Projet Slot Car Autonome

1- Introduction

Le but de ce projet est de mettre en place un algorithme de Q-Learning ainsi qu'une architecture basée sur des Raspberry Pi, afin de rendre une voiture slot-car Mario Kart autonome sur une piste. L'autonomie pourra être assurée en fonction de la tension du circuit, d'un retour caméra, ou d'une combinaison des deux.

Tout au long du projet, vous serez amenés à travailler avec des concepts variés et complémentaires tels que :

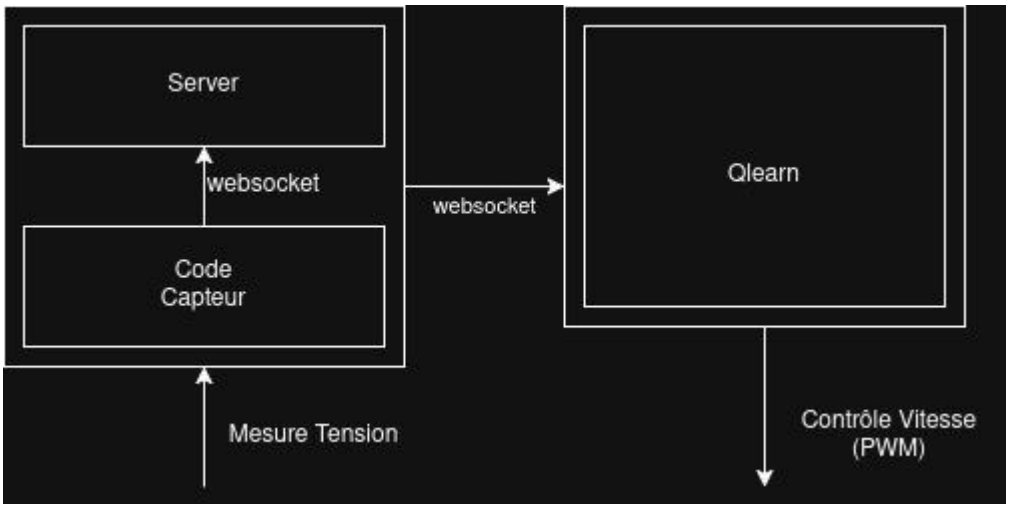
- **Électronique**
- **Q-Learning**
- **Réseaux** (architecture client-serveur, WebSocket)
- **IoT** (interfaces I2C, GPIOs)

L'architecture prévue (et fortement recommandée) est la suivante :

Deux Raspberry Pi sont utilisés :

- Le **premier** mesure la **tension du circuit** et l'envoie à un **serveur local** (hébergé sur le même Raspberry).
- Ce serveur relaie ensuite l'information au **deuxième Raspberry**, qui exécute l'algorithme de Q-Learning et contrôle la voiture en fonction des données reçues.

On a donc l’architecture suivante:



1.2 Le matériel

Voici une liste du matériel que vous utiliserez pour ce projet:

- 2 Raspberry Pi’s
- 1 Câble Ethernet
- 1 Transistor NPN (modèle **2N222**)
- 1 Analog to Digital Converter, (ADC Modèle **ADS1115**)
- 1 Capteur de tensión (de MH-Electronic)
  (Capteur de tension est un nom assez trompeur, pour ceux qui se connaissent en électronique c’est un diviseur de tension, le seul but de ce composant est de ne pas griller l’**ADC**)

- Des câbles pour tout brancher
-

Nous vous conseillons vivement de lire **(et surtout de comprendre)** les fiches techniques des différents composants, afin de savoir comment les brancher correctement et éviter de les endommager (le risque est faible, mais mieux vaut être prudent).

1.3 L’algorithme de Q-Learning

1.3-1 Le coeur de l’algorithme:

Le Q-Learning est un algorithme basé sur la recherche d'une politique optimale qui dit à un agent quoi faire dans un environnement pour maximiser une récompense à long terme.

L’idée principale est d’apprendre une **fonction de valeur d’action** appelée **Q-fonction** :

$$Q(s,a) \leftarrow Q(s,a) + \alpha[\, r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a)]$$

avec:

1. **s:** Etat actuel
2. **a:** Action choisie
3. **r :** Récompense reçue après l’action
4. **s’:** Nouvel état atteint
5. **α :** Taux d’apprentissage
6. **γ :** Facteur d’actualisation
7. **a’max Q(s', a') :** Meilleure Q-valeur possible dans l’état suivant

autrement dit:

Q(s, a) = valeur de l'action **a** lorsqu'on est dans l'état **s**

On a donc une **Q-table** qui est **une table (ou un tableau à deux dimensions)** qui mémorise les valeurs Q pour chaque couple (s,a)

1.3-2 Le problème d’exploitation/exploration:

Afin d’explorer toutes les possibilités il faut aussi implémenter un système d’exploration-exploitation:

On utilise souvent une stratégie **ε-greedy** :

- avec probabilité ε : on explore (action aléatoire),
- avec probabilité 1−ε : on exploite (action avec la plus haute Q-valeur).

