

# Hatice DAGLI

## Inspiration

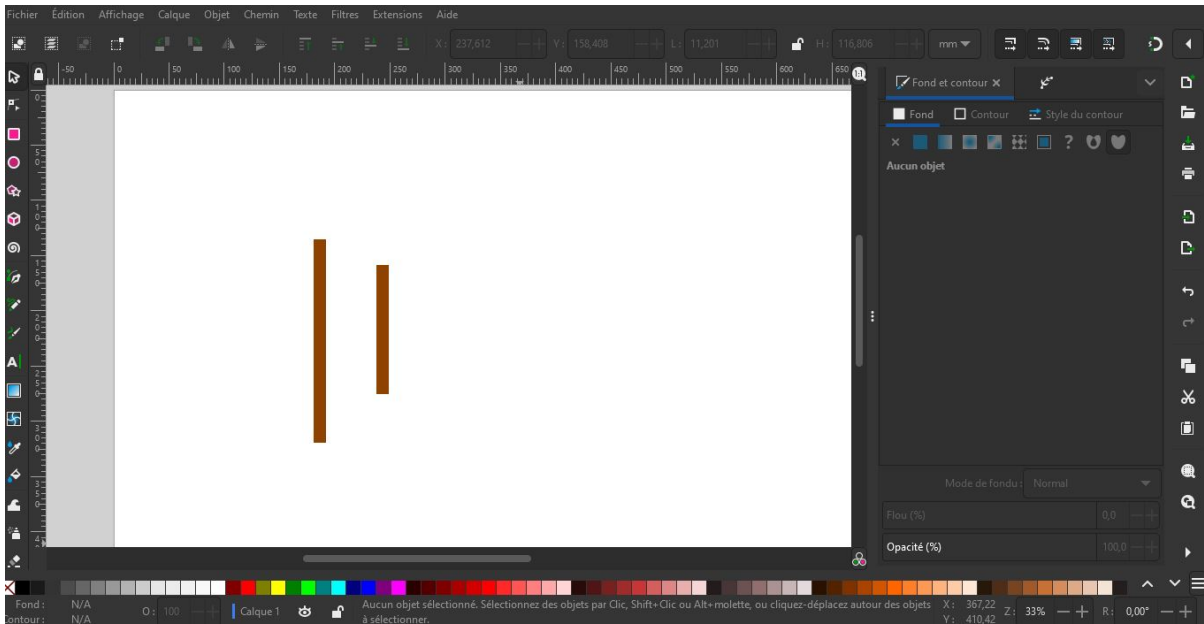
Pour créer mon porte savon, je me suis inspirée de ce modèle :



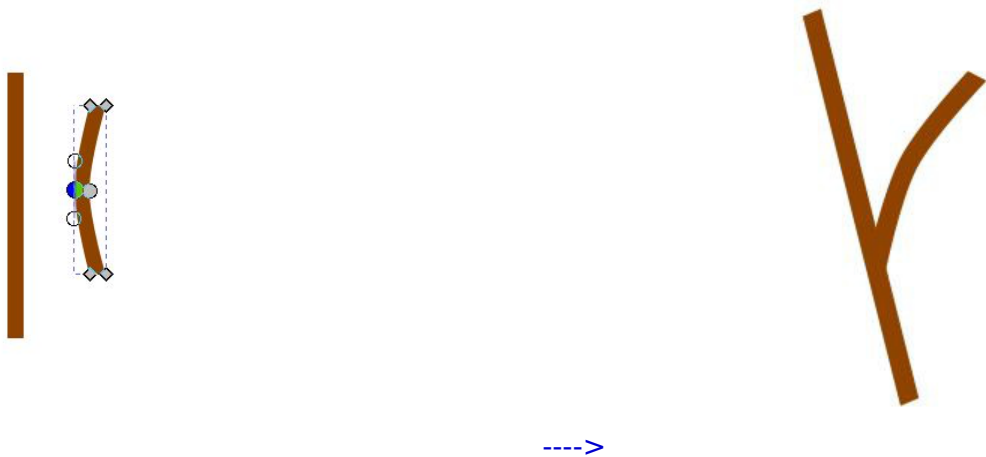
## Modélisation 2D

Toutefois, je souhaitais faire quelque chose de plus minimaliste, je me suis donc lancée à dessiner une plante sur **Inkscape** :

