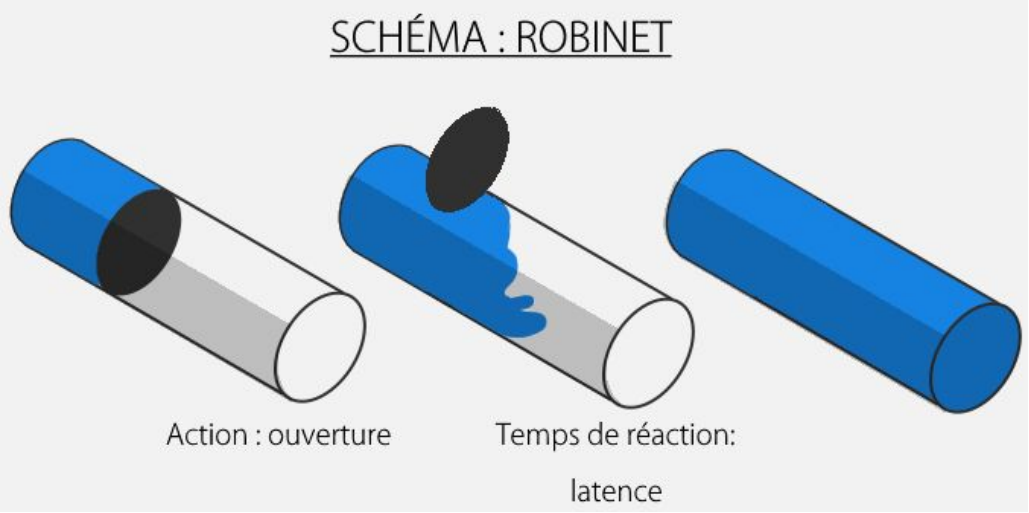
Projet Latence : Robot Flappy Bird

I. Introduction

1. La latence qu'est-ce que c'est ?

Il s'agit du temps de différence qui sépare un ordre de l'action qui en résulte. Par exemple le temps qui s'écoule entre le moment où l'on ouvre le robinet et le moment où l'eau sort.



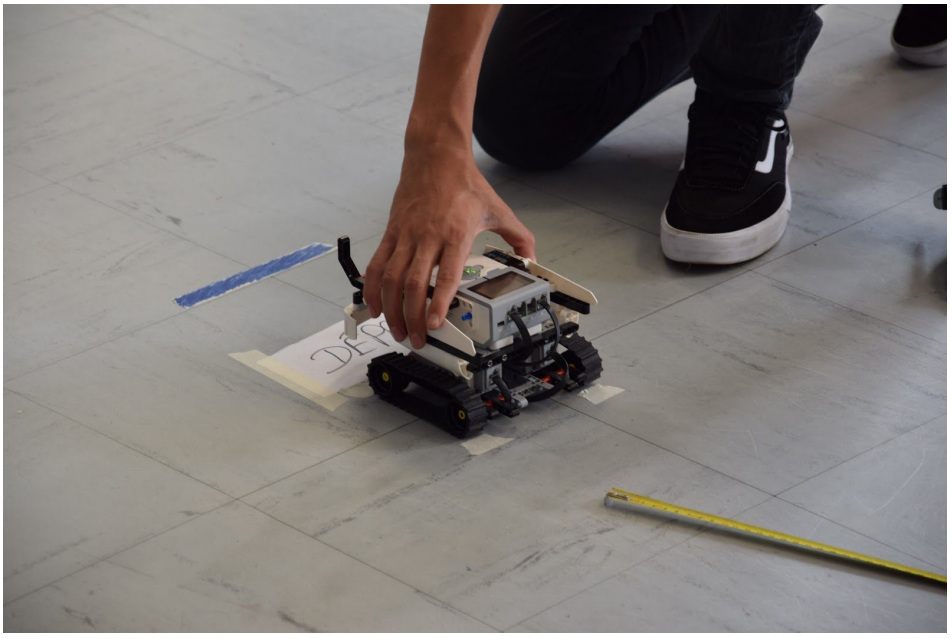
exemple de latence avec le robinet

2. Pourquoi y a-t-il de la latence ?

Tout simplement parce que l'information ne circule pas instantanément. Plus l'information circule vite (à distance égale) moins il y aura de latence. Pour la réduire il faut donc faire appel à des réseaux plus performant, qui feront circuler l'information plus rapidement et contribueront donc à réduire la latence.

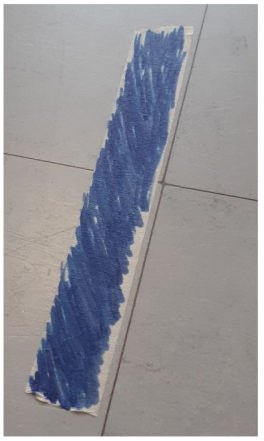
Dans ce projet nous allons chercher à montrer que la latence, même minime, peut occasionner à grande échelle des erreurs plus ou moins importantes. Notre robot Flappy Bird sera notre simulation d'une voiture autonome.

