Balancement d’un plateau

Thomas H, Emilien K, Valentin C, Benjamin V.

IA & IoT and robotics - M2 IA - 09/01/2025
University Lyon 1 Claude Bernard

Abstract

Dans ce projet, l’objectif est d’équilibrer une balle sur une planche, capable de s’orienter verticalement, en utilisant de l’apprentissage par renforcement. Pour le réaliser, nous avons dû construire l’infrastructure physique du système. Mais aussi, mettre en place l’infrastructure logiciel comprenant des flux de données permettant l'entraînement du modèle.

1. Modélisation et design

1.1. Scénario d’utilisation

<u>Scénario n°1 :</u> Stabilisation de la balle sur un plateau

Étape 1. Le plateau démarre parallèlement au sol ;
Étape 2. Un utilisateur pose la balle sur le plateau et lui donne un coup pour la diriger vers un des bords de la planche dans le but de faire tomber la balle ;
Étape 3. Le système détecte la balle et utilise son moteur pour orienter la planche afin d’empêcher que la balle ne tombe ;
Étape 4. Le système stabilise la balle sur la planche au centre ;
Étape 5. Lorsque la balle est stabilisée au centre du plateau pendant un laps de temps, l’utilisateur peut recommencer en donnant un coup à la balle.

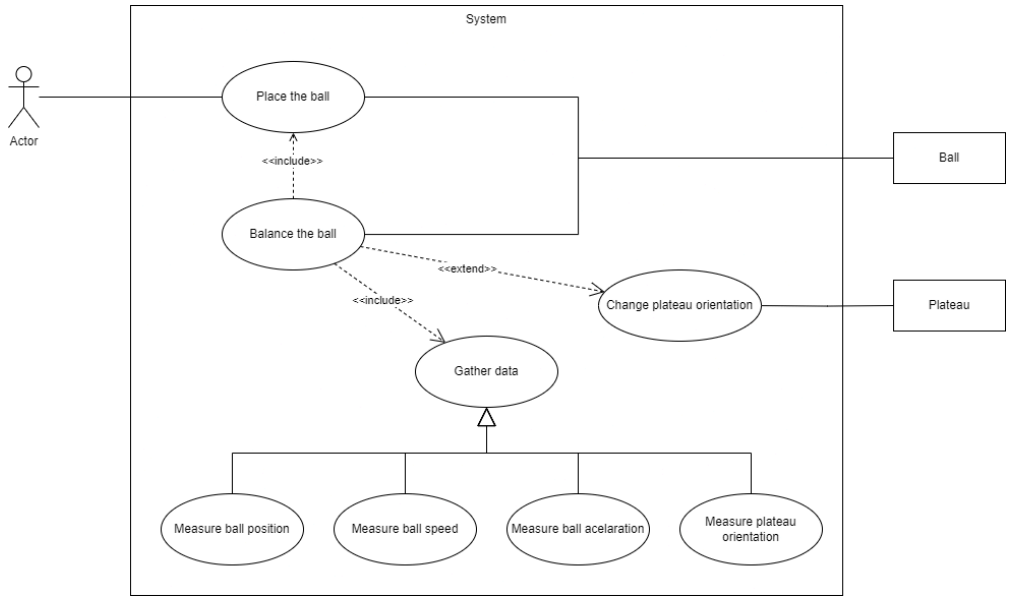

Figure 1 : Diagramme cas d’utilisation pour le scénario n°1 : Stabilisation de la balle sur un plateau

<u>Scénario n°2 :</u> Chute de la balle

Étape 1. Le plateau démarre parallèlement au sol avec une balle dessus ;
Étape 2. Un utilisateur pose la balle sur le plateau et lui donne un coup pour la diriger vers un des bords de la planche dans le but de faire tomber la balle ;
Étape 3. Le système détecte la balle et utilise son moteur pour orienter la planche afin d'empêcher que la balle ne tombe ;
Étape 4. Le système échoue à maintenir la balle sur la planche et celle-ci tombe sur un des deux côtés ;
Étape 5. L'utilisateur signale au système que la balle est tombée (En appuyant sur la touche entrée du clavier) ;
Étape 6. La moteur remet la planche parallèle au sol et l'utilisateur peut reposer la balle.

