Projet Multitâche FreeRTOS : Capteur Ultrasonique, Affichage et Buzzer sur Arduino UNO

Kennouche Abderrahmane
Spécialité : ISI
Groupe : 2

December 16, 2025

1 Introduction

Dans ce projet, nous mettons en œuvre un système multitâche utilisant **FreeRTOS** sur Arduino UNO. FreeRTOS permet le **multitâche préemptif**, ce qui signifie que plusieurs tâches peuvent être exécutées de manière quasi-parallèle même sur un microcontrôleur mono-cœur. Nous avons créé **trois tâches concurrentes** :

- **Tâche Ultrason :** mesure la distance à partir du capteur ou la simule avec un potentiomètre.
- **Tâche Affichage :** affiche la distance mesurée sur un afficheur 7 segments.
- **Tâche Buzzer :** génère un signal sonore en fonction de la distance, plus rapide lorsque l'objet est proche.

2 Matériel et Composants

Les composants suivants sont utilisés :

- Arduino UNO (microcontrôleur principal)
- Capteur Ultrasonique HC-SR04 (Trig → D3, Echo → D2, VCC → 5V(intégrer), GND → GND(intégrer))
- Potentiomètre (broche centrale → entrée IN du capteur, côtés → 5V/GND)
- Buzzer actif (D7 → signal, GND → masse)
- Afficheur 7 segments (broches 4,5,6,8,9,10,11 → segments a-g + GND)

3 Connexions du Circuit

Le circuit est câblé comme suit :

- **Capteur Ultrasonique :** Trig à D3, Echo à D2.
- **Potentiomètre :** Broche centrale vers IN du capteur (simule la distance), côtés vers 5V et GND.
- **Buzzer :** Pin D7 pour le signal, GND pour la masse.
- **Afficheur 7 segments :** Pins connectées à Arduino comme indiqué ci-dessus.

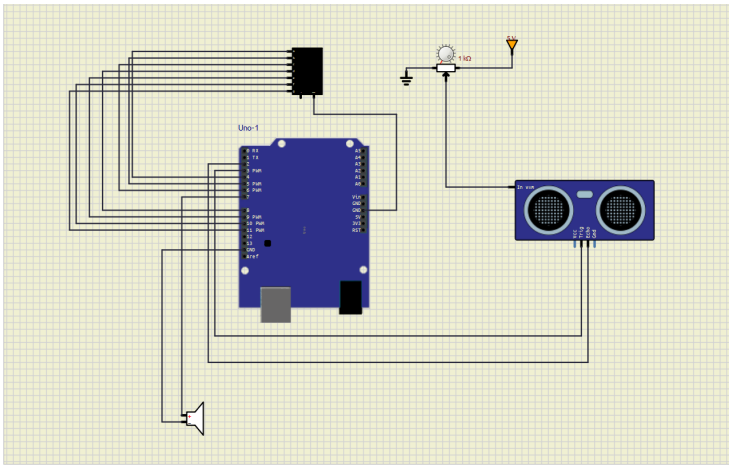

Figure 1: Schéma du circuit avant exécution.

4 Processus Réel et de Simulation

4.1 4.1 Processus Réel

Dans le monde réel, le système fonctionne comme suit :

1. **Envoi de l'impulsion :** L'Arduino envoie une impulsion de 10 µs sur la broche **Trig** du capteur HC-SR04. Cette impulsion déclenche l'émission d'ondes ultrasoniques.
2. **Propagation des ondes :** Le capteur émet une onde ultrasonique qui se propage et se réfléchit lorsqu'elle rencontre un objet.
3. **Réception de l'écho :** La broche **Echo** reçoit l'impulsion réfléchie. La durée pendant laquelle Echo reste **HIGH** correspond au temps aller-retour des ondes.
4. **Calcul de la distance :** La durée de l'écho est stockée dans la variable `duration_us` (en microsecondes).

   La distance en centimètres est calculée par la formule :

$$distance\_cm = \frac{vitesse\_du\_son \times duration\_us}{2} = 0.017 \times duration\_us$$

   où 0.017 cm/µs correspond à la moitié de la vitesse du son (340 m/s). Ainsi, chaque microseconde mesurée correspond à 0,017 cm entre le capteur et l'objet.

5. **Affichage :** La distance est affichée sur le 7 segments. Si elle est hors plage (0–450 cm), l'afficheur montre "E" pour erreur. La conversion pour l'afficheur se fait par exemple :
$$chiffre\_affiche = \frac{distance\_cm}{50}$$

6. **Buzzer :** La fréquence des bips varie selon la distance : plus l'objet est proche, plus le bip est rapide. La durée entre les bips est calculée par la fonction `map()` :
$$beepDelay = map(d, 450, 20, 500, 50)$$

4.2 4.2 Processus de Simulation (SimulIDE)

Dans SimulIDE, le déplacement réel de l'objet ne peut pas être simulé. Nous utilisons un **potentiomètre** pour générer des valeurs simulant la distance :

1. **Simulation de la distance :** Un potentiomètre génère une tension variable simulant la distance. La broche centrale est connectée à l'entrée IN du capteur. Les côtés du potentiomètre sont connectés à 5V et GND. La valeur lue par le capteur (ou Arduino) est convertie en distance simulée.

