POLYTECH NANCY

# Commande du robot 3pi+

*Erasmus Student*
*Dmytro Savchuk*
*For*
*5A IA2R*

UNIVERSITÉ DE LORRAINE | LORRAINE INP vos talents se lèvent à l'Est

04/02/2022

# Sommaire

## Table des matières

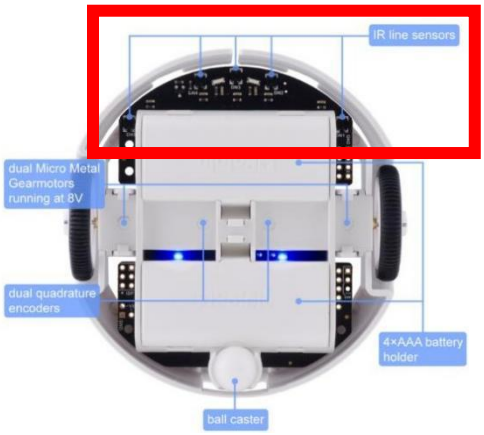
# I - Introduction

Le robot 3pi+ 32U4 est un robot programmable grâce à la carte A-Star 32U4 intégré dans ce module.


ARDUINO™

En termes des capteurs, le robot contient cinq capteurs, Les cinq capteurs de ligne qui se trouvent à l'avant du robot permettent d'aider le robot 3pi+ à distinguer une ligne non réfléchissante (noire) sur une surface réfléchissante (blanche).

Chaque capteur de réflectance se compose d'un émetteur infrarouge (IR) analogique orienté vers le bas LED (Light-Emitting Diode) associée à un phototransistor capable de détecter la lumière infrarouge réfléchie par la LED. Trois des cinq capteurs sont placés au milieu et les deux derniers se situent à l'extrémité des deux côtés




En ce qui concerne les actionneurs, le robot est entrainé par deux moteurs couplés à ses deux roues une à droite et une à gauche.

Le robot avance droit lorsque les deux roues tournent à la même vitesse. Le robot se dirige vers la droite ou vers la gauche si la vitesse de l'un de ses moteurs tourne à une vitesse plus réduite que la deuxième.

Le défi de cette étude, est de créer une commande permettant le robot de parcourir la piste de compétition de course qui a la forme de F1 stage Estoril au Portugal le plus vite possible comme le ferai un pilote professionnel.


# II - Stratégie 1 : Ajustement d'une stratégie existante

## II.a. - Explication de la stratégie

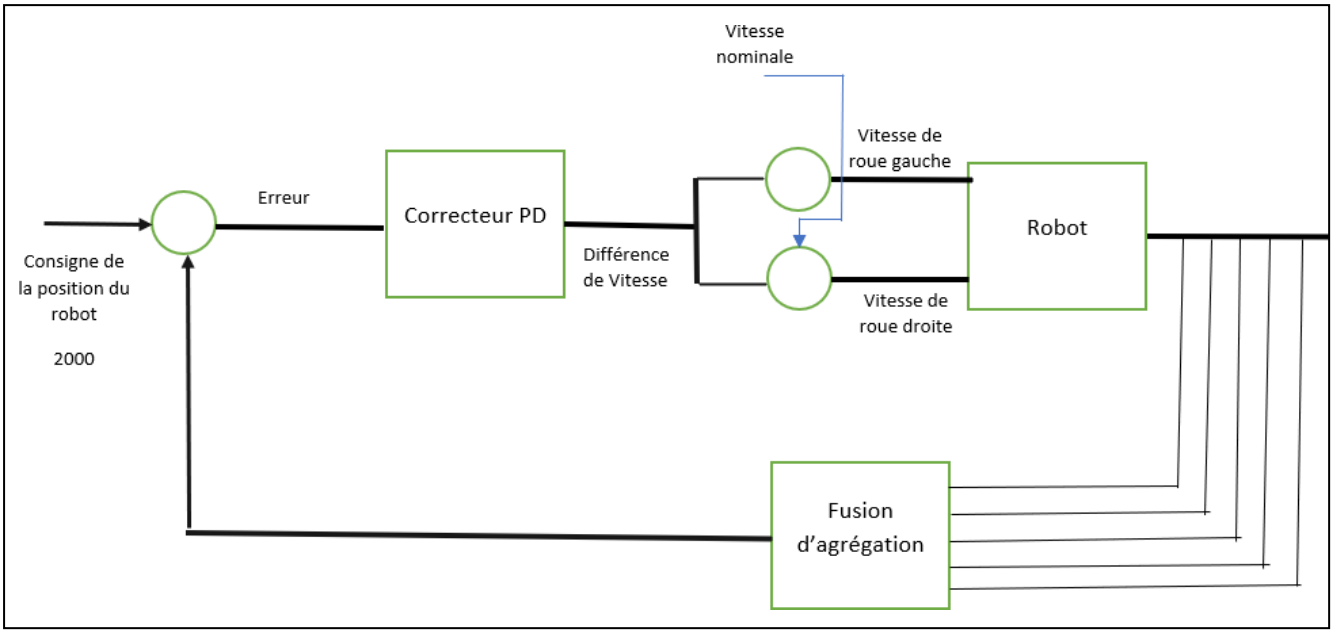
L'objectif de ce TP est d'établir la meilleure stratégie pour que le robot puisse effectuer un tour de piste dans le temps le plus court possible avec la plus grande vitesse possible.

