Rapport de projet :

Projet KYUH

Partie Personnel

Kilian FERREIRA

BTS SNIR
Lycée Costebelle

SOMMAIRE

Tables des matières

Sommaire ……..… 2
Carnet de bord ……..... 3
Introduction ……..…. 4
Fonctionnalité du système ..……..….4
Contrainte logiciel ……..... 4
Choix de composants ……...……..……..... 4
La caméra ……..…. 5
Diagramme de cas d'utilisation ……..... 6
Diagramme de séquence……..... 7
Diagramme de bloc ……...……..... 8
Algorithme……10
Création d'un model avec Edge Impulse …… 11
Physique……... 15
Conclusion……....19
Annexes ...…….…….... 20

Carnet de bord

Semaine 1 - 05 au 08 janvier	→ Analyse du dossier. → Répartitions des tâches avec le groupe de projet.
Semaine 2 - 11 au 15 janvier	→ Début de la réalisation des diagrammes de bloc et de séquence. → Listing du matériel pour le groupe. → Début du cahier des charges.
Semaine 3 - 18 au 22 janvier	→ Fin des diagrammes de bloc et de séquence. → Fin du cahier des charges. → Préparations de la revue 1.
Semaine 4 - 25 au 29 janvier	→ Fin de la revue 1. → Début du devis.
Semaine 5 - 01 au 05 février	→ Fin du devis. → Début des recherches sur OpenMV IDE.
Semaine 6 - 08 au 12 février	→ Création d'un model OpenMV.
Semaine 7 - 15 au 19 février	→ Préparation de la revue 2. → Fin de la revue 2. → Rapport de conception.
Semaine 8 - 08 au 12 mars	→ Connection avec la base de données créée par Hugo Arbona.
Semaine 9 - 15 au 19 mars	→ Début de recherche pour le stream sur l'ihm.
Semaine 10 - 22 au 26 mars	→ Concrétisation du stream sur l'ihm
Semaine 11 - 29 mars au 02 avril	→ Reprise du model avec Edge Impulse
Semaine 12 - 06 au 09 avril	→ Edition des rapports de projet en groupe et individuel.

INTRODUCTION DETECTION FRELON

Mon rôle ici sera de détecter un frelon sur l'appât. Pour ce faire, j'utiliserais une caméra OpenMV H7 Plus. Cette caméra est programmable en micro-python, et permet de faire la détection de ce qu'elle voit à l'aide d'un model créé avec une base de plusieurs images catégorisée en plusieurs labels.

Une fois la détection faite, la caméra prendra une photo du frelon, et l'enverra par liaison WiFi au RaspberryPI qui s'occupera de l'ajouter à la base de données, et de faire le nécessaire pour la protection des abeilles.

Fonctionnalité du système :

Analyse continue de l'appât à l'aide d'une caméra OpenMV H7 Plus.
Si un frelon est détecté alors envoie d'une photo du frelon au RaspberryPi

Contrainte logiciel :

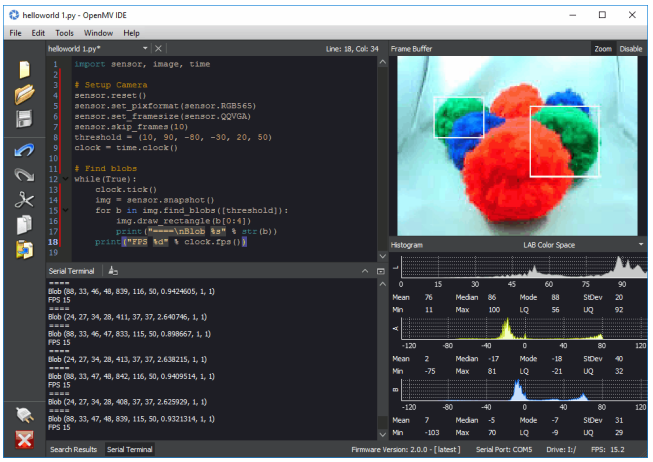
La programmation de la caméra OpenMV nécessite obligatoirement l'utilisation de son IDE par défaut. Soit OpenMV IDE utilisant du micro-python

Choix de composant :

Solution	OpenMV	MaixDuino	OpenCV
Qualité vidéo	1920x1080p	224x224p	640x480p
Carte embarqué	OpenMV+Shield	MaixDuino AIO	Raspberry Pi

J'ai donc choisi ici le composant OpenMV pour une meilleure definition

La caméra (OpenMV H7 Plus) :



l'acquisition et le traitement des images se fera à l'aide de la OpenMV H7 Plus. Cette caméra est programmable en micro-python, à l'aide de son ide disponible, l'OpenMV IDE.

Pour pouvoir faire la détection, la caméra aura besoin d'un modèle, ce dernier pourra être réalisé à l'aide d'un entraînement de type "Machine Learning" via le projet Edge Impulse. Il sera nécessaire pour cela de fournir des images positives de frelons asiatiques et des images négatives des insectes pouvant se poser sur l'appât : abeilles, mouches, etc...

Diagramme de cas d'utilisation :

