Alex MEURILLON
11/12/2023

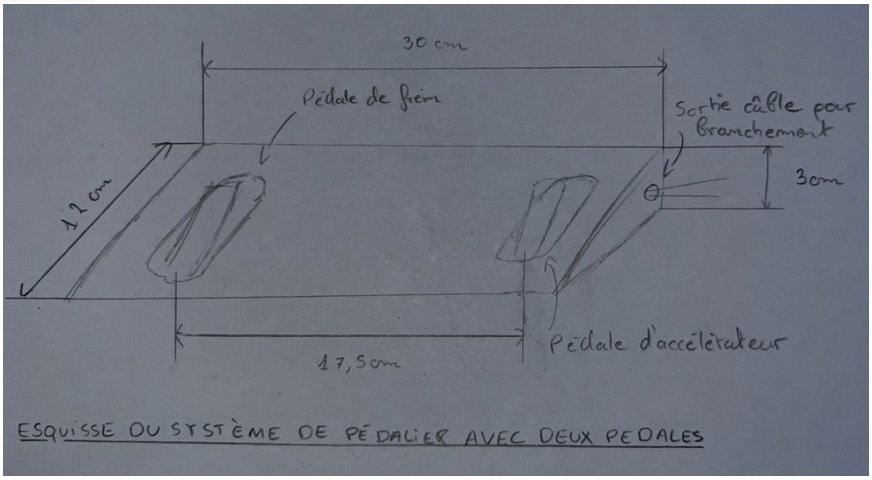
Compte rendu Séance 1 – Projet Arduino – Simulateur de course automobile

Durant la première séance, j'ai réfléchi à nombreuses problématiques dont nous n'avions pas forcément anticipé et commencer notamment à réfléchir au pédalier et aux différentes manières de le construire.

- **Réflexion du pédalier**

Après plusieurs échanges avec Mr RALLO, j'ai décidé de me renseigner sur l'achat de pédale sur le site AliExpress. En effet, j'avais aussi le choix de réaliser mon propre pédalier, dont nous en reparlerons plus tard dans le rapport.

L'achat de pédale demande également de les fixer pour reproduire un vrai pédalier. J'ai eu l'occasion de me rendre au FabLab afin de discuter avec l'ingénieur présent. Il m'a proposé de revenir plus tard, lorsque nous aurons reçu toutes les pièces et que nous pourrons les fixer. Ci-dessous, l'esquisse du pédalier souhaité que je concevrai si les deux pédales sont commandables.



Toute cette démarche a abouti à la demande de deux pédales (image ci-dessus) sur le site marchand AliExpress :

https://fr.aliexpress.com/item/1005005484751231.html?spm=a2g0o.detail.0.0.608d6b0bHI6vpK&gps-id=pcDetailTopMoreOtherSeller&scm=1007.40050.362094.0&scm_id=1007.40050.362094.0&scm-url=1007.40050.362094.0&pvid=55949556-3b09-4313-835b-51501dbaf4b2&_t=gps-id:pcDetailTopMoreOtherSeller,scm-url:1007.40050.362094.0,pvid:55949556-3b09-4313-835b-

51501dbaf4b2,tpp_buckets:668%232846%238114%231999&isseo=y&pdp_npi=4%40dis%21EUR%215.60%211.6%21%21%215.89%21%21%40210384b917023024221663375ee0d1%211120000343615077 59%21rec%21FR%21%21AB

Si la commande n'est pas possible, j'ai également réfléchi à un système de pédalier que je pourrai construire.

Son fonctionnement est assez simple : lorsque l'on appuie sur la pédale, une crémaillère permet le mouvement d'un engrenage. Cet engrenage est lui-même relié à un potentiomètre qui permet alors de récupérer les informations sur la carte Arduino. En calibrant le potentiomètre on peut ainsi déterminer la tension appliquée et donc retrouver la sensation d'une véritable pédale d'accélérateur.

Le raisonnement est analogue pour la pédale de frein.
On obtiendrait alors une pédale de cette forme :

Auteur : JOB. Design number : 18966. Source : https://cults3d.com

- **Conception matérielle du volant**

J'ai pu également réfléchir avec mon binôme sur la conception matérielle du volant. Deux solutions se sont offertes à nous :

- Assemblage et découpe laser
- Impression 3D

Cette seconde option est trop longue. En effet, après modélisation d'un volant sur un logiciel de visualisation 3D, celle-ci aurait duré 17H pour l'imprimer.