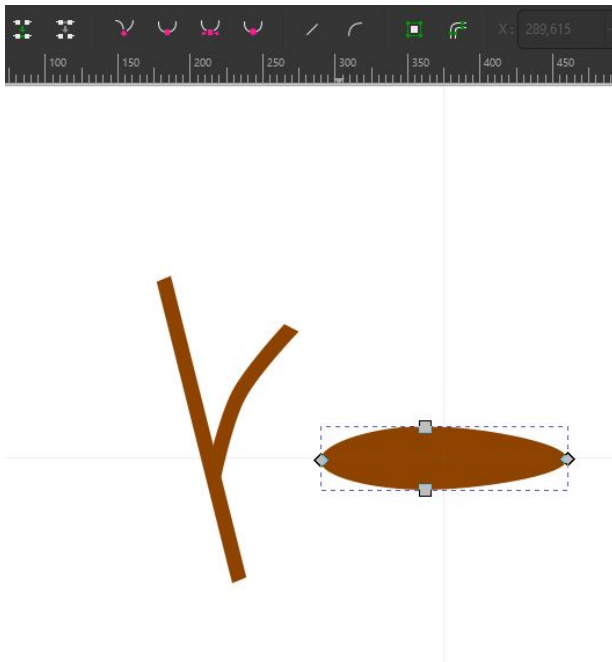
Je commence par les branches avec des rectangles fins sans contours



J'incurve ensuite l'une des deux branches afin de donner cet aspect naturel via l'outil "objet en chemin" en choisissant le point qui fera une belle courbe arrondie :



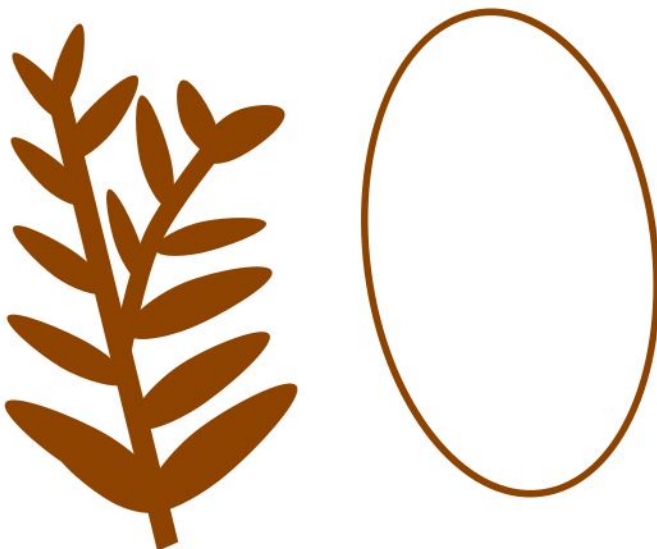
Ensuite, j'ajoute les feuilles via l'ajout d'un élément rond que j'incurve de la même façon que la branche :