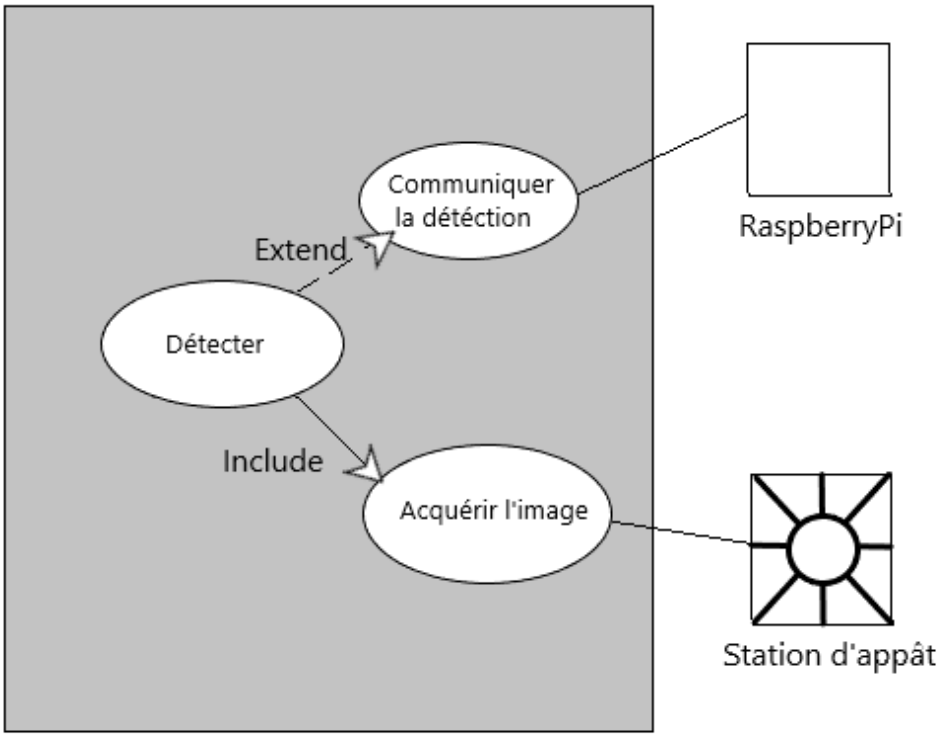
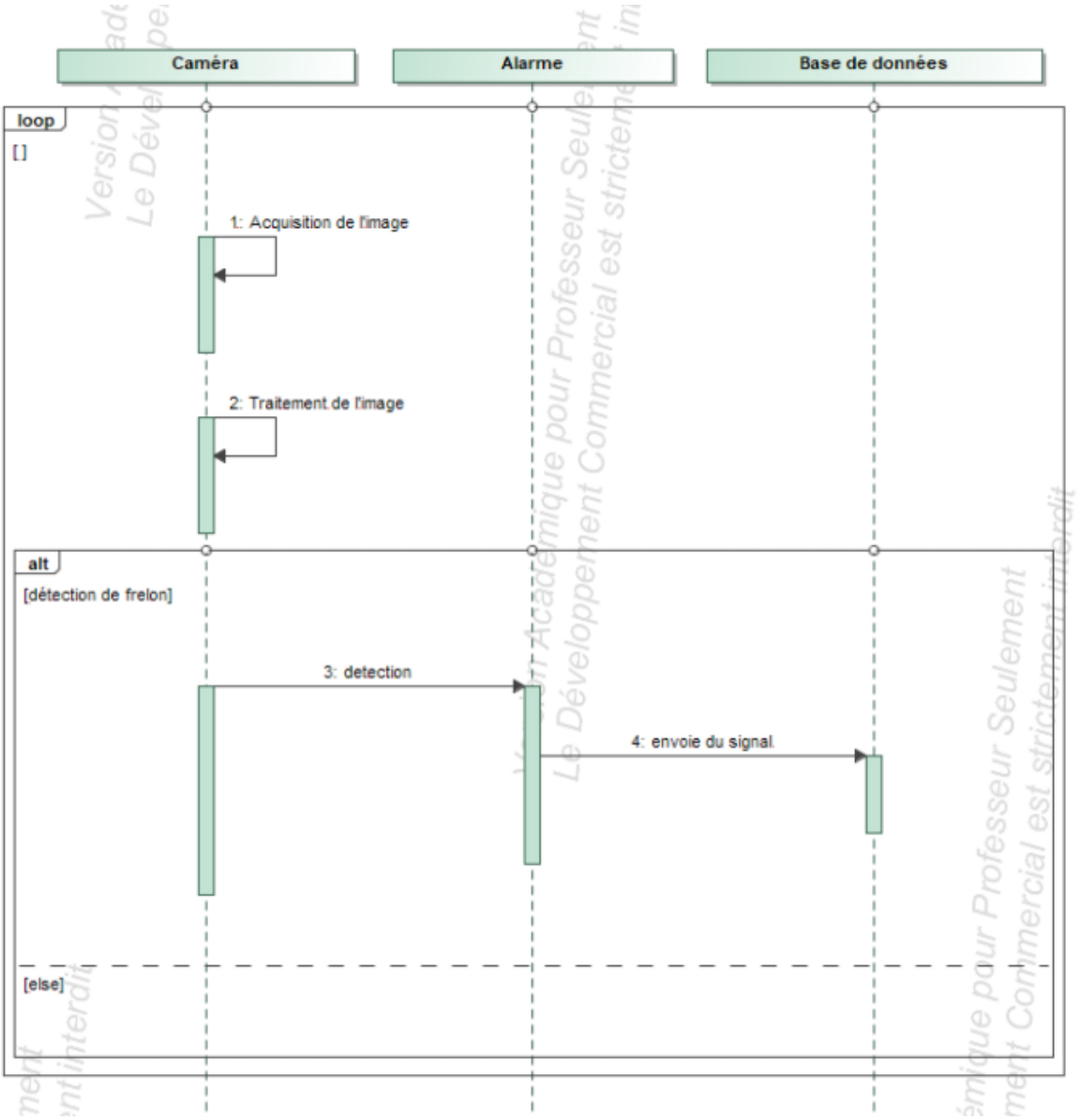
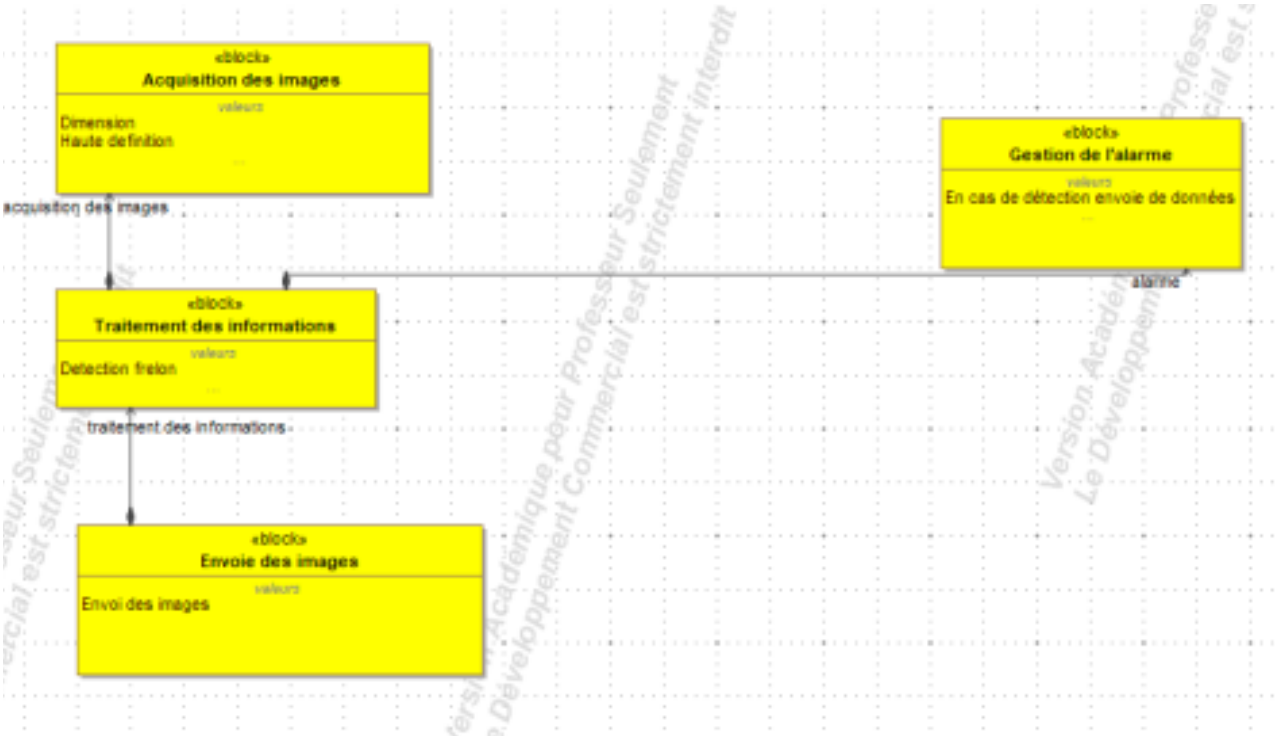


