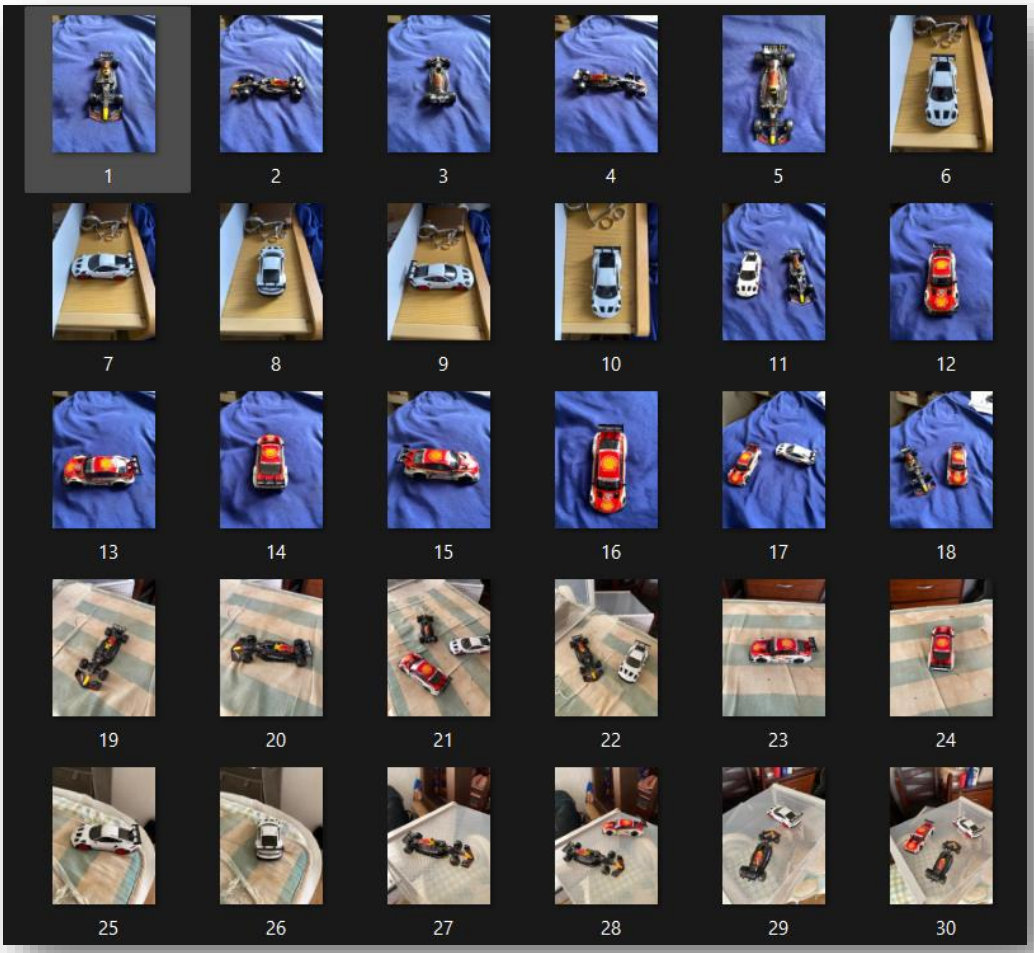
Oscar Israel Valencia Gómez – 22310210 6G

Proyecto: “*Racing Cars*”

