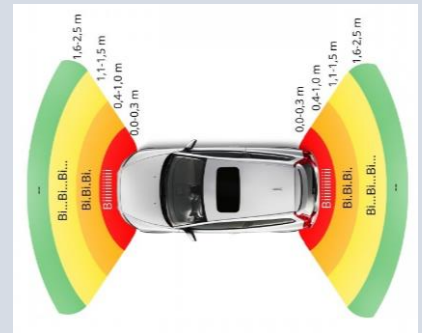


Les détecteurs de proximité sont devenus très fréquents sur les véhicules automobiles. Ils permettent d'avertir le conducteur de la présence d'un obstacle.

Ces détecteurs utilisent des ondes ultrasonores pour estimer la distance séparant le véhicule de l'obstacle.

La distance à l'obstacle est signalée sur l'écran du tableau de bord par l'indication d'une couleur (vert : tout va bien, rouge : danger) et par un signal sonore.



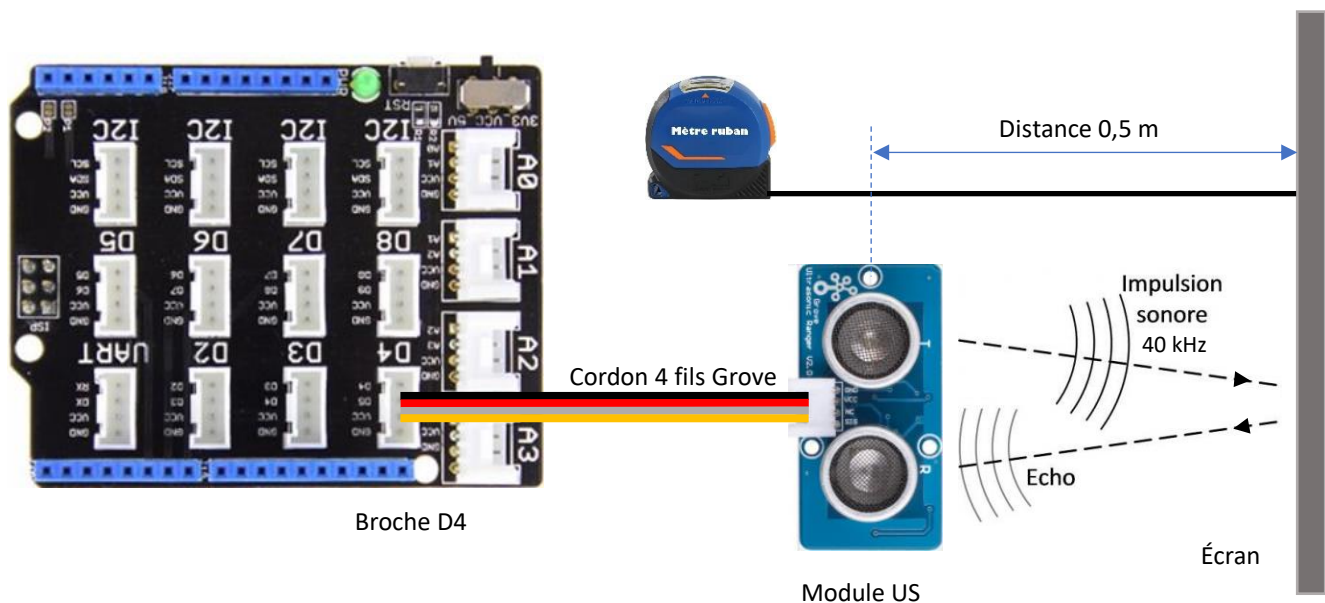
Problématique : Comment simuler le fonctionnement d'un détecteur de proximité ?

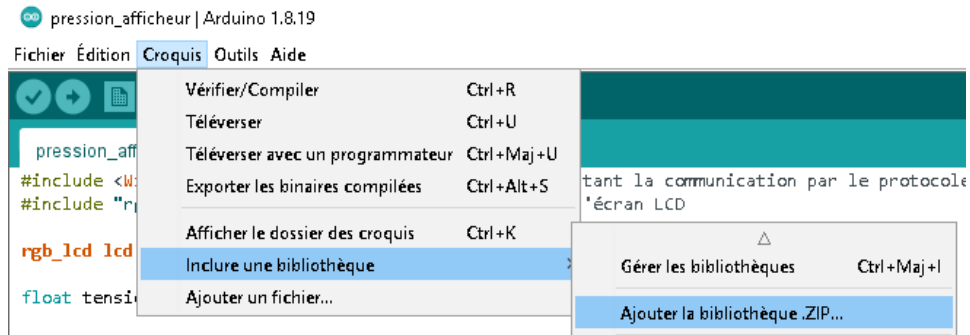
[Source de l'activité](#)

1. Ouvrir le fichier `distance_serie.ino` puis compléter la formule donnant la distance, en m, entre le module US et l'écran.