II. Le robot Flappy Bird



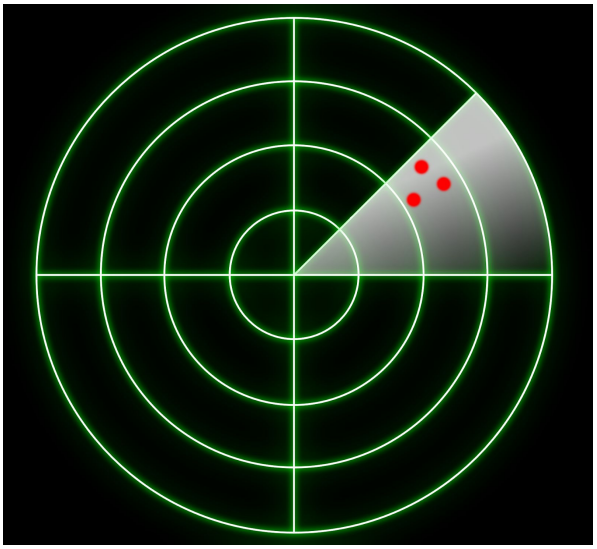
Le robot avance et doit se déplacer pour éviter les obstacles, comme dans le jeu "flappy bird".

Tout le long du parcours sont disposées des bandes de couleurs au sol. Celles-ci ont trois couleurs : bleu, rouge ou vert (on aurait pu en choisir d'autres). Ces bandes permettent au robot d'effectuer des actions en fonctions de leur couleur :



- ROUGE : droite
- BLEU : gauche
- VERT : fin

Ces indications sont donc vitales pour le robot mais peuvent être remplacées, si on veut une application plus concrète, par différents capteurs qui détectent la présence d'obstacles et permettrait au robot de visualiser l'environnement dans lequel il évolue (comme un capteur ultrason, un sonar, …).



III. L'expérience

1. Les conséquences vérifiables :

Nous nous attendons à ce que le temps mis par le robot augmente de manière proportionnelle par rapport au temps de latence programmé.

En effet, on a :

$$v = \frac{d}{t} \Leftrightarrow t = \frac{d}{v}$$

Or la vitesse du robot est constante. On a donc :

$t = \frac{1}{v} * d \Leftrightarrow t = k * d$, c'est à dire que t est proportionnel à d

Ainsi on suppose que le temps mis par le robot et la distance qu'il parcourt (qui augmente avec la latence) sont liés par une relation de proportionnalité.

2. Protocole :

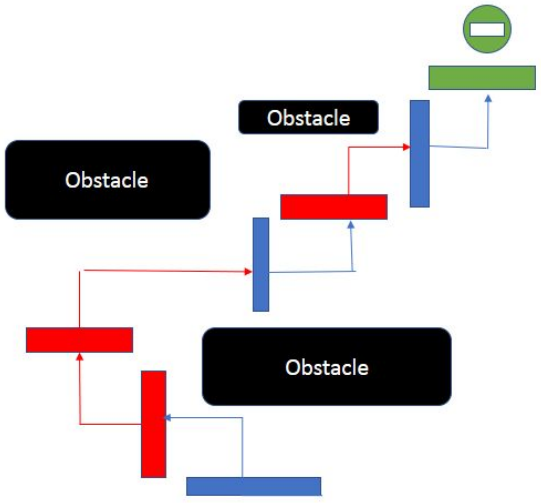
Le robot est placé sur une ligne de départ marquée, qui reste fixe entre chaque passage. On lance le programme (développé plus loin). Puis on réitère avec tous les temps de

latence que l'on veut tester, en changeant la variable. Avec un grand nombre de virages, le robot sort parfois du circuit. Nous sommes donc obligés de déplacer les marques aux sols.

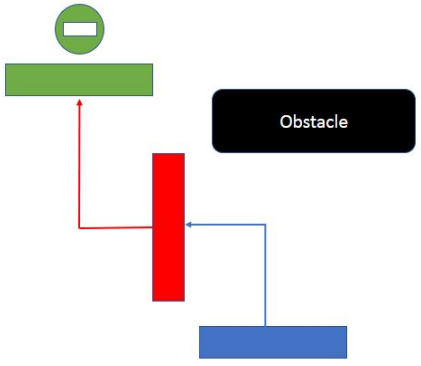
3. Circuit :

On a d'abord opté pour un circuit à 6 virages. Mais le robot en sortait trop souvent et cela devenait trop fastidieux, nous avons donc simplifié le problème avec un circuit à seulement 2 virages.

