Le Bouteiller Desforges Océane

L2 Sciences de l'ingénieur – Renforcé

2025/2026

COMPTE RENDU

PROJET PERSONNEL DE RECHERCHE

SUJET : Détection d'objets en milieu maritime par apprentissage profond

Encadrante : Nadège Thirion-Moreau, Laboratoire LIS

Table des matières

1. Description du projet

1.1. Contexte et motivation du projet

Ce projet est né d'un besoin personnel lié à ma pratique du wingfoil, un sport qui dépend fortement des conditions météorologiques, et plus particulièrement du vent.

Pour se faire une idée des conditions de vent réelles, plusieurs solutions existent :

— Les sites de prévisions météorologiques (mais il ne s'agit que de prévisions)
— Les relevés d'anémomètres (malheureusement, la plupart sont déventés par des bâtiments ou des reliefs)
— Les webcams (mais souvent mal orientées et de mauvaise qualité)

Aucune de ces solutions n'étant réellement satisfaisantes, mettre en place un système permettant de savoir si les conditions locales sont exploitables constituerait une véritable valeur ajoutée.

Mon atout : la vue de l'appartement de ma mère, situé face au plan d'eau.

Mon projet consiste donc à exploiter cette vue en y installant une caméra à la fenêtre.

Un wingfoiler

La vue de l'appartement

1.2. Présentation du projet

1.2.1. Version initiale du projet

L'idée initiale présentée en juin 2025 (1ère année de Licence) consistait au développement d'un modèle de détection automatique de wingfoil à partir d'images prises avec mon téléphone.

Ce projet est l'occasion pour moi de découvrir le domaine de l'intelligence artificielle et j'avais proposé de comparer 2 types de modèles :

— Un modèle basé sur TensorFlow (MobileNet SSD V2)

— Un modèle basé sur PyTorch (YOLOv8 nano)

Pour la prise d'images, j'ai utilisé mon téléphone Apple avec un zoom numérique x5, et pour le développement, mon ordinateur portable MacBook.

Après plusieurs mois, j'avais obtenu une base de données de 180 images et j'ai développé 2 modèles : MobileNet SSD V2 (TensorFlow) et YOLO v8 nano, mais le résultat n'était pas satisfaisant.

J'ai réalisé plusieurs choses :

— Un zoom optique serait un vrai avantage compte tenu de la distance
— Développer un modèle IA sur un Mac n'est pas une bonne idée (beaucoup de problèmes de compatibilité, d'environnement et l'obligation de passer par des machines virtuelles lors du développement)

De plus, la finalité future de ce modèle étant d'être intégré dans une caméra, il aurait été beaucoup plus judicieux de construire la base de données à partir d'images issues de cette même caméra.

1.2.2. Version finale du projet

En janvier 2026, je décide de repartir de zéro et de m'équiper de tout le matériel nécessaire pour concevoir le projet dans sa totalité et dans les meilleures conditions : un PC Windows/Linux avec carte graphique, une caméra avec zoom, une carte électronique pour la programmation du système.

L'objectif initial (*développement d'un modèle IA de détection de Wingfoil*) est toujours présent mais le projet comprendra aussi la conception du système caméra permettant :

— La récupération d'images pour le dataset.
— L'intégration d'un détecteur intelligent de wingfoil avec envoi de notification lors d'une détection.
— Une interface web pour la visualisation du flux caméra et le contrôle du système à distance de manière sécurisée.

Nouvel objectif : **réaliser une plateforme de surveillance visuelle intégrant un modèle de détection intelligent de wingfoil permettant l'envoi de notifications de détection, avec accès et contrôle à distance sécurisés.**

La caméra sera orientée vers le plan d'eau à environ 200 m de la fenêtre.

Le projet s'inscrit dans une démarche volontairement open source. Mon objectif est de contrôler chaque couche du système et d'adapter précisément le comportement de la plateforme à mes besoins spécifiques.

1.3. Objectifs fonctionnels

— **Consulter le flux vidéo en direct depuis mon téléphone à distance**

  via un tunnel sécurisé pour accéder au réseau domestique sur lequel est connecté le système de surveillance.

— **Créer un modèle IA de détection automatique de wingfoils**

Base de données issue de la caméra principalement.

La base de données sera composée de 3 types d'image :

— Dataset aléatoire (issu de capture automatique à des heures aléatoires)
— Dataset manuel (issu de capture manuelle via un bouton de contrôle à distance)

Et une fois le modèle de détection en place, ajout de :

— Dataset détection (images avec détection en sortie d'inférence)

— **Recevoir des notifications sur le téléphone lors d'une détection**

Avec l'image associée (le snapshot).

— **Améliorer en continu le modèle de détection**

En automatisant au maximum le processus (transfert, labellisation, réentraînement, redéploiement).

— **Pouvoir interagir à distance et visualiser le flux caméra sur une interface.**

En plus de visualiser la vidéo en direct, je dois pouvoir interagir avec le système de surveillance (ex : choisir la fréquence d'inférence du modèle, le nombre de captures aléatoire, prendre une capture d'image, etc...). Cela sera possible via une interface web sécurisée.

2. Conception technique

2.1. Choix technologique

— **Carte Raspberry pi**

Pour visualiser la caméra sur mon téléphone, faire tourner mon détecteur IA, stocker des images, centraliser le tout sur une interface et pouvoir y accéder à distance, il me faut une carte électronique pour système embarqué.

Plusieurs solutions matérielles existent, notamment les microcontrôleurs (Arduino, ESP32), les nano-ordinateurs alternatifs (Orange Pi, Odroid), les plateformes orientées IA (NVIDIA Jetson) ainsi que les mini-PC classiques.

Les microcontrôleurs ont été écartés car ils ne permettent pas d'exécuter un système d'exploitation complet ni de gérer du streaming vidéo et de l'intelligence artificielle.

Les plateformes plus puissantes comme les mini-PC et NVIDIA sont surdimensionnées pour ce projet, notamment en termes de coût et de consommation énergétique.

La Raspberry Pi représente le meilleur compromis entre coût, consommation, flexibilité logicielle, disponibilité des ressources open source et compatibilité avec les outils d'intelligence artificielle utilisés dans le projet.

Carte Raspberry pi

— **Caméra avec zoom 8-50 mm F1.4**

J'ai choisi une caméra de l'écosystème Raspberry Pi afin de garantir la compatibilité avec le système et de faciliter l'accès au flux vidéo et aux paramètres de contrôle.

Le zoom 8–50 mm permettra d'obtenir un niveau de détail suffisant pour la détection de wingfoils situés entre 200 et 600 mètres de la caméra.

Caméra HQ Raspberry + zoom 8-50mm

— **Support pour l'inférence : l'accélérateur IA NPU Hailo-8**

Effectuer l'inférence (l'application du modèle IA) sur le CPU de la Raspberry Pi limite énormément en puissance et en taille de modèle. On risque de saturer le CPU et rendre le fonctionnement du système complet très instable.

Or, il est possible que l'inférence s'effectue sur un autre support : l'image à traiter est envoyée sur ce support, l'inférence est réalisée, puis le résultat est renvoyé au CPU.

Plusieurs supports sont possibles :

Support de l'inférence	Avantages	Inconvénients
Accélérateur IA (ex : TPU Coral, NPU Hailo)	Système autonome	Puissance limité
Déporté sur mini serveur GPU local (ex : pc en réseau local)	Grande puissance de calcul	Le pc doit rester allumé H24
Déporté sur serveur Cloud distant (ex : Google Cloud, Amazon Web Services)	Grande puissance de calcul	Abonnement couteux (10-100€/mois)

Étant donné que mon pc n'est pas toujours à la maison ni allumé 24 h/24, et que je ne souhaite pas payer un abonnement, je décide d'utiliser un accélérateur IA ce qui me

permettra d'utiliser un modèle IA de taille raisonnable, certes, mais sans perturber la fluidité de mon système global.

Je choisis comme accélérateur IA le NPU Hailo, car il est plus puissant que le TPU Coral et permet d'inférer une plus grande variété de modèles.

AI HAT+ avec NPU Hailo-8 (accélérateur IA)

2.2. Architecture globale

Le système final comprend :

— **Un Raspberry Pi relié à une caméra** : pour le stream vidéo, la capture d'images, le détecteur IA et l'interface web et son serveur.
— **Un PC** : pour développer le modèle IA et mettre en place le système global.
— **Le téléphone personnel** : pour visualiser et interagir avec l'interface web et recevoir les notifications d'alerte, tout ça à distance et de manière sécurisée.

