalise ce programme ?

Validation du professeur :

4) Exercice 2 : Analyse de la structure du programme pour réaliser une première modification du fonctionnement.

4.1) Cahier des charges de l'exercice :

Nous allons vous aider à faire allumer la LED jaune lorsque le temps total de parcours et inférieur à 2000µs.

4.2) Analyse du programme :

Le programme à changer se situe dans la partie « loop » (la partie du programme qui boucle en permanence).

```
void loop()                                  //Programme bouclé
{
  TempsDeParcours = Capteur_ultrasonic.Timing();   //Mesure du temps de parcours

  AfficheurLCD.Posxy(1,2);                   //Placement du curseur sur la position "1" de la ligne "2"
  AfficheurLCD.print(TempDeParcours);        //Affichage de valeur de la variable "TempsDeParcours"
  AfficheurLCD.println("us");                //Affichage de l'unité "µs"

  delay(100);                                //Attente de 100ms
}
```

- La première ligne de programme mesure le temps de parcours de la salve ultrason et place le résultat dans la variable « TempsDeParcours ».
- Les trois lignes de programme suivantes placent le curseur de l'afficheur au bon endroit, affichent le temps de parcours puis l'unité.
- Il faut donc ajouter à la suite de ces quatre lignes (avant le « delay(100); ») une fonction « if » qui ressemble à celui de la séquence 1 :

```
if(digitalRead(BP_Rouge)==1)                 //Si la lecture de "BP_Rouge" donne "1" (on appuis sur le BP) :
 {
   digitalWrite(LED_Rouge, HIGH);          //Alors je place la "LED_Rouge" à l'état haut (HIGH)
 }
 else                                        //Sinon (si la lecture de "BP_Rouge" donne "0") :
 {
   digitalWrite(LED_Rouge, LOW);           //Alors je place la "LED_Rouge" à l'état bas (LOW)
 }
```

Copie de la fonction « if » de la séquence 1

4.3) Adaptation du programme au nouveau cahier des charges :

1 Pour notre cas, il faut placer une structure en « if » (comme celle de la séquence 1, ci-dessus), mais au lieu de lire l'état du bouton rouge : *if(digitalRead(BP_Rouge)==1)*, il faut lire si la variable « TempsDeParcours » est inférieure à 2000µs (cahier des charges de l'exercice).

=> Remplacez : *if(digitalRead(BP_Rouge)==1) par : if(TempsDeParcours<2000).*

Il faut aussi changer la LED à allumer en remplaçant LED_Rouge par LED_Jaune.

2 Faites les modifications puis le « téléversement ».

3 Si le programme répond au cahier des charges, **Collez une copie d'écran de votre programme ci-dessous** (ne garder seulement que les parties « setup » et « loop », avec les commentaires de droite).

Validation du professeur :

PRENONS DE L'AUTONOMIE

Jusqu'à présent, nous vous avons guidés pour faire les modifications. Dans cette deuxième partie vous devrez faire preuve d'analyse et d'initiatives pour faire les modifications demandées. Il n'est pas grave ni inquiétant de ne pas réussir du premier coup. Le résultat (et donc la note) vient en grande partie de la persévérance dans la recherche de solutions.

5) EXERCICE 3 :

5.1) CAHIER DES CHARGES DE L'EXERCICE ;

Ajoutez l'allumage de la LED bleue lorsque le temps total de parcours et inférieur à 1000µs.

5.2) ANALYSE ET MODIFICATION DU PROGRAMME

1 **Analysez puis modifiez le programme.**

2 **Faites le « téléversement ».**

3 **Vérifiez le fonctionnement du programme**.

4 Si le programme répond au cahier des charges, **Collez une copie d'écran de votre programme ci-dessous** (ne garder seulement que les parties « setup » et « loop », avec les commentaires de droite).

Validation du professeur :

6) Exercice 4 :

6.1) Cahier des charges de l'exercice :

Ajouter l'allumage des LEDs verte et rouge lorsque le temps total de parcours devient respectivement inférieur à 500µs puis à 250µs.

6.2) Analyse et modification du programme

1) Analysez puis modifier le programme.

2) Faites le « téléversement ».

3) **Vérifiez le fonctionnement du programme**.

4) Si le programme répond au cahier des charges, **Collez une copie d'écran de votre programme ci-dessous** (ne garder seulement que les parties « setup » et « loop », avec les commentaires de droite).

Validation du professeur :

POUR ALLER PLUS LOIN...

Dans cette troisième partie vos capacités d'analyse et de concentration devront être fortes. Il est donc normal de passer davantage de temps pour réussir une question. Il faut rester focalisé sur ce qui est demandé, bien analyser la situation, faire des hypothèses et des essais, puis essayer de comprendre ce qui a fonctionné et ce qui n'a pas répondu à votre attente pour pouvoir recommencer et, à force de volonté, réussir. Il n'est pas grave ni inquiétant de ne pas réussir du premier coup. Le résultat (et donc la note) vient en grande partie de la persévérance dans la recherche de solutions.