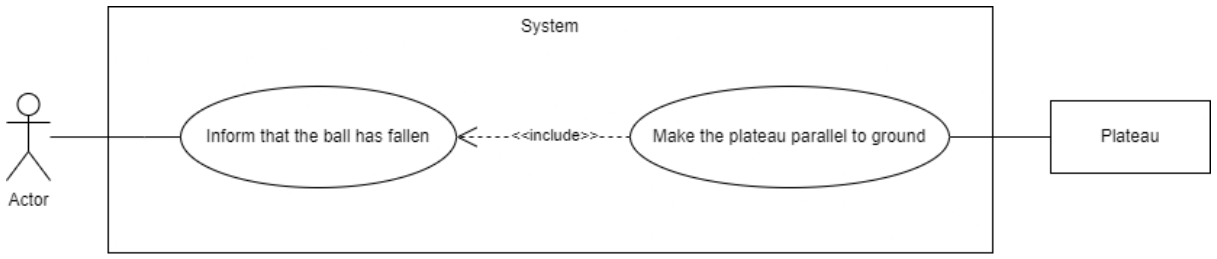

Figure 2 : Diagramme cas d'utilisation pour le scénario n°2 : Chute de la balle

Pour entraîner le modèle, nous répétons donc une série d'action. On initialise le système en mettant la planche parallèle au sol. On pose la balle et on la fait se déplacer sur la planche. Si la balle tient un certain temps sur la planche, l'épisode est réussi (le modèle est récompensé). Au contraire, si la balle tombe, l'épisode est interrompu (le modèle est puni).

