2GPJT - Réalisation d'un module roulant autonome

Thomas Gogoua[a], Ethan Clement[b], Sinthy Nimalaratnam[c]

[a]*Ecole Hexagone, thomas.gogoua@ecole-hexagone.com, France*
[b]*Ecole Hexagone, ethan.clement@ecole-hexagone.com, France*
[c]*Ecole Hexagone, sinthy.nimalaratnam@ecole-hexagone.com, France*

Résumé

Dans le cadre de ce projet, nous nous efforçons de développer un véhicule autonome capable de naviguer autour d'obstacles et de trouver la sortie d'une aire de jeu spécifique. Cette aire de jeu est représentée par un rectangle clos contenant divers objets, et notre objectif est de permettre au véhicule de localiser une sortie d'une longueur de 15 cm. Pour concrétiser ce projet, chaque groupe dispose d'une enveloppe budgétaire de 100 euros destinée à l'acquisition des composants essentiels, garantissant ainsi le bon fonctionnement du véhicule et la proposition d'algorithmes novateurs.

1. Introduction

La capacité à éviter les obstacles joue un rôle crucial dans le développement des véhicules embarqués intelligents et autonomes. Notre objectif est de permettre à ces véhicules de détecter, comprendre et éviter de manière proactive les obstacles présents dans leur environnement, qu'il s'agisse de piétons, de véhicules, d'animaux ou d'obstacles statiques tels que des bâtiments ou des arbres. En évitant ces obstacles, les véhicules peuvent se déplacer en toute sécurité, assurant ainsi une navigation fluide et efficace. Dans le présent rapport, nous décrirons en détail notre approche pour parvenir à l'algorithme final, en mettant l'accent sur les composants que nous avons sélectionnés et intégrés dans notre système.

2. Le choix des composants

L'étape de sélection des composants a été à la fois complexe et cruciale dans notre projet. Ces composants ont exercé une influence considérable sur nos choix en matière d'algorithmes. De plus, ils représentaient une contrainte budgétaire importante, ce qui nous a obligés à optimiser l'allocation des ressources pour garantir le bon fonctionnement du véhicule.

2.1. Arduino KEYESTUDIO Mega Plus 2560 R3

Au cœur de notre système, le microcontrôleur Arduino Mega Plus joue un rôle central. Ce dispositif combine les fonctionnalités essentielles d'un ordinateur, telles qu'un processeur, une mémoire et des interfaces d'entrée-sortie. Agissant comme le véritable cerveau de notre véhicule, il est chargé de traiter les données provenant des capteurs ultrasons. Sa mission consiste à contrôler les moteurs asynchrones de manière à éviter les obstacles et à rechercher activement une sortie appropriée. De cette façon, il assure un fonctionnement optimal du véhicule dans son environnement.

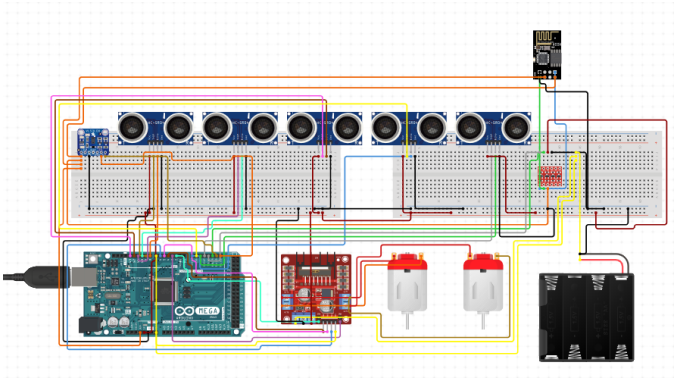

Figure 1: Schema electrique du projet

2.2. Capteurs ultrasons HC-SR04

Les capteurs ultrasons sont conçus pour émettre des ondes qui se réfléchissent sur les objets, permettant ainsi de détecter la présence d'obstacles et de mesurer la distance qui les sépare du capteur. Dans notre projet, ces capteurs jouent un rôle essentiel dans la détection et l'évitement des obstacles présents dans l'aire de jeu, tout en assurant le bon fonctionnement de l'algorithme de la "règle de la main gauche", qui sera expliqué en détail ultérieurement. Notre choix d'utiliser des capteurs ultrasons plutôt qu'un LIDAR est motivé par leur coût abordable, un facteur déterminant dans notre décision.

