Article:

Introduction:

On a construction site, the workers are exposed to several dangers, where many come from accidents. Thus, it is always important that they are properly protected by PPE, to avoid any lethal damage to the worker. However, even knowing the risks, and being mandatory, some workers are still caught without the use of it.

Given this problem, we thought of implementing an AI that could detect whether workers are wearing PPE, through photos taken by the robot's camera. This way, the robot could patrol the construction site, and if it detected someone without a helmet, it would go to the worker carrying the helmet in a box on its back and give him the equipment.

Deep Learning to detect helmets:

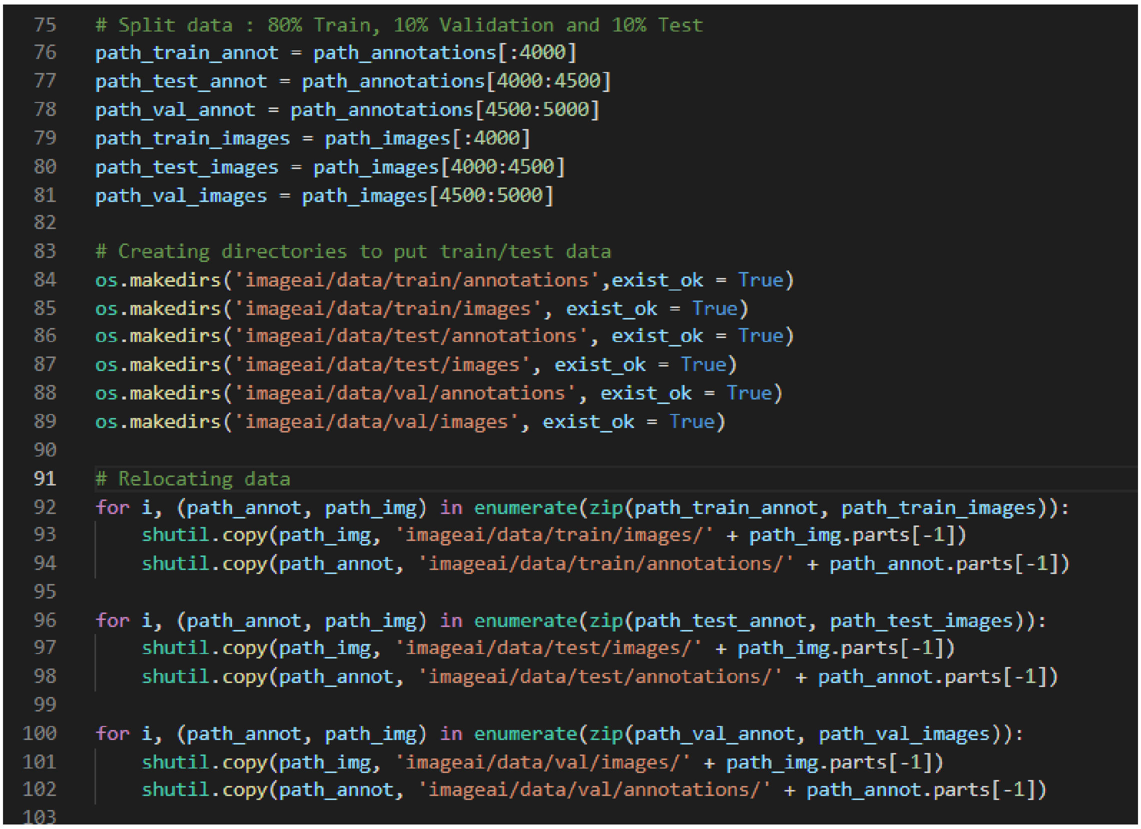
Before we begin to understand the choice of AI model, it is interesting to understand some concepts. The first of these is what a perceptron is, which is a type of artificial neural network. A perceptron is a binary classifier that maps its input x to an output value f(x).

Another important concept is CNN, which is, in a machine learning context, a convolutional neural network (CNN). The CNN is a class of artificial neural network that uses a variation of multilayer perceptrons designed to require as little preprocessing as possible. For this reason, such networks are known as space invariant neural networks (SIANN).

