Guide pour la manipulation de caméras (Perception)

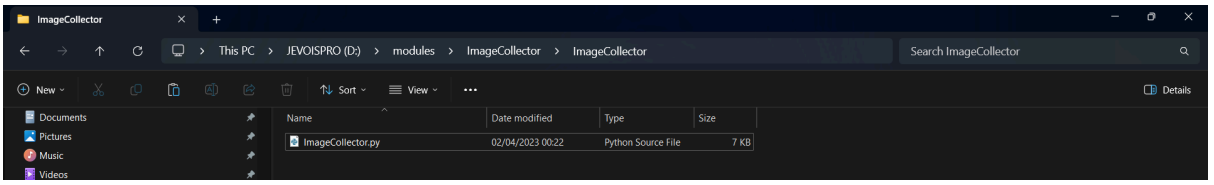
➔ Étiquetage d'image:

Prérequis : Clonez le repo https://github.com/Topkif/Object-Detection-and-Datasets

1) **Collecte des données d'images avec le robot**

   Depuis `Object-Detection-and-Datasets/JeVoisresources/ImageCollector/ImageCollector`,

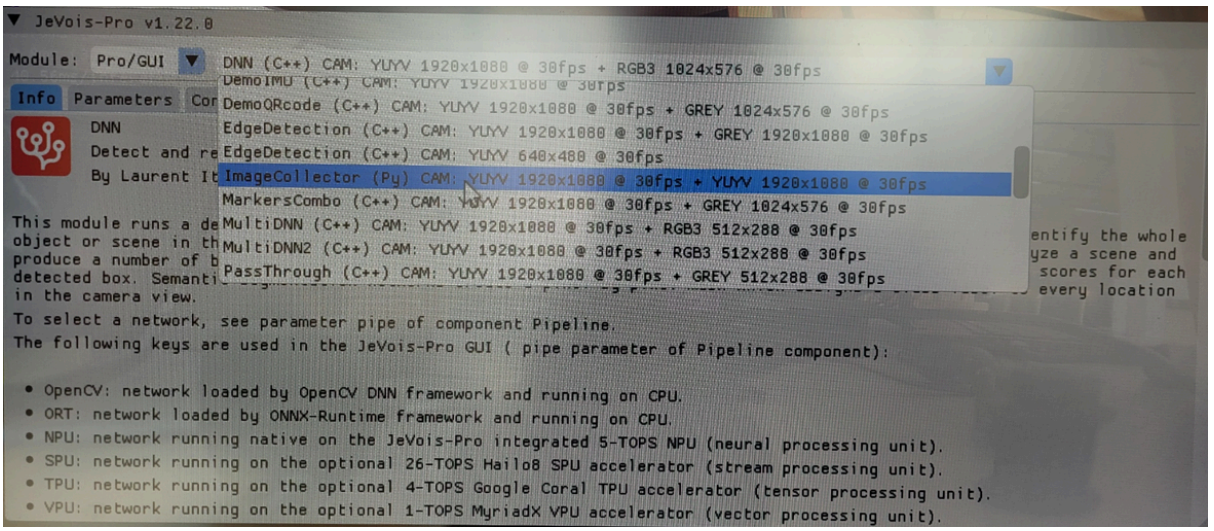
   Ajoutez le script **ImageCollector.py** dans sur la carte SD dans le répertoire : `JEVOISPRO/modules/ImageCollecor`



   Ensuite, ajoutez la ligne suivante à la fin du fichier **videomappings.cfg**, situé dans : `JEVOISPRO/config` :

```
JVUI 0 0 0 CropScale=YUYV@1920x1080:YUYV 1920 1080 30 ImageCollector ImageCollector
```

   Branchez un dispositif de stockage USB sur la caméra, puis lancez le script **ImagesCollector**

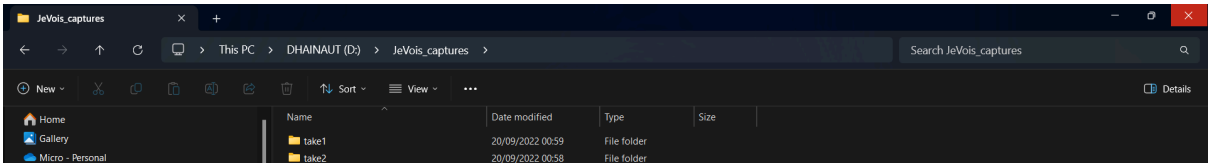


Ce script permet de capturer une image toutes les secondes.