2.3. Moteurs asynchrones TT

Les moteurs asynchrones sont responsables de la propulsion du robot en réponse aux commandes émises par le microcontrôleur. Leur contrôle est assuré par le module L298N, qui joue un rôle essentiel dans la gestion et la direction des moteurs. Grâce à cette configuration, nous sommes en mesure de contrôler avec précision les mouvements du véhicule et de lui permettre de se déplacer de manière autonome dans l'aire de jeu.

2.4. Module L298N

Ce module offre la possibilité de faciliter l'implémentation de la marche arrière du robot. En général, il permet un contrôle plus complet des moteurs, incluant la gestion de la vitesse des moteurs, entre autres fonctionnalités. De plus, il contribue à réduire la consommation de courant de l'Arduino, ce qui permet d'allouer plus de puissance aux autres capteurs sans difficulté.

2.5. Module ESP32

Grâce au module ESP32, l'Arduino est capable d'envoyer des données via un flux MQTT en utilisant une connexion Wi-Fi, permettant ainsi le traitement de ces données par un ordinateur distant. Dans le cadre de notre projet, son utilisation principale consiste à générer une carte de l'aire de jeu. En exploitant cette fonctionnalité, nous sommes en mesure de recueillir et de transmettre les informations pertinentes sur l'environnement de jeu, offrant ainsi une visualisation claire et précise de l'aire de jeu pour une meilleure prise de décision lors de la navigation autonome du véhicule.

3. L'algorithmique

L'aspect algorithmique du robot se concentre principalement sur deux aspects clés : l'évitement des obstacles et la détection d'une sortie. À cet effet, plusieurs algorithmes et techniques sont utilisés, tels que l'algorithme de la main gauche, la vision par ordinateur et la technique de localisation et de cartographie simultanées (SLAM pour Simultaneous Localization and Mapping) qui est couramment utilisée. Ces approches permettent au robot d'analyser son environnement, de prendre des décisions en temps réel et de naviguer de manière autonome tout en évitant les obstacles et en recherchant activement une sortie.

3.1. Une premiere approche : le SLAM

3.1.1. Introduction

Le concept de localisation et de cartographie simultanées (SLAM - Simultaneous Localization and Mapping) représente une avancée significative dans le domaine de la robotique mobile. Cette technique, développée en 1995, permet à un robot de créer une carte de son environnement tout en estimant simultanément sa propre position dans cette carte. Le SLAM offre des applications importantes dans divers domaines, tels que la navigation autonome, la réalité augmentée et la robotique de service. En exploitant cette technologie, les robots peuvent non seulement se déplacer de manière autonome, mais aussi interagir efficacement avec leur environnement, améliorant ainsi leurs capacités et leur utilité dans de nombreux contextes.

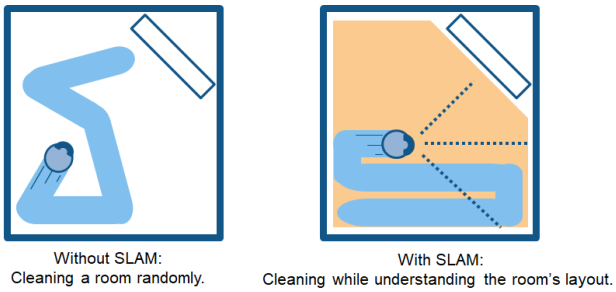

Figure 2: Les avantages du SLAM

3.1.2. L'algorithmique du SLAM

L'un des principaux avantages du SLAM réside dans sa capacité à traiter des environnements inconnus et dynamiques. Contrairement à d'autres approches, comme l'algorithme de la main gauche qui nécessite une connaissance préalable de la structure de l'environnement, le SLAM peut fonctionner efficacement dans des environnements non structurés et en constante évolution. Cela en fait une technique particulièrement adaptée aux applications de robotique autonome, où l'environnement est sujet à des changements imprévisibles. Grâce au SLAM, les robots peuvent continuellement mettre à jour leur carte de l'environnement et estimer leur position avec précision, ce qui leur permet de naviguer de manière autonome et de prendre des décisions adaptées aux conditions changeantes de leur environnement.