Circuit à 6 virages



Circuit à 2 virages :



Légende :

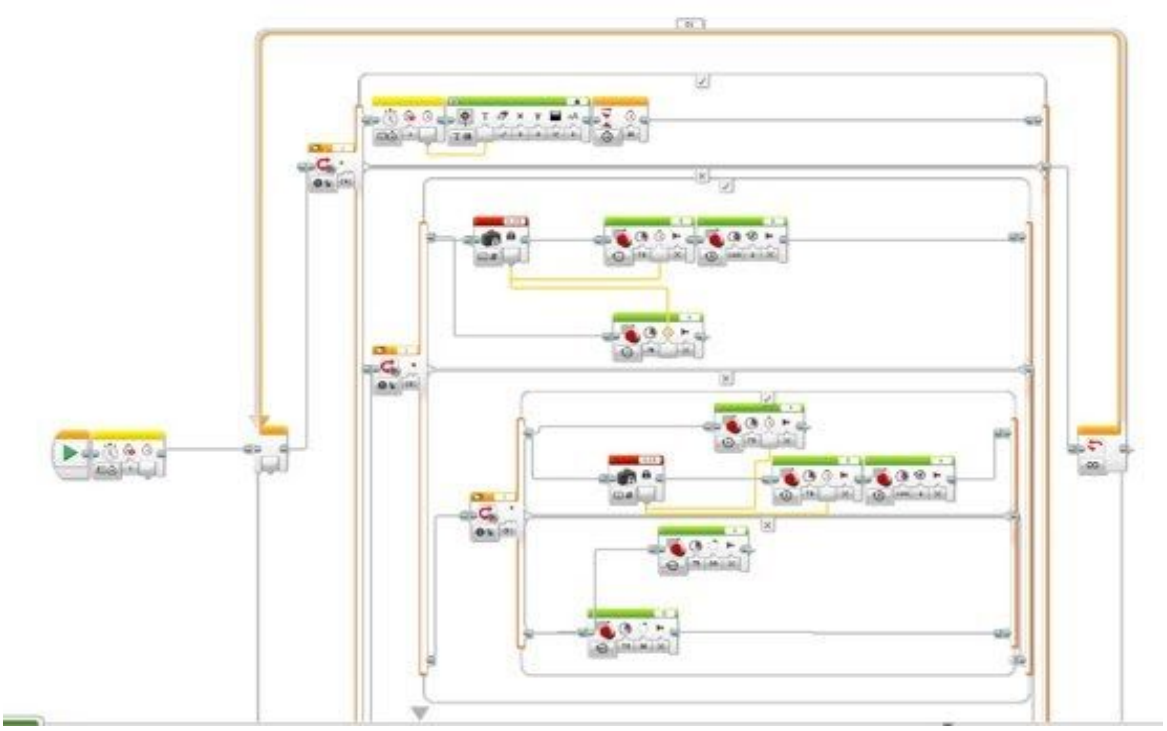
Les obstacles factices que doit éviter le robot. Ils sont uniquement à but représentatif car le but n'est pas de savoir si le robot réussira le parcours mais de voir combien de temps il met pour le faire.

Un marquage au sol de couleur qui indique au robot quoi faire. (ici de couleur rouge, qui donnera l'ordre de tourner à droite)

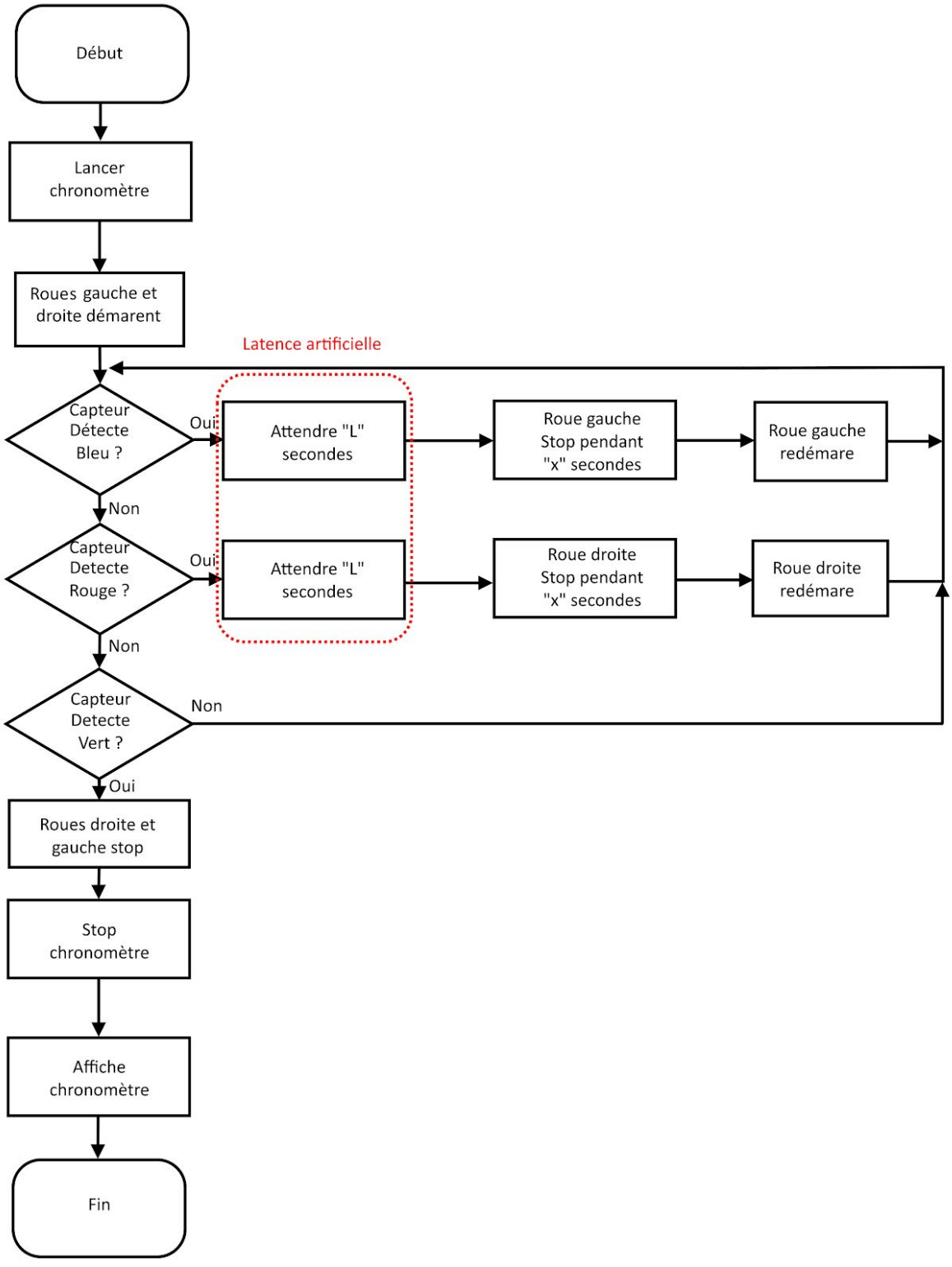
Le trajet du robot. Ici il tourne à gauche après une bande couleur bleue.