Pour constituer un jeu de données d'images, déplacez le robot dans toutes les directions afin de capturer des images sous différents angles et dans diverses situations. Ces images seront ensuite utilisées pour l'entraînement du modèle d'IA.

Pour arrêter le script, lancez n'importe quel autre module, attendez un instant que le buffer d'image soit vidé dans la carte SD, puis débranchez votre dispositif de stockage.

2) **Récupération des images depuis la carte SD**
Une fois les images capturées, récupérez les dans le répertoire `JeVois_captures/take…`



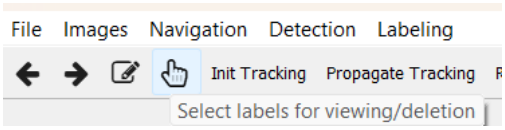
3) **Téléchargement des outils d'étiquetage**
Pour l'étiquetage, il est recommandé d'utiliser le logiciel **DeepLabel** dans `Object-Detection-and-Datasets/DeepLabel Software`
Il est également utile d'installer **GitHub Desktop** pour faciliter le travail collaboratif sur l'étiquetage.
4) **Étapes de l'étiquetage avec DeepLabel**
Voici les étapes à suivre pour étiqueter vos images avec **DeepLabel** :

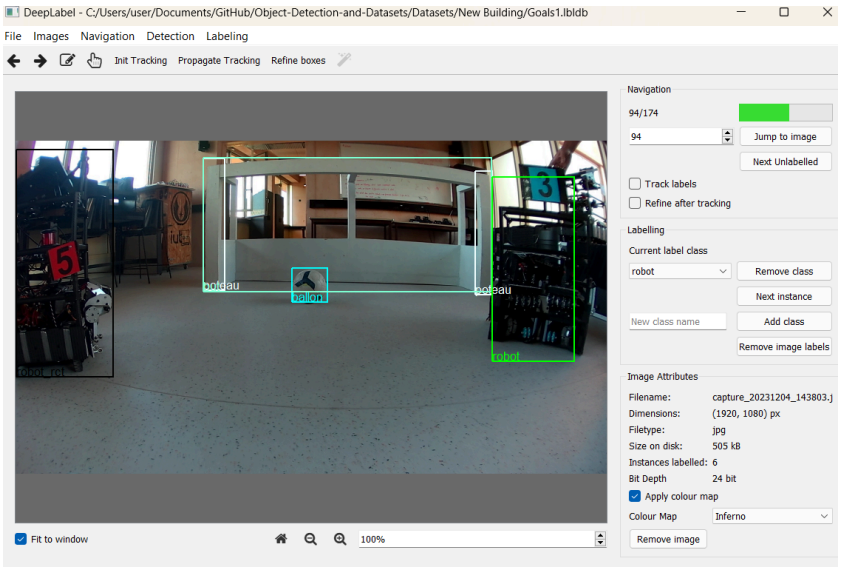
- **Créez un projet** : Créez un projet puis ajoutez les classes. Pour la Robocup, ajoutez celles contenues dans `13classes.names`. Puis importez les images.
- **Ou ouvrez un projet** : Sélectionnez le fichier projet avec l'extension `.lbldb`.

- **Vérification des bounding boxes (Si importation de labels)** : Pour chaque image, vous pouvez rencontrer trois cas :
    - Des encadrements déjà créés mais incorrects.
    - Des encadrements manquants.
    - Des encadrements corrects.

    Si l'encadrement est mal fait, il suffit de le supprimer et de créer un nouveau cadre correspondant à la bonne classe d'objet.

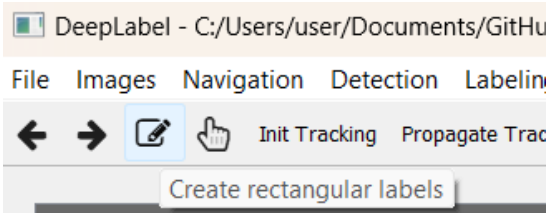
- **Suppression d'un cadre** : Sélectionnez le cadre incorrect et supprimez-le.