Le fonctionnement du SLAM repose sur la fusion de données provenant de différents capteurs, tels que des capteurs de distance et des odomètres. Ces capteurs fournissent des informations sur l'environnement du robot, ce qui lui permet de construire progressivement une carte tout en se localisant avec précision. En utilisant des techniques d'estimation probabiliste, le robot résout le problème de la rétroaction en réduisant les erreurs de localisation au fur et à mesure de l'exploration de l'environnement. Cela lui permet d'améliorer continuellement la qualité de sa carte et de sa position estimée. En combinant les données de différents capteurs et en les traitant de manière probabiliste, le SLAM permet au robot de surmonter les incertitudes inhérentes à l'environnement et d'obtenir des résultats fiables pour une navigation autonome précise.

3.2. Le filtre de Kalman

3.2.1. Introduction

Dans le SLAM, le filtre de Kalman joue un rôle essentiel dans l'estimation de l'état du robot, c'est-à-dire sa position et son orientation. Le filtre de Kalman fonctionne en deux étapes: la prédiction et la mise à jour. Pendant la phase de prédiction, le filtre utilise les données d'odométrie pour estimer la nouvelle position et orientation du robot, ainsi que l'évolution de la carte en fonction du mouvement du robot. Dans la phase de mise à jour, les mesures provenant des capteurs sont utilisées pour corriger les estimations précédentes et améliorer la précision de l'estimation de l'état du robot et de la carte.

Algorithm 1: SLAM

Initialisation: Créer une carte vide Initialiser la position et l'orientation initiales du robot
Boucle principale: while *le robot est en mouvement* **do**
    1. Acquisition des données: - Acquérir les données des capteurs (par exemple, caméra, LIDAR, odométrie) 2. Mise à jour:
    - Effectuer l'association des données entre les mesures des capteurs et les repères existants sur la carte
    - Mettre à jour la position estimée du robot en utilisant les données des capteurs et le modèle de mouvement (par exemple, odométrie)
    3. Mise à jour de la carte:
    - Ajouter de nouveaux repères à la carte en fonction des résultats de l'association des données
    - Affiner la position des repères existants sur la carte
    4. Fermeture de boucle:
    - Vérifier les fermetures de boucle (par exemple, similarité entre les observations actuelles et les observations passées)
    - Si une fermeture de boucle est détectée, raffiner la carte et ajuster les positions estimées du robot
    5. Déplacement:
    - Ordonner au robot de se déplacer vers la prochaine position en utilisant les commandes de mouvement

En utilisant le filtre de Kalman dans le SLAM, il est possible d'estimer de manière cohérente l'état du robot et la carte de l'environnement, en prenant en compte à la fois les mesures actuelles et les estimations passées. Cela permet de résoudre le problème de localisation et de cartographie simultanées, où les informations de localisation et de cartographie se complètent mutuellement. Le filtre de Kalman permet d'obtenir des estimations fiables et précises, en prenant en considération à la fois les mesures des capteurs et les informations de mouvement, ce qui est crucial pour la navigation autonome et la construction précise de la carte de l'environnement.

3.2.2. Fonctionnement du filtre de Kalman

Avant d'expliquer l'algorithme, il est important de définir certains termes afin de ne pas perdre le lecteur :

-La covariance est une mesure statistique qui quantifie la variation conjointe de deux variables.

-La pondération est un concept qui implique l'attribution de différents poids ou importances à des éléments ou des facteurs dans un calcul, une évaluation ou une décision.

-Le terme "modèle dynamique" fait référence à une représentation mathématique de l'évolution de l'état d'un système au fil du temps. Il décrit la relation entre l'état actuel du système et son état précédent, en tenant compte de la dynamique telle que la vitesse, l'accélération, etc.