Then, in the development of the AI, we used a ready-made dataset that we found on the Kaggle platform, which allows us to detect the helmet and the head as well in case the worker is not wearing a helmet. Given this, we chose to use a single pass object detection method, which uses a convolutional neural network to extract the features, because as the name of the method itself says "You Only Look Once", i.e., it only needs to see the image once to send to the neural network, unlike other methods like R-CNN or Faster R-CNN, and due to this feature YOLO is able to achieve a much higher detection speed than other methods.

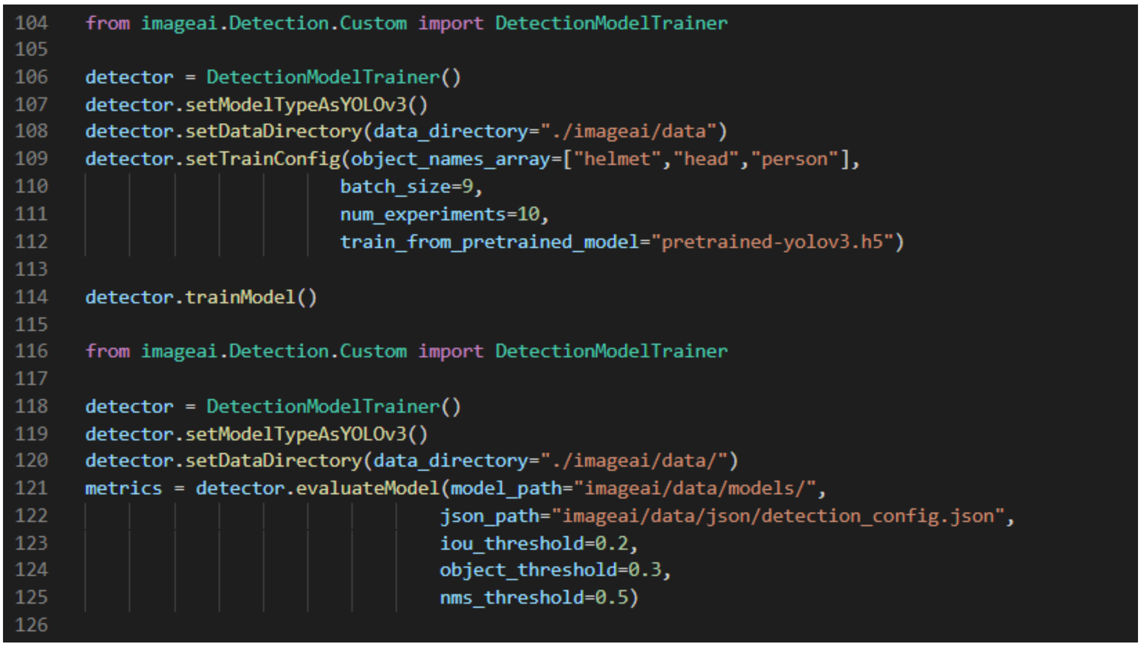
Before we started training the AI, it was necessary to prepare the environment. So, we created a virtual environment (for python) where we installed all the requirements (libraries in their required versions) for using YOLO and downloaded the database from Kaggle.

To train the AI, we first divided the data, so that 80% was for training, 10% for validation, and 10% for testing, and then we reallocated this data into folders.



```python
# Split data : 80% Train, 10% Validation and 10% Test
path_train_annot = path_annotations[:4000]
path_test_annot = path_annotations[4000:4500]
path_val_annot = path_annotations[4500:5000]
path_train_images = path_images[:4000]
path_test_images = path_images[4000:4500]
path_val_images = path_images[4500:5000]

# Creating directories to put train/test data
os.makedirs('imageai/data/train/annotations',exist_ok = True)
os.makedirs('imageai/data/train/images', exist_ok = True)
os.makedirs('imageai/data/test/annotations', exist_ok = True)
os.makedirs('imageai/data/test/images', exist_ok = True)
os.makedirs('imageai/data/val/annotations', exist_ok = True)
os.makedirs('imageai/data/val/images', exist_ok = True)

# Relocating data
for i, (path_annot, path_img) in enumerate(zip(path_train_annot, path_train_images)):
    shutil.copy(path_img, 'imageai/data/train/images/' + path_img.parts[-1])
    shutil.copy(path_annot, 'imageai/data/train/annotations/' + path_annot.parts[-1])

for i, (path_annot, path_img) in enumerate(zip(path_test_annot, path_test_images)):
    shutil.copy(path_img, 'imageai/data/test/images/' + path_img.parts[-1])
    shutil.copy(path_annot, 'imageai/data/test/annotations/' + path_annot.parts[-1])

for i, (path_annot, path_img) in enumerate(zip(path_val_annot, path_val_images)):
    shutil.copy(path_img, 'imageai/data/val/images/' + path_img.parts[-1])
    shutil.copy(path_annot, 'imageai/data/val/annotations/' + path_annot.parts[-1])
```

