Séance de TP : Conception et programmation d'un robot aspirateur évitant les obstacles

Objectifs de la séance

Objectifs pédagogiques :
- Comprendre le fonctionnement d'un capteur spécifique et son intégration dans un système embarqué.
- Savoir programmer une carte Arduino pour contrôler un robot en fonction des données du capteur.
- Développer des compétences en résolution de problèmes et en travail collaboratif.

Compétences visées :
- **CO3.1** : Identifier et caractériser les fonctions et les constituants d'un produit ainsi que ses entrées/sorties.
- **CO3.3** : Identifier et caractériser le fonctionnement temporel d'un produit ou d'un processus.
- **CO5.3** : Mettre en évidence les constituants d'un produit à partir des diagrammes pertinents.
- **CO5.5** : Proposer des solutions à un problème technique identifié en participant à des démarches de créativité.
- **CO5.8** : Proposer/choisir l'architecture d'une solution logicielle et matérielle au regard de la définition d'un produit.
- **CO7.4** : Réaliser et valider un prototype ou une maquette obtenus en réponse à tout ou partie du cahier des charges initial.
- **CO7.5** : Mettre en œuvre un scénario de validation, interpréter les résultats et qualifier le produit.

Contexte

Dans le cadre de ce TP, vous êtes chargés d'améliorer un robot aspirateur pour qu'il puisse éviter les obstacles grâce à un capteur que vous avez choisi. Vous travaillerez en groupe pour concevoir, programmer et tester votre solution.

Consignes générales

Avant la séance

- **Réflexion préalable** : Réfléchissez au type de capteur que vous utiliseriez pour permettre à un robot aspirateur d'éviter les obstacles (parmi une sélection proposée : capteur à ultrasons, infrarouge, lidar, etc.).
- **Justification du choix** : Préparez une courte justification de votre choix, en mentionnant les avantages et les inconvénients du capteur.
  - Quels sont les principes de fonctionnement de ce capteur ?
  - En quoi est-il adapté à la détection d'obstacles pour un robot aspirateur ?
  - Quels pourraient être les défis ou limitations liés à son utilisation ?

Pendant la séance

Vous suivrez les étapes suivantes :

1. **Prise en main du capteur**
2. **Conception de l'algorithme**
3. **Programmation du robot**

4. **Tests et ajustements**
5. **Rédaction du compte rendu**

Après la séance

— Finalisez votre compte rendu en incluant toutes les parties demandées.
— Préparez-vous à présenter votre travail lors de la prochaine séance (si applicable).

Plan de travail détaillé

Étape 1 : Prise en main du capteur

— **Installation du matériel** :
  — Connectez le capteur à la carte Arduino selon le schéma fourni.
  — Vérifiez soigneusement les connexions pour éviter les erreurs.
— **Chargement du code de base** :
  — Importez le programme de base pour lire les données du capteur.
  — Compilez et téléversez le code sur la carte Arduino.
— **Observation** :
  — Ouvrez le moniteur série pour visualiser les données du capteur.
  — **Questions intermédiaires :**
    1. Comment les valeurs lues par le capteur varient-elles en fonction de la distance ou de la présence d'obstacles ?
    2. Quels sont les seuils de détection pertinents pour votre application ?
    3. Y a-t-il du bruit ou des fluctuations dans les mesures ? Si oui, comment pourriez-vous les atténuer ?

Étape 2 : Conception de l'algorithme

— **Analyse du problème** :
  — Identifiez les situations où le robot doit modifier sa trajectoire.
  — Déterminez les actions que le robot doit entreprendre en fonction des données du capteur.
— **Création de l'algorithme** :
  — Dessinez un organigramme ou rédigez un pseudo-code du fonctionnement souhaité.
  — **Questions intermédiaires :**
    1. Quels sont les états possibles du robot (avancer, reculer, tourner) ?
    2. Comment le robot décide-t-il de passer d'un état à un autre ?
    3. Comment gérez-vous les situations où plusieurs obstacles sont détectés simultanément ?

Espace pour votre organigramme :

Étape 3 : Programmation du robot

- **Écriture du code** :
  - Programmez les réactions du robot en fonction des données du capteur.
  - Utilisez des structures conditionnelles pour implémenter votre algorithme.
  - **Questions intermédiaires :**
    1. Comment traduire les conditions de votre algorithme en code ?
    2. Avez-vous pensé à gérer les cas extrêmes ou les erreurs potentielles ?
    3. Comment organisez-vous votre code pour le rendre clair et maintenable ?
- **Intégration des commandes moteurs** :
  - Contrôlez les moteurs pour réaliser les mouvements (avancer, reculer, tourner).
  - Assurez-vous que les broches utilisées pour les moteurs sont correctement définies.
  - **Questions intermédiaires :**
    1. Comment contrôlez-vous la vitesse et la direction des moteurs ?
    2. Avez-vous besoin de gérer des temporisations pour les mouvements (par exemple, durée de rotation) ?
    3. Comment testez-vous individuellement les commandes des moteurs avant l'intégration complète ?