Algorithm 2: Algorithme du filtre de Kalman pour le SLAM

Input: Estimation initiale de l'état du robot $\hat{x}_0$ et de la matrice de covariance P_0
Output: Estimation de l'état du robot $\hat{x}_k$ et de la matrice de covariance P_k

$k = 1$ **to** N // Étape de prédiction
$\hat{x}_k^- = A \cdot \hat{x}_{k-1}$ *[r]Prédiction de l'état du robot
$P_k^- = A \cdot P_{k-1} \cdot A^T + Q$ *[r]Prédiction de la covariance
// Étape de mise à jour
$K_k = P_k^- \cdot H^T \cdot (H \cdot P_k^- \cdot H^T + R)^{-1}$ *[r]Gain de Kalman
$y_k = z_k - H \cdot \hat{x}_k^-$ *[r]Résidu de mesure
$\hat{x}_k = \hat{x}_k^- + K_k \cdot y_k$ *[r]Mise à jour de l'état du robot
$P_k = (I - K_k \cdot H) \cdot P_k^-$ *[r]Mise à jour de la covariance

Les variables :
$\hat{x}_k$: État estimé du robot à l'instant k. Il représente la meilleure estimation de l'état du robot, incluant sa position, son orientation, sa vitesse, etc.
P_k : Matrice de covariance à l'instant k. Elle représente l'incertitude ou l'erreur associée à l'état estimé $\hat{x}_k$. Une valeur plus élevée dans la matrice de covariance indique une plus grande incertitude dans les variables d'état correspondantes.
N : Nombre total d'itérations ou d'instantanés.
A : Matrice de transition d'état. Elle décrit la relation entre l'état courant $\hat{x}k$ et l'état prédit $\hat{x}k^-$ à l'instant suivant.
Q : Matrice de covariance du bruit de processus. Elle représente l'incertitude ou le bruit dans le modèle de processus ou de mouvement. Elle tient compte de la dynamique du système et capture l'écart entre les transitions d'état prédites et réelles.
H : Matrice de mesure. Elle fait correspondre l'état estimé $\hat{x}_k$ à l'espace de mesure. Elle définit comment l'état prédit est lié aux mesures ou lectures des capteurs.
R : Matrice de covariance du bruit de mesure. Elle représente l'incertitude ou le bruit dans les mesures obtenues à partir des capteurs. Elle capture les erreurs de lecture des capteurs ou les inexactitudes dans le modèle de mesure.
K_k : Gain de Kalman à l'instant k. Il détermine la contribution de la mise à jour de mesure à l'estimation d'état. Il pondère l'importance de l'état prédit et de la nouvelle mesure dans le processus de fusion.
z_k : Mesure ou lecture du capteur à l'instant k. Elle représente les données observées obtenues à partir des capteurs.
y_k : Résidu de mesure ou innovation à l'instant k. C'est la différence entre la mesure réelle z_k et la mesure prédite basée sur l'état estimé $\hat{x}_k^-$.
I : Matrice identité. Il s'agit d'une matrice carrée avec des uns sur la diagonale et des zéros ailleurs. Elle est utilisée dans l'étape de mise à jour de covariance pour mettre à jour la matrice de covariance.

1. L'équation $\hat{x}_k^- = A \cdot \hat{x}_{k-1}$ effectue la prédiction de l'état du robot en utilisant le modèle dynamique représenté par la matrice A. Cela permet d'estimer la nouvelle position et

l'orientation du robot en se basant sur l'estimation précédente.

2. L'équation $P_k^- = A \cdot P_{k-1} \cdot A^T + Q$ prédit la covariance de l'état du robot en prenant en compte la covariance précédente P_{k-1} et le bruit de processus représenté par la matrice Q. Elle reflète l'incertitude associée à la prédiction de l'état du robot et de la carte.

3. Le gain de Kalman est calculé à l'aide de l'équation $K_k = P_k^- \cdot H^T \cdot (H \cdot P_k^- \cdot H^T + R)^{-1}$. Le gain de Kalman détermine la pondération entre l'estimation de l'état prédit du robot et la mesure réelle provenant des capteurs. Il est calculé en comparant la covariance prédite P_k^- avec la covariance de mesure $H \cdot P_k^- \cdot H^T + R$, où H est la matrice de mesure et R est la covariance du bruit de mesure. Un gain élevé indique une confiance accrue en la mesure.