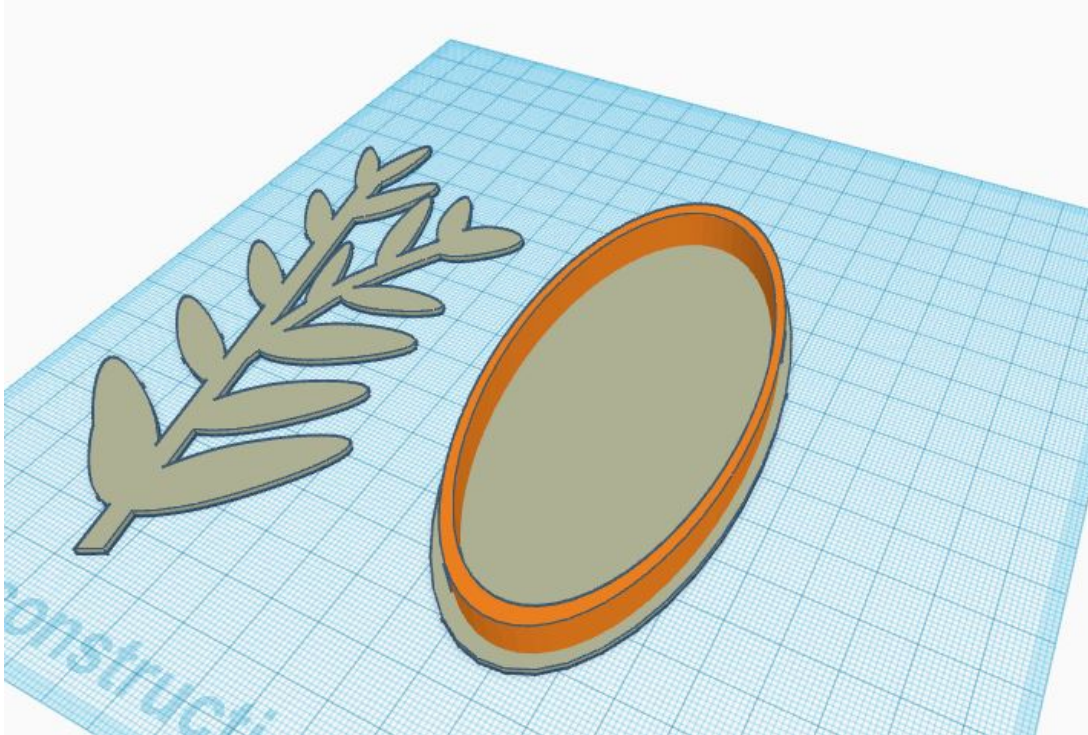
Je copie colle cette feuille plusieurs fois en plusieurs taille pour obtenir ma petite plante :



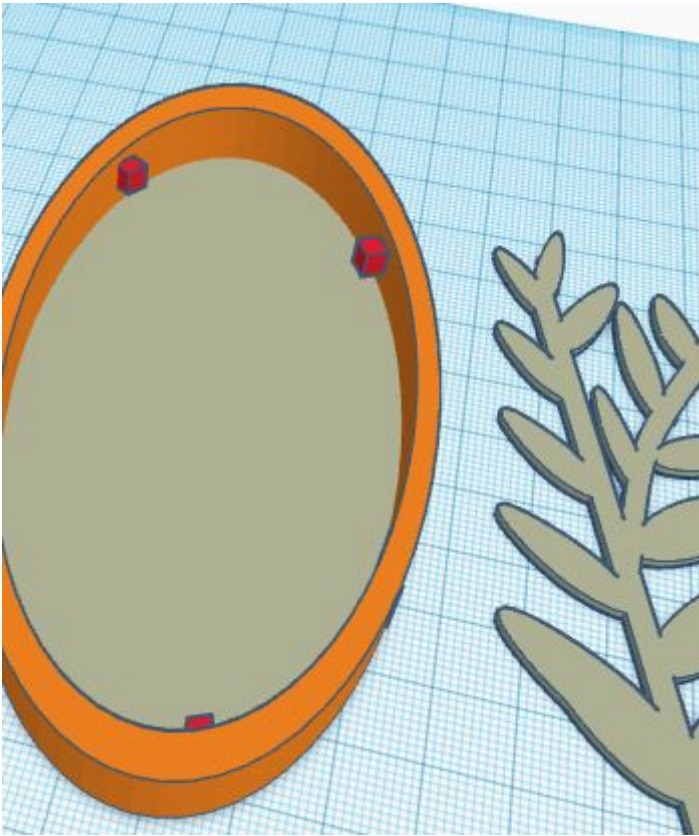
Je fais également sur Inkscape un ovale à la même dimension que ma plante afin de le mettre en 3D sur Thinkedcad (bien que ce ne fut pas l'idéal mais je détaille plus bas pourquoi).



# Modélisation 3D



Sur **Thinkercad**, j'ai commencé par posé un cylindre que j'ai mis en ovale sur mon support de base vert.



J'ai ensuite ajouté 3 petits blocs de carré afin de faire tenir "ma grille" qui n'est autre que ma plante.