La première stratégie consiste à optimiser le travail qui a été réalisé par nos collègues. La stratégie de base de nos collègues consiste donc à repérer la ligne noire à l'aide de la fusion d'agrégation des cinq capteurs. Les cinq capteurs de ligne sont orientés vers le bas et peuvent aider le robot 3pi+ à distinguer une ligne non réfléchissante (noire) sur une surface réfléchissante (blanche). La valeur du capteur est inférieure sur une surface blanche et supérieure sur une surface sombre.

À partir de la fusion d'agrégation des 5 capteurs LED, on renvoie la valeur de la position du robot. Le robot est considéré au milieu de la ligne noire quand il est à la valeur 2000. On est bien confronté à une problématique de régulation de la position du robot. Avec la consigne de la position est égale à 2000.

Pratiquement, des tests conditionnels en fonction de la valeur maximale de la somme des erreurs sur les 10 dernières mesures ont été effectués. Lorsque l'erreur est plus importante, la correction sera naturellement plus importante, ce qui entraîne une réduction de la vitesse du robot.

Le choix des valeurs $K_p$ et $K_d$, les gains proportionnelles et dérivées du correcteur PD ont été choisis en se basant sur la méthode essai-erreur.




## II.b. - Amélioration de la stratégie

La difficulté du problème est causée par la faite que le système n'était pas conçu d'une façon qui permet l'extraction et la communication des données (concernant les capteurs et les actionneurs par exemple). D'où la difficulté d'étudier la réponse du robot et de créer un modèle qui traduit son évolution et sa dynamique à partir des données.

L'unique solution pour atteindre objectif est de commander la vitesse nominale du robot selon l'erreur vu par ses capteurs.

L'idée dans cette partie est donc d'ajuster et d'affiner la solution expliquée précédemment (partie II.a). Les deux paramètres sur lesquels il est possible d'agir sont : le premier, est l'erreur qui est la différence entre la sortie des capteurs et la consigne (dans notre cas la consigne égale à 2000) et le deuxième la vitesse nominale qui est le paramètre de commande.

Plusieurs manières de calculer l'erreur ont été mise en pratiques :

### L'erreur maximale

La solution consiste à stocker dans un tableau les dix dernières valeurs d'erreurs et de prendre en compte par la suite seulement la plus grande valeur (qui représente le pire des cas).

| Case 0 | Case1 | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 | Case 9 |
|---|---|---|---|---|---|---|---|---|---|
| Erreur 0 | Erreur 1 | Erreur 2 | Erreur 3 | Erreur 4 | …… | ……. | ……. | ……. | ….. |

Axe du temps t

Le défaut de cette méthode survient lorsque l'erreur maximale se retrouve sur l'une de premières cases du tableau (ancienne erreur), cela s'explique par la faite que le robot prendra une décision selon une erreur ancienne alors que à l'instant t sa position à changer et la décision prise ne convient plus à ce dernier.