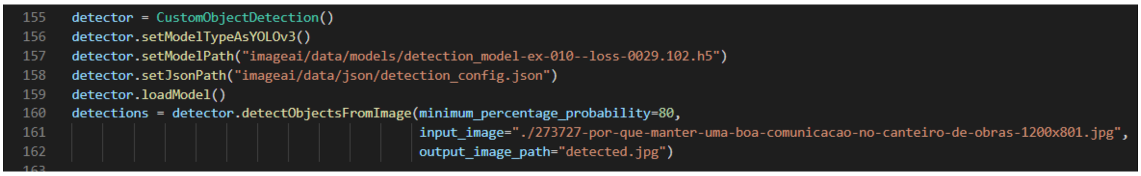
Then we start training the AI and after its completion, we start a validation routine for each of the 10 epochs of the AI.



```python
from imageai.Detection.Custom import DetectionModelTrainer

detector = DetectionModelTrainer()
detector.setModelTypeAsYOLOv3()
detector.setDataDirectory(data_directory="./imageai/data")
detector.setTrainConfig(object_names_array=["helmet","head","person"],
                        batch_size=9,
                        num_experiments=10,
                        train_from_pretrained_model="pretrained-yolov3.h5")

detector.trainModel()

from imageai.Detection.Custom import DetectionModelTrainer

detector = DetectionModelTrainer()
detector.setModelTypeAsYOLOv3()
detector.setDataDirectory(data_directory="./imageai/data/")
metrics = detector.evaluateModel(model_path="imageai/data/models/",
                                json_path="imageai/data/json/detection_config.json",
                                iou_threshold=0.2,
                                object_threshold=0.3,
                                nms_threshold=0.5)
```

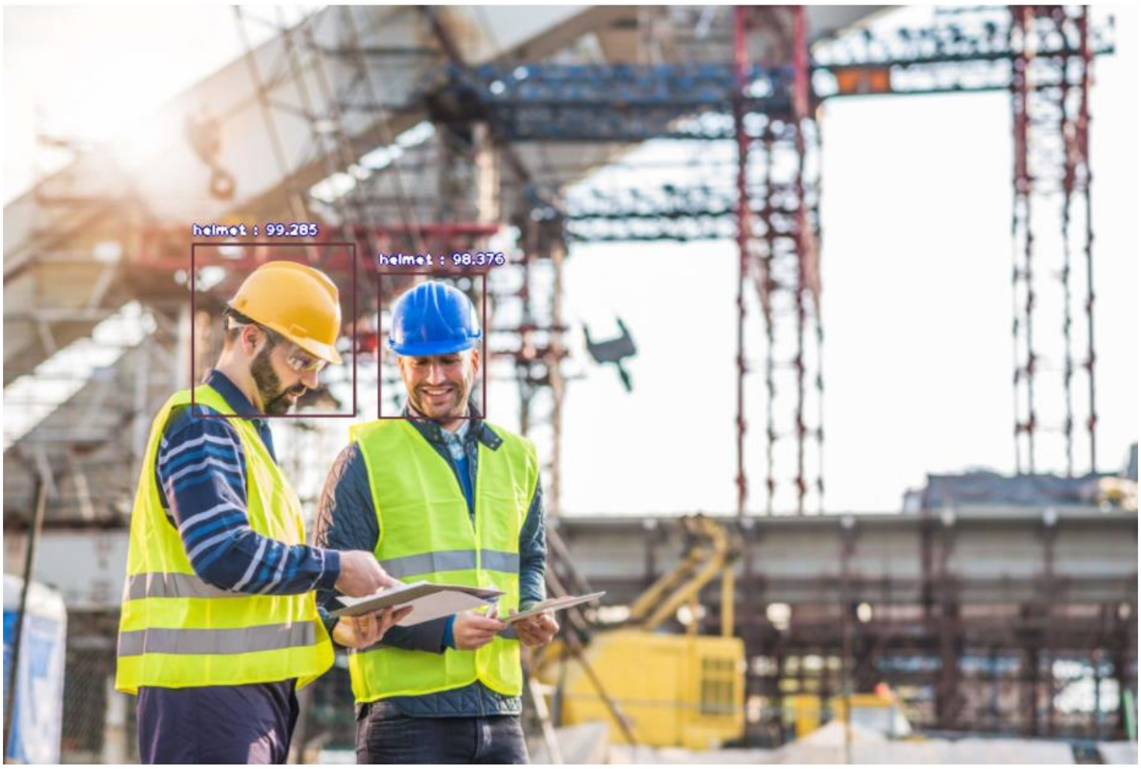
To finish, we tested the AI with a generic image from google images. Below you can see the routine and the image after detecting the helmets.