Diagramme de séquence :



Mon diagramme de séquence possède trois lignes de vie qui montrent la connexion entre la caméra, l'alarme et la base de données.
La caméra analyse en continu l'appât à la recherche de frelon. Si un frelon est détecté un signal d'alarme sera envoyé, et un screenshot du frelon sera envoyé à la base de données.

Diagramme de bloc :



Algorithme :

Initialisation des bibliothèques

```
import sensor, image, time, os, tf, network, sys, urequests
import usocket as socket
```

Il est à noter plusieurs bibliothèques particulières micropython liées à la communication réseau : network, usocket et urequests. Ainsi que des bibliothèques liées à la reconnaissance par l'IA comme tf, bibliothèque permettant d'utiliser les modèle TensorFlow de google (modèles du Machine learning réalisé avec "Edge Impulse")
Enfin des bibliothèques liées à la capture de l'image : sensor et image, qui permettent de gérer la caméra et l'objet image à reconnaître.

Connection au réseau privé WiFi

```
SSID ='raspi-webgui'      # Network SSID
KEY  ='ChangeMe'      # Network key
HOST =''      # Use first available interface
PORT = 8080   # Arbitrary non-privileged port
```

Initialisation des paramètres de ma caméra

```
sensor.reset()                          # Reset and initialize the sensor.
sensor.set_pixformat(sensor.RGB565)     # Set pixel format to RGB565 (or GRAYSCALE)
sensor.set_framesize(sensor.QVGA)       # Set frame size to QVGA (320x240)
sensor.set_windowing((240, 240))        # Set 240x240 window.
sensor.skip_frames(time=2000)           # Let the camera adjust.
```

On peut remarquer qu'on prend ici une faible résolution (240x240) afin d'éviter les pertes de performance

```
compteur=0
fiable=0
while(compteur!=10):
    img = sensor.snapshot()
    # default settings just do one detection... change them to search the image...
    for obj in tf.classify(net, img, min_scale=1.0, scale_mul=0.8, x_overlap=0.5, y_overlap=0.5):
        print("**********\nPredictions at [x=%d,y=%d,w=%d,h=%d]" % obj.rect())
        img.draw_rectangle(obj.rect())
        # This combines the labels and confidence values into a list of tuples
        predictions_list = list(zip(labels, obj.output()))

        for i in range(len(predictions_list)):
            print("%s = %f" % (predictions_list[i][0], predictions_list[i][1]))

        if predictions_list[1][1] >= 0.6 :
            print('fiable+1')
            fiable+=1
        else:
            print('fiable-1')
            fiable-=1
```

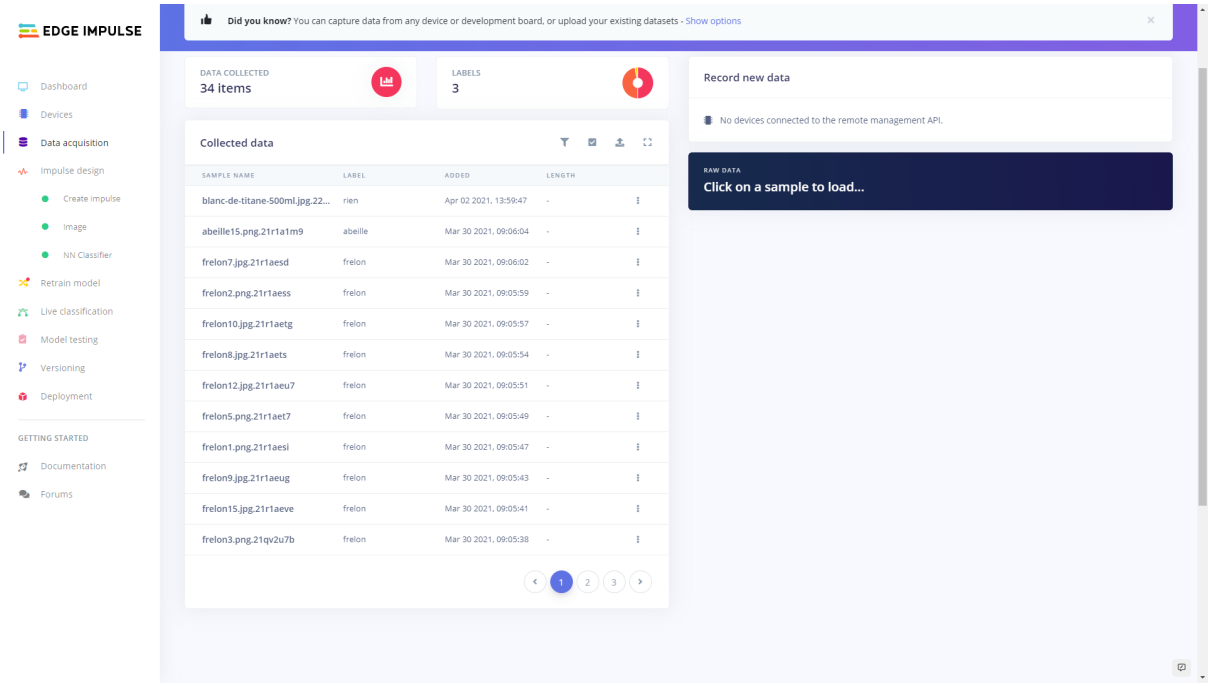
Détection des frelons avec l'aide d'une variable s'incrémentant à chaque fois que la caméra trouve à plus de 60% un frelon et se décrémentant si ce n'est pas. cette variable est modifié 10 fois puis si après les 10 incrémentations/incrémentations la valeur de la variable est comprise entre 0 et 10 alors le programme considère qu'il y a bien présence d'un frelon

```
if fiable>=0 :
    sensor.snapshot().save("example.jpg")
    files = open('example.jpg', 'rb')
     r = urequests.post(url, files)
     client.send("POST /savimage.php HTTP/1.1 \r\n"
    client.write("POST /update HTTP/1.0\r\n")
    client.write("Host: " + HOST + "\r\n")
     client.send("Content-Length: " + str(len(files)) + "\r\n\r\n")
    client.write("Content-Length: \r\n\r\n")
    client.write(files)
    print(client.read(128))

     client.send("GET /savimage.php HTTP/1.1 \r\n" \
             "Host: 10.3.143.1\r\n" \
             "Content-Type: multipart/x-mixed-replace;boundary=openmv\r\n" \
             "Cache-Control: no-cache\r\n" \
             "Pragma: no-cache\r\n\r\n")
    print("send")
```