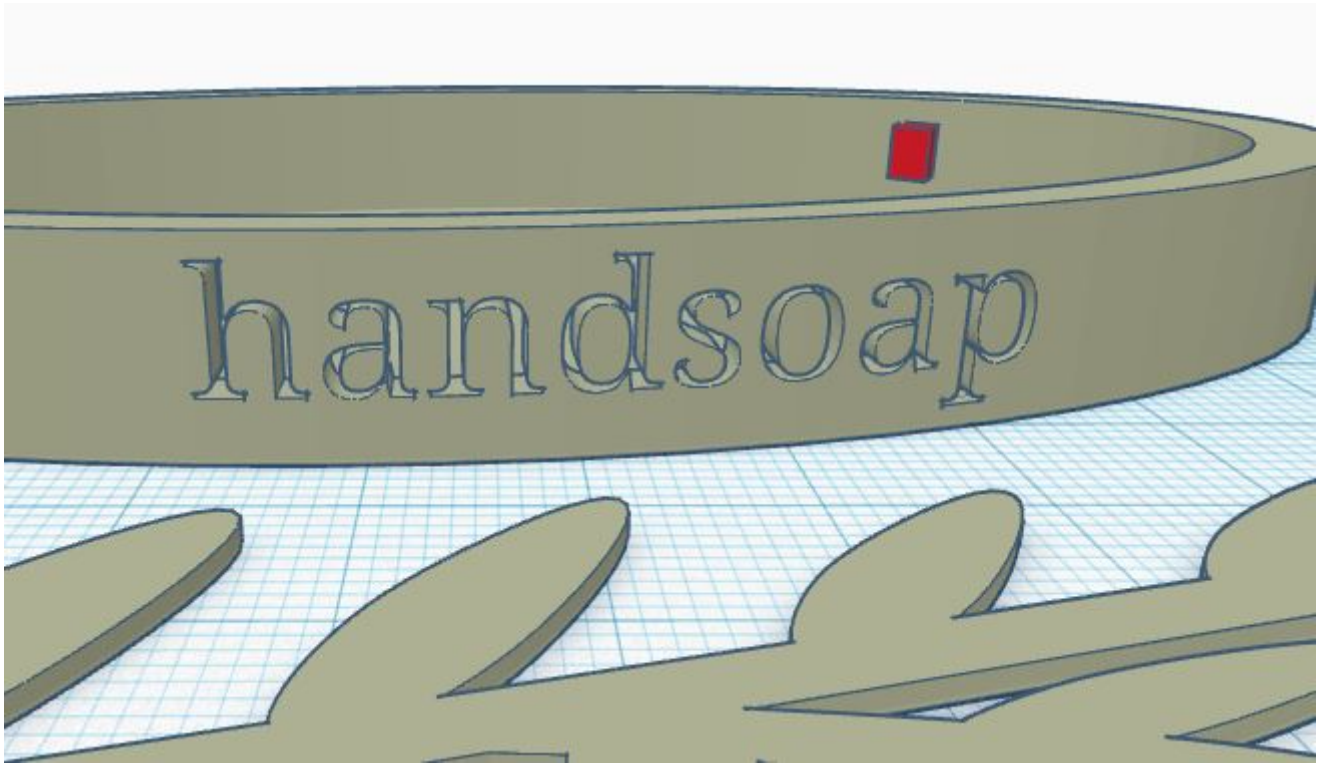
Problème : Ici, je me suis rendue compte que faire mon support de la même taille que ma plante n'était pas une bonne idée car ma plante n'aurait pas tenu sur le support sans aucun élément qui le tient en place. Il fallait donc réduire sa taille. Egalement, j'ai aussi appris qu'il était inutile que j'importe la base de ce support ovale depuis Inkscape car je pouvais le réaliser sur Thinkercad directement et plus facilement avec un élément ovale que j'aurais troué par perçage.

Enfin, j'ai décidé d'ajouter un espace par laquelle je pourrais verser l'eau qui s'accumulerait tout en ajoutant une touche décorative avec une écriture :