Fin du circuit marqué par une bande verte.

4. Programme :



On commence par démarrer le chronomètre. Ensuite dans la boucle infinie on intègre trois sous-boucles if, une pour chaque couleur. Si le capteur détecte du vert, il s'arrête et affiche le temps du chronomètre pendant 30 sec. Lorsqu'il détecte du rouge, il avance pendant un certain temps, qui représente la latence. Ce temps est défini par une variable. Ensuite il tourne la roue gauche afin de tourner à droite . De même lorsqu'il y a du bleu sous le capteur, il avance tout droit pendant un temps défini par une variable, puis fait tourner sa roue droite . Enfin, s'il ne détecte rien, le robot roule tout droit en faisant tourner les deux roues en même temps. Voici, ci-dessous, l'organigramme du programme du Robot.


Début
Lancer chronomètre
Roues gauche et droite démarent
Latence artificielle
Capteur Détecte Bleu ?
Oui
Attendre "L" secondes
Roue gauche Stop pendant "x" secondes
Roue gauche redémare
Non
Capteur Detecte Rouge ?
Oui
Attendre "L" secondes
Roue droite Stop pendant "x" secondes
Roue droite redémare
Non
Capteur Detecte Vert ?
Non
Oui
Roues droite et gauche stop
Stop chronomètre
Affiche chronomètre
Fin

IV. Résultats

1. Premiers tests, avec 6 changements :

Sans latence	Avec 100ms	Avec 200ms	Avec 300ms	Avec 400ms	Avec 500ms	Avec 600ms
16,2s	16,42s	16,62s	16,73s	16,82s	17,01s	17,1s

