Rapport du projet de Frelon

1. Description :

Il s’agit d'un dispositif électronique conçu pour protéger les ruches d'abeilles installées dans notre école contre les frelons asiatiques. Ce dispositif a pour but d’éliminer tout frelon s’approchant trop près de la ruche.   
Il comprend un laser assez puissant pour être léthale (vu d’un frelon), deux moteurs pas à pas qui permettent pour assurer la rotation du laser et un ventilateur pour éviter tout surchauffe du laser  
De plus, pour assurer la détection du frelon, nous utilisons une caméra qui prend régulièrement des photos et une IA de traitement d’image. Le tout est contrôlé par un microcontrôleur ESP32. Pour simplifier la partie hardware, nous utiliserons le esp wrover kit.  
L'alimentation est assurée une batterie 5V branché sur la carte.

1. Fonctionnement :

La carte prend régulièrement des photos. Nous utiliserons un code généré par l’IA pour effectuer la reconnaissance des images. Ce modèle, nommé Edge Impulse, traitera les images capturées par la caméra afin de déterminer les positions x et y du frelon dans l’espace. Le frelon est supposé être stationnaire. Une fois détecté, le moteur tourne pour placer le laser en face de sa position, puis nous effectuons une nouvelle confirmation si le frelon est toujours immobile. Si le signal est positif, nous neutralisons le frelon à l’aide du laser.

Après avoir éliminé le frelon, d'autres images doivent être capturées et envoyées à l'outil de l’IA pour la reconnaissance, afin de confirmer que le frelon a bien été éliminé. Ensuite, le moteur retourne à sa position initiale et la caméra détecte d'autres frelons, et le processus se répète.

1. Avancement sur le projet :
   1. Le contrôle moteur :

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquementLes moteurs choisis par nos prédécesseurs sont des servomoteurs. Ils contiennent un décodeur, qui permet de transformer un signal numérique en position à atteindre

Le problème rencontré est que nous ne connaissons pas les valeurs qu’il faut utiliser pour ces moteurs en particulier. Par ailleurs, on sait qu’avec un même signal, les 2 moteurs auront un résultat différent. Le seul point positif, c’est qu’il s’agit juste d’un simple offset.

En connaissant le temps pour 90°, vous pouvez déduire le temps de 180° comme sur l’illustration précédente.  
Je vous conseille vivement de refaire les mesures avec un rapporteur, pour être sûr que les valeurs écrites dans le code soient bien précises.   
  
Pour le code, la librairie qu’on utilise pour contrôler les moteurs n’utilisent pas un temps, mais des ticks. On décompose la période en 4096 points, et vous choisissez cb de « points » vous souhaitez à l’état haut. Reprenons l’illustration, pour une période de 20ms, si nous souhaitons 1ms à l’état haut, il faut 4096/20 = 205 ticks en valeur.  
  
Comme vous avez pu le comprendre, que l’on parle de ticks, temps à l’état haut ou degrés, cela revient au même au finale.  
Pour une meilleur visibilité et projection, dans le reste du code nous utilisons des degrés dans nos calculs, que nous convertissons en tics lors de la commande des moteurs.  
  
Par ailleurs, pour connaitre les degrés à atteindre, nous utilisons les coordonnées x y à ciblées. Nous savons que nous commençons en position « initiale », soit 90° axe x et 90° axe y. Nous sommes au centre de l’image, donc 160x 120y. Nous pouvons en déduire le déplacement à effectuer pour se centrer sur le frelon 🡪 convertion en angle souhaité

Vous pouvez trouver le code du fonctionnement du moteur, sur ce lien suivant : <https://github.com/aghrizen/HornetProject23/blob/main/Moteur/Software/testMoteur.ino>

* 1. La partie carte :

Concernant le code qui sert à faire fonctionner la carte pour qu’elle prenne des photos, il fonctionne bien. Nous l’avons testé et l’essayé avec deux cartes de ESP32 modèle Wrover. Toutefois, une de ceux cartes n’arrive pas à sauvegarder les images sur la carte SD (besoin de resouder des fils, impossible à louper). N’ayant besoin que d’une seul, nous avons utilisé celle qui fonctionne.  
Il existe un autre modèle de carte, la ESP CAMERA IA, qui serait plus adapter pour la prise de photo (détection de mouvement), mais elle ne possède pas assez de pins pour contrôler moteur + laser.  
Une des solutions envisagées serait de créer un port de communication entre la ESP camera IA et le Wrover KIT afin d’obtenir de meilleur résultat.  
Nous n’avons pas pu tenter cette solution par manque de temps.