Cf ANNEXES (6.2. Schématisation de l'architecture globale)

Description détaillée de chaque support :

2.2.1. Raspberry Pi caméra

Ce que fait la « Pi caméra » :

— Stream vidéo
— Capture et stockage d'images pour alimenter le dataset
— Interface web
— Détection IA et déclenchement de notifications lors de détections positives

Détail matériel :

— Raspberry pi 5, 8Go de RAM* (CPU* 4 cœurs ARM Cortex-A76) *Cf ANNEXE 6.1*
— SSD* NVMe 256GB en USB (plus robuste que micro-SD quand nombreuses écritures)
— AI HAT+ 26 TOPS contenant le NPU* Hailo-8 (Accélérateur IA, connecté via port PCLe)

— Caméra HQ Raspberry (capteur Sony IMX477, 12.3MP, format 4 :3)
— Zoom 8-50 mm F1.4
— Ventilateur « *active cooler* »
— Boîtier en aluminium + bouton power déporté sur boîtier
— Support motorisé pour rotation de la caméra (socle sur roues + 2 moteurs DC + carte de puissance L298N + fils Dupont)

CPU *(Central Processing Unit) exécute Linux et les différents codes qu'on lui donne (c'est le travailleur)*

***RAM** : sert à stocker temporairement les données en cours de calcul (c'est la mémoire vive)*

***SSD** ou micro-SD : sert à stocker le système Linux, les programmes, les images/vidéos (c'est le stockage)*

***NPU Hailo** (Neural Processing Unit) s'occupe uniquement de l'inférence IA, permet d'alléger la charge CPU*

Détail logiciel et librairies :

— **Raspberry Pi OS 64 bits, lite** (système d'exploitation, lite=sans interface graphique)
— **Picamera2** (accède au capteur caméra, gère résolution, envoie le flux vidéo à l'encodeur H264 du SoC, et en parallèle stocke les dernières images brutes dans un buffer en RAM)
— **MediaMTX** (serveur streaming qui récupère flux vidéo et le publie en RTSP/WebRTC/HLS)
— **FastAPI + Redis/WebSocket + Cloudflare Tunnel** (pour la diffusion de l'interface Web en local (FastAPI), la mise à jour en temps réel (Redis et WebSocket event) et l'accès à distance de manière sécurisée via Cloudflare Tunnel et Zero Trust)

2.2.2. PC

Ce que fait le pc :

— Se connecte en SSH à la Pi caméra (en phase de développement, pour la récupération du dataset, pour l'envoi du modèle mis à jour)
— Annotation des images (labellisation)
— Développement / entraînement / amélioration du modèle IA (VS Code)
— Exportation du modèle au format adéquat pour l'accélérateur IA (NPU Hailo-8)

Détail matériel :

— PC portable MSI GP66 (i7 11th gen 2.30GHz 8 coeurs, 32Go de RAM, 1To de disque dur, GPU NVIDIA GeForce RTX 3080)

Détail logiciel :

— Windows 11 + WSL (Windows Subsystem for Linux)
— Visual Studio code + l'extension Remote - WSL
— Logiciel de labellisation (CVAT)

2.2.3. Téléphone

Ce qu'il fait :

— Accès à l'interface web via un navigateur

— Réception de notifications push via Telegram

Détail matériel :

— Téléphone personnel (Android ou Apple)

Détail logiciel :

— Application de messagerie Telegram

2.3. Choix du modèle IA

Le fait d'utiliser un NPU Hailo-8 va permettre d'accélérer l'inférence mais en restant sur des modèles de type edge (conçus pour l'embarqué).

Les modèle pré-entrainés adaptés sont principalement : YOLO (*Pytorch*) et MobilNet SSD (*TensorFlow*).

Après avoir exploré les 2 durant mon projet initial en 2025, j'ai une préférence pour YOLO car il est plus simple à mettre en place pour des résultats aussi performants.

Il reste maintenant à savoir quel modèle YOLO permettra d'obtenir la meilleure précision de détection tout en ne créant aucune latence sur le système.

2.3.1. Besoins et contraintes

Le choix du modèle est conditionné par les besoins du projet et les contraintes matérielles pour l'inférence.

Nos contraintes matérielles :

— Inférence sur le NPU Hailo-8 26 TOPS. Il est capable d'effectuer plusieurs inférences en parallèle. Mémoire interne : ~26 MB
— Le capteur de la caméra HQ est au format 4 :3 et contient 4 résolutions natives. Choisir parmi l'une d'elles permettra d'éviter tout resize de la part de Picamera2.

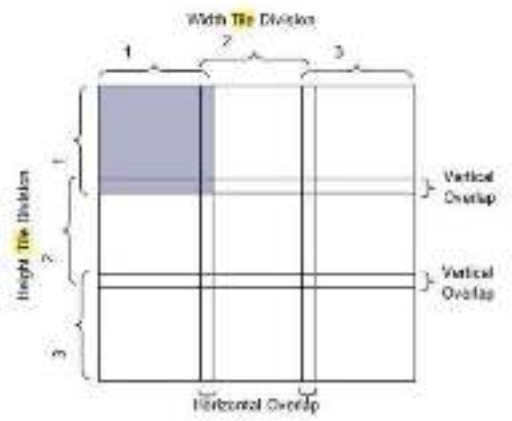
Nos besoins :

— Fréquence d'inférence limite : 1 image toutes les 5 secondes (en mode alerte)
— Détection de petits objets (une dizaine de pixels sur une image 2MP avec zoom)

En effet, dans ce projet, il n'est pas nécessaire d'avoir un détecteur en temps réel (>1fps) car le but n'est pas d'afficher le cadre de détection en temps réel sur le flux vidéo mais uniquement d'alerter par notification lors d'une détection de wingfoil.

Cette fréquence lente va permettre d'optimiser le modèle et de l'adapter à la recherche de petits objets.

2.3.2. Tiling & Overlap :



Tiling 3x3 avec overlap sur image format 1 :1

Tiling = découpe de l'image en plusieurs images (tuiles)

Overlap = recouvrement des tuiles.

Afin d'éviter toute déformation à la suite d'un resize et garder la qualité de l'image d'origine, on va découper l'image en plusieurs tuiles qui se recouvrent.

Le recouvrement permet d'éviter de couper un objet en 2 et de perdre ainsi une détection.

Si la taille de l'objet est inférieure au nombre de pixels de recouvrement (vertical et horizontal), alors l'objet sera forcément dans une des tuiles.

Dans notre cas, pas besoin d'un gros recouvrement car les objets à détecter font une dizaine de pixels en longueur et en largeur. Mais mieux vaut prendre une marge pour les objets plus près et plus gros.

Choix du Tiling selon la résolution caméra :

$$overlap = \frac{(nb\_tuiles \times 640 - taille\_image)}{nb\_tuiles - 1}$$

Tous les modèles de type YOLO sont optimisés pour des images d'entrée de résolution 640x640 pixels. Pour maintenir la qualité d'origine, il est donc préférable d'effectuer le découpage sur des tuiles de 640x640p. Le recouvrement devra être au minimum de 100 pixels (horizontal et vertical) pour s'assurer de ne manquer aucune détection.

Calcul de l'overlap :

Le tableau ci-dessous présente les grilles de tuiles minimales qui permettent de recouvrir toute l'image avec le calcul du nombre de pixels compris dans l'overlap en vertical et horizontal, pour les résolutions adaptées à notre caméra :

Résolutions natives caméra HQ	Grille de tuiles	Overlap horizontal	Overlap vertical
4056× 3040 (12.3 MP)	Trop lourd	-	-
2028 × 1520 (3.08 MP)	4 × 3	≈177 px	≈200 px
1640 × 1232 (2.02 MP)	**3 × 2**	**140 px**	**48 px**
1332 × 990 (1.32 MP)	2 × 2	≈52 px	≈290 px

La résolution R2MP (1640x1232 pixels) est la plus adaptée pour notre recherche de petits objets car elle ne contient pas un trop grand nombre de tuiles (3x2=6 tuiles) avec un overlap qui reste raisonnable.

Cependant, l'overlap vertical est un peu trop faible et trop différent de l'overlap horizontal, mieux vaut que les 2 soient similaires pour rester logique dans la détection.

Une des solutions est de supprimer les pixels du haut de l'image, dans la majorité des cas du ciel, ce qui permettra d'augmenter l'overlap sans devoir ajouter une tuile.

Ainsi, supprimer les 96 pixels du haut de l'image juste avant le tiling permettra d'avoir le même overlap en X et en Y (140 pixels) et de rester sur 3 x 2 tuiles.

