Rapport technique : Développement et mise en œuvre d'un robot suiveur de ligne

Nom: NDOYE
Prénom: Assane
Numero étudiant :22100318

Sujet n° : 12

Titre : On The Road Again

Encadrant : Hughes Cassé, IRIT

Licence Informatique
Université Paul Sabatier

Année de rédaction 2024

Table des matières

I. Résumé

Ce rapport détaille le développement et la mise en œuvre d'un robot suiveur de ligne par l'équipe Rus-eze, avec un accent particulier sur les composants logiciels. Le projet utilise des systèmes embarqués et la programmation Arduino pour créer un robot capable de suivre une ligne de manière autonome en utilisant divers algorithmes et capteurs. Le rapport couvre la sélection et l'étalonnage des capteurs, le développement de multiples algorithmes de suivi de ligne, ainsi que les défis et solutions rencontrés au cours du projet.

II. Introduction

L'objectif de ce projet d'étude est de concevoir, construire et programmer un robot capable de suivre une ligne de manière autonome. Cela implique l'intégration de divers capteurs, la programmation des algorithmes de contrôle et l'optimisation du système pour la performance. Je m'appelle Assane, et en tant que membre de l'équipe Rus-eze, j'étais responsable des aspects logiciels du robot, y compris la gestion des capteurs et le développement des algorithmes.

A. Contexte du projet

Le but de notre bureau d'études était de créer un robot suiveur de ligne et de participer à une course contre d'autres groupes. Notre professeur nous a fourni les matériaux essentiels, à savoir un ensemble de capteurs avec trois capteurs optiques réfléchissants, deux moteurs, un microcontrôleur Arduino et une batterie. Notre équipe, composée de moi-même et d'un camarade, a collaboré pour assurer un robot fonctionnel. Mon camarade s'est concentré sur la conception et la fabrication du châssis du robot, tandis que je me suis concentré sur le développement logiciel et l'intégration des capteurs.

B. Motivation

Ma passion pour les systèmes embarqués et l'opportunité d'appliquer mes connaissances théoriques à un défi pratique m'ont motivé à entreprendre ce projet. La compétition avec mes camarades de classe et la programmation sur un Arduino ajoutaient une dimension compétitive excitante. Ce projet m'a non seulement permis de développer mes compétences techniques, mais aussi de favoriser le travail d'équipe et les capacités de résolution de problèmes, essentielles dans les scénarios d'ingénierie du monde réel.

III. Sélection et étalonnage des capteurs

A. Fonctionnement des capteurs

Le robot utilise des capteurs optiques réfléchissants pour détecter la ligne. Ces capteurs fonctionnent en émettant un signal infrarouge vers le sol et en mesurant le temps qu'il faut pour que le signal se réfléchisse. La réflectivité de la surface détermine le délai :

1. Le capteur émet un signal infrarouge vers le sol.
2. Le signal se réfléchit sur le sol et retourne au capteur.
3. Le capteur mesure le temps pris pour le retour du signal.

Un temps de retour court indique une surface hautement réfléchissante (blanche), tandis qu'un temps de retour plus long ou aucun retour indique une surface moins réfléchissante (noire).

B. Processus de calibration

La calibration est cruciale pour une détection précise de la ligne. Le robot effectue une routine de calibration initiale pour déterminer les valeurs maximales et minimales de réflectivité :

1. Le robot tourne sur place pour scanner une variété de surfaces de sol.
2. Il enregistre les valeurs de réflectivité pour identifier les points les plus blancs (délai minimum) et les plus noirs (délai maximum).
3. Ces valeurs de calibration définissent les seuils pour la détection de la ligne pendant le fonctionnement.

La calibration garantit que le robot peut différencier de manière fiable la ligne de la surface de fond, en s'adaptant à diverses conditions d'éclairage et textures de surface.