* 1. Détection de frelon, utilisation du Module l’I.A Edge Impulse :

Nous avons d'abord entamé la démarche de comprendre comment utiliser le modèle de reconnaissance d'images de l'IA. Cette étape s'est avérée essentielle pour parvenir à générer un code fonctionnel. Notre objectif principal étant la reconnaissance d'images, nous avons opté pour l'utilisation de l'IA dans le cadre de la "Détection d'Objets".

Après avoir assimilé le fonctionnement de cet outil, nous avons créé plusieurs versions, chacune ayant un rôle spécifique. Les premières versions ont été déployées afin de conduire des tests multiples sur l'outil, explorant trois modèles différents.

Modèles :

Nous disposons de plusieurs modèles pour la détection d'objets. Pour notre projet, nous avons choisi l'un des trois suivants :

- MobileNetV2 SSD FPN-Lite (320x320 uniquement)

- FOMO (Faster Objects, More Objects) MobileNetV2 0.1

- FOMO (Faster Objects, More Objects) MobileNetV2 0.35

Nous préférons utiliser le deuxième, car il s'agit d'un modèle de détection d'objets basé sur MobileNetV2 (alpha 0.35) conçu pour segmenter grossièrement une image en une grille d'arrière-plan par rapport aux objets d'intérêt. Ces modèles sont conçus pour être inférieurs à 100 Ko en taille et prennent en charge une entrée en niveaux de gris ou RVB à n'importe quelle résolution.

Ensuite, les versions ultérieures ont été conçues dans le but d'exploiter au maximum les bases de données disponibles, dans l'espoir de constater une amélioration de la probabilité de détection des objets sélectionnés, qui sont dans notre cas ‘Frelon’ et ‘l'Abeille’.

Le problème initial auquel nous avons été confrontés était le manque d'images disponibles pour entraîner notre projet. Par conséquent, nous avons décidé d'utiliser une base de données existante. Malgré nos multiples tentatives de créer différentes versions, nous avons constaté qu'un problème persistait en ce qui concerne la détection.

La version qui s'est avérée la plus efficace peut être trouvée dans ce répertoire GitHub : <https://github.com/aghrizen/HornetProject23/tree/main/EdgeImpulseCode/to_use_version_V2>

Cette version présente les caractéristiques suivantes :

- Elle détecte correctement le frelon.

- Parfois, elle parvient à détecter l'abeille.

- Elle envoie le message "aucun objet détecté" lorsque rien n'est présent devant la caméra.

- Pour les tests, il faut utiliser ce code ou toute autre version nécessairemnt dedans la boîte. C’est-à-dre, d’utiliser la boîte comme arrière-plan où nous plaçons nos frelons et abeilles de test.

De plus :

* Nous avons modifié cette version manuellement en intégrant le code du moteur et en ajoutant un pour envoyer le signal du laser.
* Nous avons ajouté les pins de la carte ESP32-Wrover. Parce que, par défaut, quand le code est généré, il contient que l’I.A Thinker et un autre modèle que nous n’allons jamais utiliser.

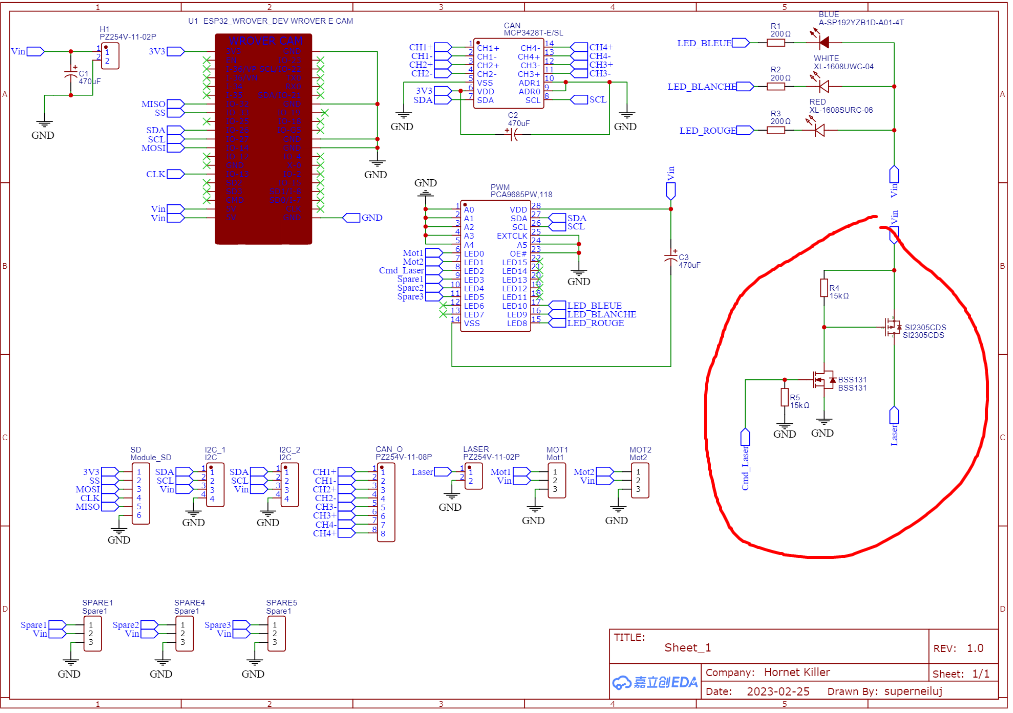
Pour l’I.A Thinker :