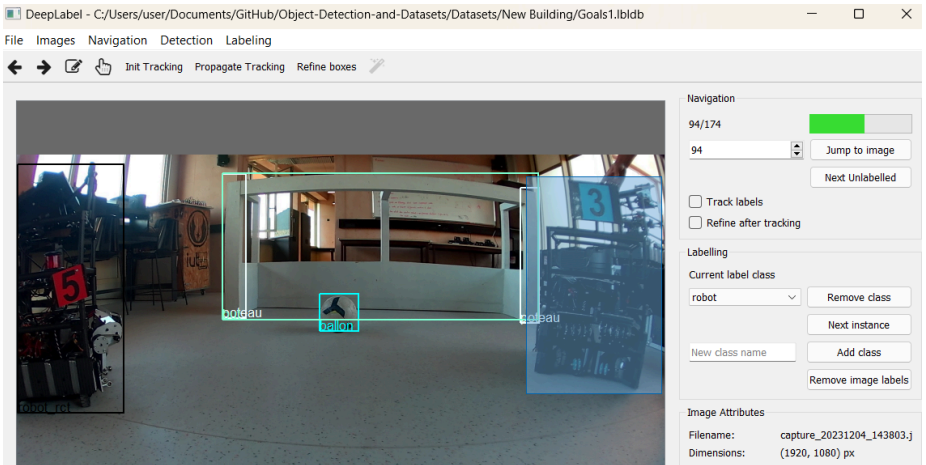


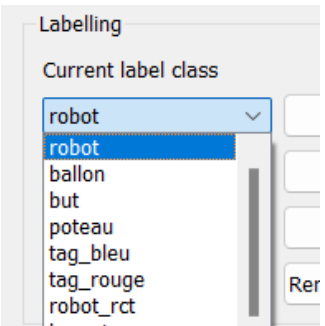


l'image sélectionnée est en vert

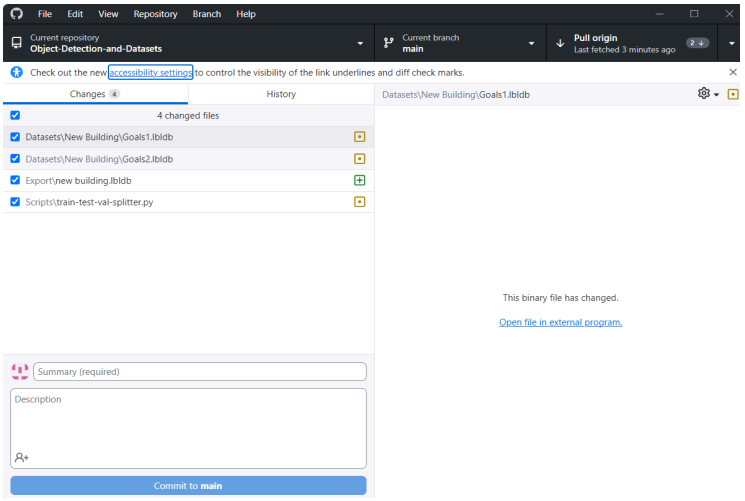
- **Ajout ou recadrage** : Dessinez un nouveau cadre autour de l'objet avec la classe appropriée.







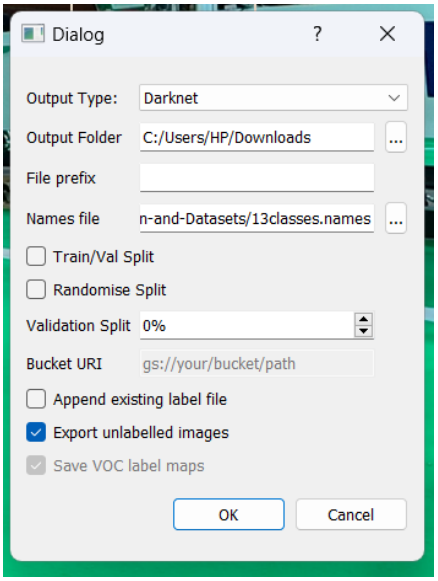
- **Archivage et versionnement sur GitHub**
  Une fois l'étiquetage terminé, vous devez archiver les changements localement et les synchroniser avec le serveur GitHub.