### L'erreur moyenne

La solution consiste à stocker dans un tableau les dix dernières valeurs d'erreurs (largeur de la fenêtre est égale à dix), et de calculer la moyenne arithmétique des erreurs :

$$Moyenne\ des\ erreurs = \frac{\sum erreurs}{Largeur\ de\ la\ fenêtre}$$

### La somme des erreurs

La solution consiste à stocker dans un tableau les dix dernières valeurs d'erreurs et de calculer la somme.

En prenant la somme des erreurs, le robot prend la vitesse nominale minimale car il se retrouve toujours dans la pire condition (le pire des cas lorsque l'erreur est très grande) et par conséquence le robot est lent sur la course.

Cette solution peut être utile dans un autre contexte où la sécurité est plus importante que la vitesse.

### L'erreur moyenne avec coefficients

La solution revient à la solution précédente (l'erreur moyenne), mais en ajoutant cette fois un coefficient qui permet de donner une plus grande importance à la dernière donnée d'erreur (la plus récente) et inversement, un poids plus faible ou moins d'importance au premiers données (donnée moins récentes).

$$\text{Moyenne pondérée} = \frac{x_1 \times p_1 + x_2 \times p_2 + \cdots + x_n \times p_n}{p_1 + p_2 + \cdots + p_n}$$

### Largeur de la fenêtre d'erreur

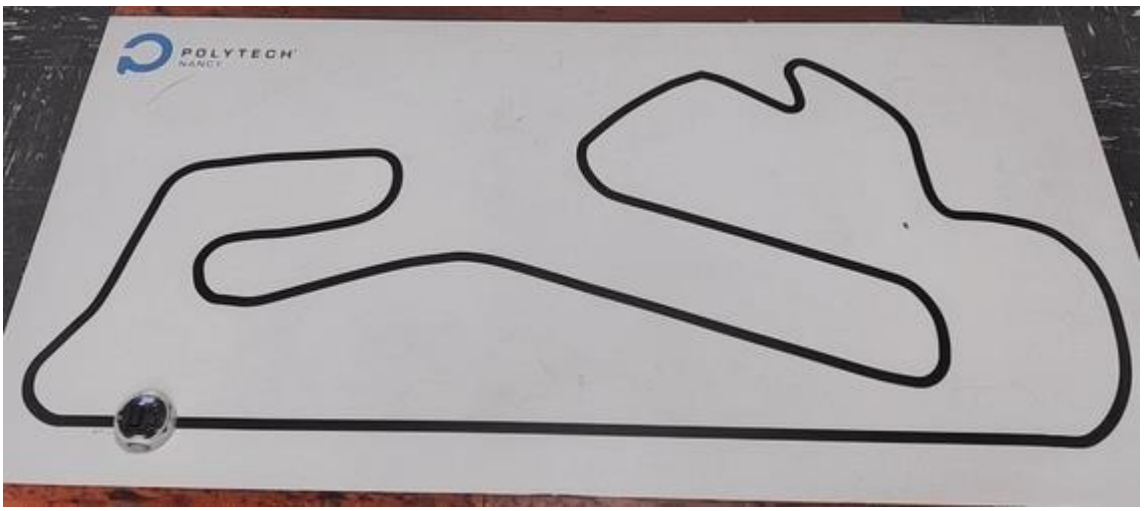
Ici, l'idée était d'étudier la taille de la largeur de la fenêtre d'erreur, comment ce paramètre peut influencer la performance de la stratégie de commande.

Il fallait tout simplement essayer et comparer les résultats obtenus en fixant à chaque fois une largeur de fenêtre différente.

Pour conclure, le robot a une meilleure performance lorsque la largeur de la fenêtre est égale à cinq. Une grande largeur de la fenêtre impacte sur la vitesse et donc sur le temps écoulé pour faire un tour. Une largeur de fenêtre petite impacte sur la stabilité du robot.

### II.c. - Résultats

Le meilleur résultat obtenu est le suivant : Une largeur de fenêtre d'une valeur de cinq, a permis le robot à faire un tour de piste dans les deux sens avec une vitesse de 3.6 m/s en 11.80 secondes.