distance =

2. Réaliser le montage.

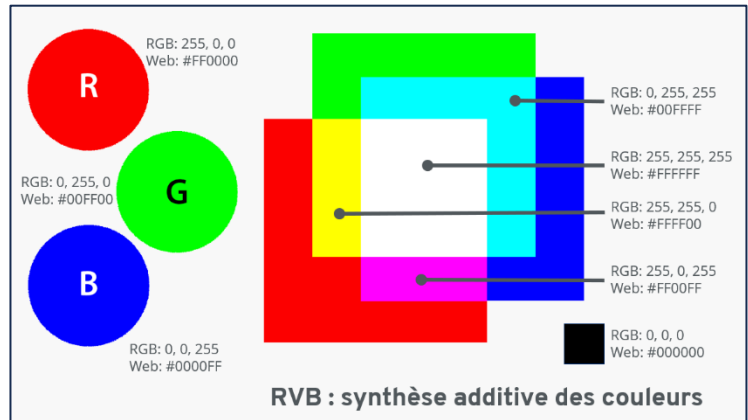




7. Ouvrir le fichier `distance_LCD.ino` puis **compléter** la formule donnant la distance, en m.

8. **Téléverser** le programme dans la carte puis **noter** la distance lue sur l'afficheur LCD.

9. **Justifier** que le rétroéclairage de l'afficheur LCD est de couleur blanche.



10. **Modifier** le programme pour qu'il prenne en compte cette consigne : « rétroéclairage rouge si distance inférieure à 0,2 m, sinon vert ». **Noter** vos modifications puis **vérifier** le rendu sur l'afficheur LCD.

```
//affichage :
if ( ..... ) {
  lcd.setRGB( ..... );
}
else {
  lcd.setRGB( ..... );
}
```

On peut ajouter un buzzer pour plus de réalisme !

11. **Brancher** le buzzer sur la broche D8 puis **ajouter** l'instruction suivante juste en dessous de la boucle principale.

```
void loop() {
  pinMode(8, OUTPUT); // buzzer sur D8 en sortie
```

12. **Préciser** le rôle des instructions suivantes puis les **ajouter** au bon endroit du programme.

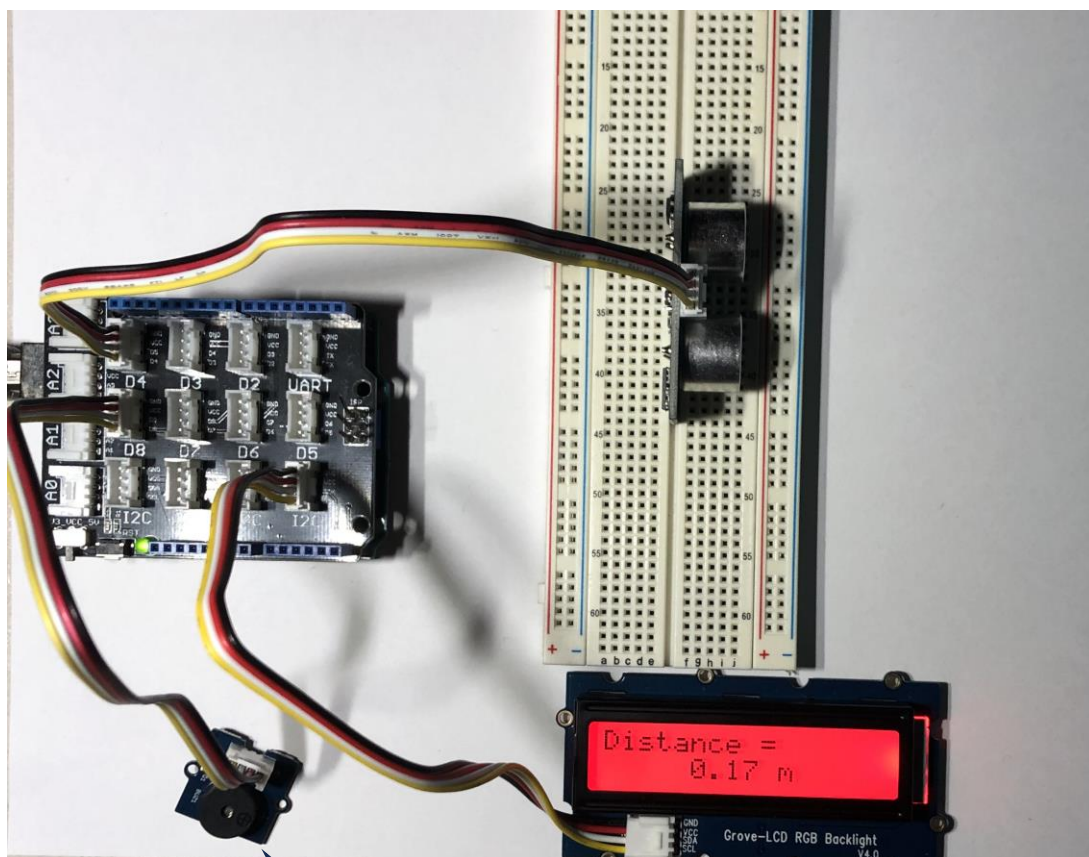
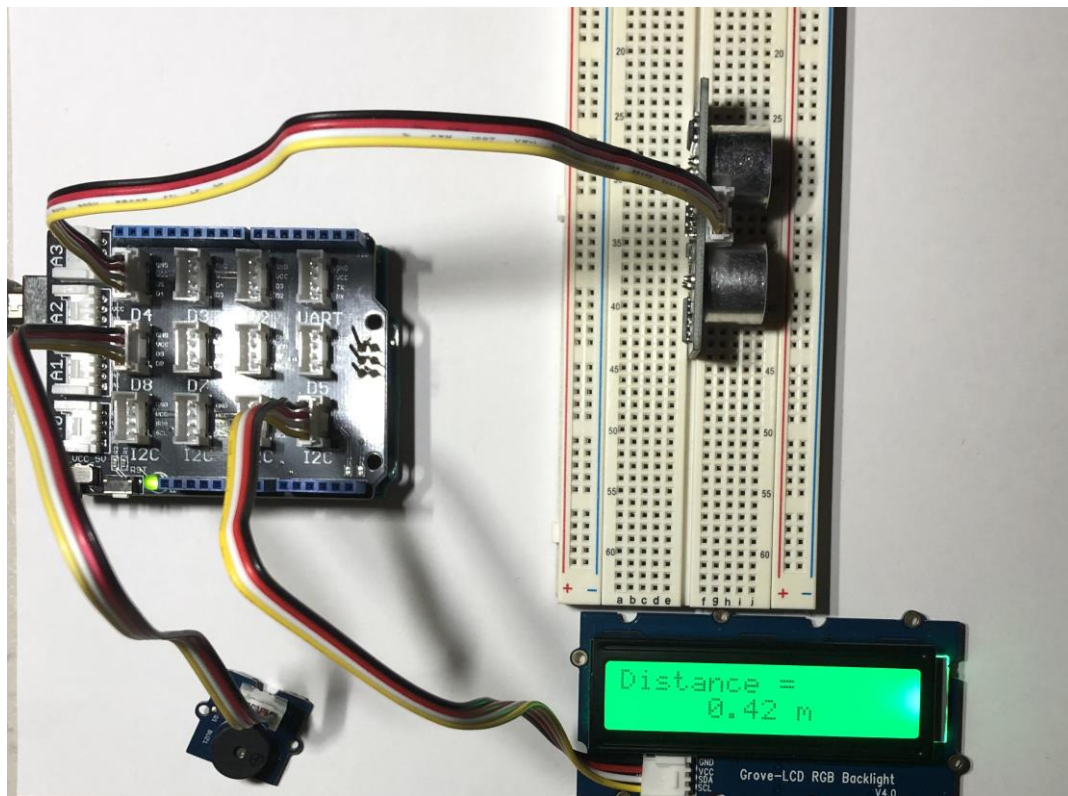
```
digitalWrite(8, HIGH);
delay(400);
digitalWrite(8, LOW);
delay(50);
```