- **Commit local** : Sur GitHub Desktop, ajoutez un nom à votre commit dans la zone au-dessus de "Description". Cliquez ensuite sur **Commit to main** pour sauvegarder les modifications localement.
- **Push vers le serveur** : À ce stade, les modifications sont encore uniquement sur votre machine. Pour les envoyer au serveur et permettre à d'autres utilisateurs d'y accéder, cliquez sur **Push origin**.
- **Vérification de l'historique** : Allez dans l'historique du projet (onglet "History") pour voir votre commit en haut de la liste. Ce commit devrait être le plus récent.

Export des annotations

Dans File>Export labels choisissez ces paramètres :



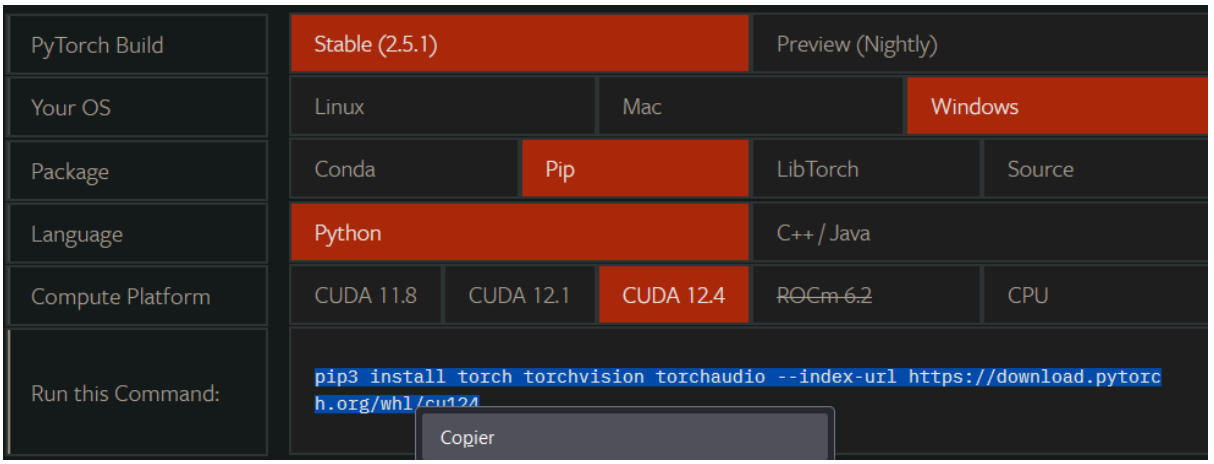
Puis lancez le script
`Object-Detection-and-Dataset/Scripts/train-test-val-splitter.py`
Choisissez les dossiers source et de destination en ayant choisi une proportion pour les datasets train, val, et test :

```
# Define the proportions for training, validation, and testing
train_prop = 0.7
val_prop = 0.3
test_prop = 0
```

On peut laisser la proportion d'images du dataset de test à 0 car nous évaluons les performances des modèles directement sur les robots et nous voulons un maximum d'images pour l'entraînement.

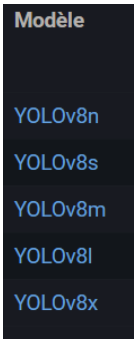
➔ Entrainement du modèle avec les images collectées:

Intallez pytorch depuis https://pytorch.org/get-started/locally/



Pour entraîner de yolov8 à yolov11 installez le CLI de ultralitics dans le même environnement que pytorch :

`pip install ultralytics`
Puis téléchargez le modèle de votre choix depuis:
https://docs.ultralytics.com/fr/models/ par exemple yolov8s.pt



Créez le fichier dataset.yaml pointant vers vos données labellisées et organisées.

```yaml
train: ../Dataset/images/train
val: ../Dataset/images/val
test: ../Dataset/images/test

# number of classes
nc: 13

names:
 0 : robot
 1 : ballon
 2 : but
...
```

Ensuite lancez la commande :

```
yolo task=detect mode=train model=yolov8s.pt data=dataset.yaml
epochs=200 imgsz=640 plots=True batch=32 workers=16 device=0
```

Ou utilisez des données de COCO :

```
yolo task=detect mode=train model=yolov8s.pt data=coco8.yaml
epochs=200 imgsz=640 plots=True batch=32 workers=16 device=0
```

Lien utile pour comprendre les paramètres :
https://docs.ultralytics.com/fr/modes/train/#train-settings

Les poids à récupérer en sortie sont enregistrés dans
`./runs/detect/train/weights/best.pt`

Pour convertir pour le modèle en .hef (Hailo) on doit convertir de .pt en .onnx:

```
yolo export model=best.pt format=onnx imgsz=640,640
```