Code:



```
detector = CustomObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath("imageai/data/models/detection_model-ex-010--loss-0029.102.h5")
detector.setJsonPath("imageai/data/json/detection_config.json")
detector.loadModel()
detections = detector.detectObjectsFromImage(minimum_percentage_probability=80,
                                             input_image="./273727-por-que-manter-uma-boa-comunicacao-no-canteiro-de-obras-1200x801.jpg",
                                             output_image_path="detected.jpg")
```

Image:



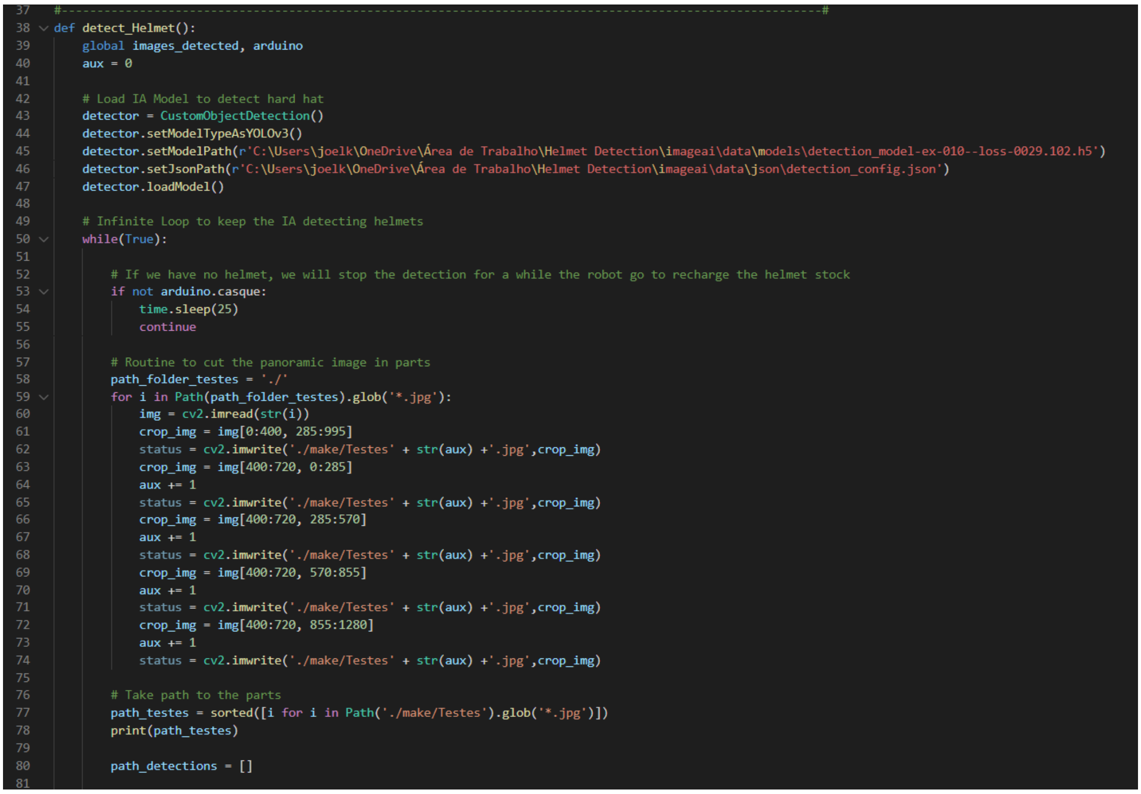
So, from the test one can see promising results, as the AI was able to detect all helmets, even if this does not imply that it will always detect them.