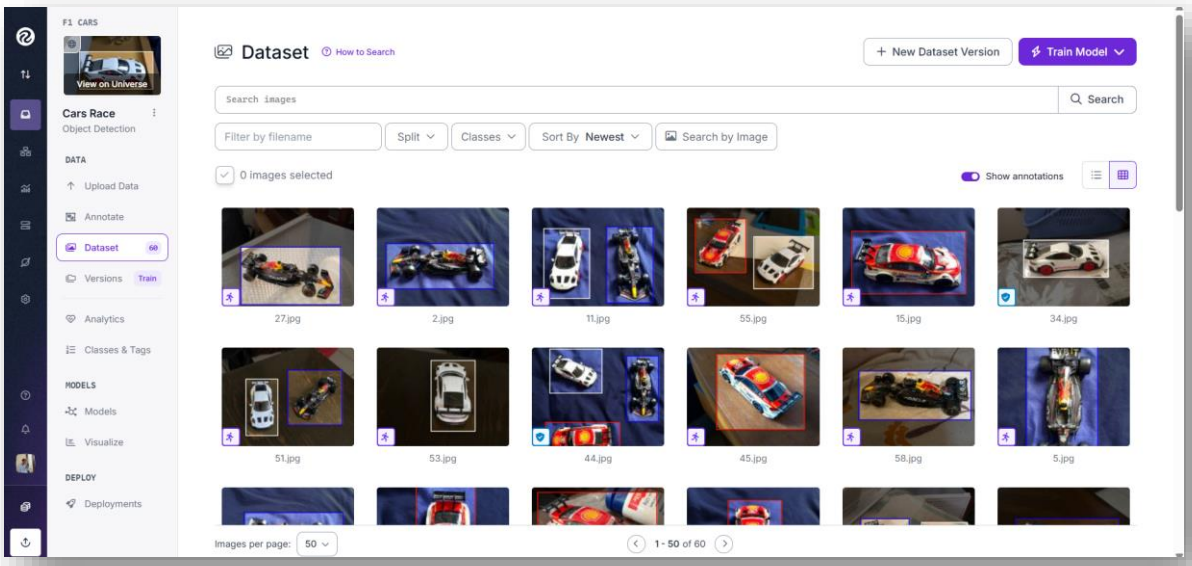
De las carreras a tu espacio de trabajo, lo que es este proyecto consiste que por medio de un entrenamiento de IA (YOLO) se detecte diferentes carritos de carreras que aparezca su modelo y especificación, también al momento de que sean detectados prenda un led por medio de un Arduino de su respectivo color.

DESARROLLO DEL PROYECTO:

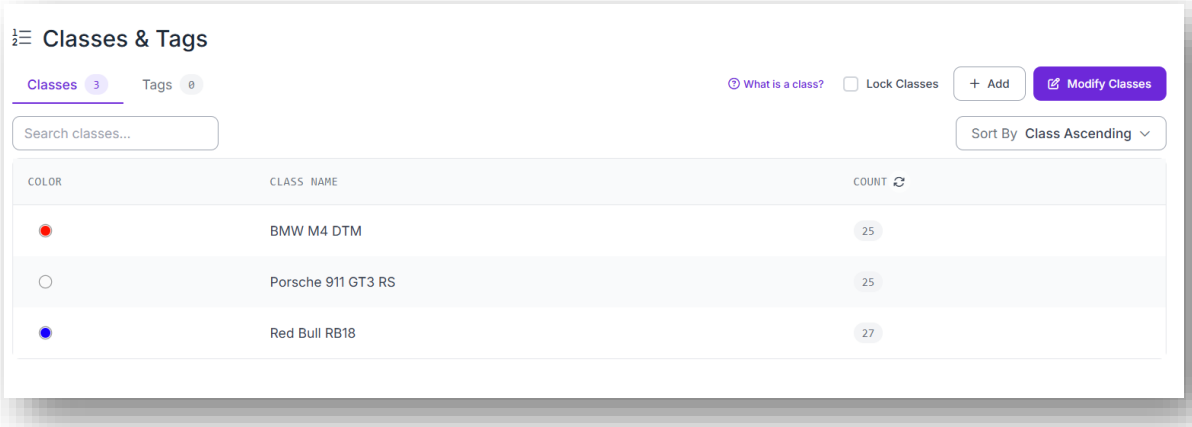
Cómo comienzo de lo que fue el desarrollo del proyecto es que al principio tenía que tomar diferentes imágenes de los carritos en diferentes posiciones, ángulos, contrastes de iluminación y brillo para que fuera de lo mejor la detección para la IA