— Résolution caméra HQ optimale pour mon projet **: 1640 × 1232 (R2MP)**
— Découpage des images R2MP : **crop des 96 pixels du haut puis tiling 3x2 en tuiles 640x640 px avec overlap X et Y de 140px (soit un ratio/tuile=0.21875)**

Représentation visuelle des tuiles et crop sur une images contenant des objets de près :

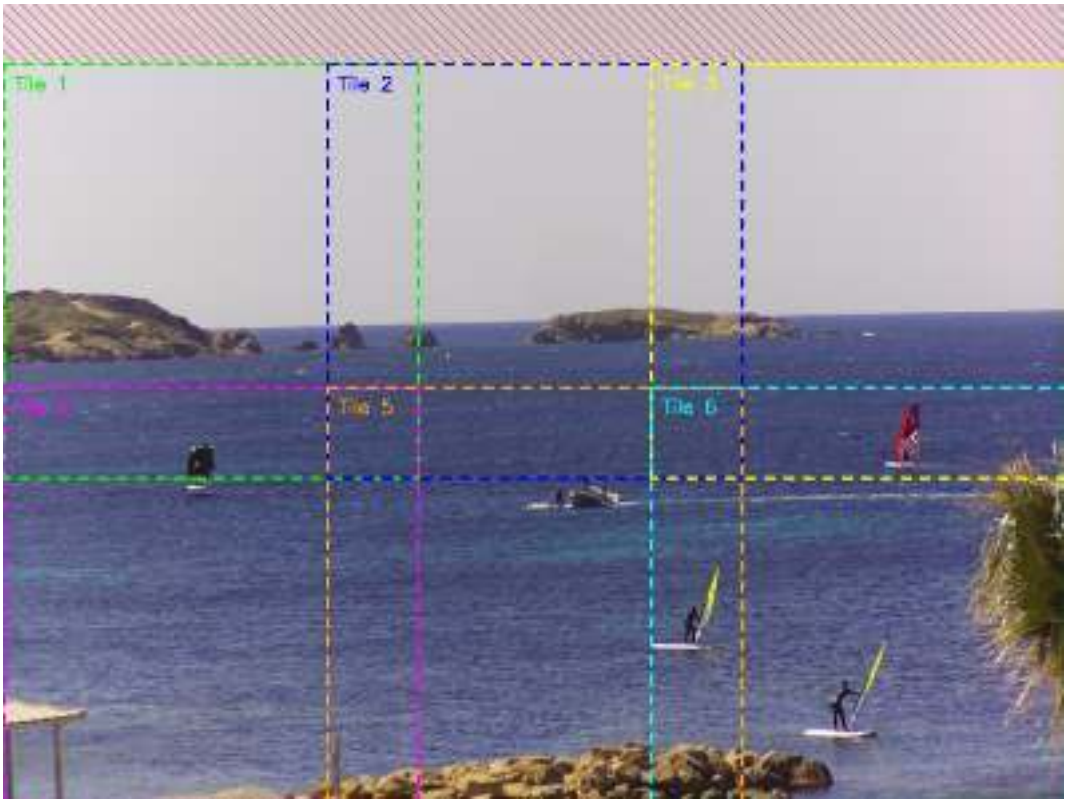


Image (1640x1232 px) avec bordure des tuiles (640x640 px) et zone à supprimer (les 92 px du haut)

2.3.3. Choix du modèle YOLO

Le modèle va être crée via du transfert learning d'un modèle YOLO pré-entrainé à la détection d'objets (base de données COCO). Ainsi, avant même de commencer son entrainement sur notre dataset, il est déjà habitué à détecter des objets sur une images.

En janvier 2026, Ultralitycs a sorti une nouvelle version de modèle YOLO : YOLO 26 (nano, small). On testera donc les modèles YOLO v5 v8 v11 et 26, nano et small car medium devient trop lourd pour de l'embarqué.

En mode alerte, le NPU devra effectuer 3x2=6 inférences en moins de 5 secondes, soit une fréquence d'inférence minimale d'environ 1 fps (frames par seconde).

— Modèle YOLO à tester : **YOLO v5, v8, v11, 26 (nano et small)**

Etant donné que tous les modèle YOLO ont les mêmes tailles d'image en entrée, il sera facile de changer de version (v5, v8, v11, 26) ou de taille du modèle (nano, small).

2.3.4. Reconstruction & fusion

Après l'inférence sur les différentes tuiles, on repositionne les détections sur l'image d'origine en reportant les boxes des tuiles dans le repère de l'image, c'est la reconstruction.

À la suite de ça, plusieurs détections redondantes peuvent apparaître pour un même objet dans les zones de recouvrement (*overlap*) entre tuiles. Afin de supprimer ces doublons, une méthode de fusion des boîtes de détection de type Non-Maximum Suppression (NMS) est appliquée.

La suppression est réalisée séparément pour chaque classe d'objet. Les boîtes proches sont d'abord triées par score de confiance décroissant et la boîte ayant le score le plus élevé est conservée comme référence, puis comparée aux autres boîtes de la même classe.

Les critères utilisés pour déterminer si 2 boîtes correspondent au même objet sont :

— **L'IoU minimal** (Intersection over Union) : si une boxe de même classe est positionnée sur la boxe de référence avec une surface minimale (ex : IoU min=0.3 = 30% de la surface de réf), il est alors considéré comme le même objet, cette boxe est supprimée.
— **La distance entre les centres des boîtes maximale** (Distance Threshold) : Si la distance du centre de la boxe avec le centre de la boxe de référence est inférieure au seuil (ex : dist_threshold=30 pixels), il est alors considéré comme le même objet, cette boxe est supprimée.

3. Réalisation du projet

Voici un résumé des différentes étapes effectuées pour réaliser le projet.

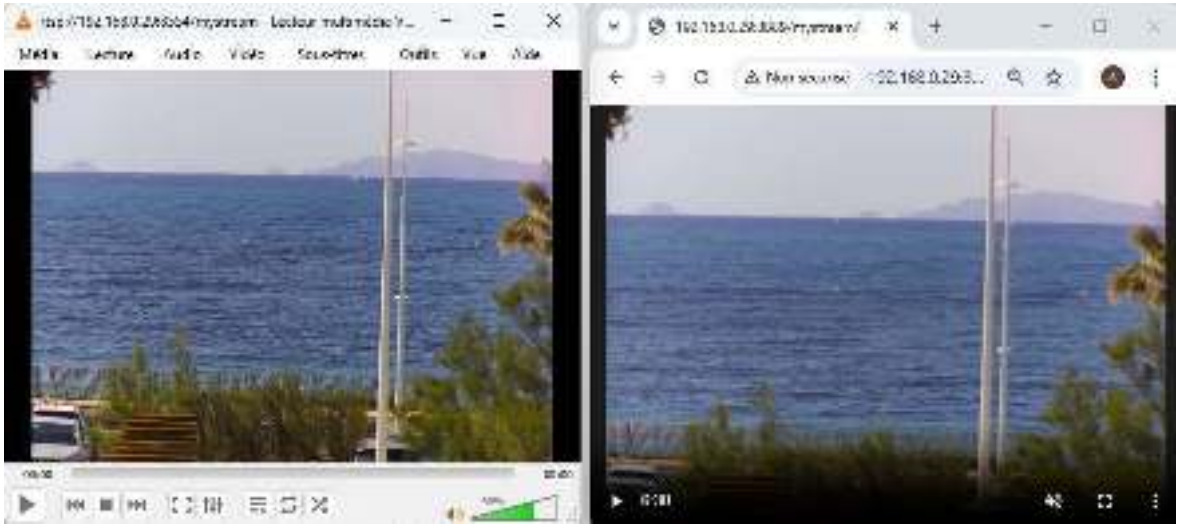
3.1. Etape 1 : Montage et flux vidéo

Après avoir installé l'*OS Raspberry Pi 64 bits lite* (système d'exploitation) sur la carte SSD, la carte Raspberry Pi est placée dans un boîtier auquel est fixée la caméra.

Montage caméra (Pi 5, ventilo, boitier, carte SSD)

Montage boitier fermé

Je crée ensuite le *service caméra* pour configurer la caméra puis le *service MediaMTX* pour visualiser le flux vidéo en local sur VLC media player (flux RTSP) ou sur une page web (flux WebRTC) :



rtsp ://IP :8554/mystream *http ://IP :8889/mystream/*

Nb : Pour obtenir un système fiable, chaque service est mis en **systemd**. Cela permet d'isoler les tâches à effectuer afin que si une d'elles bugue ou crashe, ça n'implique pas tout le système.

3.2. Etape 2 : Récupération d'un dataset

Afin de pouvoir créer une base de données conséquente, je développe une interface web accessible à distance et sécurisée afin de :

— Visualiser la caméra
— Contrôler la capture du dataset
— Planifier le démarrage/arrêt des services
— Surveiller l'état de ma caméra (charge CPU, température, volume de stockage)

Capture d'images et interface web

Afin d'éviter de provoquer des instabilités, la capture de frame ne se fait pas directement sur le flux caméra mais dans le buffer RAM.

