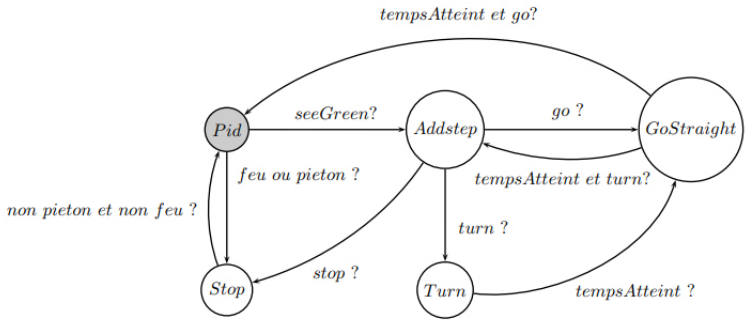
Rapport - Programmation Synchrone

MEGUENNI Amine, SANTOS DUARTE Elsa

December 18, 2022

La fonction principale du programme consiste principalement en un automate, composé de cinq états. Les transitions d'un état à l'autre se font de la façon suivante :



- L'état Pid, qui gère l'action Go dans sa quasi-majorité, et se charge de la bonne direction de la trajectoire. Pour ce faire, on utilise l'algorithme du PID tel que vu en cours en calibrant ses constantes au fur et à mesure de tests. La notion d'erreur dans l'algorithme du PID est libre d'interprétation et propre à chaque implémentation. Nous avons décidé d'utiliser le niveau de rouge ou de vert car plus l'on sort d'une trajectoire droite, plus l'on va sur les côtés de la route et plus cette valeur augmente, il nous a donc semblé que cette mesure était adéquate. Le tout en prenant en compte qu'une grande valeur de vert signifie que nous passons une bande verte. Si tel est le cas, le PID prend la valeur 0.0.

- L'état AddStep qui permet de déterminer l'avancée dans les instructions de la carte, et maintient la vitesse à la vitesse maximale demandée par la map. Il ne boucle donc jamais et est un point de passage incrémentant l'étape actuelle et permettant de passer à la suivante.

- L'état GoStraight permet de continuer en ligne droite durant le nombre d'instants nécessaires après être passé par AddStep ou Turn. Durant ce nombre d'instants, calculé en fonction de la vitesse, le programme ne détecte plus de zone verte et donc plus de nouvelle étape. Cet état permet de corriger une anomalie du programme que nous détaillerons dans la partie suivante.

- L'état Stop rabaisse la vitesse à zéro. Si la voiture est parvenue à la fin du trajet, notamment si elle n'est pas arrêtée en raison d'obstacles ou de feux et que stop est une étape de l'itinéraire, alors arriving est modifié en conséquence. Dès lors que le feu passe au vert ou que les obstacles ne sont plus captés par les senseurs avant, la voiture repart et l'exécution repasse à Pid.

- L'état Turn, qui gère l'action turn et dans lequel on utilise un compteur d'instants pour calculer le temps que la voiture doit mettre à tourner. Chaque roue prend ainsi une direction ou l'autre en fonction de l'angle donné en paramètre. Dès lors que le temps préalablement calculé est atteint, l'exécution repasse à GoStraight.

Au cours des différentes itérations nous avons rencontrés différents problèmes, dont les principaux sont :

- Lorsque l'automate détectait une zone verte, l'étape était incrémentée infiniment jusqu'à atteindre la dernière étape. Une première solution était de tester la valeur de $isGreen([\![sensor]\!]_n) \wedge \neg isGreen([\![sensor]\!]_{n-1})$ pour pouvoir passer à l'étape suivante. Mais en implémentant cette solution, nous nous sommes apperçu.es que le programme captait la zone verte de façon discontinue, causant des incréments d'étapes inattendus. Notre solution a donc été de calculer un temps pendant lequel il était impossible d'incrémenter des étapes. Ce temps est fonction de la vitesse, si cette dernière est basse, le temps sera en mode court, si elle est modérée le temps sera moyen et si elle est élevée le temps sera long. Les correspondances entre vitesse et temps ont été trouvées à la suite de tests successifs.

- Turn prend en paramètre un angle qui est négatif ou positif selon le sens du tour à faire. Cela induit que la roue ayant une valeur négative change selon le sens du tour. Après quelques problèmes d'implémentation, nous avons décidé de créer une constante qui prend la valeur 1 ou -1 en fonction de la valeur de l'angle de l'action Turn et qui est multipliée par la vitesse de chaque roue, nous permettant d'avoir une implémentation plus légère.

Certains problèmes restent cependant encore à régler, parmi lesquels :

- Lorsque la voiture passe sur une étape, la valeur du PID augmente puisque le senseur capte du vert, ce qui entraîne des excès de vitesse ponctuels.

- La solution GoStraight n'est pas la plus optimale car elle nous cause des virages un peu trop importants. La solution de "isGreen(sensor) and not isGreen(sensor)" nous paraissait plus efficace mais malheureusement difficilement adaptable à notre implémentation actuelle.