4. Le résidu de mesure est calculé par l'équation $y_k = z_k - H \cdot \hat{x}_k^-$, où z_k est la mesure réelle provenant des capteurs. Il représente la différence entre la mesure réelle et l'estimation de la mesure prédite à partir de l'état prédit du robot.

5. L'estimation de l'état du robot est mise à jour en utilisant l'équation $\hat{x}k = \hat{x}k^- + K_k \cdot y_k$. Elle utilise le gain de Kalman pondéré par le résidu de mesure pour ajuster l'estimation de l'état prédit du robot, en tenant compte de la mesure réelle. Cela permet d'améliorer la précision de l'estimation de la position et de l'orientation du robot ainsi que de la carte de l'environnement.

6. La covariance de l'état du robot est également mise à jour en utilisant l'équation $P_k = (I - K_k \cdot H) \cdot P_k^-$. Elle ajuste la covariance prédite en tenant compte de la confiance accordée à la mesure réelle. Une covariance réduite indique une réduction de l'incertitude associée à l'estimation de l'état du robot et de la carte.

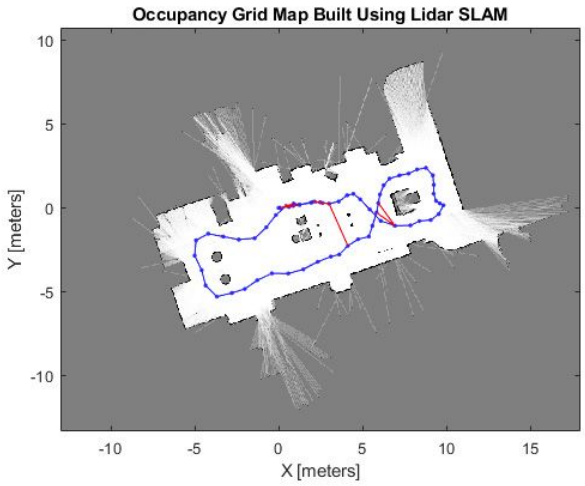

Figure 3: Une visualisation de donnees LIDAR

3.2.3. Problematique du SLAM

Cependant, le SLAM présente un inconvénient majeur lié au coût des composants nécessaires pour assurer son fonctionnement optimal. En effet, l'utilisation d'un LIDAR est souvent indispensable pour obtenir un algorithme SLAM robuste et précis. Contrairement aux capteurs ultrasons, un LIDAR utilise un laser qui offre une plus grande portée et permet d'obtenir la position de plusieurs points dans l'espace, en plus d'une vision à 360 degrés de l'environnement, contrairement à un capteur ultrasonique qui fonctionne en utilisant un cône de détection. De plus, le LIDAR n'est pas limité par des objets qui ne renvoient pas les ondes, ce qui améliore la fiabilité des données de perception. Cependant, l'acquisition d'un LIDAR représente un investissement financier important (au moins 300 euros), ce qui constitue la principale raison pour laquelle nous avons choisi une approche différente.

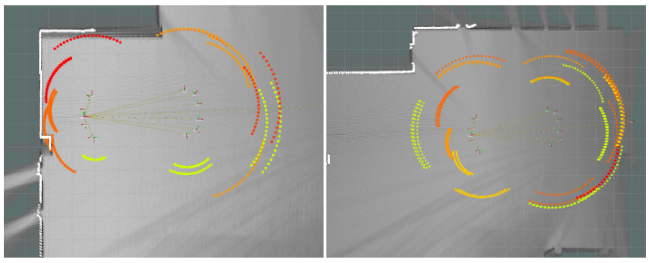

Figure 4: Une visualisation de donnees ultrasons

3.3. Une technique moins couteuse : l'algorithme de la main gauche

3.3.1. l'Algorithmique

L'algorithme de la main gauche est une méthode largement utilisée dans les domaines de la robotique et de la planification de trajectoires pour résoudre le problème de la navigation dans des environnements complexes. Son nom provient du principe consistant à suivre un mur en gardant constamment la main gauche en contact avec celui-ci.