Integration robot – IA:

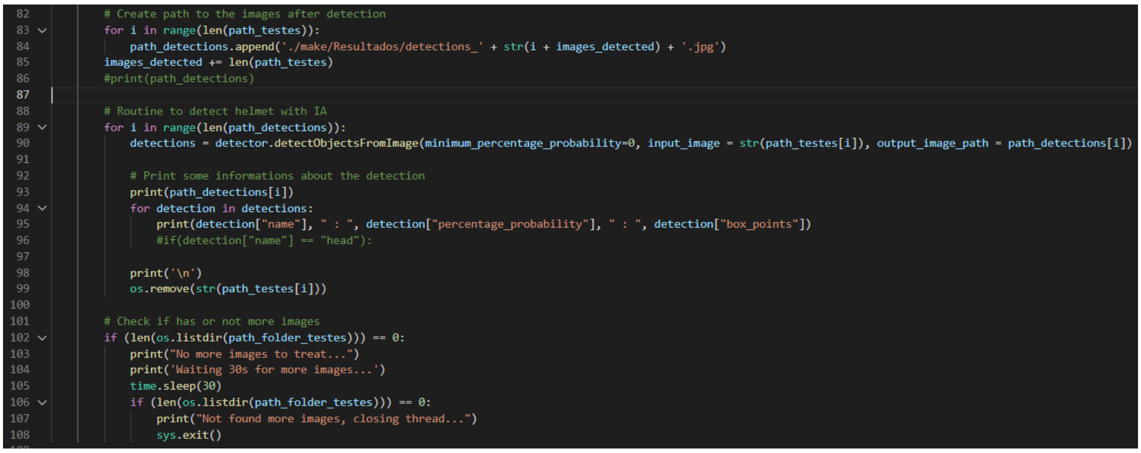
To make the integration between the scripts, being the Arduino routine to detect the helmets through a sensor, the routine to load the AI that will detect the helmets and the routine that will make the command in the shell that makes the robot send images by streaming, we use threads to create parallelism between them, making the routines run in parallel.

In the routine of helmet detection, first, we load the IA model out of the looping, to avoid execute this command in unnecessary way. Second, we start with an image treatment, because we receive the streaming image from robot, that image isn’t in the right state to apply the IA, so we need to cut in 5 images and after applies the IA in each one. After that, we create the path to the detected images and images to be detected and send all to the IA routine. To finish, we check if have more images in the folder, if no, the thread will be waiting for a short time, if after it can find more images, the thread will stop.

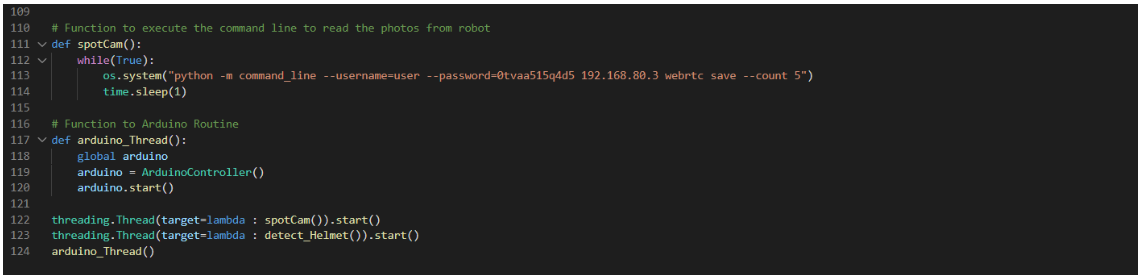
In the routine of spot cam, we only call the command line, which take images from the camera 360, each second.



