Mario Kart MINDSTORM : un jeu de course de karts modulaires

Jeremy AUGEREAU, Youssef BENDALI, Mathis CAISSON, Fatine CHEDDAD

Encadrants : Marie BABEL
François PASTEAU

2021-2022

Résumé

Mario Live Home Circuit est un jeu en réalité augmentée où il est possible de contrôler un kart dans la vraie vie à l'aide d'une console nintendo switch. Dans ce projet nous allons tenter de reproduire ce jeu de courses de karts physiques le tout à l'aide de legos Mindstorm construits sous la forme de karts contrôlables par téléphone.

1 Introduction

1.1 Contexte du projet

Nous réalisons ce projet dans le cadre de notre enseignement de 3ème année à l'INSA, notre but est donc, sur une année, de faire une étude pratique sur un sujet donné. Tout au long de cette étude pratique nous avons ainsi tenté de réaliser un jeu inspiré de Mario Kart Live Home Circuit à l'aide du matériel mis à disposition et en prenant en compte les différentes contraintes imposées. Ce sujet a été présenté aux étudiants cette année pour la première fois, nous devons donc le réaliser de zéro.

1.2 Objectifs et contraintes du projet

L'objectif primaire de ce projet est de réutiliser d'anciens legos mindstorm [1] qui se trouvaient dans les placards du département informatique au moins depuis les années 2000. Notre tâche de construire un modèle mécanique de kart viable. C'est-à-dire un kart assez fluide et rapide pour pouvoir effectuer une course. De plus, pour réaliser une course en multijoueur, il faut au minimum 2 karts . L'épreuve était donc de pouvoir construire deux bases de kart nous permettant de tourner et d'avancer grâce à nos moteurs.

Ensuite, nous devions travailler sur notre carte de contrôle : le modèle M5Stack core qui nous était imposé [2]. Nous avions également besoin d'établir une liaison entre notre carte et un client . C'est grâce à cette liaison que l'utilisateur pourra contrôler le kart. L'objectif était donc d'obtenir une connexion entre le joueur et le kart afin

1. Les Lego Mindstorms sont un projet du groupe Lego basé sur les briques « RCX » (Robotics Command System), puis « NXT », et plus récemment « EV3 »

2. Le M5Stack est un système de modules composé d'ordinateurs à cartes uniques à destination de l'Internet des objets, basé sur le microcontrôleur d'architecture ESP32-S

de pouvoir jouer aisément. Il nous fallait ainsi gérer les actions du joueur et récolter les différentes informations à travers les capteurs connectés au M5Stack .

Enfin, une fois cela en place, nous devions affiner notre projet afin d'y ajouter les règles de base du jeu Mario Kart. C'est-à-dire les différents points d'étape formant un circuit fermé, le classement des joueurs, ou encore y implémenter différents objets bonus du jeu comme par exemple la carapace que l'on pourra lancer virtuellement sur un autre joueur. Le but à la fin étant d'avoir une base de course de karts, en ajoutant des premières étapes d'interactivité .

1.3 Plan de réalisation

Nous avons, à partir des contraintes imposées, du cahier des charges réalisé et du matériel à disposition, mis en place une première esquisse des choix techniques que nous avions à prendre. En effet, il a fallu effectuer ces choix techniques au niveau de différents aspects :

1. La réalisation mécanique du kart en lego ;
2. La programmation du M5STACK pour contrôler le kart ;
3. La technologie de communication et les protocoles associés ;
4. L'interface client et la gestion des différentes interactions.

C'est autour de ces 5 principaux points que nous allons concevoir notre architecture logicielle.