Création d'un model avec Edge Impulse :

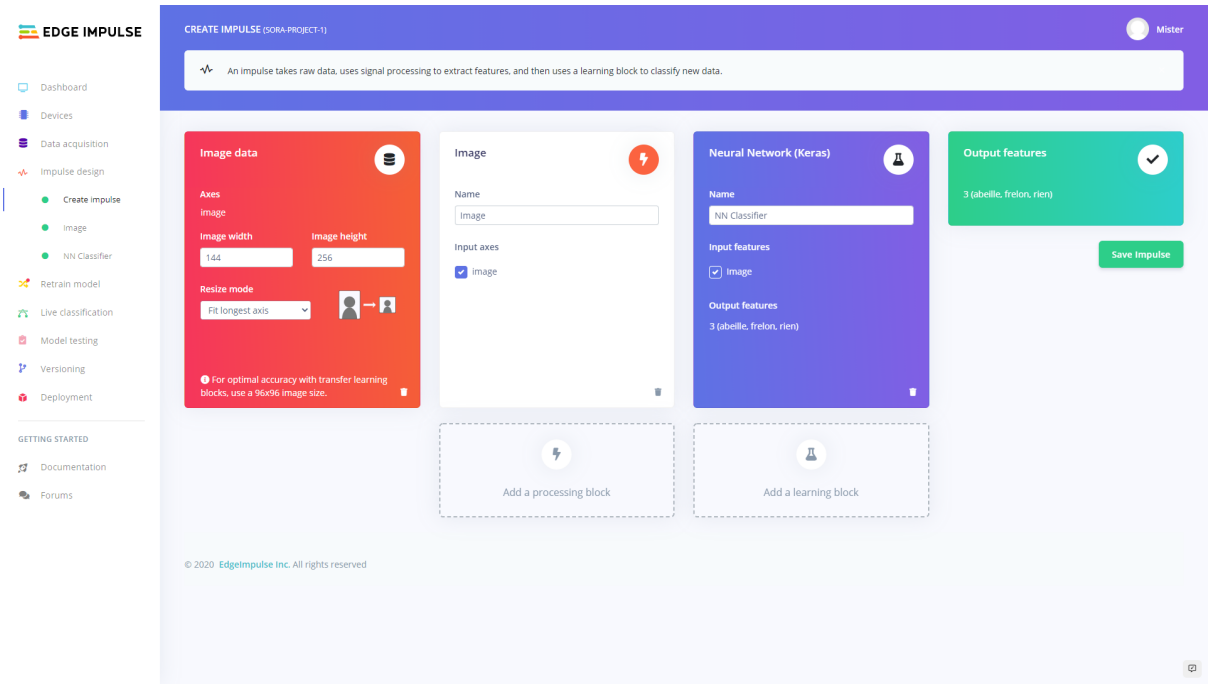
Ajouter des images de frelon/abeille sous le label "Frelon"/"Abeille" et ajouter une image d'un fond blanc pour le label "Rien"



Création du model en 3 étapes :

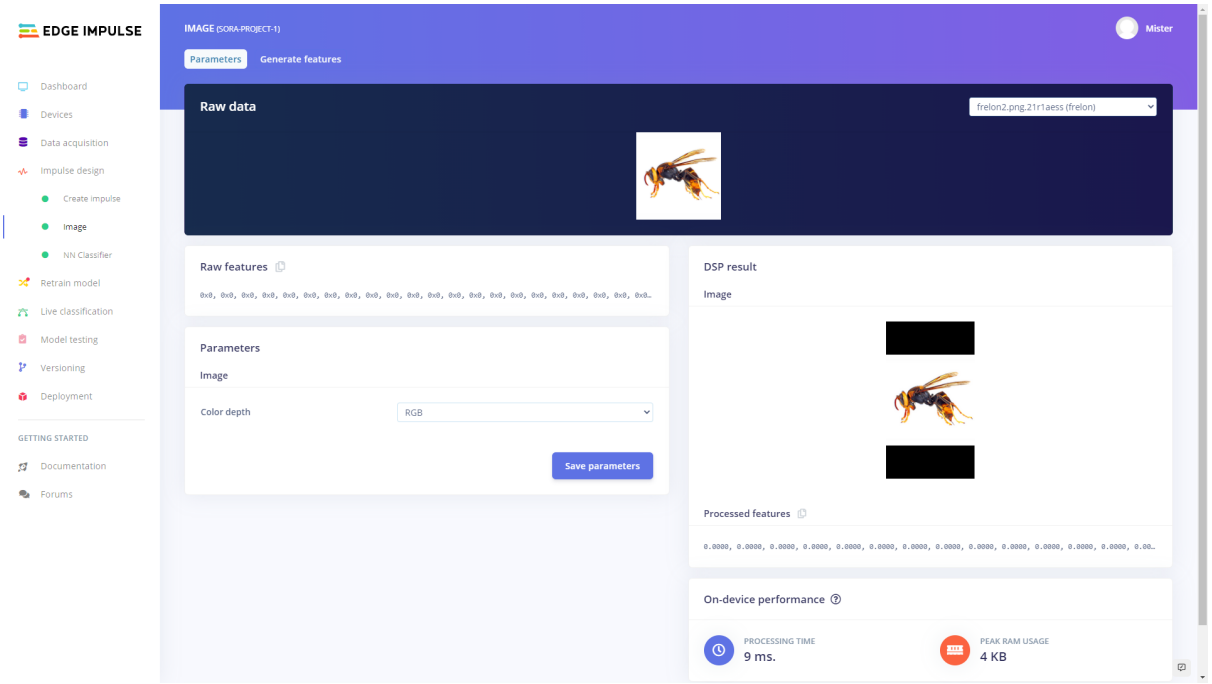
Etape 1 :

Depuis create impulse paramétrer la résolution du modèle, y ajouter NN Classifier et ajouter les 3 labels précédemment créé