# III - Stratégie 2 : gain adaptatif

## III.a. - Modélisation dynamique

### III.a.i. - Modélisation dynamique du gain proportionnel

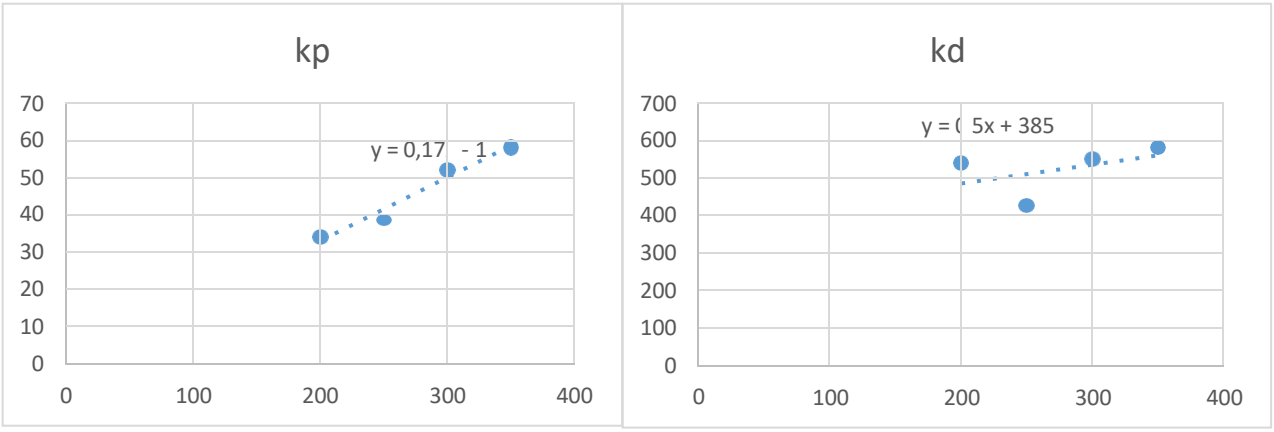
Pour contourner le problème lié au choix de Kp et Kd, la prochaine stratégie consiste à exprimer ces gains en fonction de la vitesse de consigne.

Chaque valeur de kp, kd et erreur max correspondent à une valeur de vitesse.

| Vitesse | Kp | Kd | Erreur Max |
|---|---|---|---|
| 350 | 58 | 580 | 20 |
| 300 | 52 | 550 | 30 |
| 250 | 39 | 420 | 75 |
| 200 | 34 | 540 | 400 |

Ces dernières ont été établies par « essai-erreur » et ont été réutilisés pour tracer les évolutions suivantes. Différents modèles ont donc été utilisés.

Modèle linéaire :




Les premiers modèles qui ont été estimé sont les modèles linéaires. Ce sont des modèles simples et facilement implémentables. Lors des essais, les résultats ont été concluant puisqu'avec une vitesse fixée de manière arbitraire le robot avait une correction du suivi de ligne visuellement bonne.

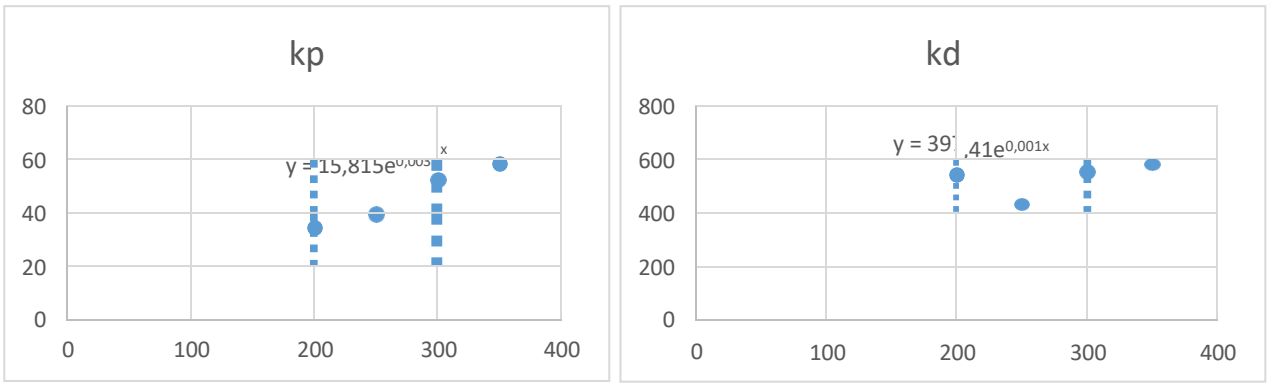
Pour essayer d'obtenir un correcteur PD plus efficace, d'autres modèles ont été utilisés pour les estimations.