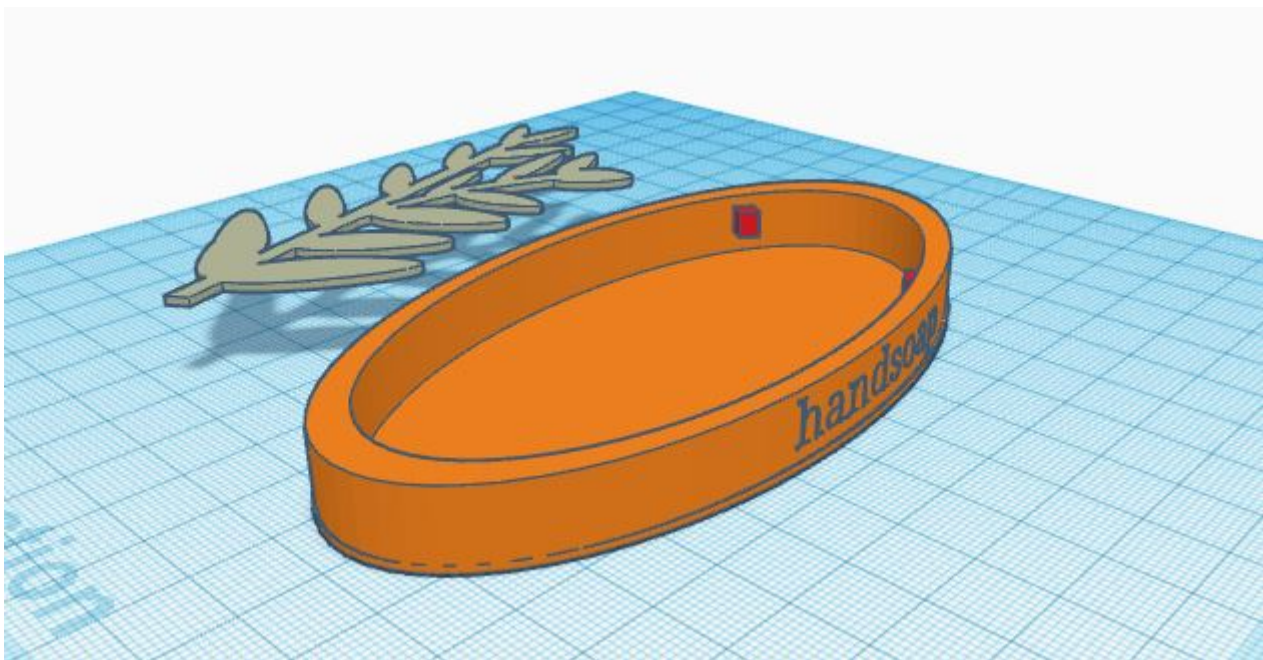


Que j'ai ensuite mis en perçage :





Et voici le plan du projet :