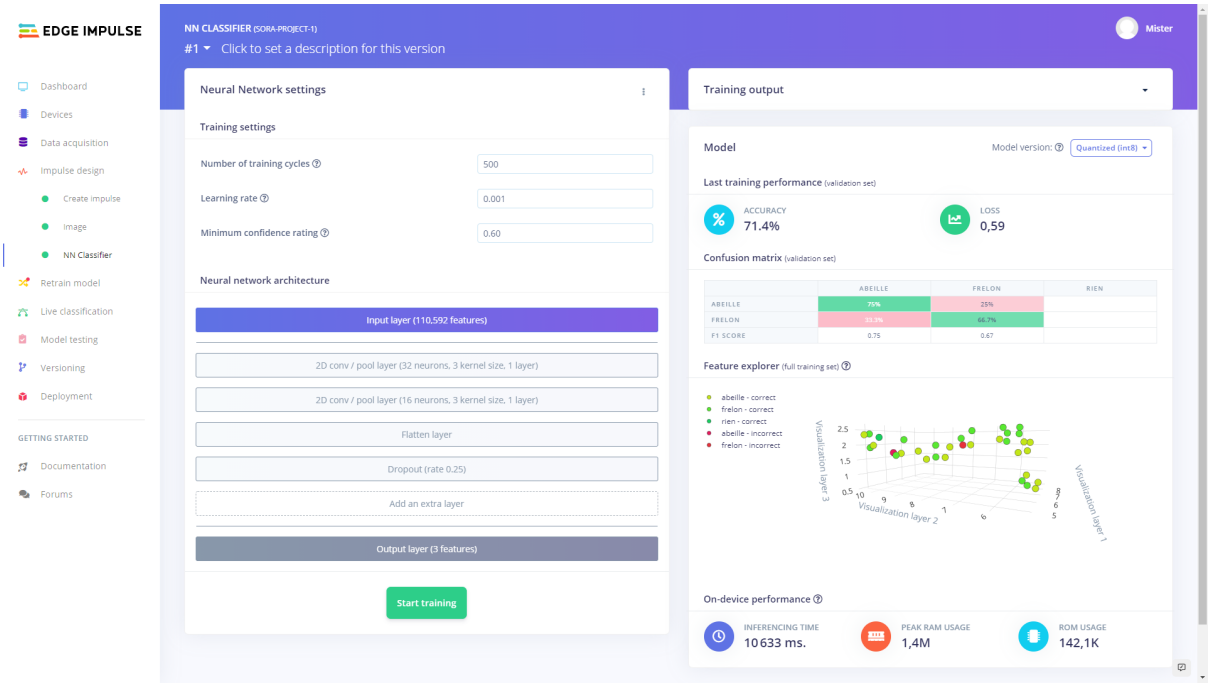
Etape 2 :

Generate features pour préparer le modèle à l'entraînement.

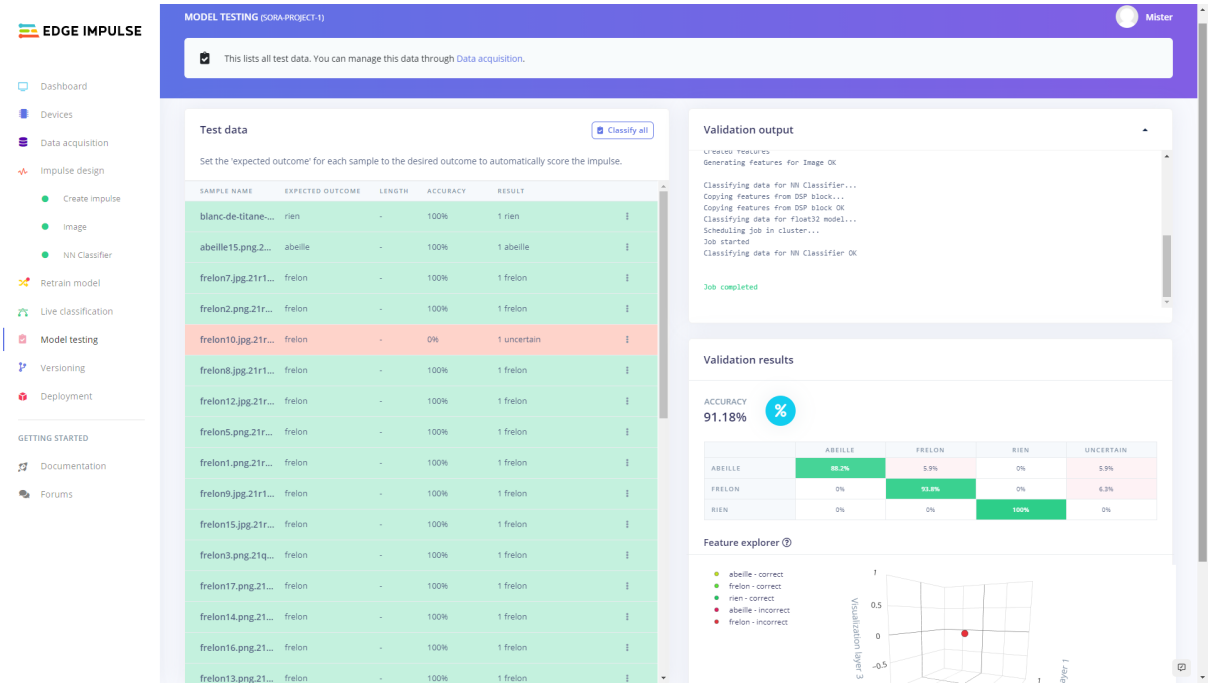


Etape 3 :

Cliquer sur start training pour lancer l'entraînement du modèle



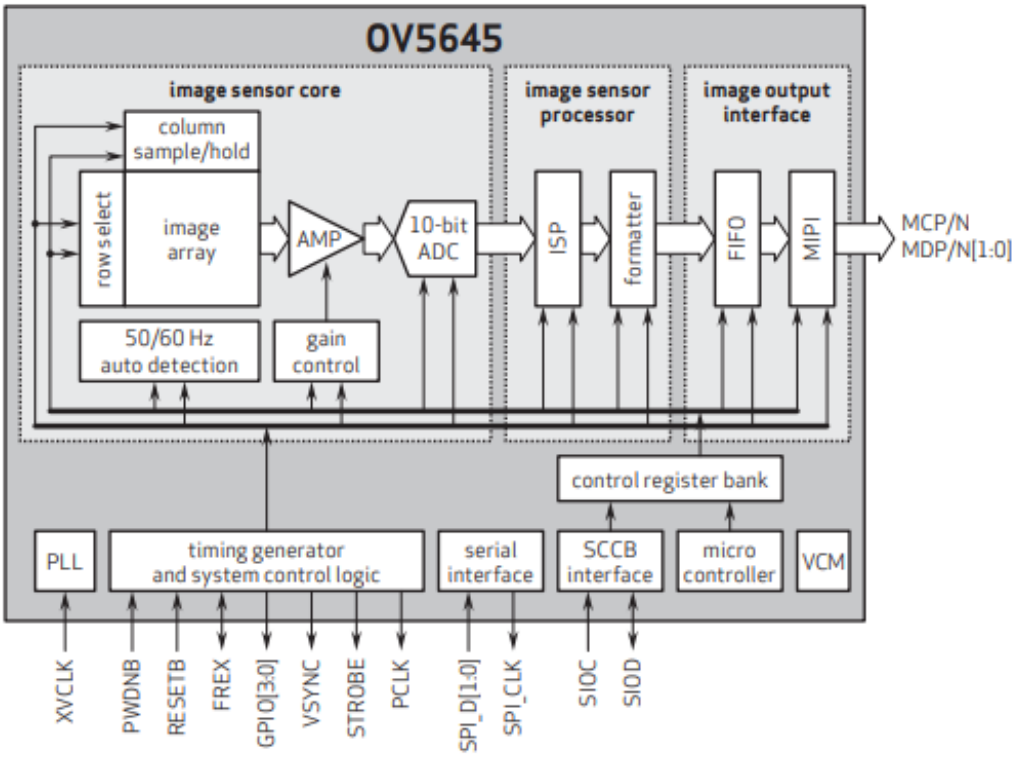
Une fois le modèle entraîné il faut faire la vérification de la fiabilité du model :
On peut ici voir qu'une des images est rouge et pour qui le résultat est incertain, mais c'est ici une parmi une seule image parmis un grand nombre et son résultat est seulement incertain et non inversé alors c'est négligeable.