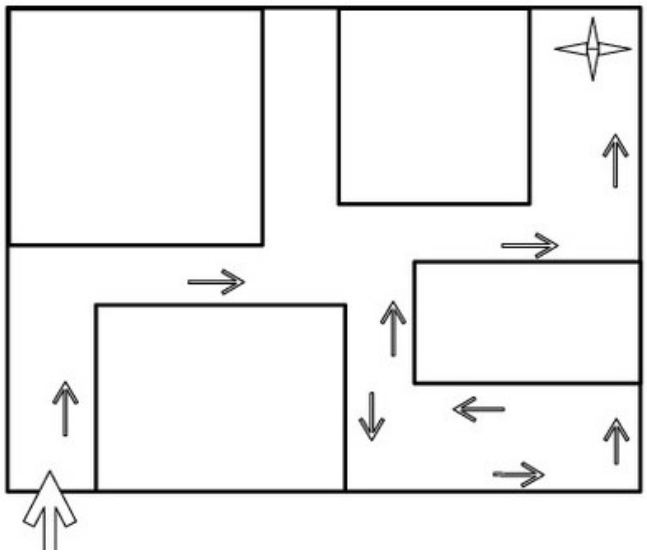

Figure 5: Une visualisation de donnees ultrasons

L'objectif principal de l'algorithme de la main gauche est de guider un robot à travers un labyrinthe ou un espace, en évitant les obstacles et en atteignant une sortie ou une destination spécifique. Cet algorithme repose sur une méthode simple et itérative où le robot suit un mur en gardant constamment ce mur à sa gauche.

Le fonctionnement de l'algorithme de la main gauche est relativement simple et efficace. Le robot commence par choisir une direction initiale et avance en maintenant le mur à sa gauche. Lorsqu'il rencontre un virage à gauche, il tourne dans cette direction et continue d'avancer en suivant le nouveau mur situé à sa gauche. Si le robot rencontre un obstacle ou un mur sur son chemin, il continue de tourner à gauche jusqu'à ce qu'il trouve un chemin dégagé pour avancer. Ce processus se répète jusqu'à ce que le robot atteigne la sortie, qui est détectée lorsque plusieurs mesures ultrasoniques ne détectent pas de retour consécutivement, grâce à l'utilisation de mousse acoustique.

3.3.2. Les limitations

L'algorithme de la main gauche présente à la fois des avantages et des limitations. Parmi ses avantages, on peut mentionner sa simplicité de mise en œuvre et sa capacité à trouver un chemin dans des labyrinthes simplement connexes. Les capteurs ultrasons sont utilisés de manière fonctionnelle, principalement pour détecter les obstacles, sans nécessiter une grande précision.

Algorithm 3: Algorithme de la main gauche

```
Input: Position de départ du robot
Output: Trajectoire du robot jusqu'à la sortie
while le robot n'a pas atteint la sortie do
    if il y a un mur à gauche then
        Tourner à droite; Avancer tout droit;
    else
        if il y a un mur en face then
            Avancer tout droit;
        else
            Tourner à gauche; Avancer d'une case;
        end
    end
end
```

Cependant, cet algorithme peut rencontrer des difficultés dans des situations plus complexes, telles que la présence de boucles ou d'espaces non simplement connexes. Il peut se retrouver coincé dans des configurations où il tourne en rond ou suit des chemins non optimaux. De plus, il ne garantit pas toujours la recherche du chemin le plus court pour atteindre la sortie, car il se base uniquement sur le maintien du mur à sa gauche.

Malgré ces limitations, l'algorithme de la main gauche reste une approche populaire pour la navigation robotique en raison de sa simplicité et de sa capacité à résoudre des problèmes de base. Dans des environnements complexes ou pour des objectifs spécifiques, des techniques plus avancées et des algorithmes plus sophistiqués peuvent être nécessaires.

3.4. Une solution aux boucles : une technique SLAM, l'estimation de la position //

Ainsi, afin d'éviter que le véhicule ne boucle indéfiniment autour d'un obstacle, nous avons pris la décision d'ajouter un module ESP32 à notre système. Ce module permet une communication sans fil entre l'Arduino et un ordinateur distant. En établissant une connexion via Wi-Fi, nous pouvons transférer les données du véhicule à l'ordinateur distant pour un traitement ultérieur.