13. **Tester** ce nouveau programme. 😊

Une amélioration est possible en tenant compte de la température de l'air avec ce capteur :



Photos du montage



Le buzzer sonne !

Le programme complet

```
distance_LCD_buzzer

#include <Wire.h>                // bibliothèque I2C
#include <rgb_lcd.h>             // pour l'afficheur
rgb_lcd lcd;                    // création de l'objet afficheur nommé lcd

//variables :
float distance;
int temps;
float celerite = 339.5;         // vitesse du son, en m/s

void setup()
{
  lcd.begin(16, 2);             // configure l'afficheur 16 lignes 2 colonnes
  lcd.clear();                  // nettoyage de l'écran
  lcd.print("Distance =");      // affichage 1ère ligne
}

void loop() {
  pinMode(8, OUTPUT);           // buzzer sur D8 en sortie

  //émission et réception de l'onde :
  pinMode(4, OUTPUT);
  digitalWrite(4, HIGH);
  delayMicroseconds(10);
  digitalWrite(4, LOW);
  pinMode(4, INPUT);

  //calculs :
  temps = pulseIn(4, HIGH);     // durée aller-retour, en µs, de l'onde
  distance = celerite*temps*0.000001/2; // calcul de la distance, en m

  //affichage :
  if (distance < 0.2) {
    lcd.setRGB(255, 0, 0);      // couleur du rétroéclairage (rouge)

    digitalWrite(8,HIGH);       // buzzer sonne
    delay(400);                  // pendant 400 ms
    digitalWrite(8,LOW);        // buzzer s'arrête
    delay(50);                   // pendant 50 ms
  }
  else {
    lcd.setRGB(0, 255, 0);      // couleur du rétroéclairage (vert)
  }
  lcd.setCursor(5, 1);          // placement curseur 2e ligne
  lcd.print(distance);          // affichage de la distance
  lcd.print(" m");
  delay(500);                   // mesure de distance toutes les 500 ms
}
```

Remarque : Avec ce module Ultrason Grove il aurait été possible d'utiliser une bibliothèque qui permet d'obtenir directement la distance. Par exemple ici : [Ultrasonic - Arduino Reference](#)