Création du *service dataset* pour la capture d'images manuelle et aléatoire et du *service app* pour l'interface web.

Technologies choisies :

— API : FastAPI (pour héberger l'interface web)
— Avec messagerie Redis (pour mise à jour de l'interface → dataset/IA en temps réel)
— Avec Websocket (pour mise à jour dataset/IA → Interface en temps réel)

Accès à distance sécurisé

Après avoir créé mon compte sur Cloudflare.com et acheté un nom de domaine pour 7€/an, je crée le service Cloudflared qui me permet désormais d'accéder à l'interface à distance et de manière sécurisé (authentification par login).

Robotisation de la caméra

Activation numérique des moteurs DC reliés aux 2 roues de la plateforme pour créer une rotation droite/gauche de la caméra à distance.

Pour alimenter et contrôler les moteurs, une carte de puissance L298N est utilisée avec sa propre alimentation. Des fils Dupond sont soudés sur le GPIO de la Raspberry Pi.

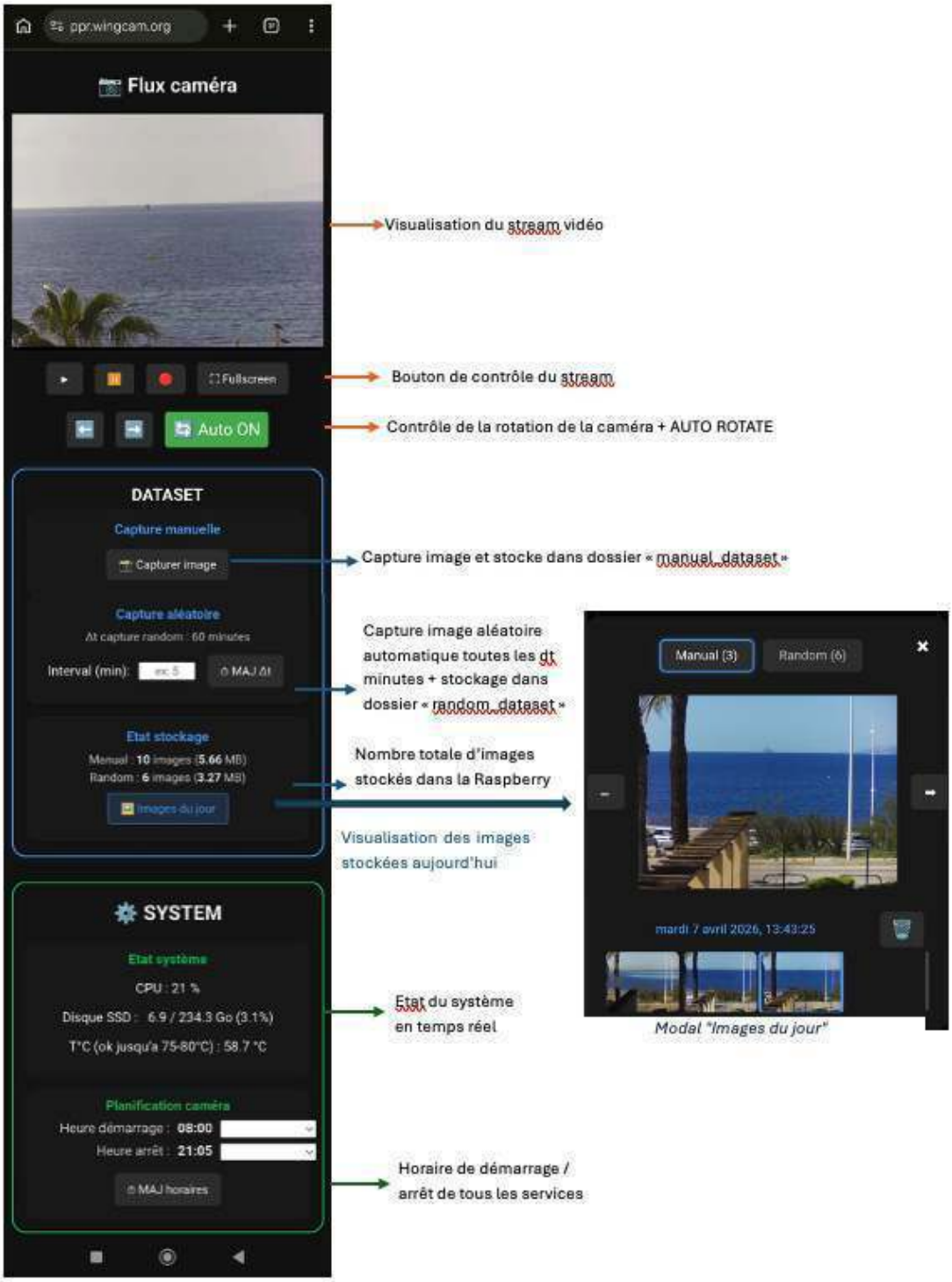
Cf ANNEXES (6.3. Câblage moteur DC)

Installation de la caméra derrière la fenêtre

Afin de ne pas gêner l'ouverture de la fenêtre, le socle est monté sur un système de bras rotatif permettant de le soulever manuellement pour le passage de la fenêtre.

Cf ANNEXES (6.4. Montage final)

Interface web obtenue :



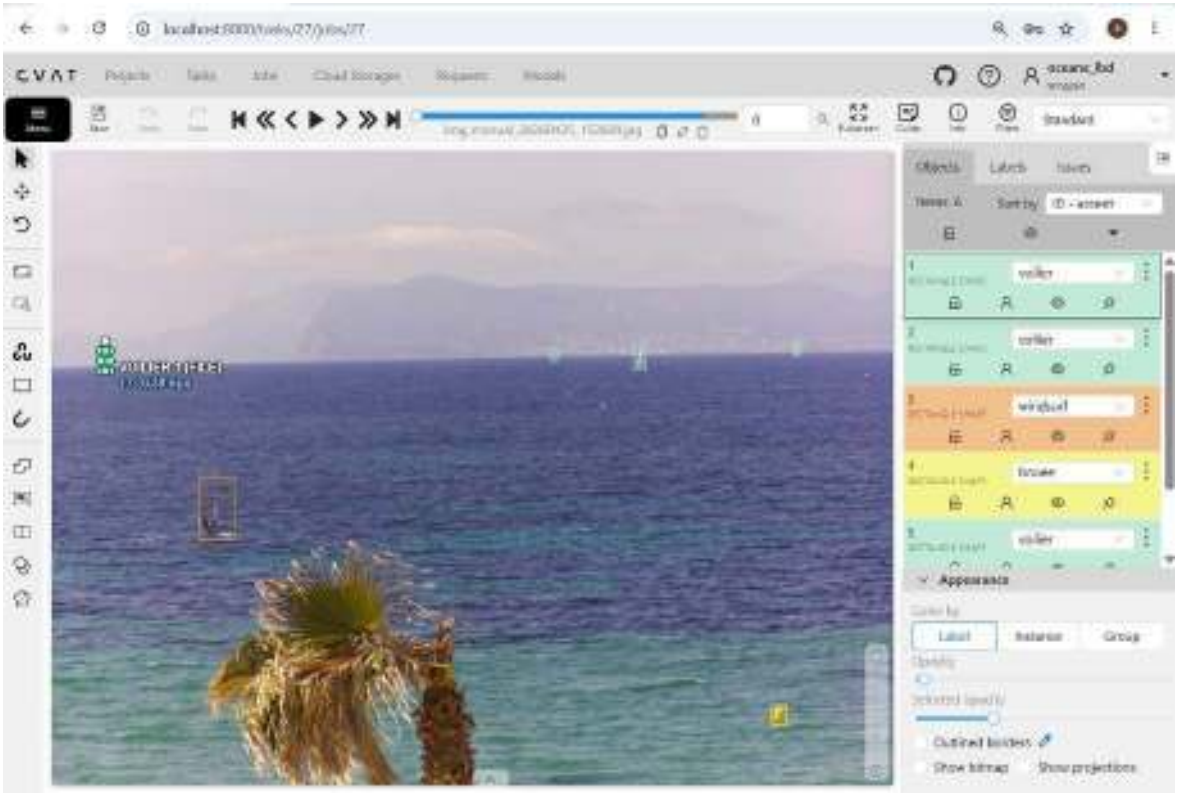
Modal "Images du jour"

Interface web obtenue

3.3. Etape 3 : Labellisation et création du modèle IA

1. **Labellisation**

Annotation des images capturées par la Pi caméra avec CVAT (Computer Vision Annotation Tool).



Interface d'annotation de CVAT

12 classes à annoter : wingfoil, windsurf, kite, paddle_surf, bateau_moteur, voilier, kayak, bouee, baigneur, oiseau, parawing, autre.

