TP : Barrière de parking automatisée avec Arduino

Problématique

Comment concevoir une **barrière de parking automatisée** capable de :

- détecter un véhicule
- afficher des messages
- demander un code d'accès
- ouvrir la barrière si le code est correct

Organisation du TP

Le TP est composé de **3 parties progressives** :

- A — Actionner : servomoteur
- B — Informer : écran
- C — Intégrer : système complet

Matériel

- Arduino Uno / Uno R4
- 1 servomoteur angulaire (SG90 / MG90S)
- (option) 1 servo rotation continue
- 1 écran LCD 16×2 (I2C) ou OLED
- 1 capteur ultrason HC-SR04
- 4 boutons poussoirs
- Breadboard + fils

PARTIE A — Servomoteurs

Objectif

Découvrir le fonctionnement des servomoteurs et piloter une barrière.

A1 — Observation

Questions

1. Quelle est la différence entre :
    - un servo angulaire
    - un servo à rotation continue
2. Quel type de servo est adapté à une barrière ? Pourquoi ?

A2 — Commande simple

Travail demandé

Programmer le servo pour :

- se placer à 0°
- puis à 90°
- puis revenir à 0°

A3 — Mouvement progressif

Travail demandé

Réaliser une ouverture progressive :

- passer de 0° à 90° par petits incréments

Question

Pourquoi ce mouvement est-il plus réaliste ?

A4 — Commande par bouton

Travail demandé

- Appui bouton → ouvrir
- Relâchement → fermer

PARTIE B — Affichage écran

Objectif

Afficher des informations à l'utilisateur.

B1 — Message simple

Travail demandé

Afficher :

```
Systeme pret
```

B2 — Interaction

Travail demandé

Afficher :

- bouton non appuyé → Attente
- bouton appuyé → Bonjour

B3 — Messages système

Travail demandé

Afficher selon les cas :

- Attente vehicule
- Bienvenue
- Entrez code

- Code correct
- Acces refuse
- Barriere ouverte

PARTIE C — Système complet

Objectif

Créer une barrière automatisée complète.

C1 — Détection véhicule

Travail demandé

Utiliser le capteur ultrason :

- distance > seuil → Attente vehicule
- distance ≤ seuil → Vehicule detecte

Question

Quel est le rôle du seuil ?

C2 — Code binaire pondéré

Principe

Bouton	Poids
B3	8
B2	4
B1	2
B0	1

Travail demandé

Lire les boutons et calculer la valeur.

✏ Exercices

1. Quels boutons pour :
    - 6 = ?
    - 9 = ?
    - 13 = ?
2. Si B2 et B1 sont appuyés → valeur = ?

C3 — Validation du code

** Travail demandé**

- Code attendu = 9
- Si correct :
    - afficher “Code correct”
    - ouvrir la barrière
- Sinon :
    - afficher “Acces refuse”

C4 — Ouverture barrière

** Travail demandé**

Créer la séquence :

1. ouverture progressive
2. attente
3. fermeture progressive

Fonctionnement global

1. Initialisation
2. Barrière fermée

3. Attente véhicule
4. Détection
5. Affichage “Bienvenue”
6. Saisie du code
7. Vérification
8. Si OK → ouverture
9. Sinon → refus
10. Retour attente

Structuration du programme

Fonctions conseillées

- ouvrirBarriere()
- fermerBarriere()
- lireDistance()
- lireCode()
- afficherMessage()

Critères de réussite

- La barrière fonctionne correctement
- L’écran affiche les bons messages
- Le véhicule est détecté
- Le code 9 est reconnu
- Les erreurs sont gérées

Pour aller plus loin

- Ajouter feu rouge / vert
- Ajouter buzzer
- Utiliser millis() (non bloquant)
- Compteur de véhicules
- Machine à états

⚡ Extension — Programmation par interruptions

🎯 Objectif

Découvrir une autre manière de programmer :

- ne plus “scruter en boucle”
- réagir à un **événement instantané**

🔁 Rappel

Jusqu’à présent, le programme fonctionne ainsi :

- boucle loop()
- lecture des entrées en continu

🔁 On appelle cela du **scrutation (polling)**

⚠ Limites du polling

- perte de temps processeur
- risque de rater un événement rapide
- programme moins réactif

🧠 Principe des interruptions

Une **interruption** permet :

- d’arrêter le programme principal
- d’exécuter immédiatement une fonction
- puis de reprendre le programme

🔁 Exemple :

- appui bouton
- détection capteur
- signal externe

Activité 1 — Bouton avec interruption

Objectif

Déclencher l'ouverture de la barrière **sans scrutation**

Travail demandé

- connecter un bouton sur une broche d'interruption (D2 ou D3)
- créer une fonction appelée automatiquement lors de l'appui

Résultat attendu

- appui → ouverture immédiate
- sans tester le bouton dans loop()

Activité 2 — Variable de signal

Principe

⚠ Dans une interruption, on évite les actions longues

On utilise une variable "flag"

Travail demandé

- créer une variable booléenne (ex : demandeOuverture)
- l'activer dans l'interruption
- traiter l'action dans loop()

Activité 3 — Capteur ultrason amélioré

Objectif

Réagir rapidement à l'arrivée d'un véhicule

Travail demandé

- détecter un changement d’état (présence / absence)
- déclencher un événement (affichage / demande code)

⚠ Bonnes pratiques

Dans une fonction d’interruption :

- ❌ pas de delay()
- ❌ pas d’affichage écran
- ❌ pas de calcul lourd

- ✅ code court
- ✅ modification de variables simples

Comparaison

Méthode	Avantages	Inconvénients
Polling	simple	lent, peu réactif
Interruption	rapide, efficace	plus complexe

Analyse

❓ Questions

1. Quelle méthode est la plus réactive ?
2. Pourquoi ne met-on pas tout le code dans l’interruption ?
3. Quel est le rôle de la variable “flag” ?

Pour aller plus loin

- gérer plusieurs interruptions
- anti-rebond logiciel
- combinaison avec millis()
- machine à états + interruptions

Conclusion

Un système embarqué performant :

- utilise peu de délais bloquants
- réagit aux événements
- sépare détection et action

C’est le principe des systèmes industriels modernes