5) EXERCICE 5 :

5.1) CAHIER DES CHARGES DE L'EXERCICE :

Pour plus de précisions et de stabilité faire une moyenne sur 10 mesures avant d'afficher le temps et les LEDs.

5.2) ANALYSE ET MODIFICATION DU PROGRAMME

Ressources :

- *Pour pouvoir faire une moyenne sur 10 mesures il faut créer une variable dans laquelle nous allons ajouter les mesures une par une. (Nous diviserons cette somme par 10 à la fin).*

  Exemple : ***float SommeDesTempsDeParcours = 0;*** *//Déclaration de la variable*

- *Ensuite pour faire un nombre fixe de mesure, nous allons utiliser une boucle « for » (les explications sur cette boucle sont dans l'exercice 5 de la séquence 3).*

  Exemple : ***for (i = 0 ; i < 10 ; i++)***

  {

  TempsDeParcours = Capteur_ultrasonic.Timing(); //Mesure du temps de parcours

  SommeDesTempsDeParcours = SommeDesTempsDeParcours+TempsDeParcours;

  delay(10); //Le delay permet au module d'avoir le temps de relancer une salve

  }

  Remarque : Ne pas oublier de déclarer la variable i avant de faire la boucle « for »

  Exemple : ***int i = 0 ;*** *//Déclaration de la variable i*

- *Attention, la variable « SommeDesTempsDeParcours » contient à la fin de la boucle « for » la somme de 10 mesures de temps de parcours. Il faudra donc la diviser par 10 pour avoir la moyenne.*

  Exemple : ***TempsDeParcours = (SommeDesTempsDeParcours/10);***

- *Il ne reste plus qu'a l'afficher.*

- *Attention, à la fin du programme, avant de recommencer la boucle de 10 mesures, il faut remettre la variable « SommeDesTempsDeParcours » à zéro...*

1 Analysez puis modifiez le programme.

2 **Faites le « téléversement ».**

3 **Vérifiez le fonctionnement du programme**.

4 Si le programme répond au cahier des charges, **Collez une copie d'écran de votre programme ci-dessous** (ne garder seulement que les parties « setup » et « loop », avec les commentaires de droite).

Validation du professeur :

6) Exercice 6 : Calcul de la distance entre le module et l'obstacle,

6.1) Comment calculer la distance en fonction du temps de parcours ?

6.1.1) D'après le site : https://fr.wikipedia.org/wiki/Vitesse_du_son#En_fonction_de_l.27altitude donner la vitesse du son à 1000 mètre d'altitude (colonne : c en m/s).

6.1.2) Déterminer par quel coefficient il faut multiplier pour passer de 336,4 « mètre par seconde » à « cm par seconde ». Que donne alors la vitesse des ultrasons en cm/s ?

6.1.3) Déterminer par quel coefficient il faut diviser pour passer de 33640 « cm par seconde » à « cm par micro-seconde ». Que donne alors la vitesse des ultrasons en cm/µs ?

6.1.4) Pour rappel, le trajet fait un allé-retour. En déduire le coefficient multiplicateur qu'il faudra utiliser pour tenir compte de cet allé-retour dans la vitesse globale.

Que donne alors la vitesse globale (allé-retour) des ultrasons en cm/µs ?

6.2) Cahier des charges de l'exercice :

Modifier le programme pour afficher en haut "Distance obst:" (après la phase d'initialisation), **puis la distance en cm sur la ligne du bas**.

6.3) Analyse et modification du programme

- *Les calculs précédents doivent donner un résultat de 0,0168 cm par micro-secondes.*
- *Aidez-vous de votre travail de physique : D = V x T*

1 **Analysez puis modifiez le programme.**

2 **Faites le « téléversement ».**

3 **Vérifiez le fonctionnement du programme**.

4 Si le programme répond au cahier des charges, **Collez une copie d'écran de votre programme ci-dessous** (ne garder seulement que les parties « setup » et « loop », avec les commentaires de droite).

Validation du professeur :

7) AJOUT DE LA SÉQUENCE 4 À VOTRE CLASSEUR NUMÉRIQUE

A la fin de chaque séquence, vous devrez intégrer votre fichier de la séquence finie (le fichier « activites.odt » dans lequel vous avez travaillé) dans votre classeur numérique.

Pour cela vous devez :

- Générer un fichier PDF à partir de votre fichier traitement de texte.
- Intégrer la séquence 4 (que vous venez de générer) à votre classeur numérique déjà existant.

Pour vous aider, vous avez le fichier « Classeur_numerique.pdf » présent dans les ressources de la séquence 0 du projet 1 (ou au début de votre classeur numérique).