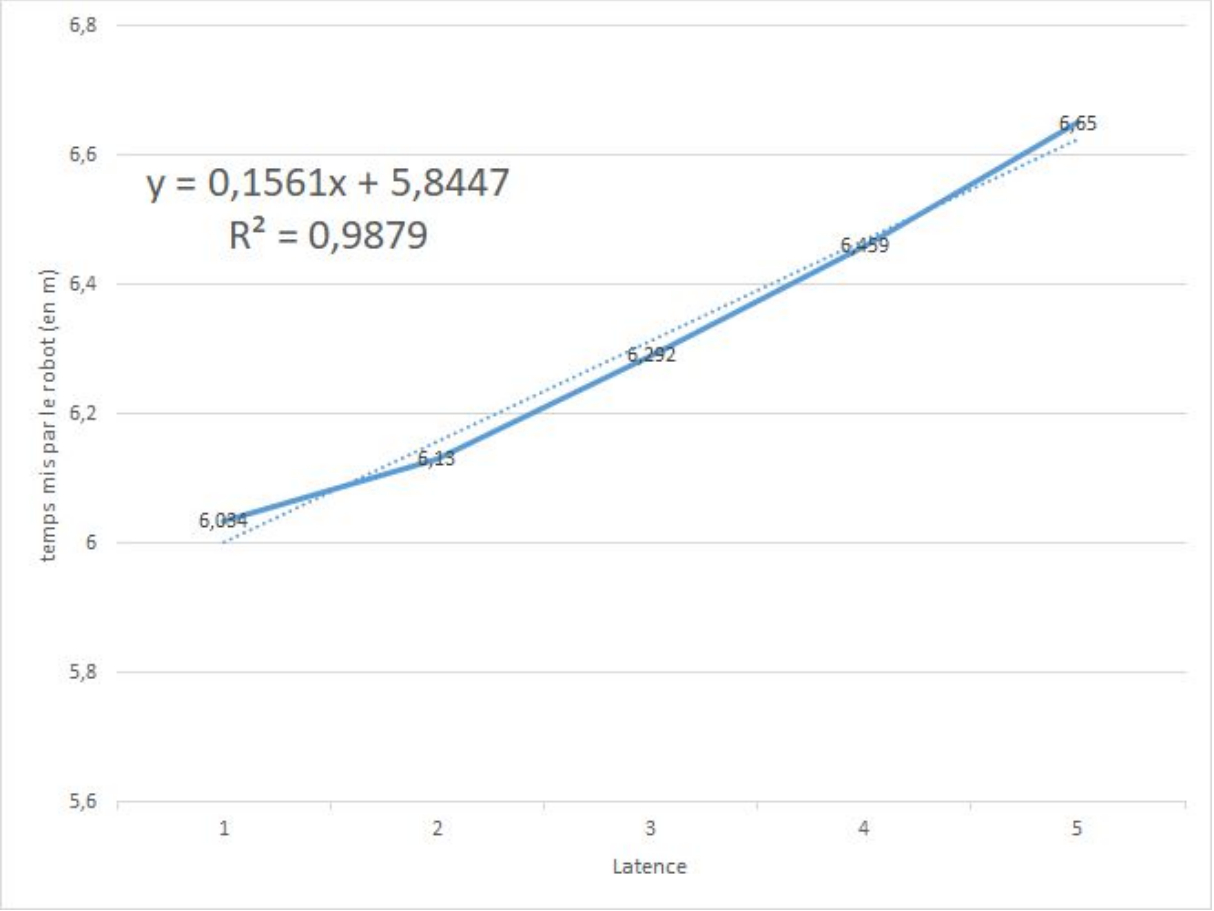
Le robot navigue dans un circuit avec 6 bandes de couleurs (3 rouges et 3 bleues).Les données préconisent un système avec une croissance proportionnelle. Cependant ce test est très fastidieux car il y faut souvent changer les emplacements des couleurs. De plus les mesures sont approximatives étant donné que le chronomètre n'est pas encore intégré au programme. On va donc basculer dans un système à seulement deux changements et avec moins de temps de latence différents.

2. Deuxième tests, avec 2 changements :

(Version 1)

Première et brève série de tests sur ce nouveau "circuit" et avec le nouveau code.

0ms	250ms	500ms	750ms	1000ms
6,034s	6,13s	6,292s	6,459s	6,65s



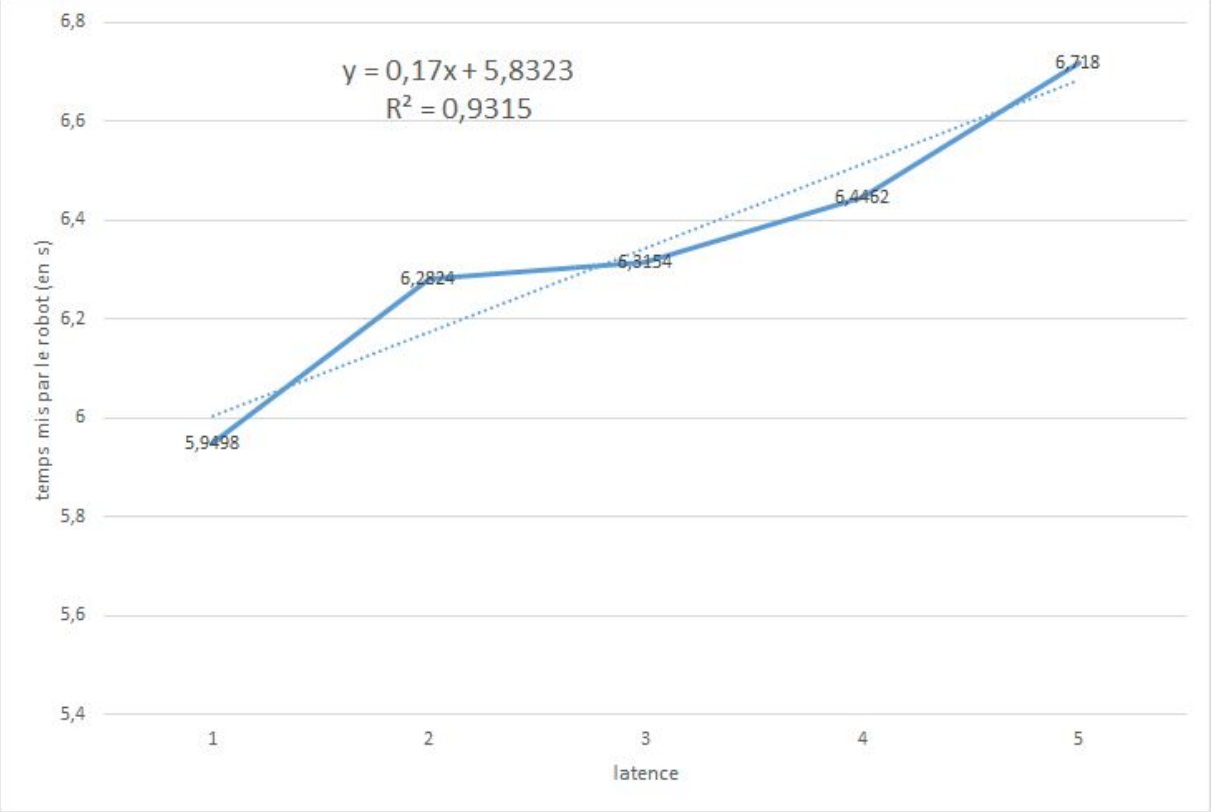
graphique des données du premier test avec 2 virages