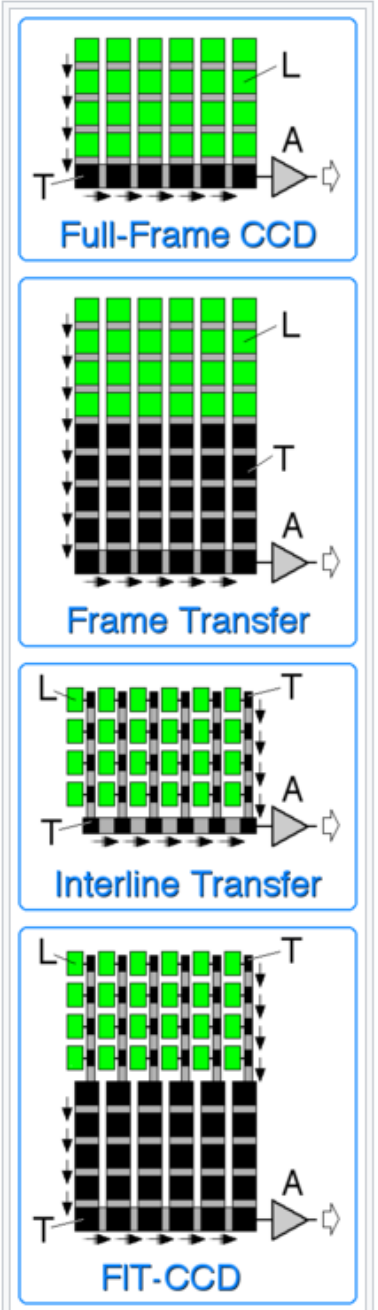


Physique

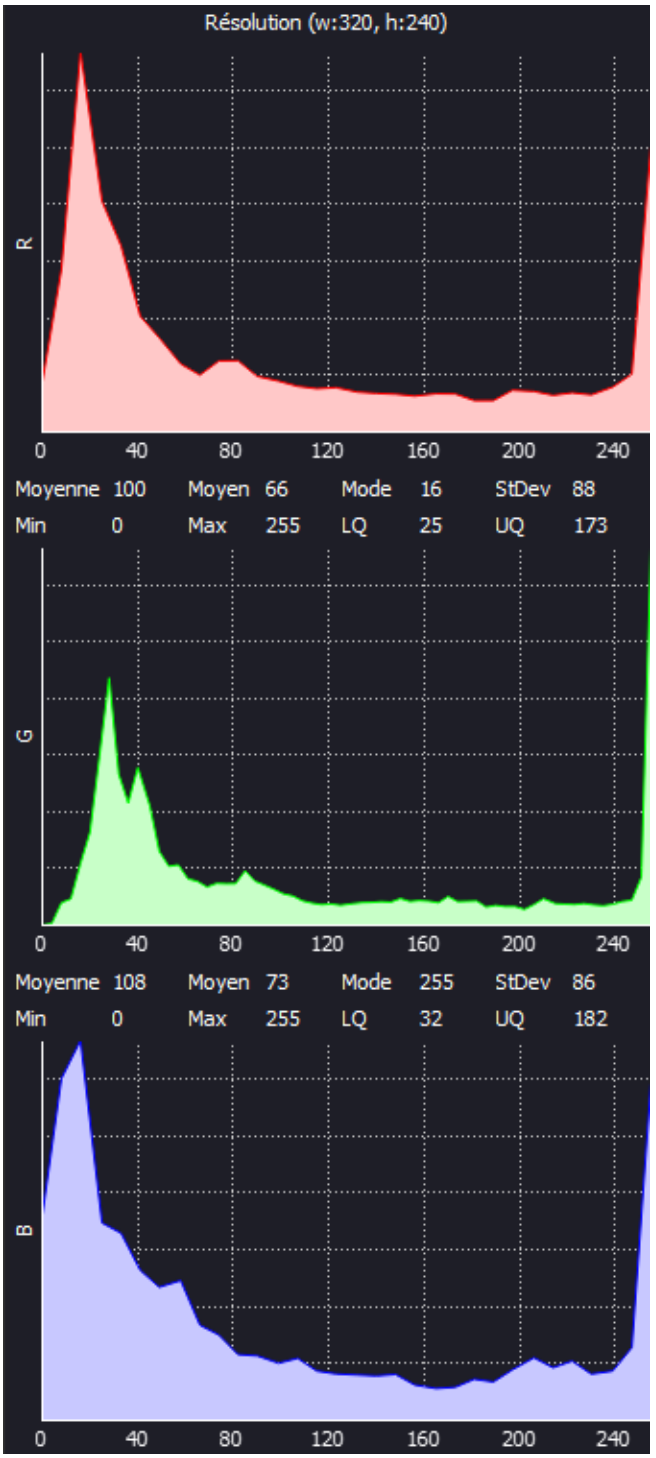
Le capteur ccd: OV5645

Un capteur photographique est un composant électronique photosensible servant à convertir un rayonnement électromagnétique (UV, visible ou IR) en un signal électrique analogique. Ce signal est ensuite amplifié, puis numérisé par un convertisseur analogique-numérique et enfin traité pour obtenir une image numérique.