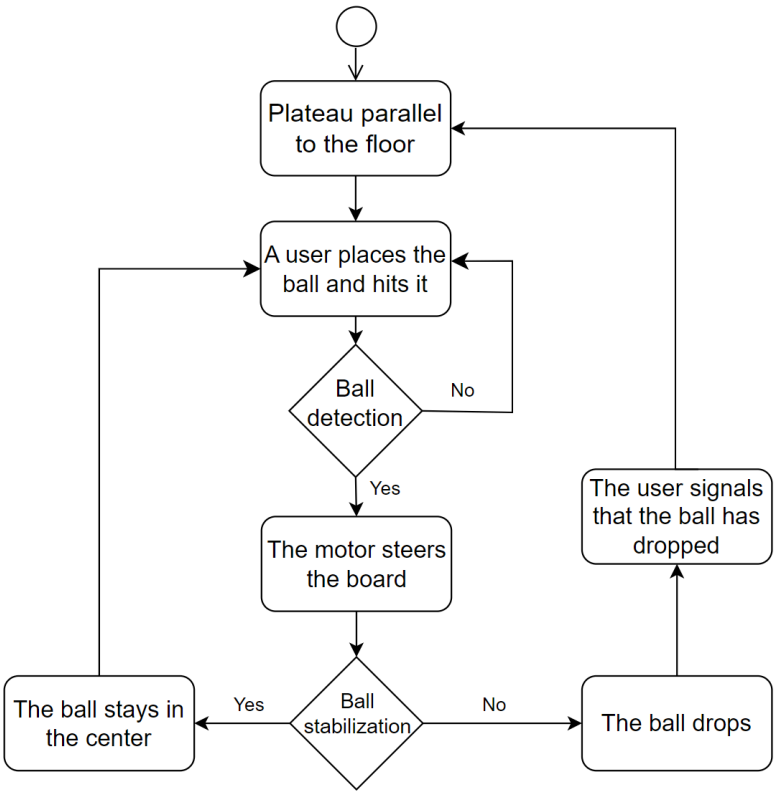

Figure 3 : Diagramme du pipeline d'entraînement

2. Architecture matérielle

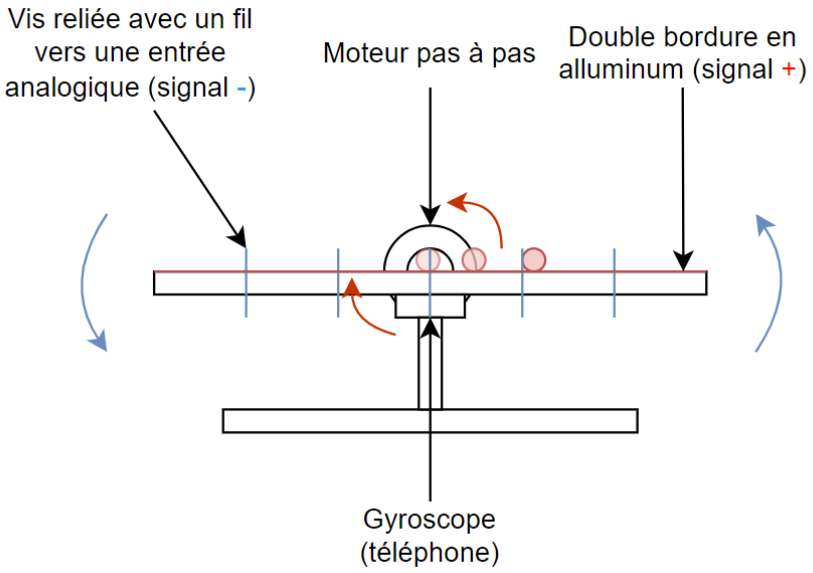

Figure 4 : Schéma de la structure construite

Notre structure est construite intégralement en bois avec une base parallèle au sol sur laquelle un plateau est posé. Des vis sont placées à intervalles réguliers le long de la longueur du plateau. Elles sont reliées avec un fil à l'aide de patafix et d'aluminium vers une entrée analogique. Des petits morceaux d'aluminium ont été collés à la surface de chacune d'entre elles pour élargir la surface de contact avec la balle.

On trouve également aux deux extrémités des longueurs du plateau des bordures construites à l'aide de pailles et d'aluminium pour transmettre le courant positif. Celles-ci ont été refaites pour que la balle passe en étant ralentie légèrement.

Nous avons aussi fixé un morceau de bois au plateau pour pouvoir poser le téléphone. Il permet d'utiliser une fonction de gyroscope qui sert aux calculs pour équilibrer la plateforme.

Nous avons démarré notre projet en choisissant d'utiliser un Arduino Due, car nous avons besoin de beaucoup d'entrées pour que nos 11 vis soient reliées à un signal négatif. Il était aussi nécessaire de relier un moteur pas à pas à l'Arduino. Celui-ci est connecté à une entrée digitale et placé sur le côté au milieu de la plateforme pour pouvoir l'orienter à gauche comme à droite.

Enfin, nous utilisons une balle en plastique que nous avons recouverte d'aluminium pour établir un courant entre les vis et les barrières. Après avoir essayé avec une balle classique, nous avons jugé qu'elle manquait de poids et qu'elle n'avait par conséquent pas la capacité de toucher les vis lorsqu'elle était en mouvement. Nous avons donc décidé d'en choisir une autre en mettant de l'eau à l'intérieur pour l'alourdir et résoudre ce problème. Pour plus de conductivité, nous mouillons la balle légèrement pour plus d'efficacité.