2 Realisation technique du projet

2.1 Réalisation mécanique du kart

Nous étions assez limités au niveau du matériel, par conséquent il fallait concevoir un kart fonctionnel et fluide avec le peu de pièces à disposition. Nous nous sommes donc penchés sur un modèle de kart unicycle comme l'illustre la figure 1 . Malgré le fait que ce modèle ne respecte pas la définition propre d'un kart, nous avons préféré partir sur celui-ci pour plusieurs raisons. Tout d'abord, c'est un modèle assez facile à construire et demandant peu de variété de pièces. Il ne nécessite pas de pièces différentielles pour les moteurs contrairement à d'autres modèles plus conventionnels. Notre système se base sur 2 roues motrices situées à l'avant permettant de tracter le kart assez rapidement. Malgré les faibles couples de nos moteurs, ceux-ci sont largement puissants pour notre utilisation. De plus, nous utilisons une roue folle à l'arrière pour stabiliser le kart comme le montre la figure 2 : nous avons utilisé un modèle unicycle et non bicycle [3]. Enfin pour pouvoir faire tourner le kart nous avons dû gérer le rapport de vitesse des deux roues du côté logiciel. Nous avons réalisé un algorithme permettant de calculer la vitesse des deux moteurs en fonction de l'angle de direction du kart, cela nous a permis notamment de facilement respecter une autre contrainte qui était de faire tourner le kart sur lui même à chaque fois qu'il se prenait une carapace d'un autre joueur. Ainsi, avec le modèle unicycle nous avons un kart qui peut avancer, reculer, tourner et le tout de manière très fluide et précise.

3. [Comparaison modèle de voiture unicyle bicycle]

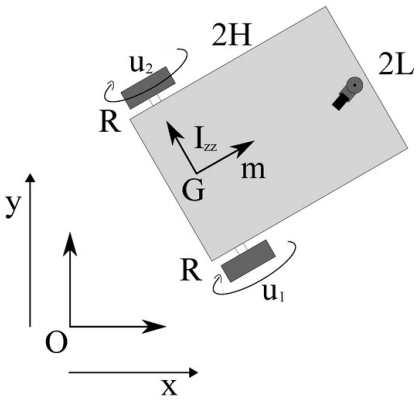

FIGURE 1 – Modèle de voiture unicycle

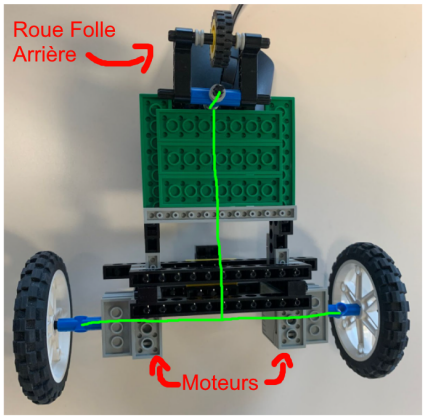

FIGURE 2 – Modèle construit de voiture unicycle

Ainsi, nous avons réalisé un modèle de kart utilisable pour notre projet . Il reste totalement modifiable grâce à sa structure lego. L'utilisateur est assez libre, et le fait d'utiliser des legos permet au joueur de modifier sa voiture comme il l'entend avec d'autres pièces legos apposées sur la base fournie.

Néanmoins, maintenant que notre structure lego est posée et que les moteurs sont placés, il nous faut réaliser la programmation et les différentes fonctions de la carte de contrôle M5Stack. Le M5Stack va en effet être un élément crucial pour permettre à l'utilisateur de contrôler le kart à distance.

2.2 La programmation du M5STACK : contrôle des moteurs et capteurs

Dans le but de piloter notre kart, il a fallu mettre en place une carte de contrôle au niveau de celui-ci. Nous avons utilisé un M5stack étant donné que cela nous a été imposé en contrainte. C'est une petite carte informatique composée d'un processeur, de mémoire vive et de mémoire interne. L'avantage de l'utilisation de ce module réside dans sa facilité d'adaptation à la captation d'information. En effet, cette carte est composée de multiples capteurs avec la possibilité d'y ajouter des modules pour réaliser les différentes tâches de contrôle qui nous intéressent. Dans le cadre de notre travail nous nous sommes servi principalement d'un module essentiel pour notre kart : le GOPLUS2 [4]. Ce module est une interface de contrôle de moteur, nous permettant de contrôler deux moteurs DC et des servomoteurs. Ce module sera utile afin de piloter les deux moteurs de nos roues avant indépendamment et ce, grâce au M5stack. Néanmoins, nous travaillons avec de vieux legos qui ne sont plus sur le marché, par conséquent les connecteurs des moteurs sont eux aussi un peu vieillissants. Pour pallier cela , nous avons dû nous mettre à l'exercice de la soudure de câbles. Ceci nous a permis de créer nous même notre connecteur DC vers les moteurs lego de 1998.