2. **Préparations des données**

Avant d'effectuer l'entraînement du modèle YOLO, il faut préparer les données :

- Répartition des images annotées en 3 ensembles d'images :
  - *Train* (images pour l'entraînement)
  - *Validation* (images utilisées pour évaluer les performances du modèle, en cours d'entraînement)
  - *Test* (images utilisées après entraînement, pour évaluer les performances)
- Découpage de chaque image en tuiles de 640x640 pixels (crop des 92 pixels supérieurs puis découpage des 3x2 tuiles avec 140 pixels de superposition)
- Conversion des bbox des images vers les coordonnées des tuiles

3. **Entraînement d'un modèle IA**

Entraînement d'un modèle IA par « deep learning » à partir d'un modèle pré-entraîné YOLO (transfert learning) avec nos tuiles labellisées.

Pour commencer j'utilise YOLOv8n puis je testerai d'autres modèles YOLO.

4. **Annotation dataset test**

L'annotation d'images ne se limite pas à l'inférence mais comprend :

— *Pre-process* : découpage des images en 3x2 tuiles 640x640 px
— Inférence sur les 6 tuiles (via mon modèle best.pt)
— *Post-process* : reconstruction + fusion NMS des boxes sur images d'origine

Pour comprendre et bien régler les paramètres relatifs à la fusion des bounding boxes je trace sur les images les bordures des tuiles et les boxes de détection avant et après la fusion :

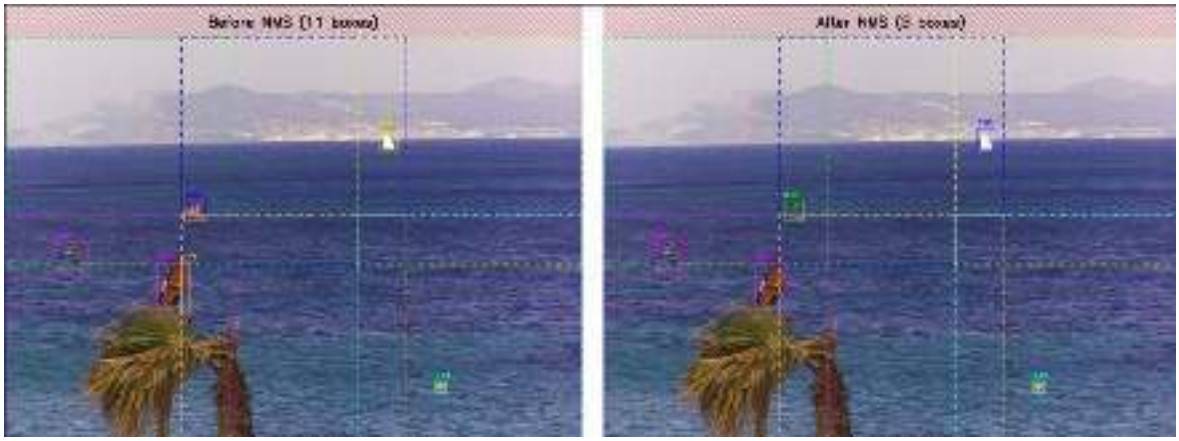


Image test avec bordures des tuiles et boxes de détections avant/après fusion NMS

5. **Labellisation assistée**

Le modèle IA entraîné va être utilisé pour aider à l'annotation.

Dans CVAT, il est possible d'importer les labels pour pré-annoter ce qui permet un gain de temps considérable pour la labellisation (il ne reste alors qu'à corriger les erreurs et compléter les oublis).

6. **Conversion du modèle pour le NPU Hailo**

Conversion d'un modèle YOLO au format *.pt* (PyTorch) vers le format .hef (*Hailo Export Format*) adapté pour le NPU Hailo.

3.4. Etape 4 : Service IA

Inférence sur NPU Hailo

Installation du runtime Hailo sur la Pi pour pouvoir effectuer l'inférence sur le NPU Hailo.

Service IA

Pipeline du Service IA :

1. Capture d'une frame (1640x1232px) dans le buffer RAM toutes les 5 minutes (modifiable dans l'interface, temps min = 2 secondes).

2. Préparation de l'image pour l'inférence *(découpage en 3x2 en tuiles 640x640px, même méthode que pour l'entraînement)*
3. Inférence du modèle best.hef sur le NPU Hailo-8 (une tuile après l'autre)
4. Si détection de wingfoil sur une des tuiles, confiance>0.7 :
   — Stockage de l'image d'origine dans dossier local « wing_dataset »
   — Envoi d'une notification push via *Telegram* (application de messagerie gratuite)

   Si détection d'object rare (baigneur, oiseaux, paddle_surf, kite) confiance>0.6 :

— Stockage de l'image d'origine dans dossier local « rareObject_dataset »

Les images avec objet rare vont permettre de nourrir le dataset d'entrainement avec des objets de faible occurrence.

Selon les logs :

— Temps d'inférence moyen sur 1 tuile = 105ms
— Temps du cycle complet de traitement d'une frame du service IA moyen = 680ms

Interface web - partie IA

La partie IA de l'interface web contient :

— Choix du temps entre 2 captures de détection
— Une galerie d'images pour visualiser les images du jour contenant des wingfoils.

Apparence du bloc IA de l'interface web

Et lorsqu'il y aura de nombreuse détection de wingfoils, il suffira d'augmenter l'intervalle entre 2 inférences dans l'interface pour limiter les notifications.

Cf ANNEXES (6.4. Interface web finale)

Notification *Telegram*

L'utilisation d'un bot Telegram permet d'envoyer directement la photographie ayant déclenché la détection, accompagnée d'un court message informatif. **L'utilisateur peut ainsi être averti de condition favorable à la pratique du wingfoil, sans avoir à surveiller la caméra.**

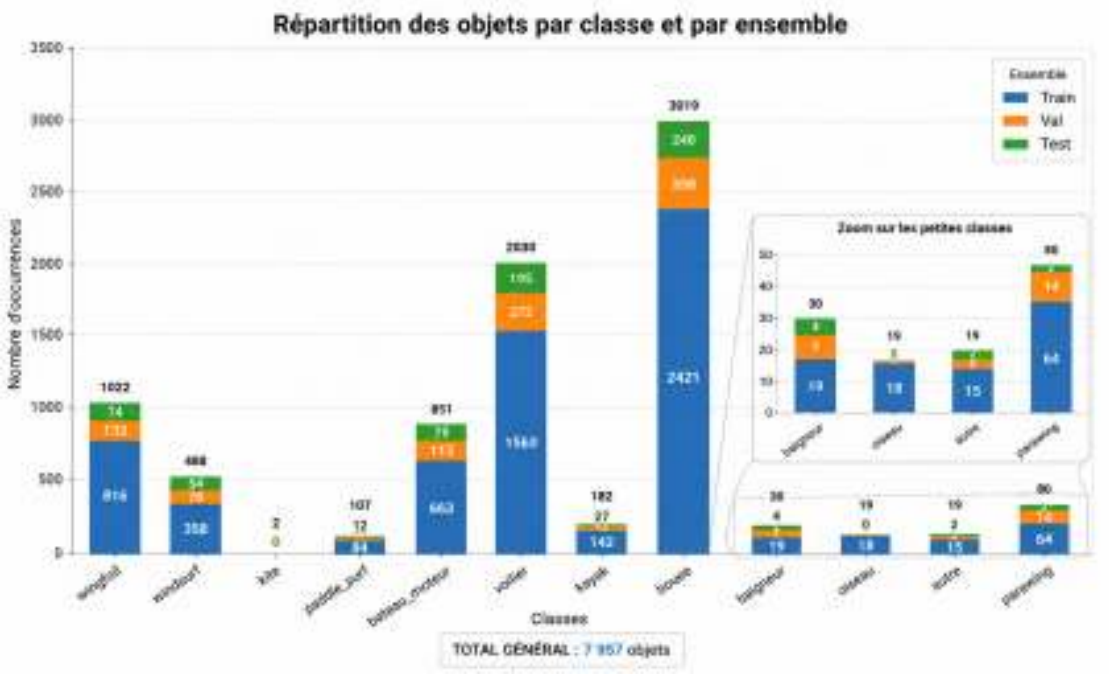
Message Telegram après détection de wingfoil

4. Résultats obtenus

4.1. Base de données

A ce jour, le 18 juin 2026, la base de données contient 1015 images réparties ainsi :

— Training dataset : 809 images (80 %)
— Validation dataset : 131 images (13 %)
— Test dataset : 75 images (7 %)



Répartition des objets par classe et par ensemble

La disproportion du nombre d'objets par classes sera rétablie dans le temps grâce aux captures automatique d'images d'objets rare.