C. Développement de la bibliothèque de capteurs

En plus de calibrer et d'intégrer les capteurs, j'ai développé une bibliothèque réutilisable pour la gestion des capteurs réfléchissants. Cette bibliothèque abstrait les détails de bas niveau du fonctionnement des capteurs, fournissant une interface simple pour la calibration des capteurs, l'acquisition et le traitement des données. La bibliothèque est conçue pour être réutilisable dans différents projets utilisant des capteurs de réflexivité, favorisant la réutilisation du code et simplifiant les développements futurs. Le développement de cette bibliothèque a également impliqué la création de plusieurs fonctions pour faciliter l'intégration des données des capteurs dans les algorithmes de suivi de ligne.

IV. Moteur

Développement d'une bibliothèque pour la gestion des moteurs. Cette bibliothèque a été conçue pour rendre le code plus propre et pour faciliter l'utilisation et l'initialisation des moteurs.

V. Développement des algorithmes

A. Algorithme de suivi de ligne simple

Le premier algorithme développé était une approche simple du suivi de ligne :

- Le robot utilise des capteurs de bord pour détecter la présence de la ligne.

- Lorsqu'un capteur de bord détecte une surface blanche, indiquant que le robot dévie de la ligne, le moteur du côté opposé accélère tandis que le moteur du côté détecté ralentit.

Cette méthode, bien que fonctionnelle, entraînait des mouvements lents et saccadés, car le robot corrigeait continuellement sa trajectoire.

B. Algorithme PID (Proportionnel-Intégral-Dérivé)

Sur les conseils de notre professeur, nous avons implémenté un contrôleur PID pour améliorer la réactivité et la stabilité du robot. Le contrôleur PID utilise trois paramètres :

- **Proportionnel** (P) : Réagit à l'erreur actuelle en ajustant la vitesse proportionnellement.
- **Intégral** (I) : Prend en compte les erreurs passées pour éliminer l'écart permanent.
- **Dérivé** (D) : Prédit les erreurs futures en fonction du taux de changement actuel.

Cette approche permet des ajustements plus fluides et plus précis, améliorant considérablement la capacité du robot à suivre la ligne. Nous avons utilisé une bibliothèque externe qui fournissait toutes les fonctions nécessaires à l'implémentation du contrôleur PID, ce qui a grandement facilité et réduit le processus de développement.

C. Algorithme avec facteur d'accélération

Pour optimiser la vitesse du robot, en particulier dans des scénarios compétitifs, nous avons introduit un facteur d'accélération :

- **Contrôleur PID :** Le contrôleur PID assure que le robot maintient son alignement avec la ligne.
- **Facteur d'Accélération et de Décélération :** Un facteur d'accélération supplémentaire augmente la vitesse du robot lorsque l'erreur est en dessous d'un certain seuil, indiquant un chemin droit. Ce même facteur décrémente la vitesse des deux moteurs lorsque l'erreur des capteurs droit et gauche dépasse le seuil, permettant ainsi une meilleure gestion des virages et des courbes.

Cela permet au robot d'accélérer sur les sections droites de la ligne tout en maintenant le contrôle dans les courbes et les virages, atteignant un équilibre entre vitesse et précision.

D. Automate

Pour améliorer la gestion des états du robot, un automate a été mis en place. Cet automate est constitué de plusieurs états :