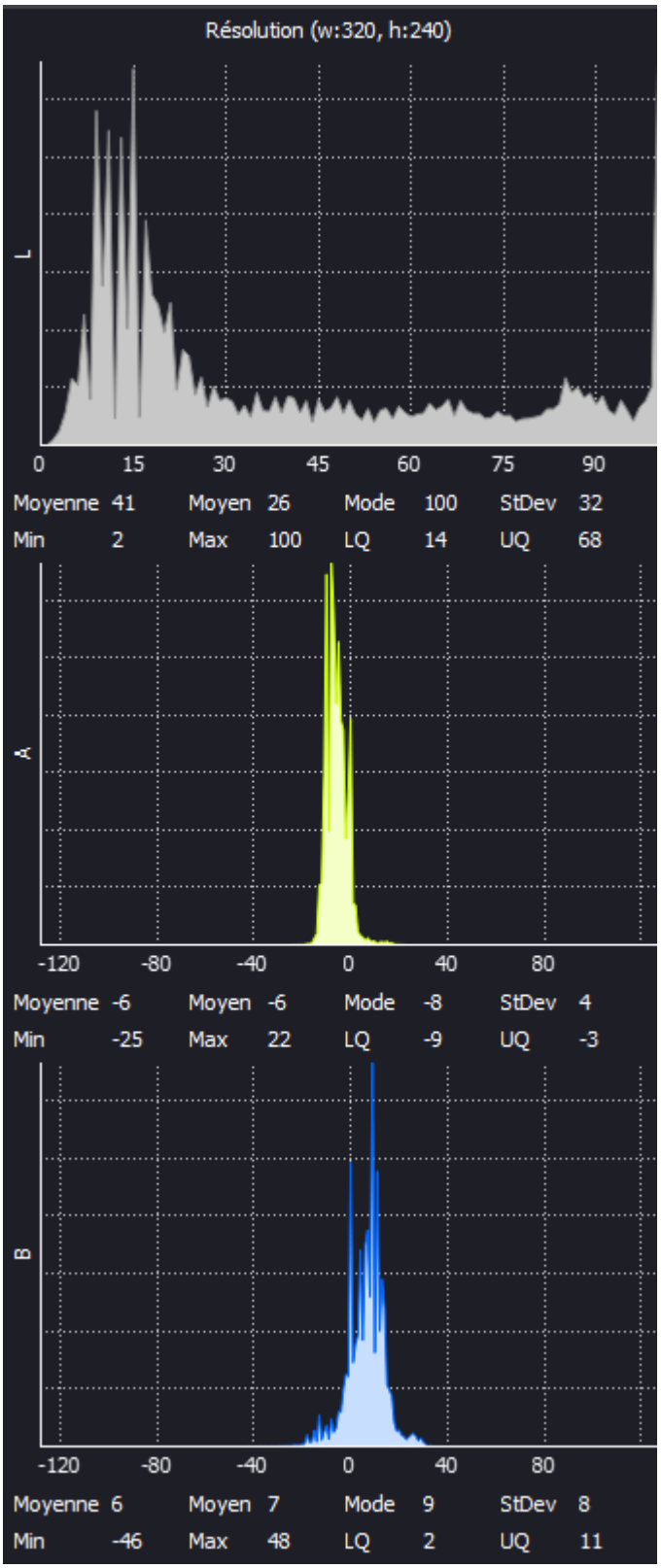


Quelques CCD :
L – pixels photosensibles,
T – à registre de transfert,
A – amplificateur de contraste.

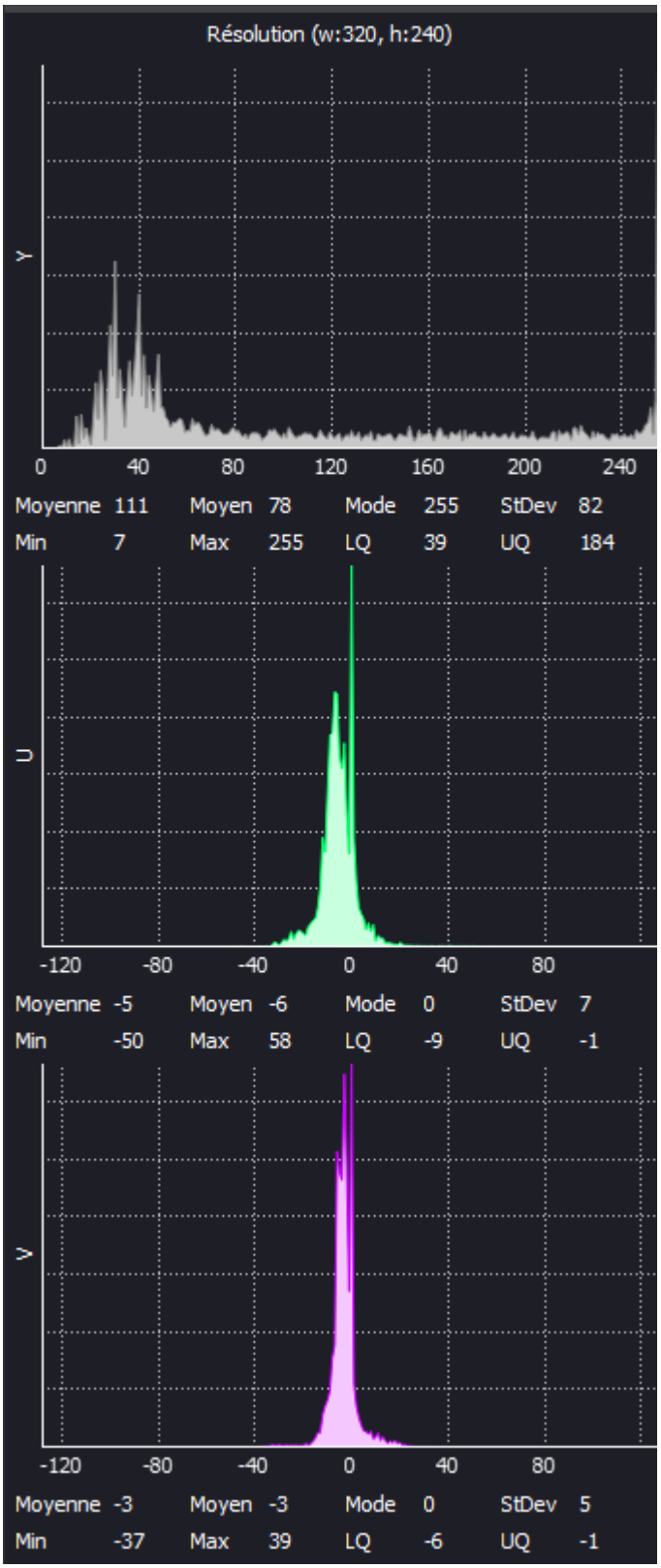


La synthèse additive des couleurs est le procédé consistant à combiner les lumières de plusieurs sources colorées dans le but d'obtenir une lumière colorée quelconque dans un gamut déterminé.

La synthèse additive utilise généralement trois lumières colorées : une rouge, une verte et une bleue (RVB ou RGB en anglais pour red, green, blue). L'addition de ces trois lumières colorées en proportions convenables donne la lumière blanche. L'absence de lumière donne du noir.