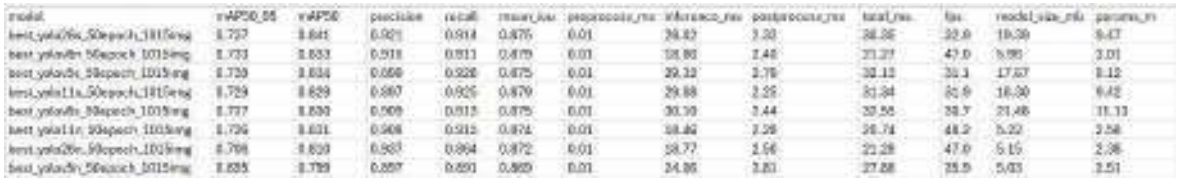
4.2. Modèle de détection

Comparaison de plusieurs modèles :

Pour comparer des modèles, il faut regarder :

— La précision globale (mAP50-95)
— La capacité à éviter les erreurs (precision),
— La capacité à détecter tous les objets (recall),
— Le temps d'inférence,
— La taille du modèle .pt (ou nombre de paramètres),
— Les confusions entre classes,
— Des exemples visuels

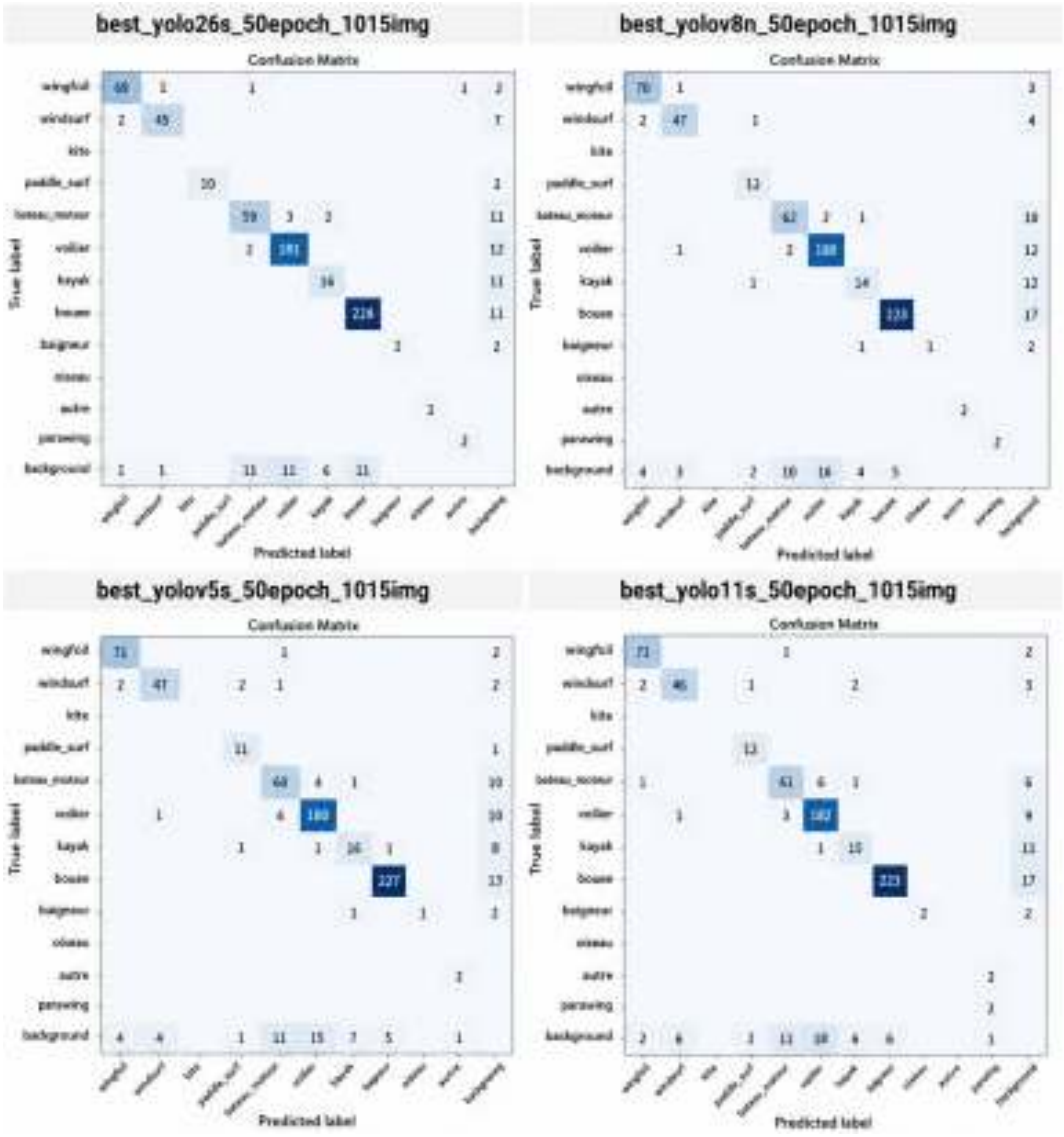
Métriques mesurées post reconstruction/fusion pour chaque modèle (y compris le temps moyen d'inférence et la taille des modèles) :



Comparaison de métriques post-reconstruction/NMS (trié selon valeur mAP50-95)

— *epoch* : Le nombre de fois que le modèle à parcouru tout le dataset
— *mAP50* : Mesure à quel point le modèle détecte correctement les objets, avec peu d'erreurs, en acceptant des boxes ayant au moins 50% de recouvrement.
— *mAP50-95* : Idem mais en acceptant les boxes avec d'avantage de recouvrement.
— *precision* : Ratio entre nombre de boxes corrects et nombre de boxe détectés.
— *recall* : Ratio entre nombre de boxes corrects et nombre de boxe à détecter.
— *mean_iou* : moyenne du taux de recouvrement sur la boxe labellisée.
— *preprocess_ms* : temps (en ms) pour effectuer le preprocessing (tiling).
— *inference_ms* : temps (en ms) pour effectuer les inférences sur tuiles.
— *post*process_ms : temps (en ms) pour effectuer le postprocessing (reconstruction).
— *total_ms* : temps (en ms) pour effectuer le cycle complet du traitement IA (preprocessing + inférences + postprocessing).
— *fps_ms* : nombre de cycle complet pouvant être effectué en 1 seconde.
— *model_size_mb* : taille (en MB) du modèle.
— *params_m* : nombre de paramètre (en Million) que contient le modèle.

Les matrices de confusion des 4 meilleurs modèles :



Comparaison matrices de confusion des 4 meilleurs modèles

Visualisation des annotations des 4 meilleurs modèles sur une image :

Une des images test avec détection des 4 meilleurs modèles

Choix du modèle à intégrer sur la Pi :

Le modèle ayant le meilleur compromis performances / taille pour notre base de données est le modèle pré-entraîné avec YOLOv8n (soit *best_yolov8n_50epochs_1015img.pt*).

Je le converti au format Hailo (.hef) pour effectuer l'inférence sur la Pi caméra.

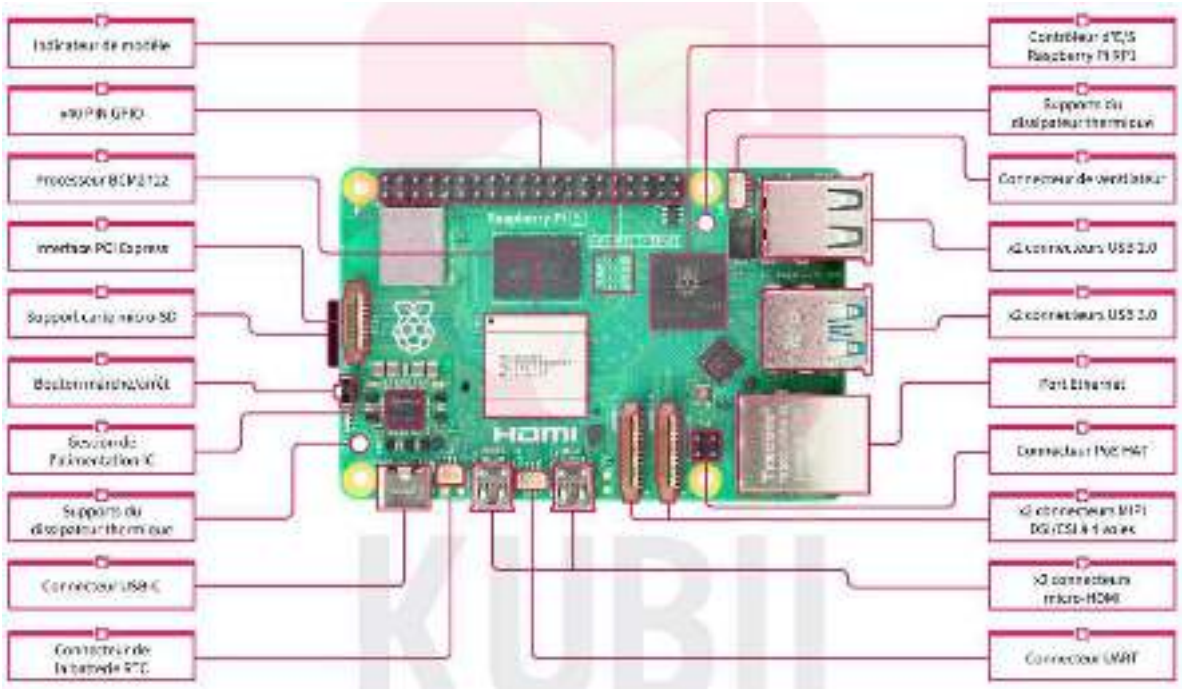
5. Conclusion

Le projet m'a permis de découvrir la programmation sur Raspberry PI, la création d'une interface web, la communication en réseau local et distant, et bien sûr, la création d'un modèle IA de type Deep Learning, via un modèle pré-entrainé.