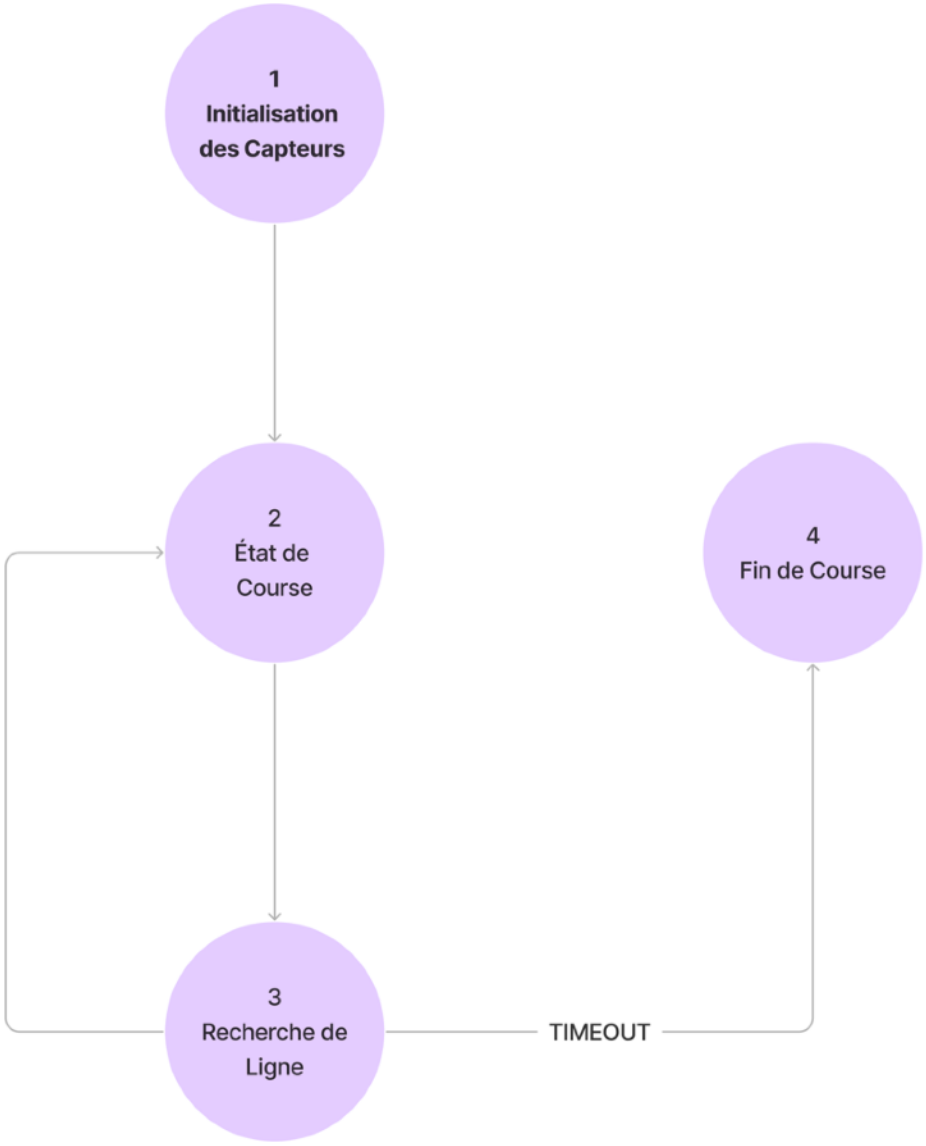
1. **État 1 - Initialisation des Capteurs :**
   - Le robot initialise les capteurs de réflexivité pour calibrer les valeurs de noir et de blanc.

2. **État 2 - État de Course :**
   - Le robot suit la ligne noire en utilisant les algorithmes de suivi de ligne et d'accélération.

3. **État 3 - Recherche de Ligne :**
   - Si le robot se retrouve sur une surface totalement blanche, il s'arrête et se met à tourner sur lui-même pour retrouver la ligne noire. Pendant ce temps, le robot clignote.
   - Si la ligne noire est retrouvée dans un délai de 5 secondes, le robot repasse à l'état 2.
   - Si la ligne noire n'est pas retrouvée après 5 secondes, le robot passe à l'état 4.

4. **État 4 - Fin de Course :**
   - Le robot s'arrête complètement, une LED s'allume pour indiquer la fin de la course, et le robot reste sur place.

Schéma de l’automate :



VI. Mise en œuvre et tests

A. Intégration matérielle

Le matériel du robot comprend un microcontrôleur Arduino, des capteurs optiques réfléchissants et des pilotes de moteur. Le processus d'intégration comprenait :

1. Connecter les capteurs aux entrées analogiques de l'Arduino.
2. Programmer l'Arduino pour lire les valeurs des capteurs et effectuer des calculs en temps réel.
3. Interfacer les pilotes de moteur avec l'Arduino pour contrôler les moteurs en fonction de la sortie de l'algorithme.