Le programme utilise la bibliothèque paho-mqtt pour récupérer les messages envoyés via le protocole MQTT par un module ESP. Les messages possibles sont "avant", "arrière", "gauche" et "droite", qui représentent les mouvements d'un robot. Les mouvements "avant" et "arrière" sont basés sur une distance fixe de 3,71 cm, tandis que les mouvements "gauche" et "droite" correspondent à une rotation de 4 degrés dans la direction respective.

Le code utilise les coordonnées initiales (0:0) du robot et effectue des calculs trigonométriques pour estimer sa position actuelle en fonction des mouvements précédents. Les coordonnées sont enregistrées dans un tableau à deux dimensions pour détecter si le robot est coincé dans une boucle. Dans

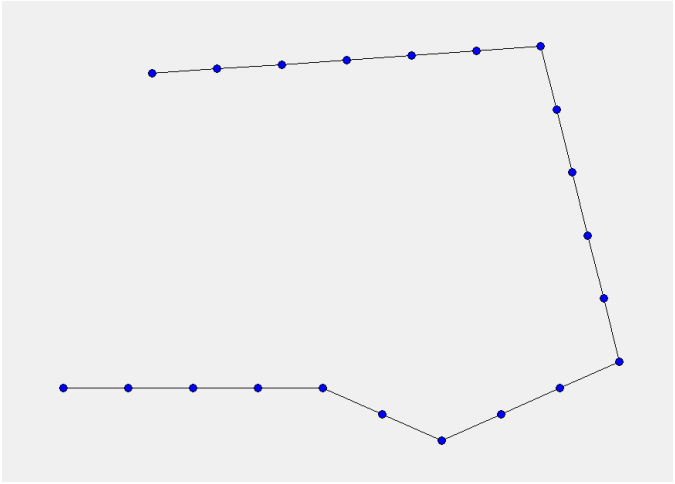

Figure 6: Tableau décrivant le trajet du robot

ce cas, un message MQTT est envoyé pour lui demander de tourner et d'éviter de revisiter les zones déjà parcourues.

Bien que le code n'implémente pas directement du SLAM ou des filtres de Kalman, il présente des concepts et des techniques liés au suivi de position et au contrôle de mouvement. En enregistrant les positions précédentes et en détectant les boucles, le code peut éviter au robot de se retrouver coincé dans un mouvement répétitif. Cela permet d'améliorer l'efficacité et la fiabilité des déplacements du robot. Cependant, pour des applications plus avancées nécessitant une localisation et une cartographie précises, l'utilisation de techniques telles que le SLAM ou les filtres de Kalman peut être nécessaire.

4. Conclusion

En conclusion, ce projet a nécessité une réflexion approfondie tant sur le plan algorithmique que sur le choix des composants. Nous avons dû faire preuve d'adaptabilité à plusieurs reprises afin de respecter le budget qui nous avait été attribué. De plus, nous avons exploré et adapté des algorithmes tels que le SLAM et le filtre de Kalman à une échelle réduite, afin d'éviter les limitations des approches plus simples.

Cette expérience nous a permis de comprendre les défis spécifiques liés à l'implémentation de ces techniques dans notre contexte particulier. En adaptant les algorithmes à une plus petite échelle, nous avons pu mieux appréhender les contraintes et les limitations auxquelles nous avons été confrontés, et cela nous a également poussés à prendre des décisions stratégiques pour résoudre des problèmes complexes tout en tenant compte des contraintes budgétaires et des spécificités du projet.

Dans l'ensemble, ce projet nous a permis d'acquérir une expérience précieuse dans le domaine de la robotique autonome et de développer nos compétences en matière de programmation, de conception de systèmes et de résolution de problèmes. Nous sommes fiers d'avoir réussi à réaliser un système fonctionnel malgré les contraintes et les obstacles rencontrés. Nous espérons que cette expérience servira de base solide pour de futurs projets et nous permettra de continuer à explorer et à innover dans le domaine de la robotique.