3. Architecture logicielle

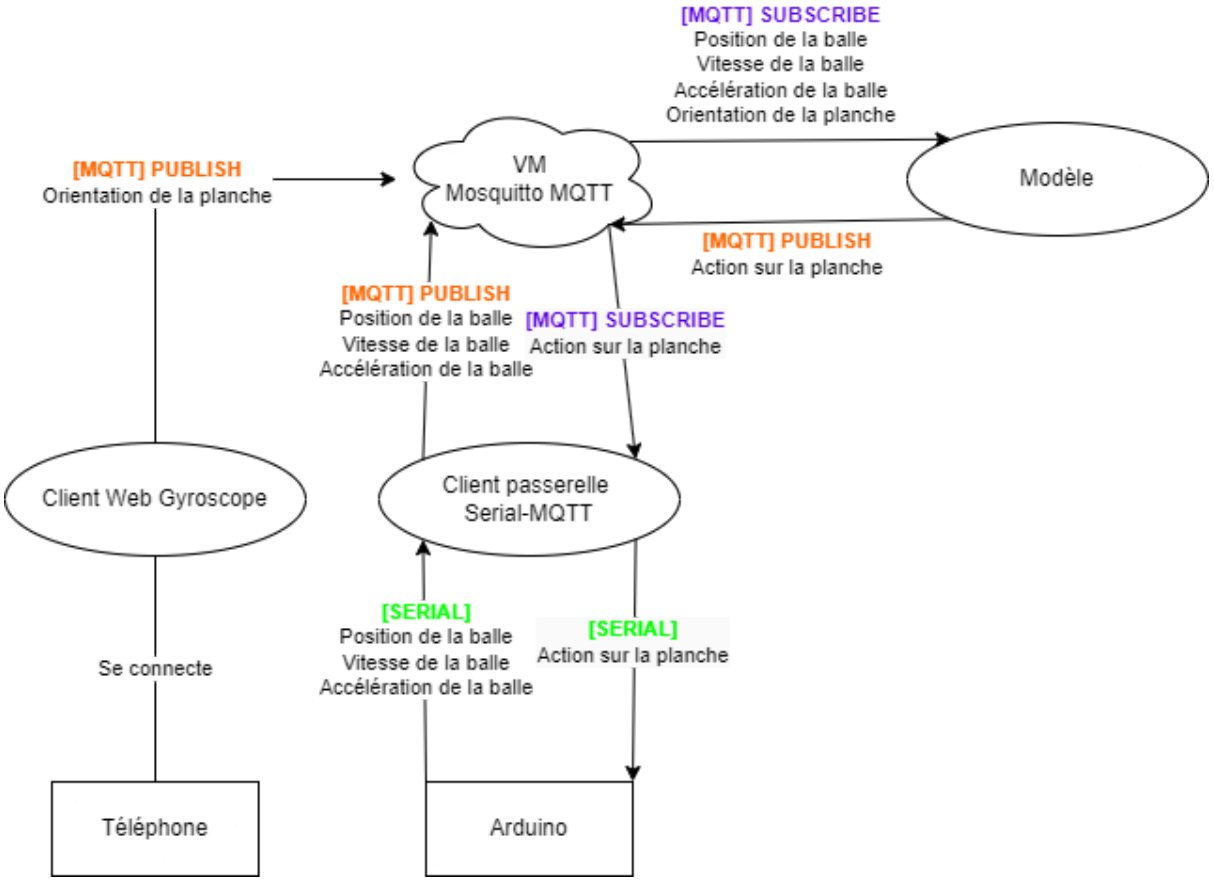

Figure 5 : Schéma architecture logicielle avec flux de données

3.1. Flux de données

A l'aide de notre système nous récupérons plusieurs données : l'Arduino fournit des données sur la balle tel que sa position, sa vitesse et son accélération. A l'aide du gyroscope du téléphone, on récupère l'orientation de la planche.

3.2. Protocole de communication : MQTT

Les différentes données que l'on obtient au travers des sensors (Gyroscope, Tension électrique avec les clous) seront communiquées via le protocole MQTT. Les données seront respectivement publiées sur les topics (*/gyro*, */position, /speed* et */acceleration*). Ensuite le modèle pourra les récupérer avec deux abonnements à ces topics (*/gyro*, */position, /speed* et */acceleration*).

Sur la VM, nous avons mis en place le broker MQTT Mosquitto qui sera donc le serveur qui recevra les informations des capteurs et qui les transmettra au modèle.

3.3. Capteur position de la balle

Nous récupérons au travers d'un script Python les données concernant la position de la balle reçue directement depuis l'Arduino via le port Serial. En se servant de la position de la balle, nous calculons aussi sur l'Arduino sa vitesse et son accélération. Les données sont ensuite publiées via MQTT sur les topics */position, /speed* et */acceleration*.

3.4. Gyroscope du téléphone

Pour récupérer l'orientation du téléphone, nous avons développé et déployé un client en VueJS simple qui fait un appel aux Sensor API (https://developer.mozilla.org/en-US/docs/Web/API/Sensor_APIs). Ce client est déployé et

accessible via le réseau de l'université, comme nous devons accéder au sensors de l'appareil, il a été nécessaire de rendre le site accessible en HTTPS.

On obtient trois valeurs correspondant à l'orientation du téléphone (alpha, beta, gamma) pour chacune des dimensions, comme la rotation ne se fait que dans un sens, on garde seulement la valeur *beta*. Les données sont ensuite transmises au serveur MQTT via le topic */gyro*.

3.5. Modèle d'apprentissage par renforcement

Avec toutes les données que nous récupérons sur les différents topics MQTT, nous avons développé plusieurs modèles d'IA à base d'apprentissage par renforcement pour donner un comportement intelligent au système.

Nous avons essayé plusieurs modèles avec différents algorithmes (cf. partie Intelligence Ajoutée) mais chacun des modèles prennent en entrée : la position de la balle, sa vitesse, son accélération et l'orientation de la planche et donnent en sortie un angle pour la planche.

La sortie du modèle est ensuite publiée sur le topic MQTT /action. Il est possible d'entraîner/utiliser le modèle aussi bien sur un PC que comme sur la VM. Pour des raisons pratiques, l'ensemble de l'entraînement a été réalisé sur le PC.