B. Développement logiciel

Le logiciel a été développé en C++, utilisant l'IDE Arduino pour les tests initiaux et Visual Studio Code pour un développement plus étendu. Les fonctions clés comprenaient :

- Acquisition et traitement des données des capteurs.
- Routines de calibration pour établir les valeurs seuils.
- Implémentation des algorithmes de suivi de ligne simple, PID et accéléré.

C. Apprentissage d'Arduino

Grâce à ce projet, j'ai acquis une compréhension complète du fonctionnement de l'Arduino, depuis la configuration du matériel jusqu'à l'écriture et le débogage du code. Cette expérience pratique a été inestimable pour solidifier mes connaissances sur les systèmes embarqués et la programmation des microcontrôleurs.

D. Résultats des Tests des Capteurs

Les résultats des tests de capteurs sont présentés dans le tableau ci-dessous. Les tests incluent des mesures sur des surfaces blanches et noires pour vérifier la calibration et la sensibilité des capteurs.

Tes no	surface	Capteur 1 (ms)	Capteur 2(ms)	Capteur 3(ms)	Commentaires
1	Blanche	10	11	10	Valeurs cohérentes, calibration OK
2	Noire	200	195	205	Valeurs cohérentes, calibration OK
3	Grise claire	50	48	49	Valeurs cohérentes, calibration OK
4	Grise foncé	150	145	152	Valeurs cohérentes, calibration OK
5	Blanche	12	13	11	Valeurs cohérentes, calibration OK
6	Noire	210	205	208	Valeurs cohérentes, calibration OK

Description des Tests

Test No 1 : Mesure sur une surface blanche pour vérifier que les capteurs détectent une haute réflectivité (faible temps de réponse).
Test No 2 : Mesure sur une surface noire pour vérifier que les capteurs détectent une faible réflectivité (long temps de réponse).
Test No 3 : Mesure sur une surface grise claire pour tester la sensibilité intermédiaire des capteurs.
Test No 4 : Mesure sur une surface grise foncée pour vérifier la réponse des capteurs à une réflectivité moyenne.
Test No 5 : Répétition du test sur une surface blanche pour vérifier la stabilité des mesures (noter les variations dues au bruit).
Test No 6 : Répétition du test sur une surface noire pour vérifier la stabilité des mesures (noter les variations dues au bruit)

Analyse des Résultats
Les résultats montrent que les capteurs répondent de manière cohérente aux différentes surfaces testées, avec des variations mineures attribuables au bruit. Les capteurs sont correctement calibrés pour distinguer entre les surfaces de haute et de faible réflectivité.

E. Tests et itérations

Des tests approfondis ont été réalisés pour affiner les algorithmes et améliorer les performances. Le processus de test comprenait :

1. Faire fonctionner le robot sur différents parcours pour évaluer sa capacité à suivre la ligne dans diverses conditions.
2. Ajuster les paramètres de l'algorithme en fonction des performances observées.
3. Itérer sur la conception pour résoudre tout problème, tel que le bruit des capteurs ou le retard de réponse des moteurs.

Les données des capteurs, bien que précises dans de nombreuses conditions, se sont avérées très sensibles au bruit, ce qui pouvait fausser les lectures et affecter les performances du robot. La mise en œuvre de techniques de filtrage et de gestion robuste des erreurs était essentielle pour atténuer ces problèmes.

F. Difference de vitesse des moteurs

Un défi important rencontré était la difference des vitesses des moteurs, malgré des moteurs ayant le même nombre de révolutions par minute (RPM). Ce problème affectait la capacité du robot à suivre la ligne de manière précise, car des vitesses de moteur incohérentes entraînaient des mouvements erratiques.