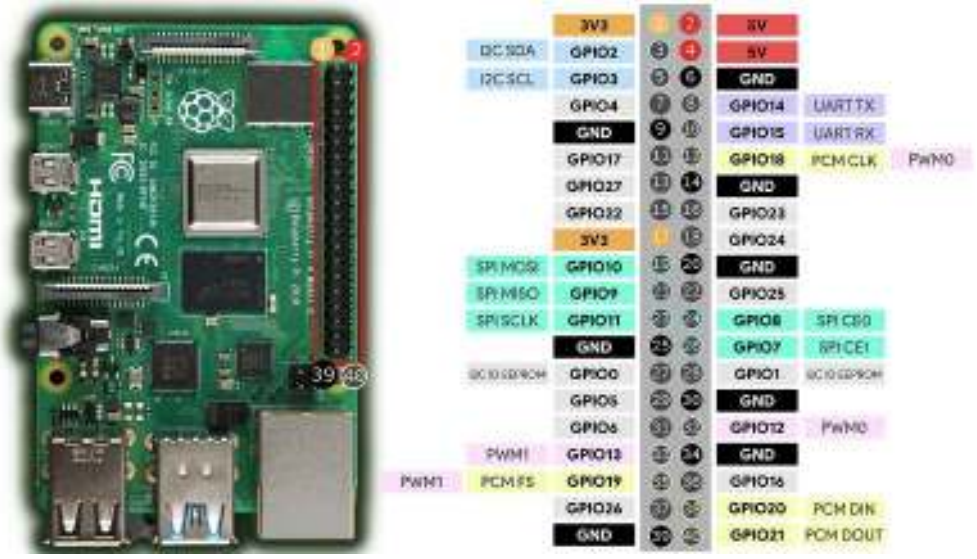
J'ai aussi réalisé l'importance de la base de données pour l'obtention d'un modèle de qualité. Elle doit être réalisée avec soin et contenir des images variées (météo, objets, luminosité, environnement). Dans mon cas, même si la plupart des images ont été prises du même endroit, c'est le fait de pourvoir régler l'orientation de la caméra qui a pu diversifier mon dataset.

6. ANNEXES

6.1. Carte Raspberry pi

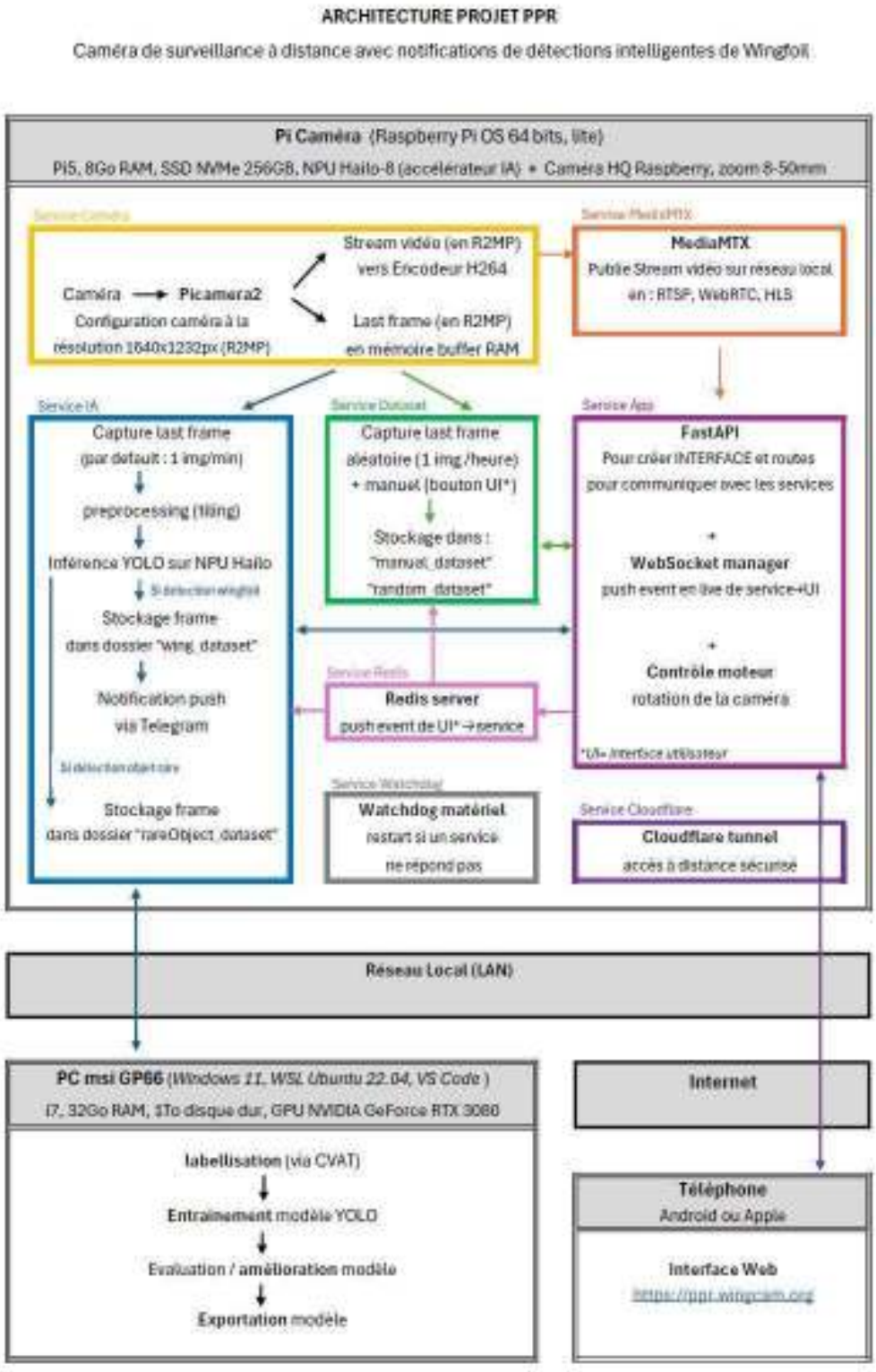


Raspberry pi 5



Broches GPIO Raspberry pi 5

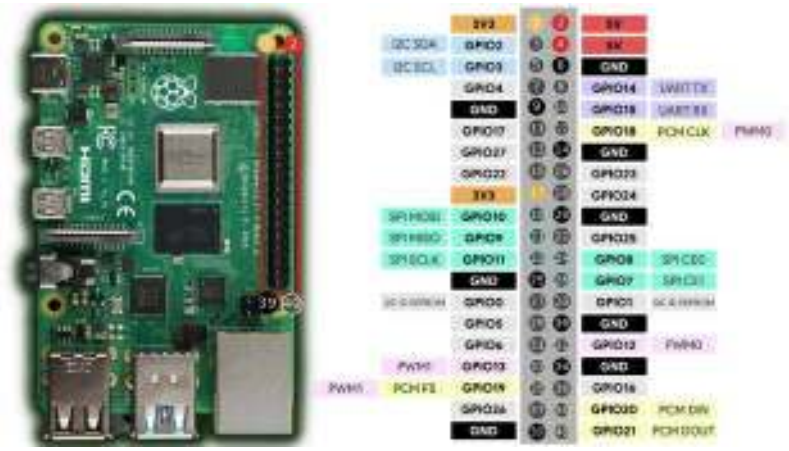
6.2. Schématisation de l'architecture globale