Il confirme les hypothèses réalisées avec le test précédent, mais il manque un nombre conséquent de données. Toutefois, par observations nous avons l'impression que la batterie joue un rôle important. Nous allons donc réaliser une nouvelle série de tests, avec plus de données et avec une légère différence.

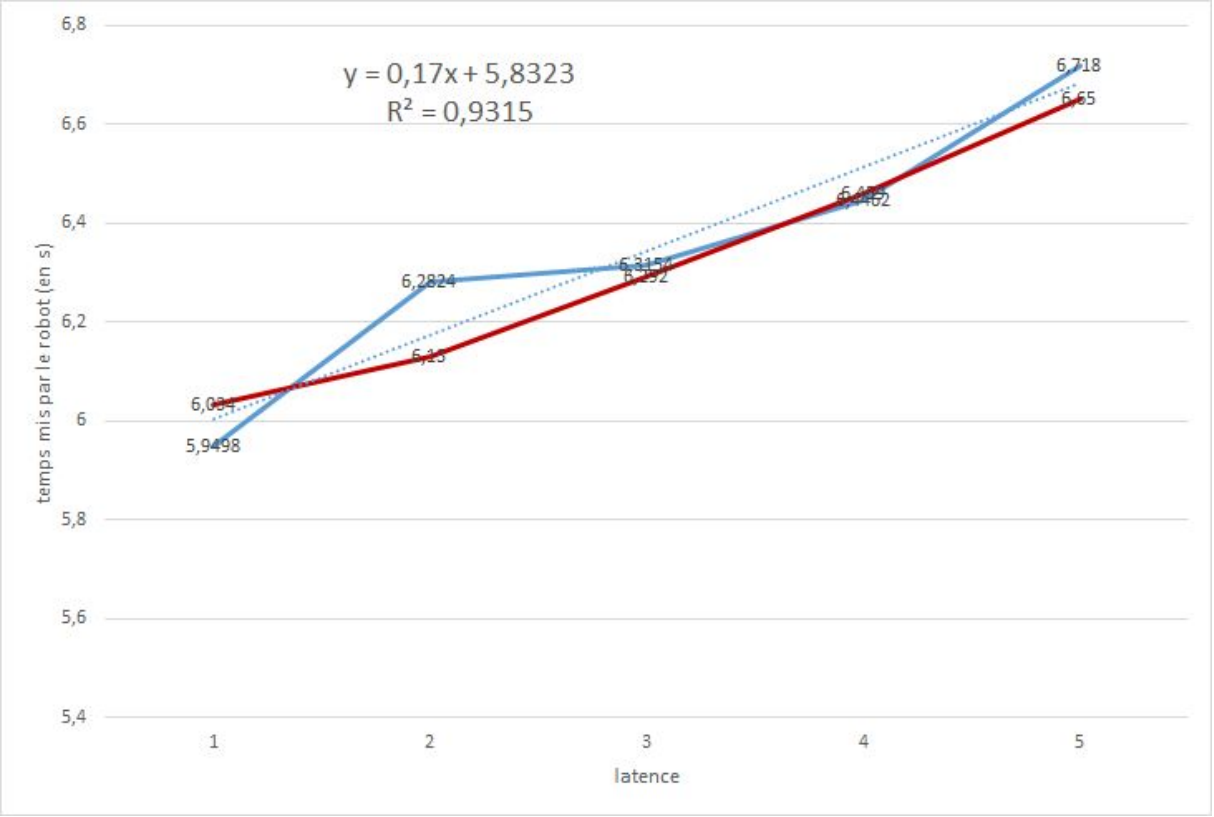
(Version 2)

Cette fois ci, les tests, au lieu d'être faits dans l'ordre croissants (0ms puis 250 puis 500 ...) les tests sont faits dans le désordre.

Latence	Sans latence	Avec 250ms	Avec 500ms	Avec 750ms	Avec 1000ms
Temps (en s)	5,984	6,298	6,236	6,459	6,741
	5,911	6,3	6,324	6,443	6,705
	5,931	6,233	6,367	6,495	6,772
	5,992	6,276	6,383	6,469	6,645
	5,931	6,305	6,267	6,365	6,727
Moyenne	5,9498	6,2824	6,3154	6,4462	6,718
Ordre de passage	1	4	3	2	5



On se rend compte ainsi que les temps de ceux réalisés en premiers (sans latence, 750ms et 500ms) sont anormalement bas alors que ceux réalisés en derniers sont anormalement élevés. Voici ci-dessous une superposition des courbes Version1 (en rouge) et Version2 (en bleu).