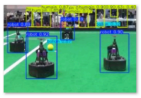
Test de l'entraînement avec les poids (non quantizés)

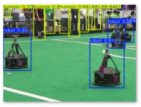
Lancez la commande :

```
yolo task=detect mode=predict model=best.pt
source="Chemin/Vers/Dossier/Images" imgsz=640 device=0 plots=True
```

Dans le dossier `./runs/detect/predict` vous trouverez les images avec les prédictions. Vérifiez que les bounding boxs et les classes soient correctes. Par exemple :

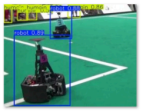

2f0df.jpg


0006.jpg

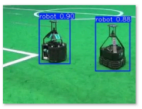

0009.jpg


0011.jpg

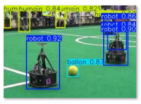

0015.jpg

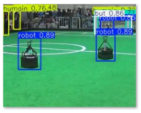

0022.jpg


0023.jpg

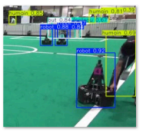

0035.jpg


0037.jpg

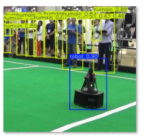

0038.jpg

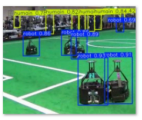

0045.jpg


0050.jpg


0067.jpg


0077.jpg


0081.jpg


0084.jpg

➔ Conversion de fichier en .hef:

Tuto : http://jevois.org/doc/UserDNNspu.html

Prérequis:

- Télécharger ubuntu 20.04 ou plus récent.
- **Version Hailo sur JeVoisPro :**
  La version de Hailo utilisée sur JeVoisPro est la **Hailo 8.13**. Pour vérifier cela sur la caméra, il vous suffit de taper la commande suivante dans la console de la caméra:
  `!dpkg --list | grep hailo`

  Ensuite, il faudra télécharger cette même version (Hailo 8.13) sur Ubuntu pour garantir la compatibilité.

1-Installation et configuration du logicielle Hailo

1. **Demandez un compte développeur** sur hailo.ai, connectez-vous et rendez-vous sur https://hailo.ai/developer-zone/sw-downloads/.
2. **Téléchargez les éléments suivants** :
   - **Hailo Software Suite - Docker**.
   - **HailoRT – Paquet Ubuntu (deb) pour amd64**.
   - **Pilote PCIe HailoRT – Paquet Ubuntu (deb)**.

   Installez les pilotes PCIe et la bibliothèque runtime. Bien que cette étape ne soit pas strictement nécessaire, elle permet de réduire les avertissements ultérieurs. Lors de l'installation, acceptez l'option DKMS (pour maintenir le pilote à jour avec les mises à jour du noyau). Si l'installation échoue, vous devrez peut-être installer des dépendances comme `dkms`, `kernel-headers`, etc. :
   sudo dpkg -i ~/Downloads/hailort-pcie-driver_4.19.0_all.deb
   sudo dpkg -i ~/Downloads/hailort_4.19.0_amd64.deb

3. **Installez Docker** (si ce n'est pas déjà fait) :

```
sudo apt install docker.io

sudo usermod -aG docker ${USER}  # Donne les droits Docker à
l'utilisateur ; redémarrez pour appliquer.
```

4. **Installez le NVIDIA Container Toolkit** (si vous avez un GPU NVIDIA). Cela permet d'accélérer l'optimisation du modèle dans Docker. Suivez les instructions les plus récentes. Voici un exemple :

```
curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | \
```

```
sudo gpg --dearmor -o
/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg \
&& curl -s -L
https://nvidia.github.io/libnvidia-container/stable/deb/nvidia-conta
iner-toolkit.list | \
sed 's#deb https://#deb
[signed-by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg]
https://#g' | \
sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list

sudo apt update
sudo apt-get install -y nvidia-container-toolkit
sudo systemctl restart docker
```

5. **Décompressez la Hailo Software Suite** que vous avez téléchargée :

```
mkdir hailodev

cd hailodev
unzip ~/Downloads/hailo_ai_sw_suite_2024-10_docker.zip
./hailo_ai_sw_suite_docker_run.sh
```

6. **Interface du conteneur Docker** :
   Une fois le conteneur configuré, vous verrez le message suivant :

```
   Welcome to Hailo Software Suite Container

To list available commands, please type:
-------------------------------------------------
HailoRT:                hailortcli -h
Dataflow Compiler:      hailo -h
Hailo Model Zoo:        hailomz -h
TAPPAS:                 hailo_run_app -h
-------------------------------------------------
(hailo_virtualenv) hailo@mypc:/local/workspace$
```

7. **Manipulation des modèles** :

   Il est possible maintenant d'utiliser le modèle entraîné en suivant la documentation Hailo.