1.3-3 Brouillon de l'algorithme:

De façon assez grossière on fini avec:

Initialiser Q(s, a) à 0 pour tous les états s et actions à

Pour chaque épisode (boucle principale) :
Initialiser l'état s

Tant que l'état s n'est pas terminal :
- Choisir une action a (ex: avec une stratégie ε-greedy)
- Exécuter l'action a
- Observer la récompense r et le nouvel état s'

- Mettre à jour Q(s, a) :
Q(s, a) ← Q(s, a) + α * [r + γ * max(Q(s', a')) - Q(s, a)]

- Mettre à jour l'état s ← s'

1.3-4 Les parametres:

Pendant ce projet, vous serez amenés à expérimenter avec différents paramètres dans le but d'optimiser le système.

En ajustant ces paramètres, vous pourrez adapter le comportement du système selon vos objectifs : par exemple, privilégier un apprentissage plus lent mais plus stable (sans sorties de piste), ou au contraire, accélérer l'apprentissage au prix d'un comportement plus risqué.

2- Organisation et Configuration

2.1- mise à jour des raspberries

Pour commencer, il est toujours préférable de mettre à jour les Raspberry Pi afin de disposer des dépôts logiciels les plus récents et pouvoir installer vos outils préférés.

1. **Téléchargez d'abord le Raspberry Pi Imager :**

   https://www.raspberrypi.com/software/

2. **Insérez la carte MicroSD** (avec ou sans adaptateur, selon ce que vous avez à disposition).

3. **Dans le Raspberry Pi Imager :**

   - Sélectionnez **Raspberry Pi 3** comme modèle.

    - Sélectionnez **Raspberry Pi OS 64 bits** comme système d'exploitation.

    - Sélectionnez la carte MicroSD.

4. **Écrivez l'OS sur la carte** et insérez-la ensuite dans votre Raspberry Pi.

5. **Plusieurs options s'offrent à vous**, mais la méthode la plus pratique consiste à brancher le Raspberry Pi à un écran via un câble HDMI et à y connecter un clavier. Vous pourrez alors procéder à la configuration, notamment pour la connexion au Wi-Fi.

6. Une fois la configuration terminée, lancez la commande suivante pour mettre à jour tous les paquets installés : ***sudo apt update && sudo apt upgrade -y***

2.2- Configuration des raspberries

Ensuite, il reste quelques éléments qu'il est impératif de configurer :

Lancez la commande suivante :

sudo raspi-config

Puis, effectuez les réglages suivants :

Options système → Hostname → Entrez un nouveau hostname.
(Conseil : Choisissez un nom unique parmi tous les Raspberry Pi du projet, et veillez à ce qu'il permette de différencier le Raspberry qui sert de capteur de celui qui est utilisé comme contrôleur.)

Options d'interface → SSH → Activer.
(Cela permet d'activer l'accès à distance via SSH pour contrôler votre Raspberry Pi à partir d'un autre ordinateur.)

Options d'interface → I2C → Activer.

(Cela permet d'activer le bus I2C pour la communication avec des capteurs ou d'autres composants électroniques.)

2.3- Acces au Raspberry

Il existe de nombreuses façons de s'organiser et de travailler avec les Raspberry Pi, donc si vous avez des préférences, n'hésitez pas à vous adapter et à explorer différentes solutions.

En règle générale, la connexion se fait via le protocole SSH (Secure Shell Protocol), qui vous permet d'accéder au terminal des Raspberry Pi et d'exécuter du code à distance.

La commande pour se connecter est la suivante :

ssh username@hostname.local

(Assurez-vous que votre ordinateur et les Raspberry Pi sont sur le même réseau.)

Si vous êtes à l'aise avec des éditeurs de texte comme nano ou vim, vous pouvez les installer et programmer directement via le terminal.

Alternativement, vous pouvez utiliser l'extension VSCode: Remote - SSH pour vous connecter à distance et modifier les fichiers directement depuis l'interface de VS Code.

3- Code et organisation du projet.

Nous vous recommandons de diviser le projet en plusieurs tâches plus simples afin de mieux comprendre les différents aspects du projet et d'éviter de commencer par une tâche trop complexe et générale.

La seule contrainte est d'utiliser Python et Flask-SocketIO pour mettre en place le serveur Flask et assurer la communication entre les Raspberry Pi.

Cependant, vous êtes libres de vous organiser comme bon vous semble.

Voici quelques étapes à suivre pour structurer votre travail :

1. **Mesurer la tension de la piste pendant que la voiture tourne** afin de comprendre la relation entre tension et vitesse.

2. **Mettre en place le circuit et le code** pour mesurer la tension avec le Raspberry Pi qui servira de capteur.

3. **Mettre en place le circuit et le code** pour contrôler la voiture avec le Raspberry Pi qui servira de contrôleur.

    - Il est utile de garder ces deux premiers codes simples et fonctionnels pour vérifier que le matériel fonctionne correctement à chaque séance ou en cas de problème.

4. **Mettre en place le serveur Flask avec des WebSockets** pour transmettre les données du capteur vers le contrôleur.

5. **Implémenter l'algorithme de Q-Learning** avec les données reçues pour contrôler la voiture, puis expérimenter avec différentes configurations et paramètres pour trouver un apprentissage efficace.

Bien sûr, vous êtes également invités à ajouter des **fonctionnalités bonus** à votre projet, par exemple :

- Créer une **page web** pour visualiser toutes les données en temps réel.

- Configurer les Raspberry Pi pour **communiquer via Ethernet**.
- Mettre en place un système pour **sauvegarder la Q-table** et tester rapidement différentes configurations.
- Expérimenter avec des **solutions alternatives au Q-Learning**.
- Faire en sorte de **réduire l'exploration** au fur et à mesure que l'algorithme apprend.