PROGRAMMATION SYNCHRONE

PROJET VÉHICULE AUTONOME

NILAM DE OLIVEIRA-GILL, LEONARD NAMOLARU

Notre implémentation pour le nœud **controller** est principalement basée sur un correcteur PID. Pour cela, nous calculons le taux d'erreur : une valeur comprise entre -1 et 1 qui représente à quel point le véhicule s'écarte de l'itinéraire prévu (dans notre cas, l'itinéraire prévu est représenté par la ligne bleue sur la route).

Etant donnée que l'espace colorimétrique RGB a trois dimensions (R, G, B), sa distance euclidienne serait définie comme suit[1] : $\mathbf{distance} = \sqrt{(R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 - B_1)^2}$

Par conséquent, nous calculons la distance entre l'emplacement où la voiture est censée se trouver (la ligne bleue) et l'emplacement où elle se trouve réellement (informations que nous recevons via les capteurs de couleur sous la voiture). Ensuite, nous divisons la valeur que nous obtenons par 255 afin d'obtenir un taux d'erreur compris entre 0 et 1, et nous vérifions si la voiture dévie davantage du cote droit (couleur rouge) ou du cote gauche (couleur verte), déterminant ainsi que si la voiture dévie davantage du cote droit, la valeur d'erreur que nous avons calculée sera positive et si la voiture dévie davantage vers la gauche, l'erreur sera négative.

Afin d'effectuer le calcul du PID, nous devons également déterminer les valeurs des trois coefficients : **Kp**, **Ki**, **Kd**. La valeur de **Kp** est égale à la valeur de la vitesse maximale autorisée et la valeur de **Kd** augmente progressivement tant que l'augmentation ne provoque pas d'erreur. Une fois la valeur de **Kd** déterminée, on augmente progressivement la valeur du coefficient **Ki** tant que l'augmentation ne conduit pas à une erreur. Nous réinitialisons les coefficients **Ki** et **Kd** au début du parcours et après chaque virage.

Dans tous les cas, le taux du coefficient **Kd** ne sera pas inférieur à 14, car à partir des tests que nous avons effectués, nous avons découvert que des valeurs inférieures ne conviennent pas pour effectuer une conduite suffisamment stable, et des valeurs supérieures ne conviennent pas à toutes les situations.

Pour l'intégration, on suppose que notre intégrale est échantillonné tous les **dt** instants. Si **dt** est très petit, cela nous donne une bonne approximation de l'intégrale. On fait une hypothèse sur la variation du signal qu'on intègre, dans notre code **dt** est fixe :

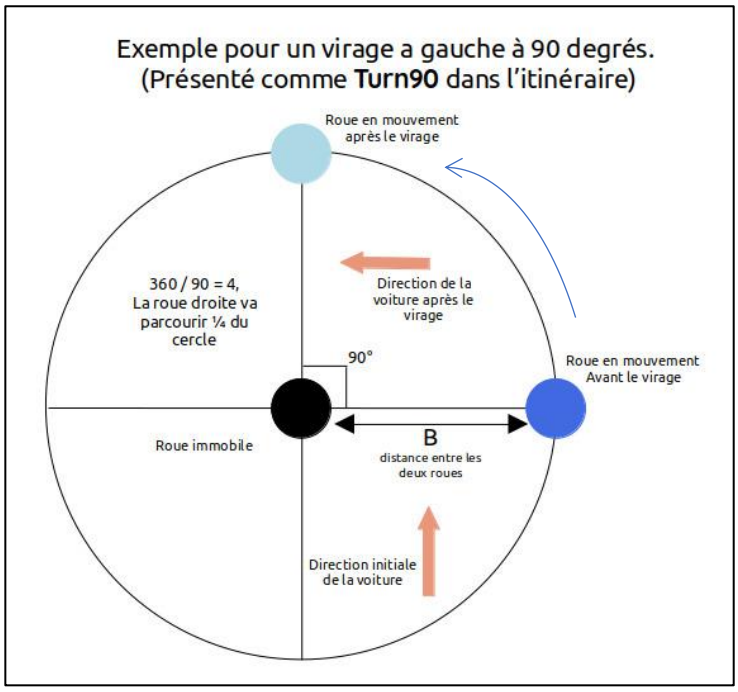
```
const dt : float = 0.01
```

Cela signifie que **dt** secondes (instants physiques) s'ecoulent entre deux instants logiques.

[1] Euclidian Distance in RGB Color : https://observablehq.com/@luciyer/euclidian-distance-in-rgb-color-space

Un autre point important de notre implémentation est la façon dont la voiture roule lorsqu'il y a un besoin de faire un virage : pour les virages, nous avons décidés de tourner sur une roue, celle du côté inverse de la direction ou nous devons aller. Par exemple, si nous voulons aller vers la gauche, la roue gauche de la voiture sera à l'arrêt pendant que celle de droite tournera jusqu'à atteindre la direction souhaitée. La roue qui est en mouvement ira à la vitesse maximale, pendant une durée calculée grâce à l'angle donné par l'itinéraire. Le formule utilisé est celle-ci : $T = D/V$. **T** est un compteur qui est initialisé au temps nécessaire pour le virage, **V** est la vitesse maximale avant le virage.

D est la distance : nous savons que le périmètre d'un cercle est **2 * PI * Rayon**.
Le rayon de ce cercle est la distance entre les deux roues indiqué par **B**. Cependant, nous ne voulons pas le périmètre du cercle entier, c'est pourquoi nous divisons cette donnée par **360/angle du virage**, qui permet d'avoir la distance que la roue en mouvement doit parcourir pour se mettre dans le bon axe.



Dernier point : l'arrêt d'une voiture à un feu de circulation se fait uniquement lorsque les capteurs sous la voiture captent une ligne rouge et lorsque dans le même temps les capteurs à l'avant de la voiture indiquent que le feu est rouge. Ainsi, On peut remarquer sur le graphique qui représente la carte 3 (à gauche) que ces deux conditions sont remplies peu après le 500ème instant et donc la voiture s'arrête. Par contre, la voiture ne s'arrête pas au feu de circulation à la fin du parcours de la carte 7 (le graphique de droite) puisque ces deux conditions ne sont pas remplies en même temps. C'est-à-dire que jusqu'à ce que la voiture soit complètement stabilisée uniquement sur la ligne rouge sur la route, le feu n'est plus rouge

Remarque : pour la carte numéro 8, par exemple, nous obtenons un *output* sur la sortie standard comme si la voiture roulait soi-disant trop vite ou pas dans la bonne direction. Nous ne comprenons pas pourquoi car nous suivons l'itinéraire demandé et selon la sortie standard, qui, par défaut, indique également la vitesse de la voiture, nous ne dépassons pas la limite de vitesse.