Nous avons maintenant notre carte de contrôle composée de tout ce qu'il faut. Á cela nous avons ajouté les divers modules et capteurs nécessaires à la réalisation de notre projet. Nous avons produit tous les câbles qu'il nous fallait. Il ne nous reste plus qu'à effectuer la programmation et relier logiquement tous nos composants. Pour cela, nous utiliserons l'IDE fourni par Arduino et programmerons en langage

4. Le M5Stack GoPlus2 est un moteur multifonctionnel empilable et un module de commande servo.

Arduino, un langage similaire au C++ fait pour les cartes de contrôle comme notre M5Stack.

Du côté programmation : pour nous aider dans la programmation logique de notre kart, nous avons utilisé beaucoup de librairies C++ disponibles sur internet. Nous avons donc pu travailler plus rapidement , avec notamment des librairies pour le capteur RFID et le GOPLUS 2. Pour ce dernier, cela nous a permis d'actionner les moteurs à une certaine vitesse sans avoir à changer la valeur de sortie des pins des moteurs directement dans la mémoire.

En ce qui concerne la partie structure de code nous avions dans un premier temps une fonction de setup, nous permettant de démarrer et d'instancier nos différents composants . Ensuite nous avons utilisé des fonctions extérieures nous permettant de réaliser diverses opérations logiques et d'ajouter certaines parties de calcul indépendantes. Enfin nous avions une boucle dans notre programme appelant périodiquement nos fonctions logiques permettant d'abord de récupérer les différentes valeurs des capteurs de la carte et dans un deuxième temps , à l'aide de prédicats, de pouvoir déterminer les mouvements de notre kart et les actions à effectuer.

2.3 La technologie de communication et les protocoles associés

Notre construction de kart est posée, nous savons comment programmer notre carte de contrôle, récupérer les informations des capteurs et envoyer des instructions à nos moteurs. Toute la base de fonctionnement d'une voiture mobile y est construite, néanmoins nous aimerions bien pouvoir contrôler notre kart à distance. Pour cela il va nous falloir trouver un moyen d'interagir avec notre M5Stack via le protocole de communication de notre choix.

2.3.1 technologie de communication

Nous avons décidé d'utiliser du wifi pour le M5stack de sorte que celui-ci ait accès à un réseau de machine et puisse interagir avec, chose qui aurait été bien plus compliquée à mettre en place si l'on avait opté pour une autre solution de communication comme le bluetooth. Notre première idée était alors de raisonner sur un système de leader et suiveur : un M5Stack aurait émis, en point d'accès, son propre réseau wifi et tous les autres se seraient connectés, un leader pour plusieurs suiveurs. Ce système nous aurait permis d'avoir très peu de contraintes matérielles avec un réseau constitué uniquement à l'aide des M5Stacks. Néanmoins, celui-ci posait plusieurs problèmes. Premièrement : certes tous les M5stacks auraient été en réseau, mais nous n'aurions pas eu accès à internet. Or, nous avons besoin d'avoir accès à certains serveurs externes, notamment des serveurs NTP [5] pour la synchronisation des horloges. Deuxièmement, le problème majeur que nous avons est que les M5Stacks sont des appareils très petits, qui possèdent de petites antennes wifi et qui ne nous permettaient pas d'avoir une qualité de connexion optimale pour avoir le minimum de latence au niveau de nos paquets. Plus on ajoute de karts, plus le réseau devient lent et inutilisable de par le système de leader et suiveur qui redirige tous les paquets vers un seul M5Stack. C'est pourquoi, nous avons opté pour une deuxième option : laisser les utilisateurs connecter les différents karts à leur réseau wifi le tout grâce à un portail captif. La librairie WifiManager [6] nous a permis d'implémenter ce portail plus facilement.