Étape 4 : Tests et ajustements

- **Mise en pratique** :
  - Testez votre programme sur le robot dans un environnement contrôlé.
  - Observez le comportement du robot face aux obstacles.
- **Débogage** :
  - Identifiez les problèmes éventuels (réactions inattendues, détection inefficace, etc.).
  - **Questions intermédiaires :**
    1. Les actions du robot correspondent-elles aux conditions définies dans votre algorithme ?
    2. Comment pouvez-vous améliorer la réactivité ou la précision du robot ?
    3. Les capteurs fonctionnent-ils correctement dans différentes conditions de luminosité ou de surface ?
  - Ajustez votre code en conséquence et retestez jusqu'à obtenir un fonctionnement satisfaisant.

Étape 5 : Rédaction du compte rendu

Votre compte rendu doit inclure :

- **Description du fonctionnement du capteur utilisé** :
  - Expliquez le principe de fonctionnement du capteur.
  - Présentez les caractéristiques techniques importantes pour votre projet.
- **Étapes de votre programmation et défis rencontrés** :
  - Décrivez votre démarche de programmation.
  - Mentionnez les problèmes rencontrés et les solutions apportées.
- **Auto-évaluation de votre solution et améliorations possibles** :
  - Évaluez l'efficacité de votre robot.
  - Proposez des pistes pour améliorer le système (matériel, logiciel, algorithme).

Livrables attendus

- **Programme fonctionnel** chargé sur la carte Arduino.
- **Compte rendu écrit** comprenant les éléments demandés.
- **Participation** active lors de la présentation finale (si applicable).

Ressources supplémentaires

— **Documentation Arduino** : `https://www.arduino.cc/reference/fr`
— **Tutoriels sur les capteurs** : Consulter les fiches techniques fournies et les ressources en ligne pour comprendre en profondeur le fonctionnement du capteur.
— **Support du professeur** : N'hésitez pas à demander de l'aide ou des clarifications si vous rencontrez des difficultés.

Bonus : Introduction au filtrage numérique

Pour les élèves qui souhaitent aller plus loin, cette section présente le filtrage numérique et comment l'appliquer aux données du capteur avant de commander les moteurs.

Contexte

Lors de l'observation des données du capteur, vous avez peut-être remarqué des fluctuations ou du bruit qui peuvent affecter les performances du robot. En cours, vous avez étudié le filtrage analogique (passe-bas) pour atténuer ces perturbations. Le filtrage numérique est une approche similaire, mais réalisée par traitement logiciel.

Filtrage numérique vs. filtrage analogique

— **Filtrage analogique** : Utilise des composants électroniques (résistances, condensateurs) pour filtrer les signaux en temps réel.
— **Filtrage numérique** : Traite les signaux après leur conversion en valeurs numériques, en appliquant des algorithmes pour atténuer le bruit.

Filtre numérique simple : Filtre Moyenne Mobile

Un filtre moyenne mobile est un moyen simple de lisser les données en calculant la moyenne des N dernières mesures.

Formule du filtre moyenne mobile :

$$\text{Valeur filtrée} = \frac{1}{N} \sum_{i=1}^{N} \text{Valeur}_{\text{mesure précédente } i}$$

Implémentation dans votre programme

1. **Déclarez un tableau pour stocker les N dernières mesures :**

```
const int N = 5; // Nombre de mesures pour la moyenne
int mesures[N];
int index = 0;
```

2. **Initialisez le tableau dans la fonction `setup()` :**

```
void setup() {
    // ... votre code existant ...
    for (int i = 0; i < N; i++) {
        mesures[i] = 0;
    }
}
```

3. **Dans la boucle principale `loop()`, mettez à jour les mesures et calculez la moyenne :**

```
    void loop() {
        int valeurCapteur = lireCapteur(); // Fonction pour lire le
            capteur
        mesures[index] = valeurCapteur;
        index = (index + 1) % N;

        // Calcul de la moyenne
        int somme = 0;
        for (int i = 0; i < N; i++) {
            somme += mesures[i];
        }
        int valeurFiltree = somme / N;

        // Utilisez valeurFiltree pour vos decisions
        // ... votre code pour controler le robot ...

        delay(50); // Ajustez en fonction de vos besoins
    }
```

4. **Remplacez les utilisations de la valeur brute du capteur par la valeur filtrée :**
   — Dans vos conditions et calculs, utilisez `valeurFiltree` au lieu de `valeurCapteur`.

Questions pour approfondir

1. **Impact du nombre N de mesures :**
   — Comment le choix de N affecte-t-il la réactivité et le lissage des données ?
   — Que se passe-t-il si N est trop petit ou trop grand ?
2. **Limites du filtre moyenne mobile :**
   — Le filtre moyenne mobile est-il efficace pour tous les types de bruit ?
   — Quels pourraient être les inconvénients de ce filtre dans certaines situations ?
3. **Autres types de filtres numériques :**
   — Explorez d'autres filtres simples, comme le filtre exponentiel lissé.
   — Comment pourraient-ils être implémentés et quels avantages offrent-ils ?

Conclusion

Le filtrage numérique est un outil puissant pour améliorer la qualité des données issues des capteurs. En appliquant un filtre simple comme la moyenne mobile, vous pouvez réduire l'impact du bruit sur les performances de votre robot. Cela vous permet d'obtenir un comportement plus stable et prévisible.