Séance n°3
Activité numérique

Capacités exigibles (BO 1ère) :

« Déterminer une distance à l'aide d'un microcontrôleur. »
Mesurer et traiter un signal au moyen d'une interface ou d'un microcontrôleur.

Objectif : Programmer le microcontrôleur d'un télémètre à ultrasons pour déterminer et afficher une distance.

Situation déclenchante :

SANTÉ ET SECURITÉ
Réduire les risques des quais de chargement.

Les signaux sonores sont utiles dans un véhicule pour alerter le conducteur de la proximité d'un obstacle mais cela ne suffit pas aux chauffeurs de poids lourds qui, lorsqu'ils reculent, doivent positionner l'arrière de leur camion à quelques centimètres seulement du quai de chargement ! Certes les webcams arrière qui équipent aujourd'hui leur véhicule sont pratiques, mais connaître la distance (au centimètre près) leur serait d'une grande aide complémentaire.

Problématique 3 :

Comment mesurer et afficher la distance entre l'arrière du camion et le quai ?

Programmation :

Le microcontrôleur d'un télémètre à ultrasons mesure des durées de propagation d'une onde entre son émission et sa réception par le capteur du dispositif.
Avec le **programme_2**, le microcontrôleur ARDUINO™ ordonne l'affichage de cette durée.

Le **but** de cette séance est de modifier le **programme_2** afin de **mesurer** et d'**afficher** la **distance** entre le capteur et l'obstacle.

Partie A Analyse rapide d'un langage de programmation (Compétences S'APPROPRIER, ANALYSER, RAISONNER)

1) Prends connaissance du programme Arduino ci-dessous téléversé dans le microcontrôleur de ta maquette **(Programme_2)**.
2) Repère et **encadre** les différents blocs de programmation du **Programme_2** (même logique de structure que le langage Python déjà rencontré en 2nde).
3) Repère et **entoure** les différ

```
#include <LiquidCrystal.h>                  // on importe la bibliothèque.
LiquidCrystal EcranLCD(12, 11, 5, 4, 3, 2);  // on crée l'objet EcranLCD, c'est l'afficheur LCD de la maquette PB100,
                                            // il est relié aux broches 12, 11, 5, 4, 3 et 2 de l'Arduino.

void setup()                          // La fonction setup s'execute une seule fois au démarrage du programme.
{
    EcranLCD.begin(16, 2);            // on initialise la communication entre l'Arduino et l'afficheur (16 colonnes et 2 lignes).
    pinMode(8, OUTPUT);               // La broche 8 de la carte Arduino est reliée à l'Emetteur "E" (déclencheur de salves d'ondes ultrasonores).
    pinMode(9, INPUT);                // La broche 9 de la carte Arduino reliée au Récepteur "R" (reçoit et mesure le temps de vol de la salve émise par E).
    digitalWrite(8, LOW);
}

void loop()                           // La fonction loop s'execute en boucle.
{
   digitalWrite(8, HIGH);             // met la broche 8 TRIGGER à l'état haut pendant 10 microsecondes, puis envoi une salve d'ondes ultrasonores.
   delayMicroseconds(10);
   digitalWrite(8, LOW);              // met la broche 8 TRIGGER à l'état bas, en attendant la prochaine boucle.

   unsigned long dureear = pulseIn(9, HIGH);    // Le dispositif reçoit l'ECHO de la salve sur la broche 9, mesure le temps de vol Aller-Retour en microsecondes,
                                                // et enregitre la valeur dans la variable nommée ici "dureear"(=durée aller retour).
```

entes positions du paramètre physique « t » mesuré et affiché sur l'écran LCD de la maquette PB100.

```
EcranLCD.clear();                     // Efface l'écran
EcranLCD.setCursor(0, 0);             // Positionne le curseur Colonne 0, Ligne 0
EcranLCD.print("temps A-R onde");     // Ecrit "temps A-R onde"
EcranLCD.setCursor(0, 1);             // Positionne le curseur Colonne 0, Ligne 1
EcranLCD.print(" t = ");              // Ecrit " t = "
EcranLCD.setCursor(5, 1);             // Positionne le curseur Colonne 5, Ligne 1
EcranLCD.print(dureear/1000.0);       // Afiche la mesure du temps de vol de l'onde en millisecondes.
EcranLCD.setCursor(10, 1);            // Positionne le curseur Colonne 10, Ligne 1
EcranLCD.print("ms");                 // Ecrit l'unité de la mesure "ms"
delay(1000);                          // Petite pause de 1 seconde avant de faire une autre mesure (= une autre boucle ici).
}
```

Partie B **Expérimentation numérique (Compétences REALISER / RAISONNER / VALIDER)**

4) Propose une **modification** du **Programme_2** pour **calculer** et **afficher** sur l'écran LCD **la distance d (en cm)** qui sépare la maquette d'un obstacle.

APPEL n°1 : APPELLE LE PROFESSEUR
POUR LUI PRÉSENTER LES MODIFICATIONS OU AJOUTS QUE TU PROPOSES
OU EN CAS DE DIFFICULTÉS

5) Modifie le **Programme_2**, renomme-le **Programme_3_NOM**.
6) Téléverse ton nouveau **Programme_3_NOM** dans la maquette et valide (ou recommence !) ton travail.

APPEL n°2 : APPELLE LE PROFESSEUR
POUR LUI MONTRER LE BON FONCTIONNEMENT DE TA MAQUETTE

Partie C	Expérimentation numérique (Compétences RAISONNER / RÉALISER / VALIDER)	Pour aller plus loin …

7) Propose une **modification** du **Programme_3_NOM** pour ajouter le **déclenchement d'un signal lumineux** si l'obstacle est trop près.
8) Modifie le **Programme_3_NOM**, renomme-le **Programme_31_NOM**.
9) Téléverse ton nouveau Programme dans la maquette et valide (ou recommence) ton travail.

APPEL n°3 : APPELLE LE PROFESSEUR
POUR LUI MONTRER LE BON FONCTIONNEMENT DE TA MAQUETTE

10) Propose une **modification** du **Programme_31_NOM** pour ajouter le **déclenchement d'un signal sonore** si l'obstacle est trop près.
11) Modifie le **Programme_31_NOM**, renomme-le **Programme_32_NOM**.
12) Téléverse ton nouveau Programme dans la maquette et valide (ou recommence) ton travail.

APPEL n°4 : APPELLE LE PROFESSEUR
POUR LUI MONTRER LE BON FONCTIONNEMENT DE TA MAQUETTE