5. [Documentation et exemples NTPClient]
6. [WiFiManager librairy]

Nous nous sommes servis d'une bibliothèque déjà existante permettant de réaliser un portail de connexion pour le M5Stack sur un réseau domestique. Cela nous permet ainsi de résoudre le problème de latence, le routeur externe possédant une quantité de calcul bien suffisante, mais aussi le problème de connexion à internet. Le point négatif de tout cela est l'ajout de la contrainte de devoir posséder un réseau wifi domestique ou un téléphone en point d'accès pour pouvoir contrôler les karts. Ainsi, pour pouvoir paramétrer un kart chez vous, vous aurez simplement à vous connecter au point d'accès produit par le M5stack de SSID le nom du kart. À partir de là, il ne vous reste plus qu'à sélectionner votre wifi et vous serez automatiquement redirigés vers la page de connexion, il faudra entrer le mot de passe et le M5stack va alors redémarrer et conserver les informations de connexion pour pouvoir se connecter à votre réseau domestique automatiquement la prochaine fois. À partir de là, il suffira de vous reconnecter à votre wifi et de scanner le QRcode présent sur l'écran du kart à l'aide d'un téléphone pour ouvrir la page de contrôle du kart.

2.3.2 protocoles de communication réseau

Le broadcast UDP : Maintenant que tous nos karts sont dans un même réseau constitué d'internet, il nous faut pouvoir repérer notre kart parmi les autres. Pour cela nous utiliserons un système de protocole maison basé sur du broadcast UDP [7] : imaginons qu'un kart se connecte sur le réseau, celui ci va alors envoyer un paquet au niveau de l'adresse de broadcast, c'est à dire à toutes les adresses du réseau local avec pour message "Are you here mario ?". Si un autre kart est dans le réseau, il va alors recevoir le paquet et ajouter l'IP de l'émetteur du paquet reçu au niveau de sa liste d'adresses IP de karts, on signifie donc qu'un nouveau kart est entré en jeu et on lui renvoie "It's me Mario !". L'émetteur va alors recevoir le message de retour et va ajouter l'IP du kart qui lui a répondu. Par conséquent, que ce soit les karts déjà présents ou bien celui qui se connecte au réseau, ils constituent une même liste d'adresses IP correspondant à la liste des IPs des karts connectés au réseau local.

Le websocket asynchrone : Afin qu'un joueur puisse communiquer avec son kart, nous avons besoin de faire passer continuellement des informations entre 2 IPs que sont notre kart et le téléphone de contrôle. Ainsi, pour obtenir une communication bilatérale fiable, nous avons utilisé le principe de websocket asynchrone [8]. En effet, notre kart va agir comme un serveur web et le téléphone de l'utilisateur comme un client.

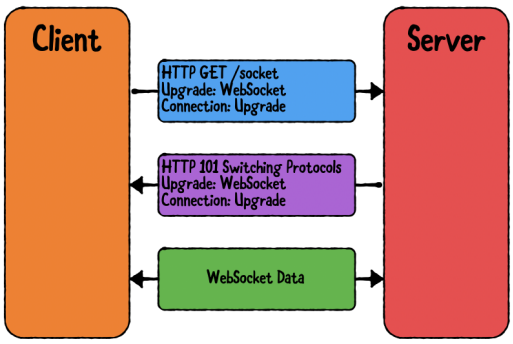

FIGURE 3 – schéma de communication WEBSocket

7. [ESPAsyncUDP librairy]
8. [ESPAsyncWebServer librairy]

l'utilisateur va se connecter à l'aide de son téléphone et d'un navigateur sur une url précise : **http ://IPKart/game/**.Le kart va envoyer à l'utilisateur la page web stockée dans la mémoire du kart. Cette page web possède une partie logique en JavaScript qui va s'occuper d'établir une connexion TCP stable entre notre téléphone et notre kart en établissant un lien de connection : le WebSocket dont le protocole y est décrit en Lorsque le kart récolte une nouvelle information, comme un nouvel objet obtenu ou un point d'étape passé, il peut alors notifier le client du changement grâce au canal TCP déjà ouvert. Le joueur peut donc envoyer des informations quand il le souhaite au kart et inversement. Enfin lorsque l'un des deux bouts interrompt la connexion, on ferme le socket TCP et donc notre WebSocket. figure 3