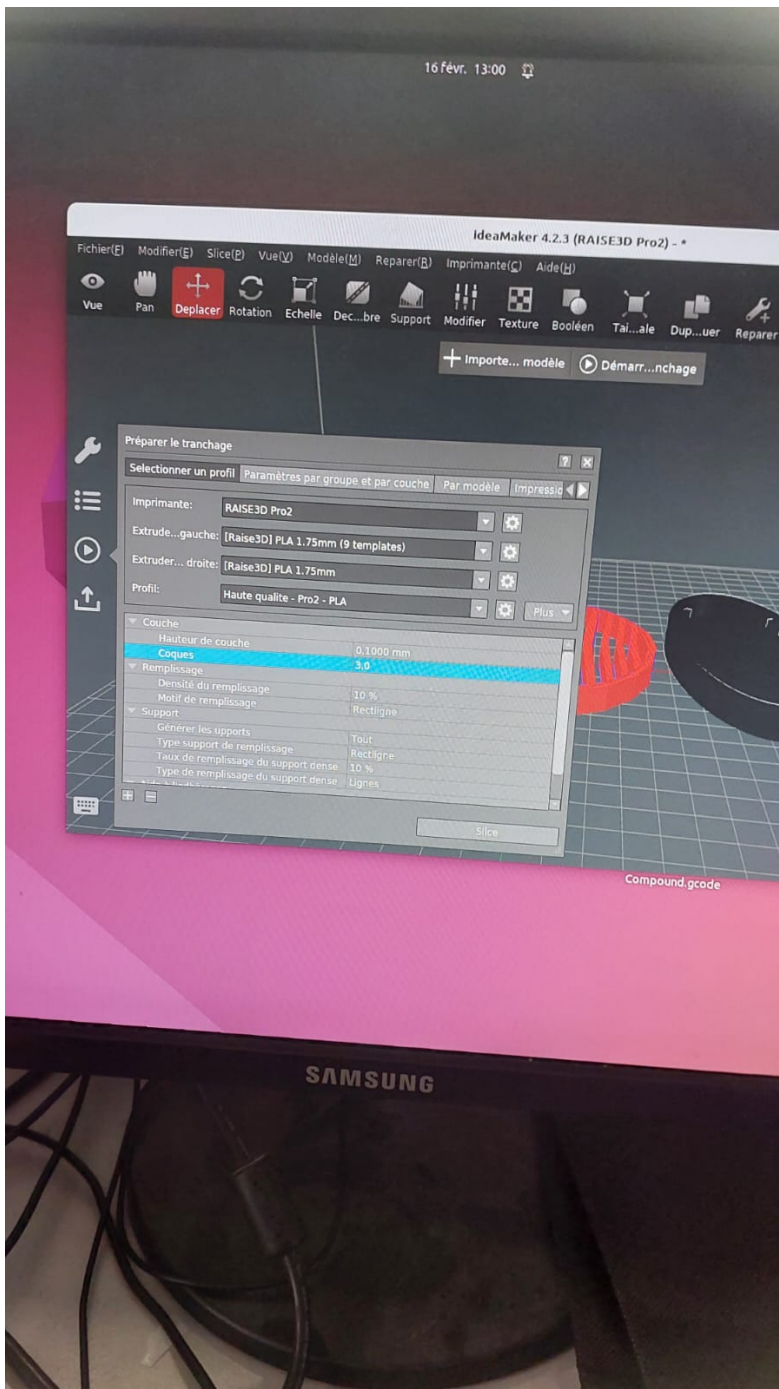
Exportation sous .SVG

Problème : Je me suis rendue compte sur IdeaMaker que les lettres o, a et p n'allaient peut-être pas bien sortir et que j'aurais du penser à ajouter un élément d'union pour faire tenir ces petits blocs.

## Impression au FabLab SU

Pour pouvoir imprimer, il faut passer nos fichier en .GCODE via **IdeaMaker**

Attention : il faut bien mettre tous les éléments à plat (ma plante était en hauteur ici)



Nous réglons les paramètres et enregistrons le fichier pour enfin insérer la clé USB dans l'imprimante 3D.

Temps estimé : 11h d'impression

# Résultat final

Remarques générales : Au fur et à mesure du projet, surtout lors de la modélisation 3D, je me suis rendue compte que ma grille plante n'était pas très pertinente car il était assez difficile de trouver un moyen simple de le faire tenir sur le support. En effet, j'aurais du réfléchir dès la modélisation 2D à comment je planifiais designer mon support.