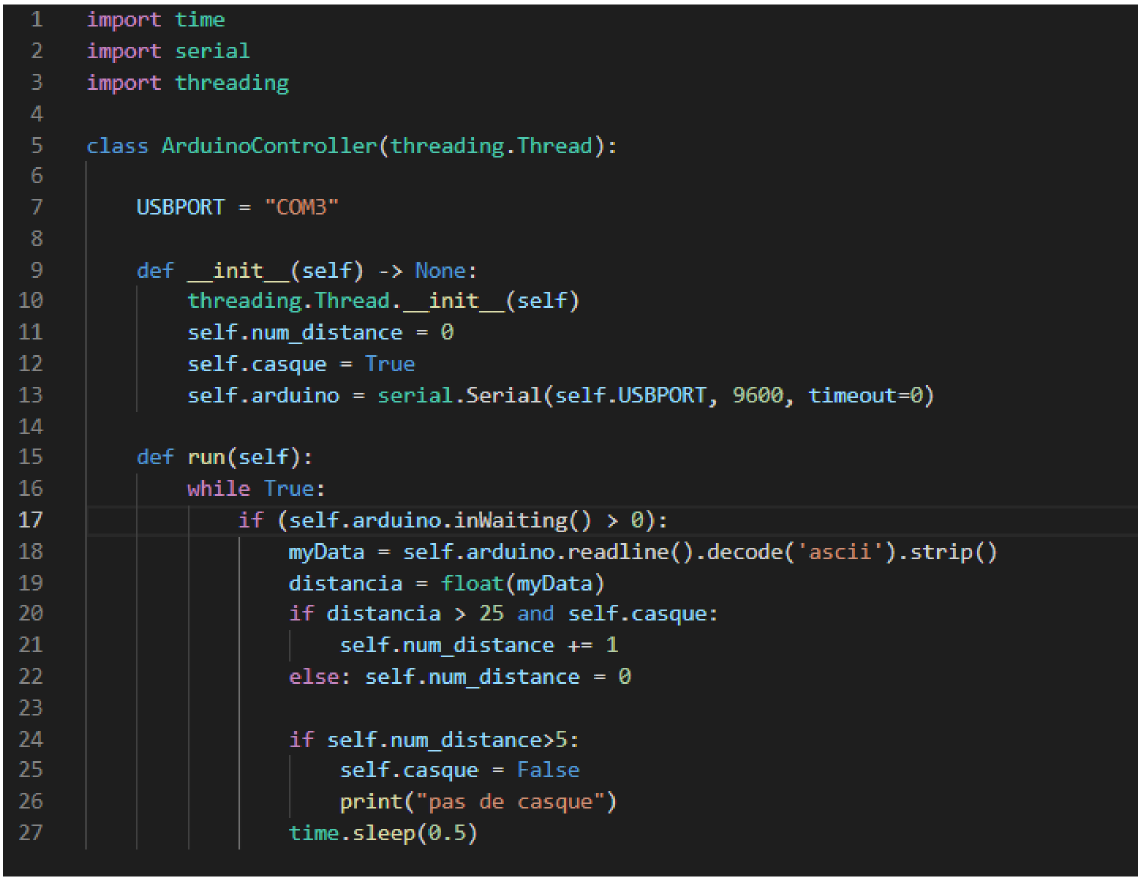
```python
#----------------------------------------------------------------------------------------------------------#
def detect_Helmet():
    global images_detected, arduino
    aux = 0

    # Load IA Model to detect hard hat
    detector = CustomObjectDetection()
    detector.setModelTypeAsYOLOv3()
    detector.setModelPath(r'C:\Users\joelk\OneDrive\Área de Trabalho\Helmet Detection\imageai\data\models\detection_model-ex-010--loss-0029.102.h5')
    detector.setJsonPath(r'C:\Users\joelk\OneDrive\Área de Trabalho\Helmet Detection\imageai\data\json\detection_config.json')
    detector.loadModel()

    # Infinite Loop to keep the IA detecting helmets
    while(True):

        # If we have no helmet, we will stop the detection for a while the robot go to recharge the helmet stock
        if not arduino.casque:
            time.sleep(25)
            continue

        # Routine to cut the panoramic image in parts
        path_folder_testes = './'
        for i in Path(path_folder_testes).glob('*.jpg'):
            img = cv2.imread(str(i))
            crop_img = img[0:400, 285:995]
            status = cv2.imwrite('./make/Testes' + str(aux) +'.jpg',crop_img)
            crop_img = img[400:720, 0:285]
            aux += 1
            status = cv2.imwrite('./make/Testes' + str(aux) +'.jpg',crop_img)
            crop_img = img[400:720, 285:570]
            aux += 1
            status = cv2.imwrite('./make/Testes' + str(aux) +'.jpg',crop_img)
            crop_img = img[400:720, 570:855]
            aux += 1
            status = cv2.imwrite('./make/Testes' + str(aux) +'.jpg',crop_img)
            crop_img = img[400:720, 855:1280]
            aux += 1
            status = cv2.imwrite('./make/Testes' + str(aux) +'.jpg',crop_img)

        # Take path to the parts
        path_testes = sorted([i for i in Path('./make/Testes').glob('*.jpg')])
        print(path_testes)

        path_detections = []