L'UDP fiabilisé : Il faut avoir une communication unicast entre deux karts particuliers. Par exemple, si on veut envoyer une carapace à l'adversaire , nous devons être capable de communiquer entre deux karts distincts. Nous devions au départ connecter en websocket chaque kart les uns aux autres pour pouvoir envoyer des messages d'un kart à un autre, néanmoins nous nous sommes rendus compte que les communications étaient très lentes et coûteuses en mémoire. C'est pour cela que nous avons décidé de partir sur de l'UDP unicast pour envoyer nos messages. Cependant, ce protocole n'est pas assez fiable et si un paquet est perdu , on se retrouve avec un message qui n'atteindra jamais sa cible. Pour remédier à cela nous avons, à la main, fiabilisé le protocole UDP grâce à une structure de message comportant un numéro d'acknowledgment et le moment d'envoie du message. La figure 4 montre les transitions des différents paquets entre 2 Karts.

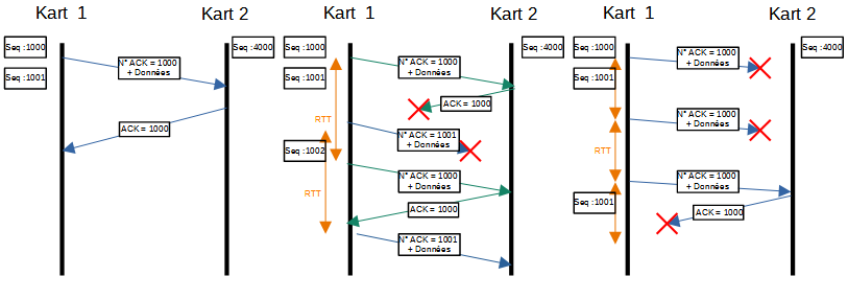

FIGURE 4 – Protocole de communication entre deux Karts

Tout cela nous permet de mettre en place certaines étapes de fiabilisation de TCP adaptées pour le protocole UDP tout en n'ayant aucune socket à ouvrir ou fermer à l'envoi de messages. Le fonctionnement du protocole se résume donc de la sorte : on envoie un message, on stocke le moment d'envoie, et si après un certain temps on ne reçoit pas de réponse, on renvoie le paquet et ainsi de suite jusqu'à 5 fois maximum. Si tout s'est bien passé, le kart destinataire reçoit le paquet et nous renvoie une réponse nous signifiant sa bonne réception.

3 Interface client et gestion des différentes interactions

L'interaction entre les Karts et les utilisateurs se fait à travers une interface web. Celle-ci est divisée en 3 parties : à droite il y a un joystick qui permet de faire avancer et reculer le Kart, au centre de l'interface apparaissent les objets , le joueur doit cliquer dessus pour les envoyer à son adversaire. À gauche se trouve un 2ème joystick qui permet de faire tourner la voiture à droite et à gauche. En bas

sont affichées des informations utiles à l'utilisateur : son classement dans la course, le checkpoint qu'il a dépassé, ainsi que le nombre de tours qu'il a fait. La figure 5 représente une capture d'écran de notre interface web sur un téléphone :

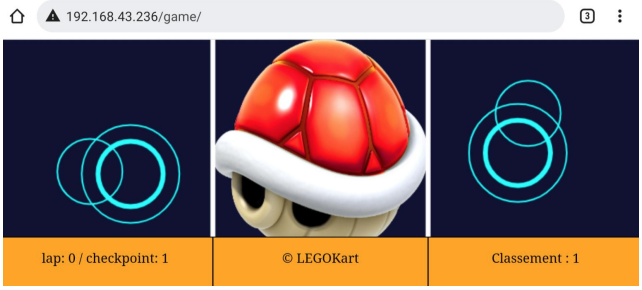

FIGURE 5 – Interface Graphique

3.1 Implémentation des points d'étapes

Notre circuit est constitué de 3 points d'étapes. Un point d'étape est formé à partir d'une chaîne de tags NFC. Le kart , quant à lui, possède des capteurs. La lecture d'un tag par le kart représente le passage d'un kart sur un checkpoint. Il faut un moyen de stocker les informations relatives au classement des karts tout le long de la course. Dans notre implémentation, chaque kart possède son propre tableau de classement. C'est grâce à la communication entre les karts participant à la course que le classement s'actualise. Si le kart passe sur un point d'étape, il actualise son tableau de classement. De plus, il envoie un message à tous les karts participant à la course, les informant qu'il a dépassé le point d'étape x à l'instant t. Les karts utilisent ces informations reçues pour actualiser leur tableau de classement. Tous les autres Karts de la course procèdent de la même façon , pour que tout le monde ait un même tableau de classement actualisé tout le long de la course.