Autres modèles :

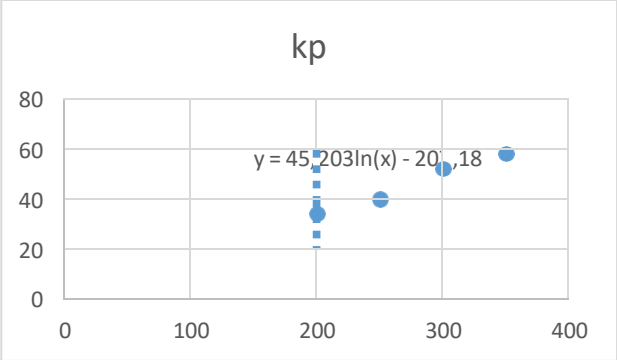
En utilisant l'outil Excel, d'autres modèles ont pu être utilisé pour estimer les variations des gains proportionnels et dérivés en fonction de la vitesse.

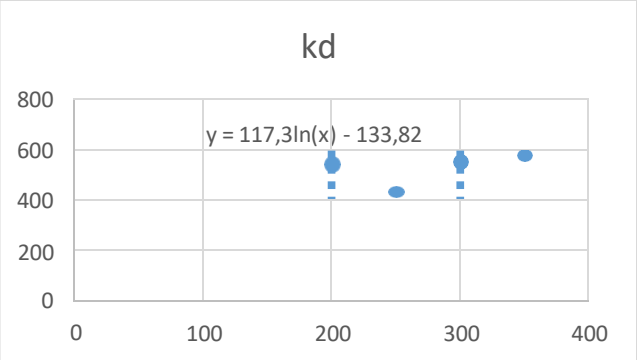
On note par exemple :

L'estimation exponentielle

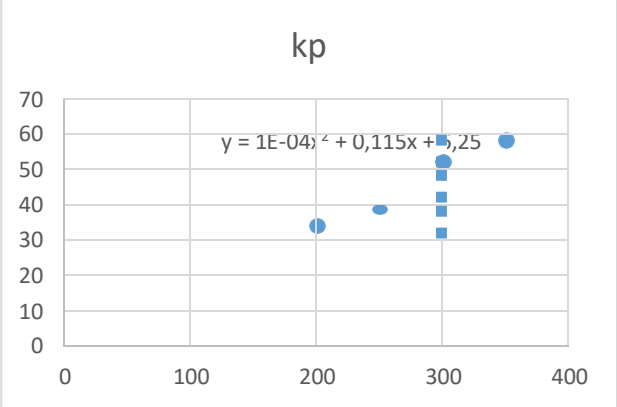


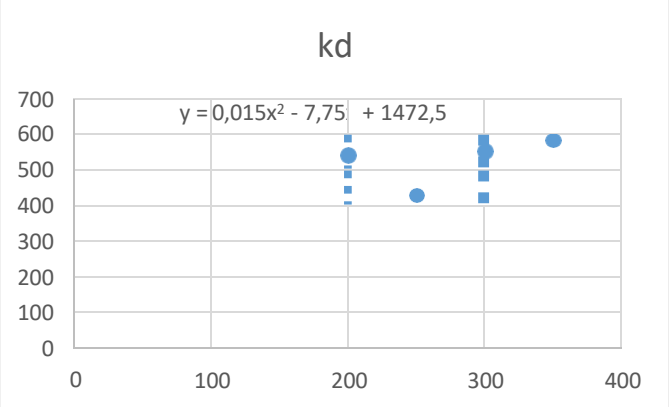
L'estimation logarithmique :