Nous décidons donc de monter le volant à l'aide de l'assemblage de pièces et de découpe laser.

Pour plus de détail sur ce paragraphe, veuillez-vous référer au travail d'Ugo PRESSENDA.

- **Réflexion sur la communication avec le pc : filaire, Bluetooth ou Wi-Fi ?**

J'ai réfléchi à la possibilité d'une communication entre la carte Arduino et le PC en sans-fil. Ce type de communication me semble inapproprié pour notre projet. En effet, lorsque l'on joue à un jeu automobile sur PC, nous nous tenons forcément proche du PC. Donc la communication sans-fil ne semble pas nécessaire dans notre cas. De plus, avec ce type de communication, nous devions installer des piles pour assurer le bon fonctionnement des composants et s'assurer de la bonne transmission des radios fréquences qui peuvent être parasitée.

Nous avons donc retenu la communication filaire entre le PC et la carte Arduino. Cette solution présente également un avantage d'un point de vue de la réactivité, qui est important dans un simulateur de course.

- **Liaison logicielle entre la carte Arduino et le PC**

Après quelques recherches, certaines solutions sont possibles à l'aide d'API (Application Programming Interface) permettant de réaliser la communication entre la carte Arduino et le PC. L'API que nous avons retenue se nomme XInput. C'est une librairie statique (« statique » car elle est compilée avec le programme qui en a besoin) qui est compilée dans le jeu exécuté.

XInput est fourni sous la forme d'une DLL (Dynamic Link Library) qui est installée dans les dossiers du système d'installation.

Cette API est la solution la plus pratique car celle-ci est téléchargeable sur le logiciel Arduino INE directement. Cela nous permettra lors de la prochaine séance de tester les premières pièces sur un jeu vidéo sur un PC.

J'ai pu commencer à me renseigner sur l'API XInput à l'aide de sa documentation et des liens ci-dessous :

Disposition du contrôleur

Les contrôleurs compatibles ont deux bâtons directionnels analogiques, chacun avec un bouton numérique, deux déclencheurs analogiques, un pavé directionnel numérique avec quatre directions et huit boutons numériques. Les états de chacune de ces entrées sont retournés dans la **structure XINPUT_GAMEPAD** lorsque la **fonction XInputGetState** est appelée.

Le contrôleur a également deux moteurs de vibration pour fournir des effets de commentaires à l'utilisateur. Les vitesses de ces moteurs sont spécifiées dans la **structure XINPUT_VIBRATION** passée à la **fonction XInputSetState** pour définir les effets de vibration.

Si vous le souhaitez, un casque peut être connecté au contrôleur. Le casque a un microphone pour l'entrée vocale et un casque pour la sortie sonore. Vous pouvez appeler la **fonction XInputGetAudioDeviceIds** ou XInputGetDSoundAudioDeviceGuids **héritée** pour obtenir les identificateurs d'appareil qui correspondent aux appareils pour le microphone et le casque. Vous pouvez ensuite utiliser les API Core Audio pour recevoir une entrée vocale et envoyer une sortie audio.

Utilisation de XInput

L'utilisation de XInput est aussi simple que l'appel des fonctions XInput comme nécessaire. Les fonctions XInput permettent de récupérer l'état du contrôleur, d'obtenir les identifiants audio du casque et de définir les effets de grondement du contrôleur.

Plusieurs contrôleurs

L'API XInput prend en charge jusqu'à quatre contrôleurs connectés à tout moment. Les fonctions XInput nécessitent tous un paramètre *dwUserIndex* transmis pour identifier le contrôleur en cours de définition ou interrogé. Cet ID se trouve dans la plage de 0 à 3 et est défini automatiquement par XInput. Le nombre correspond au port auquel le contrôleur est connecté et n'est pas modifiable.

Documentation de l'API XInput

https://learn.microsoft.com/fr-fr/windows/win32/xinput/xinput-game-controller-apis-portal
https://www.arduino.cc/reference/en/libraries/xinput/
https://github.com/dmadison/ArduinoXInput#compatible-boards