(fichier gcode trop lourd pour mettre ici)

```
#include <LiquidCrystal.h> // inclusion de la librairie LCD
```

```
// initialise la librairie LCD avec les broches utilisées
```

```
LiquidCrystal lcd(1, 2, 4, 5, 6, 7); // déclare une variable LiquidCrystal appelée lcd avec mode 4 bits  
- RW non connectée (le plus simple!)
```

```
//déclaration des nouveaux caractères
```

```
//création du caractère  $\mu$ 
```

```
byte micro[8]={
```

```
  B10001,  
  B10001,  
  B10001,  
  B10011,  
  B11101,  
  B10000,  
  B10000,  
  B10000
```

```
};
```

```
//création du caractère petit s de seconde en position haute pour tomber en face du  $\mu$  lors de  
l'affichage.
```

```
byte seconde[8]={
```

```
  B01110,  
  B10000,  
  B01110,  
  B00001,  
  B01110,  
  B00000,  
  B00000,  
  B00000
```

```
};
```

```
const int trigPin = 9; // déclare une variable à valeur constante de type const int (variable qui ne  
peut être modifiée, juste en lecture seule) appelée trigPin et valant 9 trig (Trigger = déclancheur)  
connectée à la broche numérique n° 9
```



```
const int echoPin = 10; // déclare une variable à valeur constante de type const int (variable qui ne peut être modifiée, juste en lecture seule) appelée echoPin et valant 10
trig echo (Echo = récepteur) connectée à la broche numérique n° 10
long duration; // déclare une variable de type long appelée duration (Taille de 32bits).
```

```
// Instruction d'initialisation de la carte Arduino
//***** FONCTION SETUP = Code d'initialisation *****
// La fonction setup() est exécutée en premier et 1 seule fois, au démarrage du programme
```

```
void setup() {
```

```
  //Attribution des numéros des caractères
  lcd.createChar(1, micro); // attribution d'un numéro au caractère créé
  lcd.createChar(2, seconde); // attribution d'un numéro au caractère créé.
```

```
  lcd.begin(16,2); // initialise le LCD en mode 16 colonnes x 2 lignes
```

```
  pinMode(trigPin, OUTPUT); //Configure la Broche spécifiée (ici la broche 9 digital) pour qu'elle se comporte en sortie avec OUTPUT.
```

```
  pinMode(echoPin, INPUT); //Configure la Broche spécifiée (ici la broche 10 digital) pour qu'elle se comporte en entrée avec INPUT.
```

```
}
```

```
//***** FONCTION LOOP = Boucle sans fin = coeur du programme *****
// la fonction loop() s'exécute sans fin en boucle aussi longtemps que l'Arduino est sous tension
```

```
void loop() {
```

```
  digitalWrite(trigPin, LOW);
  /*Met un niveau logique LOW (BAS en anglais) sur une broche numérique 9.
  Si la broche a été configurée en SORTIE avec l'instruction pinMode(), sa tension est mise à la valeur correspondante : 5V pour le niveau HAUT, 0V (masse) pour le niveau BAS.*/
```

```
  delayMicroseconds(2);
  //La fonction delay() permet de mettre en pause le programme pendant un certain nombre de millisecondes. C'est une fonction bloquante.
  //La fonction delayMicroseconds() accepte un unique paramètre obligatoire qui correspond à la durée en microsecondes de la temporisation. Cette fonction accepte uniquement des nombres entiers.
```

```
  digitalWrite(trigPin, HIGH);
  /*Met un niveau logique HIGH (HAUT en anglais) sur une broche numérique 9.
  Si la broche a été configurée en SORTIE avec l'instruction pinMode(), sa tension est mise à la valeur correspondante : 5V pour le niveau HAUT, 0V (masse) pour le niveau BAS.*/
```

```

delayMicroseconds(10);
//La fonction delay() permet de mettre en pause le programme pendant un certain nombre de
millisecondes. C'est une fonction bloquante.
//La fonction delayMicroseconds() accepte un unique paramètre obligatoire qui correspond à la
durée en microsecondes de la temporisation. Cette fonction accepte uniquement des nombres
entiers.

digitalWrite(trigPin, LOW);
/*Met un niveau logique LOW (BAS en anglais) sur une broche numérique 9.
Si la broche a été configurée en SORTIE avec l'instruction pinMode(), sa tension est mise à la valeur
correspondante : 5V pour le niveau HAUT, 0V (masse) pour le niveau BAS.*/

duration = pulseIn(echoPin, HIGH)/2;// affiche la durée du trajet entre l'émetteur et l'objet .

lcd.clear();// efface l'écran et met le curseur en haut à gauche
lcd.setCursor(0,0); // 1ère colonne - 1ère ligne - positionne le curseur à l'endroit voulu (colonne,
ligne) (1ère=0 !)
lcd.print("Temps: "); // affiche la chaîne texte - message de test (Ici Temps:)

lcd.setCursor(0,1);// 1ère colonne - 2ème ligne - positionne le curseur à l'endroit voulu (colonne,
ligne) (1ère=0 !)
lcd.print(duration);// affiche le calcul de la durée en µs entre l'émetteur et l'objet
lcd.print(" "); // affiche la chaîne texte - message de test (Ici un espace)
lcd.write(1); // affiche le caractère µ pour micro
lcd.write(2); // affiche le caractère s pour seconde

delay(5000);//La fonction delay() permet de mettre en pause le programme pendant un certain
nombre de millisecondes. C'est une fonction bloquante. Ici 5 secondes pour permettre d'avoir une
lecture stable du résultat.

}
// fin de la fonction loop() - le programme recommence au début de la fonction loop sans fin
//

```

---

Revision #6

Created 15 February 2023 09:43:12 by Dagli Hatice

Updated 6 March 2023 14:51:15 by Dagli Hatice