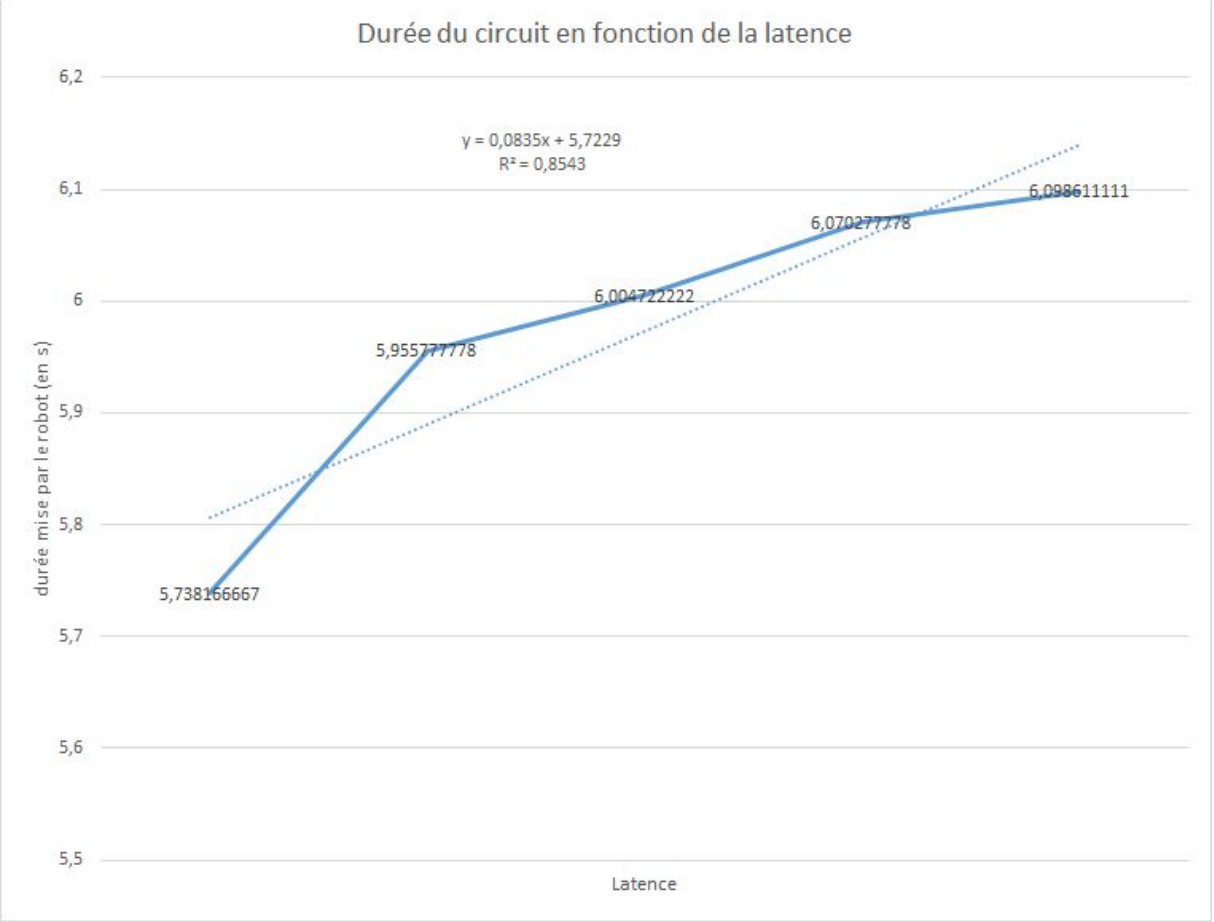
Par conséquent le niveau de batterie influence le temps mis par le robot. Donc le niveau de batterie devra rester constant pour les tests suivants.

(Version 3)

Pour cette série de tests nous avons accumulé 25 mesures pour chaque latence. Grâce à la magie des piles rechargeables, nous avons également réalisé les tests avec uniquement un niveau de batterie maximale. La case rouge correspond à une recharge. Les tests ont été réalisés dans l'ordre croissant.