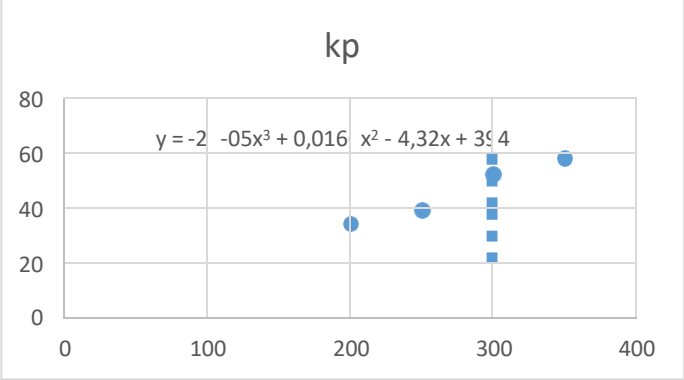



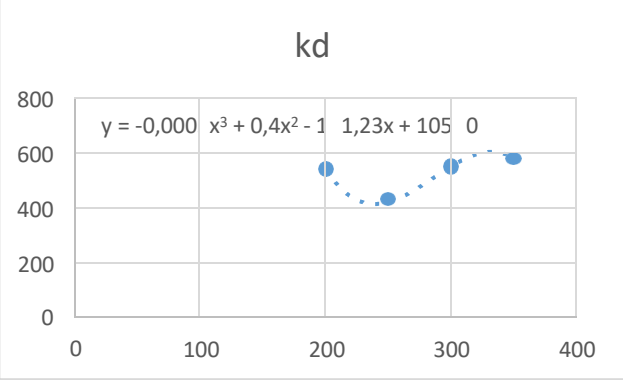

L'estimation polynomiale d'ordre 2 :




L'estimation polynomiale d'ordre4 :




Tous les modèles précédents ont été implémentés. Lors des essais, une analyse visuelle de la correction effectuée avec les différents modèles a montré que le correcteur PD basé sur un modèle linéaire est le plus efficace.

Ces résultats sont à nuancer :

Les combinaisons de vitesse, Kp et Kd ont été établis par la méthode « essai-erreur ». Elles possèdent donc des erreurs d'estimation. Chaque combinaison a donc une part d'erreur inconnue.

Les modèles ne passant pas exactement par les points sont à privilégier. Ces modèles, prennent en compte une erreur dans les estimations des mesures.

Une phase d'essai-erreur plus long pourrait être effectuer pour obtenir soit des combinaisons avec moins d'erreurs soit un plus grand nombre de combinaisons repartie sur une plage de vitesse plus grande.

La nouvelle problématique pour avoir un suivi de ligne le plus efficace et le plus rapide possible est le choix de la vitesse.

Dans cette partie, l'idée est de faire varier la vitesse en fonction du modèle suivant :

$$\text{vitesse} = \text{f}(\text{erreur})$$

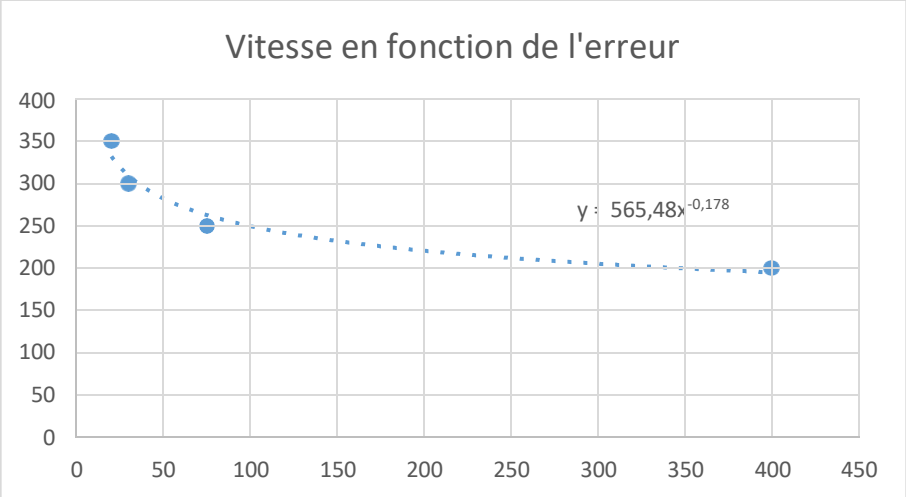
Au lieu de se baser sur des intervalles d'erreurs (voir stratégie 1).