* Nous avons utilisé le code fourni par le professeur spécifiquement conçu pour ce modèle de carte. Cette carte présente des fonctionnalités plus avancées, notamment la détection de mouvement et la possibilité de prendre des images avec flash. Cependant, nous avons rencontré une limitation : nous ne pouvons pas utiliser la partie du moteur et du laser avec cette carte. Par conséquent, nous avons décidé de nous en tenir au modèle ESP32-Wrover pour notre projet.
  1. Laser :

Afin de contrôler le laser, nous avons testé la solution proposée par le groupe d’avant.

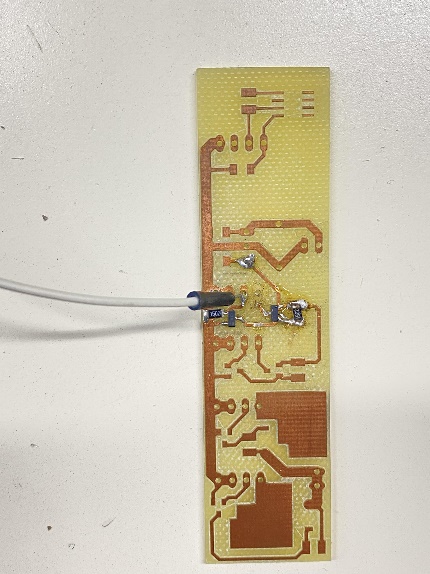
La solution sert à utiliser deux transistors pour contrôler l’allumage du laser.

Elle est présentée sur ce schéma suivant :



* Lien pour trouver le schématique : <https://github.com/aghrizen/HornetProject23/blob/main/Hardware/useful_datasheet/Schematic_hornetkiller.pdf>

Nous avons donc soudé une carte avec les composants au-dessus :



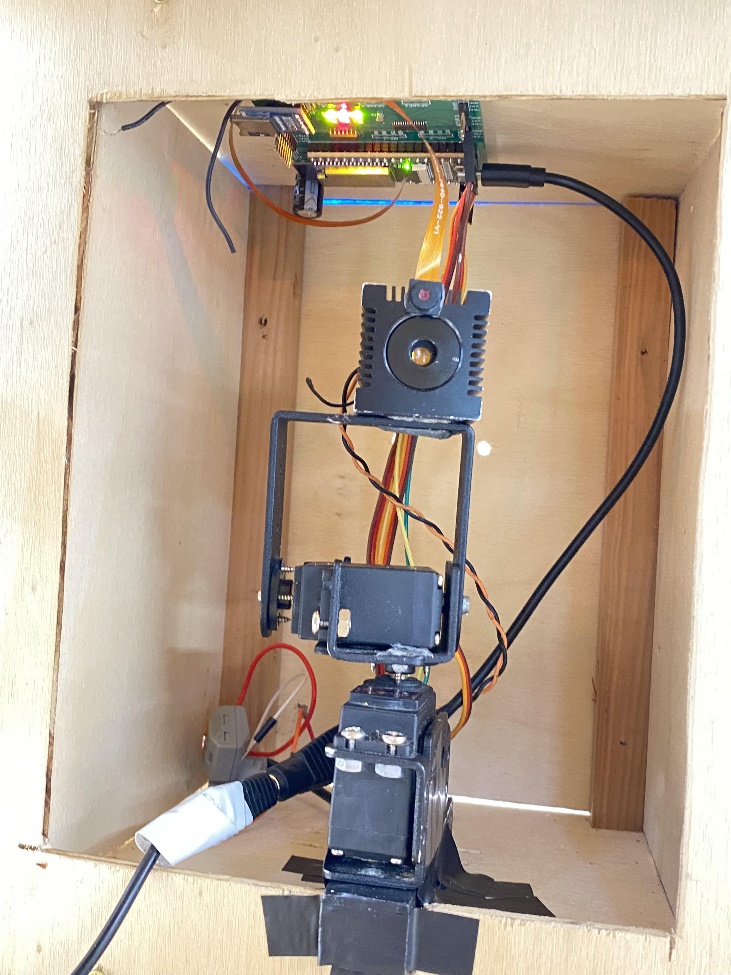
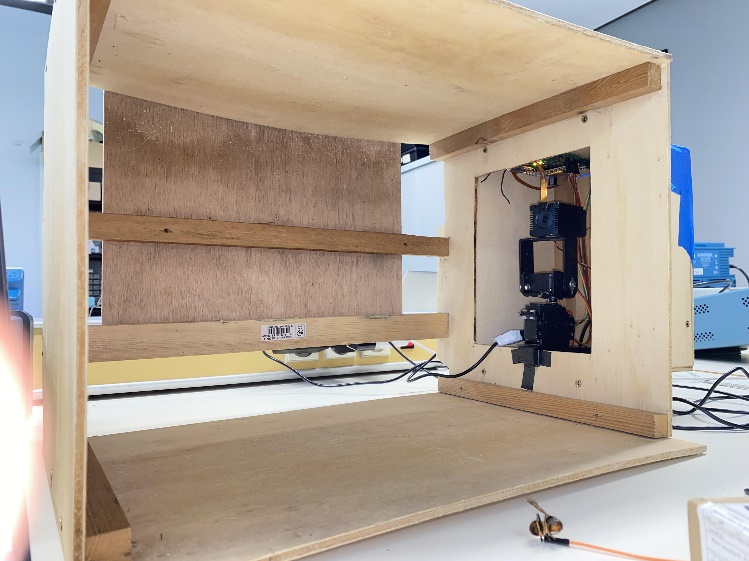
Nous avons bien testé la solution et ça fonctionné comme ce qu’il faut.