2. **Affichage :** La distance simulée est traitée exactement comme dans le système réel et affichée sur le 7 segments. Les valeurs hors plage affichent "E".

3. **Buzzer :** Le buzzer fonctionne de la même manière que dans le système réel : le temps entre les bips varie selon la distance simulée.

4. **Avantages :** Cette simulation permet de tester l'ensemble des tâches FreeRTOS (ultrason/potentiomètre, affichage, buzzer) sans nécessiter de déplacement physique.

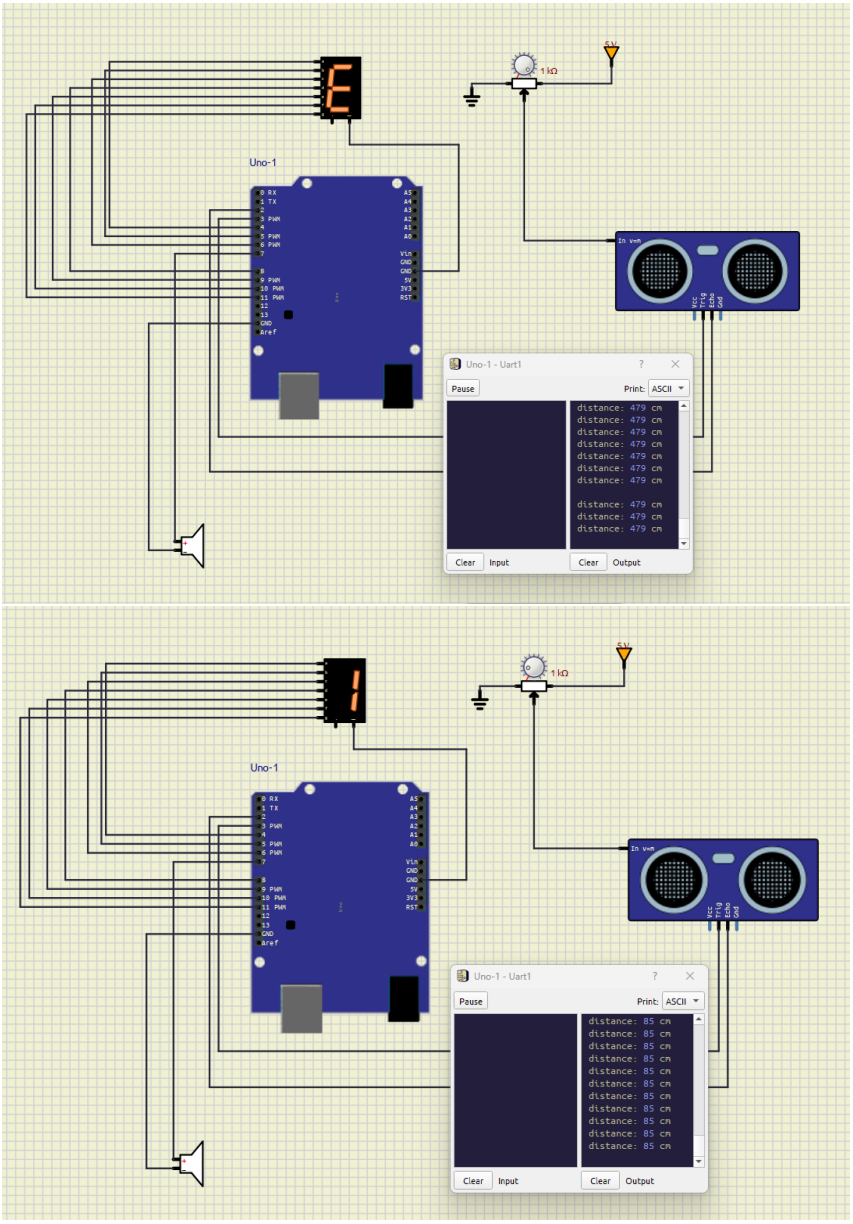

Figure 2: Simulation du circuit avec potentiomètre dans SimulIDE.

4.3 4.3 Comparaison

- **Réel :** L'Arduino mesure réellement le temps aller-retour des ondes ultrasoniques pour calculer la distance.
- **Simulation :** Le potentiomètre fournit une valeur analogue représentant la distance, reproduisant le comportement du capteur réel.
- **Affichage et buzzer :** Identiques dans les deux cas, assurant la cohérence des tâches FreeRTOS.

Conclusion

Le projet illustre l'utilisation de FreeRTOS pour gérer plusieurs tâches sur un Arduino. Chaque tâche (mesure de distance, affichage sur le 7 segments, et contrôle du buzzer) est créée avec la même priorité, ce qui permet à FreeRTOS d'ordonnancer leur exécution de manière équitable. Grâce à `vTaskDelay()`, chaque tâche se met en sommeil temporairement, laissant le processeur disponible pour les autres tâches. Ainsi, bien que l'Arduino soit mono-cœur, FreeRTOS permet de simuler une exécution quasi-parallèle et garantit que toutes les fonctionnalités fonctionnent correctement sans blocage. Cette gestion permet de montrer comment FreeRTOS contrôle le temps CPU attribué à chaque tâche et assure que toutes les opérations (mesure, affichage, buzzer) réagissent en temps réel.