Remarques:

Pour quitter et reprendre le conteneur :

Tapez `exit`.

Reprendre plus tard :

```
./hailo_ai_sw_suite_docker_run.sh --resume
```

Recommencer avec un nouveau conteneur :
`./hailo_ai_sw_suite_docker_run.sh --override`

Transfert de fichiers entre l'hôte et le conteneur Docker :
À l'intérieur du conteneur, le dossier `/local/shared_with_docker/` est partagé avec le répertoire `shared_with_docker/` sur l'hôte. C'est la méthode la plus simple pour transférer des fichiers vers/depuis le conteneur. Sinon :

Lister les conteneurs Docker :
`sudo docker container ls -a`
Copier un fichier de l'hôte vers le conteneur :
`sudo docker cp myfile <ID_du_conteneur>:/local/workspace/`
Copier un fichier du conteneur vers l'hôte :
`sudo docker cp <ID_du_conteneur>:/local/workspace/myfile .`

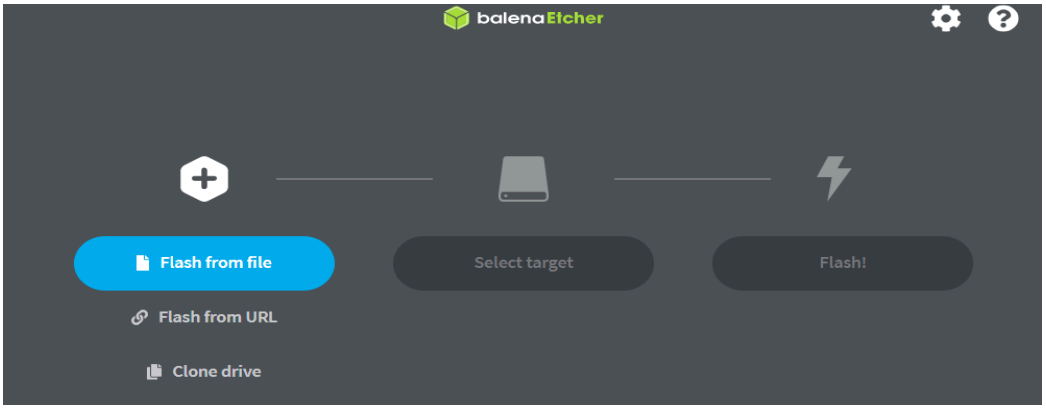
Documentation complémentaire :
Consultez la documentation complète sur Hailo Developer Zone. (Connexion requise).

➔ Insertion du nouveaux model dans les cartes SD des caméras des robots:

- Copier le fichier .hef du modèle entraîné dans JEVOISPRO ⇒ share⇒dnn⇒custom.
- Modifier le fichier .yaml en pour qu'il pointe sur le fichier ajouté.

➔ Flasher les caméras:

- Installer : https://etcher.balena.io/
- Télécharger la dernière image ici
- Insérer le fichier de l'image en cliquant sur: Flash from file



pour avoir les listes sur la caméra: console =>type je vois commande here=>listmappings => rechercher le num de la ligne je vois DNN C++ 30fps , une fois le num retrouver on l'ajoute dans le fichier initscrip du DNN (jevois prp/je vois config/dnn):(25 ici)

```
# JeVois initialization script
#
# This script is run upon statup of the JeVois main engine. You can
here specify commands (like you would type them to
# the JeVois command-line interface) to execute upon startup, even
before a module is loaded.

# Example: load the SaveVideo with no USB out module (check its mapping
index, it was 0 at the time of writing this),
# start streaming, and start saving:
setmapping 25
#numero de la ligne
setpar serout None
setpar serstyle Normal
#streamon
#start
```

proxy network sur linux : settings=>network proxy=>manuel=> HTTP proxy = proxy.univ-tln.fr / 3128 -+

le cloud : https://valetteonline.net/s/dnn?path=%2F