* 1. Le ventilateur :

Comme mentionné précédemment, nous devons utiliser le ventilateur pour prévenir la surchauffe du laser. Nous avons déjà installé un ventilateur derrière le laser. Un câble était initialement séparé du ventilateur. Après l'avoir soudé et testé, nous pouvons confirmer que le ventilateur fonctionne correctement, sans aucun problème.

* 1. L’emplacement des éléments

Pour positionner notre dispositif, nous avons recherché l'emplacement optimal qui permettrait à la caméra de couvrir le plus d'espace possible. À cet effet, nous avons identifié l'emplacement suivant :



1. Les étapes restantes :

\* Effectuer de nouvelles mesures avec les moteurs pour être sûr que la commande software 90° nous permet bien d’obtenir un angle de 90° réellement. Nous n’avions pas utilisé de rapporteur, donc actuellement les valeurs utilisées sont +- 5° d’écart

\* Générer un nouveau code d’IA. Les résultats qu’on obtient avec le modèle actuel sont très peu satisfaisants. Nous arrivons à détecter un frelon, mais les coordonnées x et y mesurées sont aléatoires. De plus, il y a beaucoup de faux positif. Si vous mettez votre main à coté de la caméra, l’ia la considérera comme un frelon. Ce qui peut être embêtant…

Plusieurs axes d’améliorations :   
- Changer la base de données, essayer de reprendre des photos avec des frelons et la même caméra que celles utilisée.  
- Changer les paramètres d’entrainement. Pour ça, il suffit juste de tester, pas de solution miracle

- Changer le langage de programmation. Actuellement, le projet est codé en arduino, mais edge impulse par défaut génère du code en C. Sur le site, il est possible de demander de générer du code en arduino, mais il y a des chances que cela influe sur le résultat.  
Petit bémole : Il faut traduire la totalité du code contrôle moteur existant en C, ce n’est pas très compliqué, c’est juste long..

\* Refaire une carte commande. Actuellement, la carte possède 6 emplacements pwn alors qu’on en a besoin que de 2. Par ailleurs, elle ne possède pas de simple pin DigitalIn/DigitalOut accessible, alors qu’on a besoin d’une pin DigitalIn pour le laser et d’une autre pour le ventillateur. La solution temporaire qu’on a trouvée est de mettre la PWN à 255 ( tout le temps à l’état haut ) ou à 0 ( tout le temps à l’état bas) pour simuler une I/O. Mais c’est un peu ridicule.  
De plus, il se peut qu’il y ait des problèmes de câblage. J’ai l’impression que les masses ne sont pas communes entre les différents emplacement PWN, ce qui peut poser un problème.

\*Pour la carte du laser, il faudrait refaire les tests. Une des résistances était mal soudé (pas assez bien soudé) et est tombée. Nous n’avons pas retesté la carte depuis que nous l’avons resoudé.