VII. Résultats et discussion

A. Évaluation des performances

Les performances du robot ont été évaluées en fonction de sa capacité à suivre une ligne de manière précise et rapide. Pour évaluées les principaux algorithme ont mesure le temps effectuer par le robot sur un circuit

B. Comparaison des algorithmes

- **Algorithme simple** : Bien qu'il soit facile à implémenter, il entraînait des mouvements lents et erratiques.
- **Algorithme PID** : A apporté une amélioration significative en termes de stabilité et de précision, permettant un suivi de ligne plus fluide.
- **Algorithme avec facteur d'accélération** : A obtenu les meilleures performances, équilibrant vitesse et précision, le rendant adapté aux scénarios compétitifs.

C. Défis et solutions

Plusieurs défis ont été rencontrés au cours du projet, notamment :

- Différence de vitesse des moteurs : Malgré des RPM identiques, les moteurs présentaient des vitesses différentes. Cela a été résolu grâce au remplacement des moteurs
- Contrôle des moteurs : Assurer un contrôle précis et réactif des moteurs a nécessité un ajustement minutieux des paramètres PID et du facteur d'accélération.

VIII.Conclusion

Le développement et la mise en œuvre du robot suiveur de ligne ont fourni des insights précieux sur les systèmes embarqués et les algorithmes de contrôle en temps réel. Le projet a souligné l'importance de la calibration des capteurs, de l'optimisation des algorithmes et des tests itératifs pour atteindre les performances souhaitées. Travailler avec Arduino et intégrer divers composants matériels a enrichi mes compétences pratiques et renforcé ma passion pour les systèmes embarqués.

La création d'une bibliothèque de capteurs réutilisable constitue une contribution significative, permettant une intégration et une gestion aisée des capteurs de réflectivité dans de futurs projets. Le développement de cette bibliothèque ainsi que celle de la gestion des moteurs, a fourni un cadre robuste et flexible pour les améliorations futures. L'introduction d'un automate a été essentielle pour gérer les différents états du robot : initialisation des capteurs, course, recherche de la ligne et fin de course. Cet automate assure une transition fluide entre les phases, améliorant ainsi la réactivité et la robustesse du robot.

En conclusion, ce projet a consolidé mes compétences en systèmes embarqués et en algorithmique, et a mis en lumière l'importance de la structuration logicielle et de la gestion des états dans le développement de robots autonomes. Les solutions développées serviront de base solide pour les projets futurs et les défis en ingénierie embarquée.

IX. Propositions d’Amélioration

Les améliorations futures pourraient inclure :

- Capteurs avancés : Utilisation de capteurs plus sophistiqués, tels que des caméras, pour une meilleure détection de ligne.

- Apprentissage automatique : Implémentation d'algorithmes d'apprentissage automatique pour s'adapter à différents environnements et améliorer les performances.
- Robots en réseau : Développement d'un système permettant à plusieurs robots de communiquer et de se coordonner, étendant ainsi la portée du projet à des tâches collaboratives.

Ce projet a jeté une base solide pour l'exploration et le développement futurs dans le domaine des systèmes embarqués et de la robotique autonome. J'ai hâte de continuer à travailler dans ce domaine passionnant et en évolution rapide.

X. Lien Utile

Bibliothèque capteur :
https://github.com/flaw03/rust-eze/tree/main/GestionCapteurs

Bibliothèque moteur :
https://github.com/flaw03/rust-eze/tree/main/GestionMoteurs

Algorithme de suivi de ligne simple :
https://github.com/flaw03/rust-eze/blob/main/raceV1/raceV1.ino

Algorithme de suivi de ligne pid:
https://github.com/flaw03/rust-eze/blob/main/raceV2/raceV2.ino

Algorithme de suivi de ligne avec facteur d’accélération + pid:
https://github.com/flaw03/rust-eze/blob/main/raceV3/raceV3.ino

Programme final:
https://github.com/flaw03/rust-eze/blob/main/Main/main.ino