IUT nancy Charlemagne

Projet tutoré

Sujet : Robot joueur de billard

Etude préalable

Groupe : CAPAR - DESSAULX - LATH - RETTER
Tuteur : M. Pierre-André Guenego

Année scolaire : 2023-2024

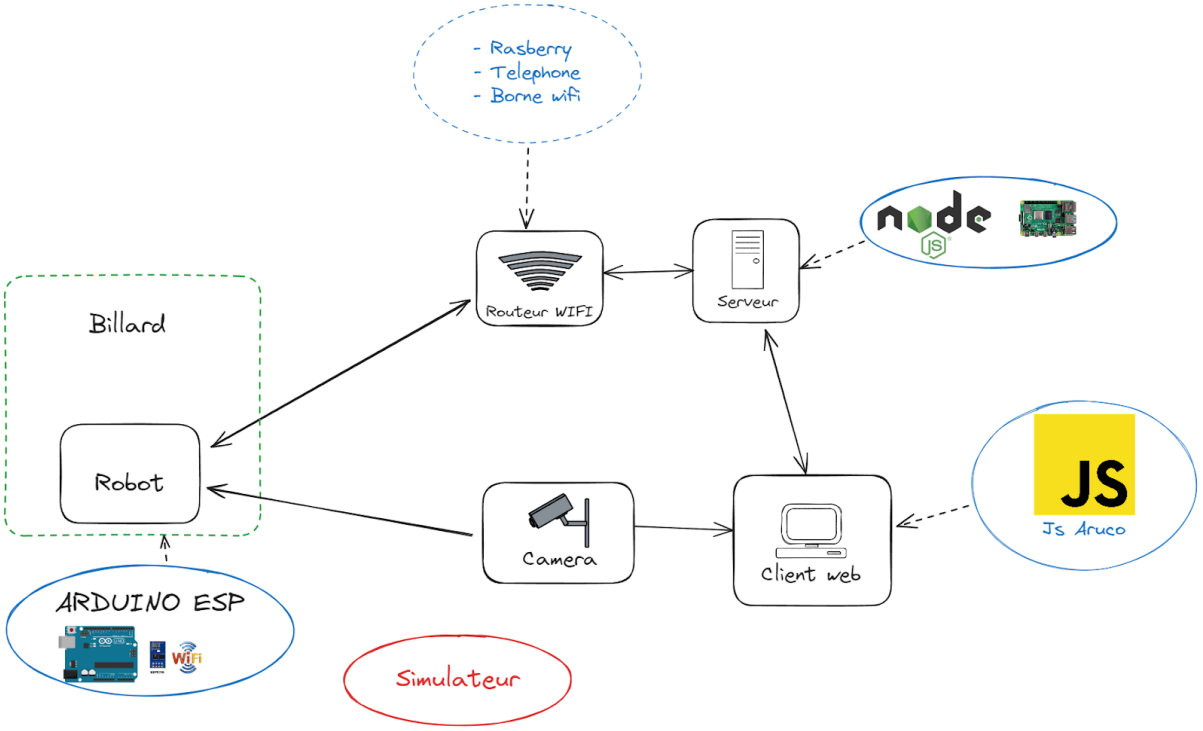
Sommaire

1. Description du sujet

Le projet consiste à développer un robot autonome qui joue au billard. Cependant, selon l'avancement du projet, d'autre objectifs et fonctionnalités pourront être définie et ajouter (cf. 6.Partie exploratoire).
Ce robot sera équipé d'un marqueur Aruco permettant d'obtenir la position du robot grâce à une caméra. Le robot sera guidé avec une caméra fixe au plafond et un serveur web. En effet, ce serveur web va gérer la communication entre le client et le robot.

2. Etude technique :

Architecture du projet :



- Le robot sera équipé d'une carte wifi (arduino esp) permettant de communiquer avec le serveur via un routeur wifi.
- Les images de la caméra seront envoyées vers le client pour ensuite être traitées par la librairie js aruco.
- Le serveur sera sous NodeJs.
- Le serveur, le client et le robot communiquent à l'aide des websockets pour maintenir un flux continu.

- Le simulateur est une partie indépendante mais importante qui vont permettre de tester les programmes avant de les mettre en place sur le robot.

NodeJS:

NodeJS est un environnement d'exécution JavaScript open source et multiplateforme qui se concentre sur les applications côté serveur et réseau.
Dans le cadre de notre projet, nous allons l'utiliser pour créer notre serveur. Ce serveur va recevoir les différentes informations du client web, effectuer tous les calculs et piloter les robots.

Arduino :

Un Arduino représente des cartes électroniques regroupant plusieurs composants électroniques afin de réaliser des objets électroniques interactifs.

Dans notre projet, tous les robots seront équipés d'un arduino esp8266. Nous allons utiliser ce modèle car celui-ci peut se connecter à un réseau WiFi. Ils seront connectés à la borne WiFi où est déployé le serveur NodeJS.