Le prochain objectif est de réaliser un modèle de vitesse en fonction des données disponibles sur l'environnement du robot. Dans ce cas, les seules données disponibles sont les erreurs transmissent par les capteurs de suivis de ligne.

Par la suite, la valeur d'erreur prise en compte sera l'erreur maximale de N dernières erreurs mesurées.

En reprenant les seuils d'erreurs précédents et les vitesses correspondantes, une estimation de vitesse en fonction de l'erreur a été modélisé avec la même méthode que les modèles des correcteurs PD.

Nous avons donc le modèle suivant :



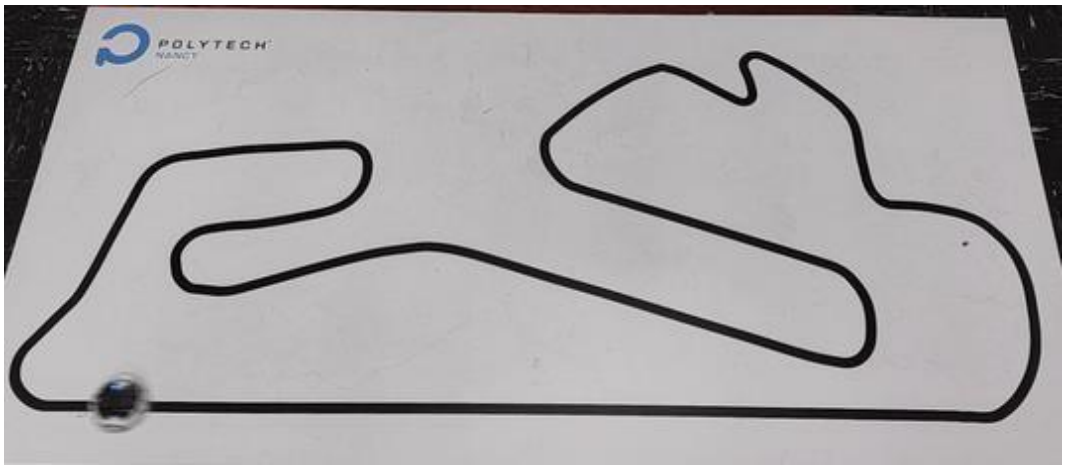

Lors des essais, seul le modèle exponentiel a été implémenté.

Ce modèle permet visuellement mieux de suivre la trajectoire de la piste que la stratégie de la partie II mais la vitesse au tour est moins grande. Le temps est d'une valeur de 12.4 secondes contre 11.80 secondes.

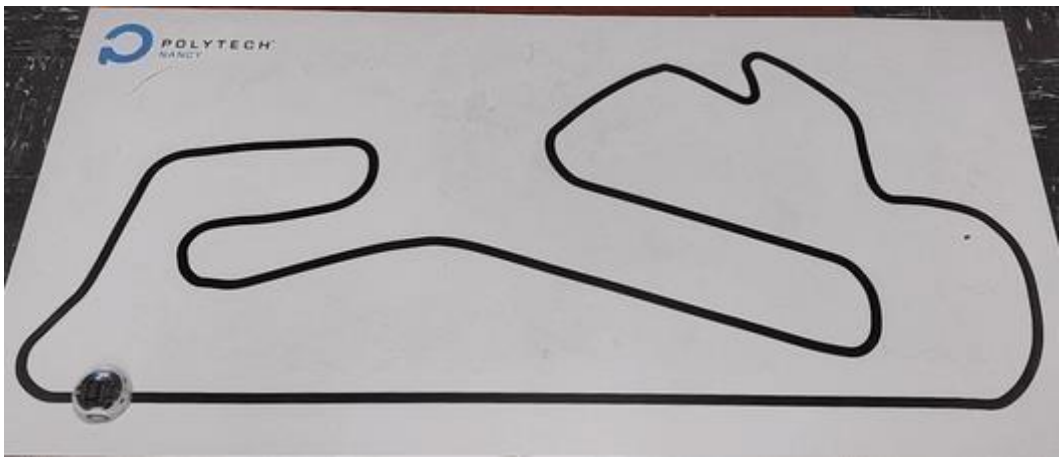
Pour progresser en temps, les vitesses des couples utilisées ont été augmentée de 10 points en gardant la même valeur de gain Kp et Kd.