3.6. Contrôleur et script de l'Arduino

Un script est lancé sur le PC qui est connecté à l'Arduino afin de faire la passerelle entre l'Arduino et les données sur MQTT. Ce script permet de recevoir les données envoyées par l'Arduino sur le serial port et le transmet au restant du système via MQTT. Aussi, il reçoit les sorties du modèles via le topic */action* de MQTT qu'il transmet à l'Arduino via le Serial Port.

Le script de l'Arduino s'occupe de récupérer la position de la balle via les ports analogiques. On regarde pour chaque port analogique connecté la tension et si celle-ci dépasse un certain seuil, on considère que la balle se trouve à la position correspondante. En utilisant les données stockées (position de la balle précédente, moment où la balle a été détectée), on calcule aussi la vitesse et l'accélération de la balle. Toutes ces données sont transmises via le port Serial.

En plus de cela, lorsque l'Arduino reçoit des valeurs numériques via le Serial Port depuis le PC, il change l'orientation de la planche en fonction de la valeur.

4. Intelligence Ajoutée

Dans cette partie, nous allons décrire quelles méthodes nous avons employé et testé pour donner un comportement intelligent à notre système.

4.1. Algorithme Magicien d'Oz

Dans le but de tester notre système et sa robustesse, nous avons dans un premier temps développé un algorithme simple qui adaptait l'orientation de la planche en prenant juste la position de la balle.

4.2. Algorithme d'apprentissage par renforcement

Afin de rendre notre système plus intelligent, nous avons essayé divers algorithmes d'apprentissage par renforcement. Pour cela, nous avons utilisé la librairie Gymnasium sur Python qui nous a permis de formaliser notre système.

Dans un premier temps, nous avons utilisé Direct Search car c'est un algorithme simple où l'apprentissage est rapide. Cependant, celui-ci a produit de mauvais résultats avec le modèle qui finissait toujours par prédire la même valeur. Nous supposons que notre système est trop complexe pour cet algorithme. Nous avons donc par la suite implémenté deux autres approches à base des algorithmes Reinforce et DQN.

4.3. Fonction de récompense avec LLM

Nous avons dans ce projet besoin de créer une fonction de récompense. Ce qui est une tâche compliquée pour un humain. Dans le cadre du cours de LLM, nous avons développé VIRAL qui est un pipeline permettant de générer des fonctions de récompense avec les LLM. Nous avons donc choisi de réutiliser notre travail pour générer la fonction de récompense de notre environnement.

5. Bilan

Nous avons donc testé deux algorithmes d'apprentissage par renforcement : DirectSearch et Reinforce.

Nous avons obtenu de mauvais résultats avec DirectSearch car notre système est trop complexe pour l'algorithme, en effet, un simple produit matriciel ne peut pas permettre de prédire une valeur entre 60 et 120 (les bornes pour la rotation de la planche). Nous obtenons de meilleurs résultats avec Reinforce, cependant en raison de l'instabilité de l'environnement (base pas fixe, ni droite, conductivité et faible délai) nous aurons besoin de beaucoup de temps pour entraîner le modèle.

6. Répartition des tâches

Nous avons débuté le projet en nous séparant en 2 équipes. Une équipe "software", constituée d'Emilien et Benjamin et une équipe "hardware", composée de Valentin et Thomas.

Liste des tâches	Contenu	Affectation
Mise en place serveur	Déploiement site, mqtt + https, service d'entraînement	Emilien, Benjamin
Mise en place de Thingsboard	Maj docker de la config, mais pas garder dans la version final	Emilien, Benjamin
Implémentation MQTT	Topic, scripts, client	Emilien, Benjamin
Gestion gyroscope	Client téléphone + connexion mqtt	Emilien, Benjamin
Création structure	Découpe des planches de bois, assemblage de la structure, placement des vis et des rambardes	Valentin, Thomas
Connexions entre le code et la structure	Premier test	Valentin, Thomas, Emilien, Benjamin
Fichiers de configuration	Mosquitto, nginx	Emilien, Benjamin
Implémentation DirectSearch	/	Valentin, Benjamin, Emilien
Implémentation PPO	Pas gardé dans la version final	Valentin, Emilien
Algorithme Magicien d'Oz	/	Valentin, Thomas
Création de l'environnement	/	Valentin, Thomas,

Gymnasium	/	Emilien, Benjamin
Implémentation Reinforce & DQN	/	Valentin, Emilien
Réalisation de l'apprentissage	Placer la balle sur la planche et appuyer sur la clavier quand elle tombe	Valentin, Benjamin
Rédaction des rapports	/	Benjamin, Thomas