Routeur Wifi :

En tant que routeur Wifi, nous pouvons utiliser un simple téléphone. Ainsi, chaque partie du projet (le serveur NodeJS, le client web, arduino) sera connecté au même réseau WiFi. On pourra par la suite remplacer le téléphone par un Raspberry Pi.

Websocket :

Les websockets est une technologie qui permet d'ouvrir un canal de communication bidirectionnelle entre un client et un serveur. Dans notre projet, les websockets vont permettre des intéractions en temps réel et en flux continu entre le client et le serveur.

Js Aruco :

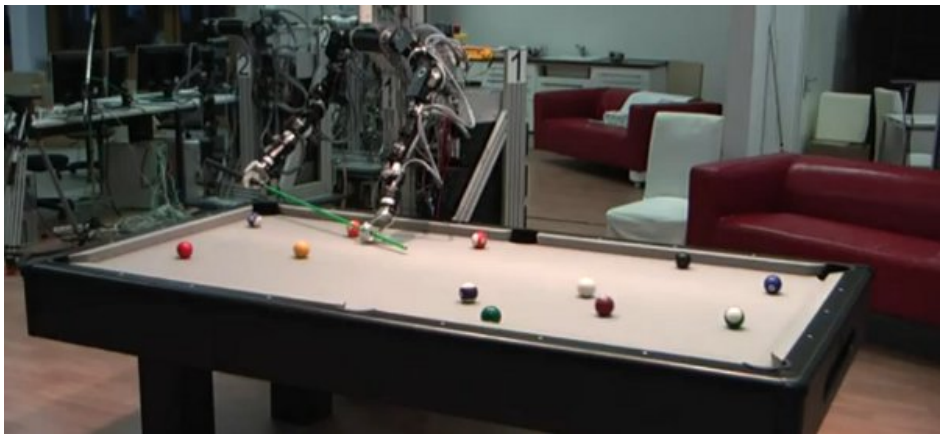
Librairie de réalité augmentée basée sur OpenCV utilisant des marqueurs. Ces marqueurs seront positionnés sur les robots. Cela permettra d'obtenir la position et l'orientation du robot dans la réalité.

JavaFX :

Framework Java permettant de créer des interfaces graphiques. Ce sera la technologie utilisée derrière le simulateur 2D.

3. Etude de l’existant :

1. Robot joueur de billard à deux bras :



Robot à deux bras conçu par Thomas Nierhoff

Un des premiers existants que nous avons trouvé est ce robot joueur à deux bras. Contrairement à notre sujet, ce robot possède une forme humanoïde mais comporte tout de même beaucoup de similitudes.

Il a été conçu par Thomas Nierhoff, lors de sa thèse à Technische Universität München (TUM). Ce robot est composé de deux bras disposant chacun de 7 degrés de liberté. Il est capable de jouer une partie de billard d’une manière autonome et totalement optimisé. Grâce au calcul, il est capable de jouer à la perfection et rate très rarement sa cible.

Une caméra haute définition est placée sur le dessus de la table de billard, qui permet d’analyser d’une manière précise la position des boules afin de calculer le coup optimum à réaliser.

Il peut se déplacer autour de la table, chercher et frapper la boule d’une manière précise tout en tenant compte du pourcentage de réussite.

Dans ce projet, il est aussi possible de projeter sur la table, le coup optimum afin de le réaliser d’une manière humaine.

Évidemment c’est un projet bien plus poussé que le nôtre mais il y a des similarités comme par exemple l’utilisation d’une caméra haute qualité pour avoir la position des boules.

2. Robot Sumo :

Photo d’un robot sumo

Un autre type de projet qui ressemble beaucoup au nôtre sont les robots sumo.
Ce sont des robots qui sont conçus pour participer à des compétitions de sumo robotique. Ces compétitions impliquent généralement 2 robots autonomes qui s’affrontent dans un petit ring. L’objectif est de pousser l’adversaire hors du ring.

Les robots sont équipés de différents types de capteurs comme des capteurs infrarouges ou des capteurs ultrasons pour mesurer la distance entre l’adversaire mais aussi la position du robot par rapport au ring.
Avec ces différentes données, le robot calcul la meilleure trajectoire à prendre pour pousser son adversaire hors du cercle mais de faire attention à ne pas sortir du ring lui-même.

Le robot intègre le programme, contrairement à notre projet où c’est le serveur qui fait les différents calculs et envoie l’ordre au robot. Contrairement au robot sumo, nos robots sont aveugles.

3. **Simulateur pour robot sumo :**

Aperçu du simulateur 2D

Le simulateur qu’on a trouvé est un simulateur utilisé pour entraîner le programme des robots sumo. L’auteur l’a rendu public sur github et explique sur son blog son utilisation.

Ce simulateur vise à tester les robots dans un environnement virtuel à deux dimensions. Il a été développé en C++.