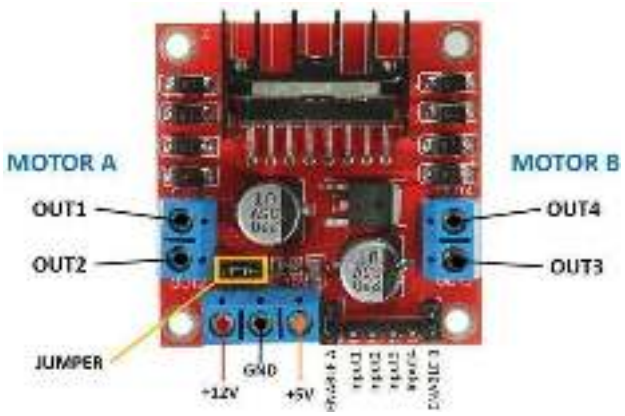
Schématisation de l'architecture globale

6.3. Câblage moteur DC

Câblage Moteurs DC → L298N → GPIO Raspberry Pi 5 :



Carte de puissance L298N



Câblage GPIO Raspberry Pi 5

Moteur A :

— OUT1 (jaune) + OUT2 (vert) → moteur A
— ENA → GPIO 19 (fil gris) (PWM1 matériel)
— IN1 → GPIO 6 (fil bleue)
— IN2 → GPIO 5 (fil violet)

Moteur B :

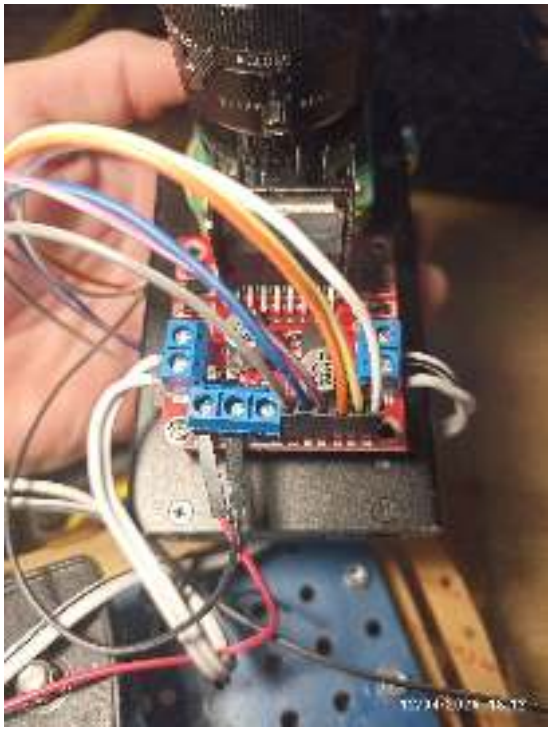
— OUT3 (jaune) + OUT4 (vert) → moteur B
— IN3 → GPIO 23 (fil orange)
— IN4 → GPIO 24 (fil jaune)
— ENB → GPIO 18 (fil blanc) (PWM0 matériel)

Masse et alimentation :

Carte de puissance L298N

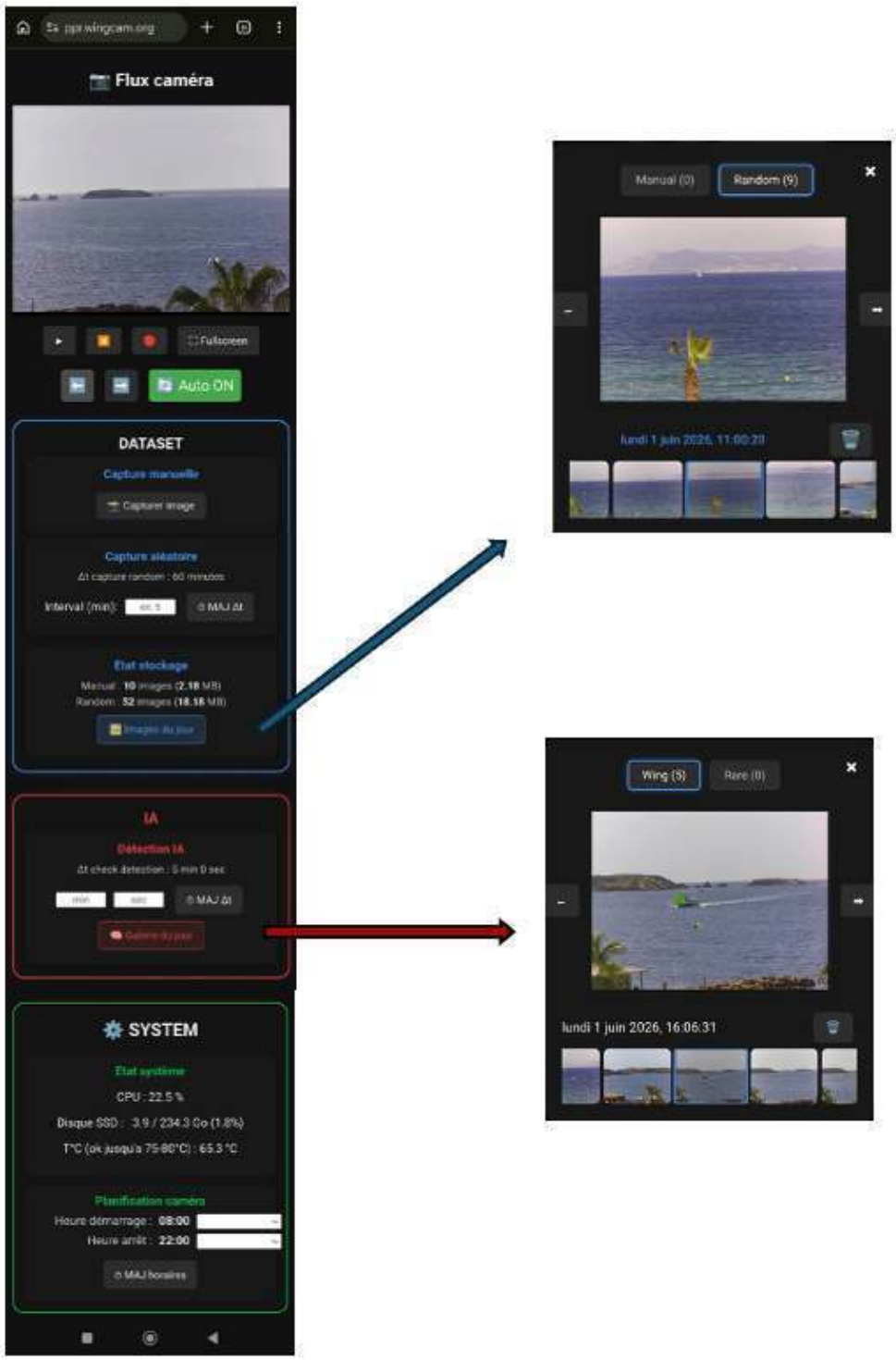
— GND L298N → GND GPIO pin 6 (fil noir)
— 12V L298N (fil rouge) + GND GPIO pin 39 (fil noir) → alim externe 12V

Soudure sur rallonge GPIO



Carte de puissance

6.4. Interface web finale



Interface web finale

6.5. Montage final

Fixation murale au-dessus de la fenêtre

Vue depuis la fenêtre avec la caméra installée

Bras articulé permettant le passage de la fenêtre

7. BIBLIOGRAPHIE

Documentations techniques :

Raspberry Pi Documentation.

https://www.raspberrypi.com/documentation/

Picamera2 Manual, documentation officielle de la librairie Picamera2.

https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf

Documentation officielle Raspberry Pi AI HAT+ pour l'accélérateur IA Hailo-8

https://www.raspberrypi.com/documentation/accessories/ai-hat-plus.html

Documentation officielle Hailo pour l'utilisation des accélérateurs IA Hailo.

https://hailo.ai/developer-zone/documentation/

Documentation Ultralytics, modèles YOLO pour la détection d'objets.

https://docs.ultralytics.com/

Documentation CVAT (Computer Vision Annotation Tool) pour l'annotation.

https://docs.cvat.ai/

Documentation MediaMTX pour le serveur de streaming

https://github.com/bluenviron/mediamtx

Documentation Cloudflare Tunnel pour l'accès distant sécurisé.

https://developers.cloudflare.com/cloudflare-one/connections/connect-networks/

Documentation Telegram Bot API pour l'envoi de notifications via un bot Telegram.

https://core.telegram.org/bots/api

Ressources communautaires et tutoriels :

Stack Overflow, échanges techniques autour de Python, FastAPI, Raspberry Pi, YOLO et du traitement d'images.

https://stackoverflow.com/

Reddit, discussions autour de Raspberry Pi, YOLO, Hailo et projets embarqués.

https://www.reddit.com/

GitHub, exemples de projets open source liés à YOLO et Raspberry Pi.

https://github.com/