Latence	Sans latence	Avec 250ms	Avec 500ms	Avec 750ms	Avec 1000ms
Temps (en s)	5,984	6,298	6,236	6,459	6,741
	5,911	6,3	6,324	6,443	6,705
	5,931	6,233	6,367	6,495	6,772
	5,992	6,276	6,383	6,469	6,645
	5,931	6,305	6,267	6,365	6,727
	6,034	6,13	6,292	6,459	6,65
	5,647	5,874	5,938	6,263	6,143
	5,764	6,042	6	6,156	6,054
	5,765	6,05	5,858	6,186	6,031
	5,791	5,991	6,105	6,136	6,136
	5,686	6,03	5,995	6,162	6,109
	5,664	5,837	5,981	6,147	6,191
	5,648	5,989	5,968	6,087	6,097
	5,726	5,93	5,952	6,072	5,916
	5,673	5,912	6,075	6,055	6,053
	5,665	5,887	6,106	5,93	6,035
	5,696	5,997	5,964	6,083	6,122
	5,804	5,913	5,909	6,029	6,041
	5,753	5,975	6,055	5,952	6,142
	5,847	5,938	6,022	5,867	6,061
	5,793	5,876	6,028	6,023	6,154
	5,72	6,033	6,097	6,057	6,18
	5,864	5,882	6,035	6,039	6,154
	5,781	6,048	5,997	6,021	6,156
Moyenne	5,73816667	5,95577778	6,00472222	6,07027778	6,09861111

Ordre de passage	1	2	3	4	5



Notre conséquence vérifiable semble être fausse (“ Nous nous attendons à ce que le temps mis par le robot augmente de manière proportionnelle par rapport au temps de latence programmé. ”). Cela ne ressemble pas à une fonction affine, mais plus à une fonction logarithmique… Alors, appliquons une courbe de tendance logarithmique :

On se rend compte qu'elle est beaucoup plus proche que si on applique une courbe de tendance affine. On se rend compte en effet que le R^2 est plus élevé, c'est à dire que le taux d'erreur est plus faible.

(Version 4)

Latence	0ms	500ms	1000ms
Temps (en s)	12,711	11,993	12,239
	12.455	11,85	11,415
		11,786	11,551.
		11,753	11.845
		11.993	11.455
		11.973	12.625
		11.815	11.339
		11.638	11.806
		11.629	12.119
		11.739	12.719
Moyenne de temps	12.583	11.79844	11.9203

Il s'agit des derniers tests effectués pour l'instant. On place le robot au centre d'un cercle d'1m50 de rayon et on cherche à savoir si la position de la bande verte sur ce cercle modifiera le temps du robots et surtout comment elle le fera. Ainsi on veut placer la fin du circuit à 45°, 90° et 0° du robot, à droite et à gauche. Dans un souci pratique, on modifie un peu le code du robot. En effet, au lieu qu'une seule roue du robot lui serve à tourner, on lui fait actionner ses deux roues dans le sens contraire. Il va donc rester sur place lorsqu'il va tourner, ce qui permettra de prédire plus facilement et précisément où il arrivera. Toutefois, les résultats étaient incohérents : on a obtenu des temps plus courts pour des temps de latence plus élevés. C'est pour cela que nous nous sommes arrêtés à 2 mesures pour 0ms.

V. Conclusion

Finalement, notre hypothèse de départ, c'est à dire que le temps mis par le robot pour faire le circuit augmente proportionnellement avec la distance, était fausse. Il y a plusieurs raison à cela :

- La vitesse que nous croyions constante ne l'est pas (à cause de la batterie notamment).
- La latence n'augmente pas proportionnellement avec la distance. Les virages ne se font effectivement pas à 90° donc lorsque le robot continue tout droit à cause de la latence il continue à se rapprocher de la ligne finale.

Puisque le temps du robot augmente en fonction de la latence selon une fonction logarithmique, un petit temps de latence entraîne rapidement une grande conséquence. On peut parler d'effet " boule de neige " .

VI. Les problèmes rencontrés

Nous avons rencontrés une grande multitude de problèmes :

- Comme dit précédemment lorsque le robot subit de la latence il se rapproche de l'arrivée, sa latence lui est "utile". Ainsi la position de l'arrivée joue beaucoup dans les mesures et peut être mauvaise.

- Le matériel lego en général a certaines limites :

  - Le détecteur de couleur n'est pas précis à 100% . Ainsi il arrive qu'il parvient à finaliser le circuit mais qu'à la fin il détecte du bleu à la place du vert et ne s'arrête pas.

  -Les moteurs ne peuvent pas être programmés pour faire tourner le robot à 90°, ainsi on perd forcément en précision et les trajectoires des robots peuvent être faussées.

  - Les robots sont très énergivores et nécessitent d'être chargés au maximum pour la précision des tests. Il fallait donc les recharger très souvent.

  -Le logiciel de programmation en bloc est peu clair et a nécessité du temps d'apprentissage.

  -On ne pouvait uniquement que téléverser et non récupérer le programme dans le robot ce qui est peu pratique pour les déplacements.

VII. L'avenir du projet

Nous avons encore plein d'idées, autres que le flappy bird pour le projet latence mais nous préférons pour l'instant travailler sur celui-ci.

Nous allons dorénavant nous concentrer sur des situations où deux robots, un avec latence et l'autre sans, feront le même circuit en même temps. Il s'agira d'un circuit simple, tel que des allers retours. De cette manière, le robot avec latence mettra beaucoup plus de temps. Ce format nous permettra de pouvoir aller le présenter et ainsi faire découvrir le projet latence.