Le simulateur prend en compte les principes physiques, ce qui offre un environnement réaliste pour tester les mouvements. L'auteur utilise des librairies open sources mais le développe lui-même pour que la physique soit le plus fidèle possible à son robot.

Contrairement à d'autres simulateurs comme Unity par exemple, ce simulateur est selon l'auteur plus simple d'utilisation.

Ce simulateur est conçu pour tester des robots autonomes qui ont leur propre capteur.

4. Principales difficultés du sujet :

Modélisation des Déplacements :
Définir avec précision les mouvements du robot en tenant compte de la cinétique et de la dynamique. En effet, il est difficile de programmer par nous-mêmes et de zéro les mouvements physiques d'un robot en javafx. Il nous faut définir une échelle réaliste du plan de jeu pour y placer un robot de la même taille qu'un vrai robot de même pour la boule. Ensuite, nous pourrons définir les mouvements de ce robot.

Interaction avec les Éléments Physiques :
Programmer les réponses du robot à son environnement, comme la détection d'obstacles, en utilisant des principes physiques. Cela sera fait avec la caméra et js-Aruco. Ici la difficulté est de bien initialiser le client pour qu'il puisse obtenir les bonnes positions des différents objets pour pouvoir les envoyer au serveur pour les calculs. De plus, dans le but d'un robot joueur de billard, il faudra analyser l'interaction entre le robot et la boule.

Problème de Latence :
Gérer les défis liés à la latence dans la communication entre le serveur et l'Arduino pour assurer des réponses en temps réel. Mais aussi entre la caméra, le client et le serveur. Il est connu que la connexion wifi est souvent instable et que de nombreux éléments peuvent interférer avec celle-ci. Ainsi, si le robot a de la latence avec le serveur cela pourrait créer des problèmes. Par exemple, le robot pourrait recevoir l'ordre d'avancer mais qu'il ne reçoive jamais l'ordre de s'arrêter. De même avec la caméra, le client et le serveur car si le client envoie des positions qui ne sont plus bonnes/effectives alors le serveur fera de mauvais calculs et le résultat ne sera pas au rendez-vous.

Différence entre programmation et réalité :
Les ordres effectués au robot peuvent être différents de la réalité. Tout cela est relié aux problèmes de latence par exemple, mais aussi par rapport au simulateur. On sait que beaucoup d'éléments extérieurs peuvent changer l'attendu du résultat. Dans les calculs l'effet de friction ne sera pas pris en compte alors il se pourrait que le robot n'aille pas à la vitesse voulu par exemple. Aussi, il suffit d'un petit caillou sur l'espace de jeu du robot pour le faire changer de trajectoire inopinément. Donc de nombreux détails peuvent

complètement changer les mouvements du robots et ils ne seront pas forcément prévisible ou pris en compte dans le projet.

5. Objectifs à atteindre

Le but principal du projet est de faire un robot qui puisse se déplacer et toucher une boule grâce à un système intermédiaire. Cela est bien sûr vu comme l'objectif final, mais l'attendu est différent. Il nous est possible de changer de technologies ou de changer une partie du projet en cours de route en fonction de notre avancement et de nos exigences. C'est un projet à but expérimental sans un seul chemin possible.

Pour pouvoir accomplir cette objectif final nous avons déterminé des objectifs intermédiaires tels que :

- Faire fonctionner le robot
- Réussir à connecter le robot au serveur
- Développer le client js pour visualiser l'espace de jeu avec une caméra
- Initialiser le serveur
- Développer le simulateur
- Intégrer les calculs de déplacement du robot dans le serveur

6. Partie exploratoire

Lorsque nous aurons atteint l'objectif final, il nous est possible de continuer le projet vers une version plus poussée tel que des robots joueur de football si on le veut. Nous pourrons aussi partir vers un projet un peu différent comme un projet de robots sumo ou robots joueurs du loup par exemple.
Aussi, si dans le déroulement de notre projet il nous est difficile de faire quelque chose ou si l' on préfère partir vers une autre direction alors on pourra un peu changer le sujet. Ce point met en valeur le côté exploration du sujet.

Sources :

- Robot joueur à deux bras : https://www.semageek.com/un-robot-a-deux-bras-capable-de-jouer-au-billard-a-la-perfection/
- Simulateur 2D : https://www.artfulbytes.com/bots2d-blogpost
- Robot sumo : https://fr.wikipedia.org/wiki/Robot_sumo
- Robot sumo arduino : https://projecthub.arduino.cc/AhmedAzouz/how-to-make-arduino-sumo-robot-afb0d8
- Information NodeJS : https://fr.wikipedia.org/wiki/Node.js
- Serveur NodeJS Arduino : https://www.locoduino.org/spip.php?article216
- Js-Aruco exemple : https://www.robot-maker.com/forum/topic/13723-comment-detecter-des-codes-aruco-avec-vigibot/