Le modèle Lab caractérise une couleur avec un paramètre d'intensité correspondant à la luminance et de deux paramètres de chrominance qui décrivent la couleur.
La combinaison L* est la clarté, qui va de 0% (noir) à 100% (blanc).
La composante a* représente la gamme de l'axe du vert (-128) au rouge (+128).
La composante b* représente la gamme de l'axe du bleu (-128) au jaune (+128).



Le modèle YUV ou plus précisément Y'UV définit un espace colorimétrique en trois composantes. La première, Y', représente la luma (à ne pas confondre avec la luminance relative notée Y, le symbole prime de Y' indiquant une correction gamma) et les deux autres, U et V, représentent la chrominance.

$$Y \simeq R + G + B \qquad U \simeq B - Y \qquad V \simeq R - Y$$

Conclusion

Ce projet m’a appris comment travailler et s’entraider dans le cadre d’une équipe avec un objectif commun. Cela m’a également appris le fonctionnement de l’Intelligence artificielle s’appuyant sur du Machine Learning et son utilisation, ce qui pourra certainement m’être utile dans les années à venir dans mon projet professionnel, notamment au vu du développement des applications mobilisant l’IA.

Ce fut l’occasion d’être confronté à différentes difficultés. J’ai dû notamment faire face à un sujet totalement nouveau et inconnu pour moi : L’intelligence artificielle et le Machine Learning. Ces notions étant très abstraites, il m’a fallu du temps afin de bien comprendre son utilisation. Mais cette difficulté m'a permis de comprendre les enjeux et le poids d’avoir une partie importante dans une équipe et très dépendante pour les coéquipiers.

Annexes

Code Détection de Frelon

```
import sensor, image, time, os, tf, network, sys, urequests
import usocket as socket
SSID ='raspi-webgui'    # Network SSID
KEY  ='ChangeMe'    # Network key
HOST =''    # Use first available interface
PORT = 8080  # Arbitrary non-privileged port
print("Trying to connect... (may take a while)...")
wlan = network.WINC()
wlan.connect(SSID, key=KEY, security=wlan.WPA_PSK)
print(wlan.ifconfig())
url = '10.3.143.1/post'
sensor.reset()                      # Reset and initialize the sensor.
sensor.set_pixformat(sensor.RGB565)   # Set pixel format to RGB565 (or GRAYSCALE)
sensor.set_framesize(sensor.QVGA)     # Set frame size to QVGA (320x240)
sensor.set_windowing((240, 240))      # Set 240x240 window.
sensor.skip_frames(time=2000)         # Let the camera adjust.
net = "trained.tflite"
labels = [line.rstrip('\n') for line in open("labels.txt")]
clock = time.clock()
while(True):
  addr = socket.getaddrinfo('10.3.141.1', 80)[0][-1]
  client = socket.socket()
  try:
    client.connect(addr)
    conn = 1
  except:
    conn = 0
  if conn == 1:
    clock.tick()
    compteur=0
    fiable=0
    while(compteur!=10):
      img = sensor.snapshot()
      for obj in tf.classify(net, img, min_scale=1.0, scale_mul=0.8, x_overlap=0.5, y_overlap=0.5):
        print("**********\nPredictions at [x=%d,y=%d,w=%d,h=%d]" % obj.rect())
        img.draw_rectangle(obj.rect())
        predictions_list = list(zip(labels, obj.output()))

        for i in range(len(predictions_list)):
          print("%s = %f" % (predictions_list[i][0], predictions_list[i][1]))

        if predictions_list[1][1] >= 0.6 :
          print('fiable+1')
          fiable+=1
        else:
          print('fiable-1')
          fiable-=1
      print(clock.fps(), "fps")
      compteur+=1
    print("\n\n\n fiabilité :")
    print(fiable)
    if fiable>=0 :
      sensor.snapshot().save("example.jpg")
      files = open('example.jpg', 'rb')
      client.write("POST /update HTTP/1.0\r\n")
      client.write("Host: " + HOST + "\r\n")
      client.write("Content-Length: \r\n\r\n")
      client.write(files)
      print(client.read(128))
      print("send"
    client.close()
```

Code streaming site web

```
import sensor, image, time, network, usocket, sys

SSID ='raspi-webgui'    # Network SSID
KEY  ='ChangeMe'    # Network key
HOST =''    # Use first available interface
PORT = 8080  # Arbitrary non-privileged port
sensor.reset()
# Set sensor settings
sensor.set_framesize(sensor.QVGA)
sensor.set_pixformat(sensor.RGB565)
# Init wlan module and connect to network
print("Trying to connect... (may take a while)...")
wlan = network.WINC()
wlan.connect(SSID, key=KEY, security=wlan.WPA_PSK)
# We should have a valid IP now via DHCP
print(wlan.ifconfig())
# Create server socket
s = usocket.socket(usocket.AF_INET, usocket.SOCK_STREAM)
#ftp = FTP('10.3.141.1') # connect to host, default port
#ftp.login() # default, i.e.: user anonymous, passwd anonymous@
# Bind and listen
s.bind([HOST, PORT])
s.listen(5)
# Set server socket to blocking
s.setblocking(True)
def start_streaming(s):
  print ('Waiting for connections..')
  client, addr = s.accept()
  # set client socket timeout to 2s
  client.settimeout(2.0)
  print ('Connected to ' + addr[0] + ':' + str(addr[1]))
  # Read request from client
  data = client.recv(1024)
  # Should parse client request here
  # Send multipart header
  client.send("HTTP/1.1 200 OK\r\n" \
          "Server: OpenMV\r\n" \
          "Content-Type: multipart/x-mixed-replace;boundary=openmv\r\n" \
          "Cache-Control: no-cache\r\n" \
          "Pragma: no-cache\r\n\r\n")
  client.send("GET /savimage.php HTTP/1.1 \r\n" \
        "Host: 10.3.143.1\r\n" \
        "Content-Type: multipart/x-mixed-replace;boundary=openmv\r\n" \
        "Cache-Control: no-cache\r\n" \
        "Pragma: no-cache\r\n\r\n")
  # FPS clock
  clock = time.clock()
  # Start streaming images
  # NOTE: Disable IDE preview to increase streaming FPS.
  while (True):
    clock.tick() # Track elapsed milliseconds between snapshots().
    frame = sensor.snapshot()
    cframe = frame.compressed(quality=35)
    header = "\r\n--openmv\r\n" \
          "Content-Type: image/jpeg\r\n"\
          "Content-Length:"+str(cframe.size())+"\r\n\r\n"
    client.send(header)
    client.send(cframe)
    print(clock.fps())
while (True):
  try:
    start_streaming(s)
  except OSError as e:
    print("socket error: ", e)
```