Les karts possèdent ce tableau qui donne, pour chaque joueur x, le temps de franchissement du checkpoint y. Les valeurs -1 montrent que le Kart n'a tout simplement pas franchi le checkpoint en question. C'est à partir de ce tableau descriptif de l'état de la course qu'on déduit le tableau du classement.

J1	J2	J3	J4
3	1	2	4

N° CP	J1	J2	J3	J4
0	t1	t2	t3	t4
1	t5	t6	t7	-1
2	-1	t8	t9	-1
3	-1	t10	-1	-1

FIGURE 6 – exemple de tableaux de classement utilisés dans notre programme

Dans l'exemple de la figure 6, tous les joueurs ont franchi le checkpoints 0, le joueur 4 ne franchit même pas le checkpoint 1, donc il est d'ores et déjà classé dernier. Le joueur 1 s'arrête au 2ème checkpoint, il est donc 3ème. De la même façon on déduit que le joueur 3 est 2ème et que le joueur 2 est le gagnant de la course.

3.2 Implémentation des carapaces

Chaque joueur peut avoir l'occasion de ralentir son adversaire durant la course. En effet, le circuit contiendra des carapaces sous forme de tags nfc, tout comme les

checkpoints. Si un Kart passe sur un tag nfc représentatif d'une carapace, un objet choisi au hasard sera envoyé à l'interface web. L'interface va afficher cet objet à l'utilisateur et va créer une zone cliquable. Dès que l'utilisateur cliquera dessus, un message sera envoyé à son M5Stack. Ce dernier enverra un message au Kart de son adversaire qui est devant. Le kart adverse va se mettre à tourner sur lui même, ce qui va le ralentir dans sa course.

Organisation du projet

Au-delà de l'aspect technique, l'aspect humain prend une place importante dans cette étude pratique. Tout au long de l'année, nous avons appris à mieux travailler en équipe et à s'aider les uns les autres. Nous avons fait des réunions régulièrement tout au long de l'année. Cela nous permettait de mettre en commun notre travail, de se fixer de nouveaux objectifs à court-terme et de se répartir les tâches en fonction de notre avancement.

Conclusion

Nous avions pour objectif de créer une course de karts en legos respectant les règles et les contraintes du jeu Mario Kart Live Home Circuit. Ce jeu regroupe plusieurs karts dans un circuit fermé constitué d'objets et de points d'étapes. Nous avons réussi à mener à bien une première version fonctionnelle de ce projet, avec deux karts construits qui peuvent s'affronter et s'envoyer des carapaces dans un circuit constitué de 3 points d'étapes, les deux karts peuvent aussi connaître leur position dans la course. Concernant la continuité de ce projet, il faudrait d'abord construire plus de karts et finaliser la partie lancement et arrêt du jeu. Pour aller plus loin, il faudrait rajouter divers capteurs sur le kart afin d'avoir plus d'interactions avec le circuit. Par exemple des capteurs de proximité pour la détection de collisions. On pourrait également implémenter de nouveaux objets comme des champignons accélérateurs.

Références

[ESPAsyncUDP librairy] Me-no-dev,
https ://github.com/me-no-dev/ESPAsyncUDP

[ESPAsyncWebServer librairy] Me-no-dev,
https ://github.com/me-no-dev/ESPAsyncWebServer

[WiFiManager librairy] Tzapu,
https ://github.com/tzapu/WiFiManager

[Exemple d'implémentation d'un websocket] RNT team,
https ://randomnerdtutorials.com/esp32-websocket-server-arduino/

[Documentation et exemples NTPClient] aAentinger team,
https ://github.com/arduino-libraries/NTPClient/blob/master/NTPClient.h

[Comparaison modèle de voiture unicyle bicycle] University of Michigan
http ://scholarships.engin.umich.edu/wp-content/uploads/sites/36/2020/02/Final-Presentation-Slides.pdf