```



```python
        # Create path to the images after detection
        for i in range(len(path_testes)):
            path_detections.append('./make/Resultados/detections_' + str(i + images_detected) + '.jpg')
        images_detected += len(path_testes)
        #print(path_detections)

        # Routine to detect helmet with IA
        for i in range(len(path_detections)):
            detections = detector.detectObjectsFromImage(minimum_percentage_probability=0, input_image = str(path_testes[i]), output_image_path = path_detections[i])

            # Print some informations about the detection
            print(path_detections[i])
            for detection in detections:
                print(detection["name"], " : ", detection["percentage_probability"], " : ", detection["box_points"])
                #if(detection["name"] == "head"):

            print('\n')
            os.remove(str(path_testes[i]))

        # Check if has or not more images
        if (len(os.listdir(path_folder_testes))) == 0:
            print("No more images to treat...")
            print('Waiting 30s for more images...')
            time.sleep(30)
            if (len(os.listdir(path_folder_testes))) == 0:
                print("Not found more images, closing thread...")
                sys.exit()
```



```python

# Function to execute the command line to read the photos from robot
def spotCam():
    while(True):
        os.system("python -m command_line --username=user --password=0tvaa515q4d5 192.168.80.3 webrtc save --count 5")
        time.sleep(1)

# Function to Arduino Routine
def arduino_Thread():
    global arduino
    arduino = ArduinoController()
    arduino.start()

threading.Thread(target=lambda : spotCam()).start()
threading.Thread(target=lambda : detect_Helmet()).start()
arduino_Thread()
```



```python
import time
import serial
import threading

class ArduinoController(threading.Thread):

    USBPORT = "COM3"

    def __init__(self) -> None:
        threading.Thread.__init__(self)
        self.num_distance = 0
        self.casque = True
        self.arduino = serial.Serial(self.USBPORT, 9600, timeout=0)

    def run(self):
        while True:
            if (self.arduino.inWaiting() > 0):
                myData = self.arduino.readline().decode('ascii').strip()
                distancia = float(myData)
                if distancia > 25 and self.casque:
                    self.num_distance += 1
                else: self.num_distance = 0

                if self.num_distance>5:
                    self.casque = False
                    print("pas de casque")
                time.sleep(0.5)
```

Links:

https://viso.ai/deep-learning/object-detection/

https://machinelearningmastery.com/object-recognition-with-deep-learning/

https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb#scrollTo=hkAzDWJ7cWTr

https://imageai.readthedocs.io/en/latest/detection/index.html