Pendant les essais, il a été observé que la correction du véhicule était moins efficace. Ce résultat était prévisible puisque les valeurs de Kp et Kd n'ont pas été optimisé pour les vitesses décrites. Mais il a été observé qu'à une vitesse maximum >= 360 le robot finissait par sortir de la piste à cause d'une inertie trop grande à l'entrée des virages.

D'autres essais ont été réalisé avec des vitesses max plus basse, La vitesse Max a été fixé à 355 points pour garder un suivi de ligne avec peu de déviation (visuellement observé).



Pour améliorer la réaction du robot et ainsi gagner en précision de suivi de ligne, des essais ont eu lieu avec une valeur de N (soit N la largeur de la fenêtre) plus petite pour déterminer l'erreur. Le cas le plus efficace observé est lorsque la largeur de la fenêtre est égale à trois (trois prélèvements d'erreurs successives).




## III.a. - Evolution possible

Pour améliorer ce modèle il est possible d'agir sur :

- La méthode de calcul des erreurs différentes en utilisant d'autres approches (les dérivés des erreurs, moyenne mobile et moyenne mobile exponentielle).
- Sur la précision des combinaisons (vitesses, Kp, Kd, erreur) pour limiter les erreurs des gains proportionnel et dérivé ainsi que les modèles de vitesse en fonction de l'erreur.
- Sur le nombre des combinaisons (vitesses, Kp, Kd et erreur) pour améliorer les gains kp et kd ainsi que les modèles de vitesse en évitant des extrapolations de modèle. Des données plus nombreuses peuvent aussi aider à choisir un modèle adapté et plus fiable si les valeurs sont suffisamment précises.
- Sur le type de correcteur en essayant un correcteur LQR. Il sera aussi possible d'ajouter des contraintes pour trouver la valeur optimale entre consommation et vitesse.

D'autres améliorations sont possibles mais nécessite des changements physiques sur le robot.

Ajouter des capteurs pour obtenir d'autres informations :

- Traitement d'image (camera),
- Détection de virage (capteur de couleur + ligne colorer),
- Capteur TOF ou lidar si bordure autour de la piste,
- Cartographie du circuit + moyen de repérage (triangulation, GPS (pas assez précis), gyroscope, ou autres odométrie relatif & absolue)

Ajouter une méthode d'acquisition pour modéliser le comportement du robot avec une méthode d'identification (Broïda par exemple) :

- Sans fil (Zigbee, wifi, Bluetooth, HF, …)
- Enregistrement direct (interface USB, SD, EEPROM)

# IV - Conclusion

Le but de ce TP est d'utiliser les connaissances acquises dans le domaine d'automatique et de la robotique pour proposer une solution permettant au robot d'effectuer un tour de piste dans le temps le plus court.

Pour cela, deux stratégies ont été proposées. La première stratégie d'ajustement de seuil, qui a pour objectif d'améliorer un travail précédemment effectué. L'application de cette stratégie a permis au robot d'avoir une meilleure performance sur la course avec une largeur de fenêtre est d'une valeur de cinq. On constate qu'une grande largeur de la fenêtre influence sur le temps écoulé pour faire un tour de piste. Et une petite largeur de la fenêtre influence sur la stabilité du robot. Avec une valeur de cinq, le robot arrive à faire un tour de piste dans les deux sens avec une vitesse de 3.6 m/s en 11.80 secondes.

En outre, la deuxième stratégie de la modélisation du gain dynamique consiste à exprimer les gains en fonction de la vitesse de consigne. Cette stratégie a permis au robot de bien suivre la trajectoire de la piste avec une vitesse de 3.55 m/s en 12.4 secondes. La stabilité du robot avec la deuxième stratégie est meilleure que celle de la première stratégie.

Dans le but d'améliorer ce TP, une troisième stratégie est proposée. L'objectif est d'extraire les données de l'erreur en temps réel. Pour cela, il est possible d'utiliser les outils suivants :

- La communication avec un grand câble USB : manque du matériel. En plus, cela peut impacter la dynamique du robot. Le robot n'a pas le même comportement avec ou sans fils.
- Bluetooth, Wifi : bande est perturbé et le réseau le wifi ne capte pas très bien en salle.
- Zigbee : long à mettre en place