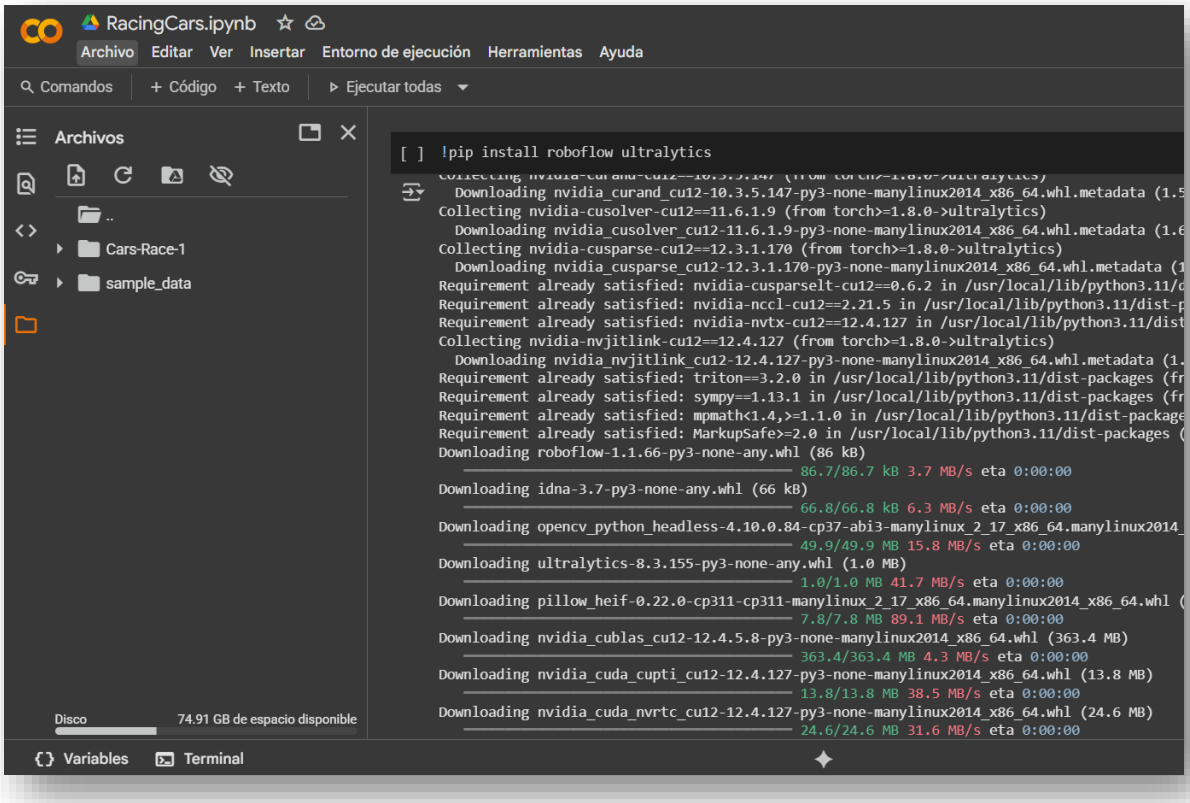
Ya teniendo esas imágenes, lo que hice después fue importar todas esas imágenes a un programa que es “roboflow” con el cual este me iba dar mi dataset para comenzar el entrenamiento de la IA.



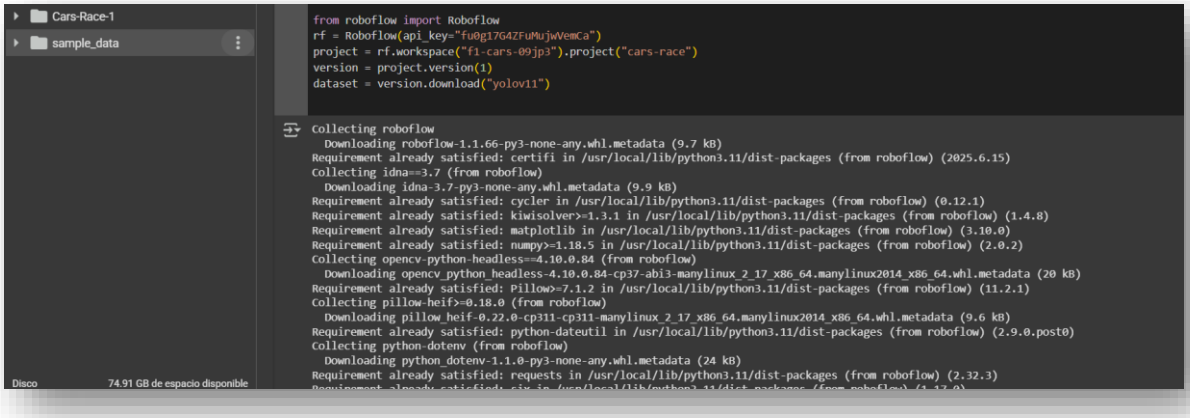
Con base en eso tuve que crear las clases para lo que iba ser cada carro y tomar las medidas para la detección de cada imagen y así hacerlo con cada una de las imágenes que tomé, ya con esto lo entrene de tal que era 70% train 20% test y 10% valid, haciendo que ya el programa me generara mi dataset para comenzar a entrenarlo.



Me pase a “Google Colab” para entrenar ahí la IA y crear un nuevo cuaderno con el nombre del proyecto para descargar las librerías y pasar ahí la dataset que me genero el anterior programa



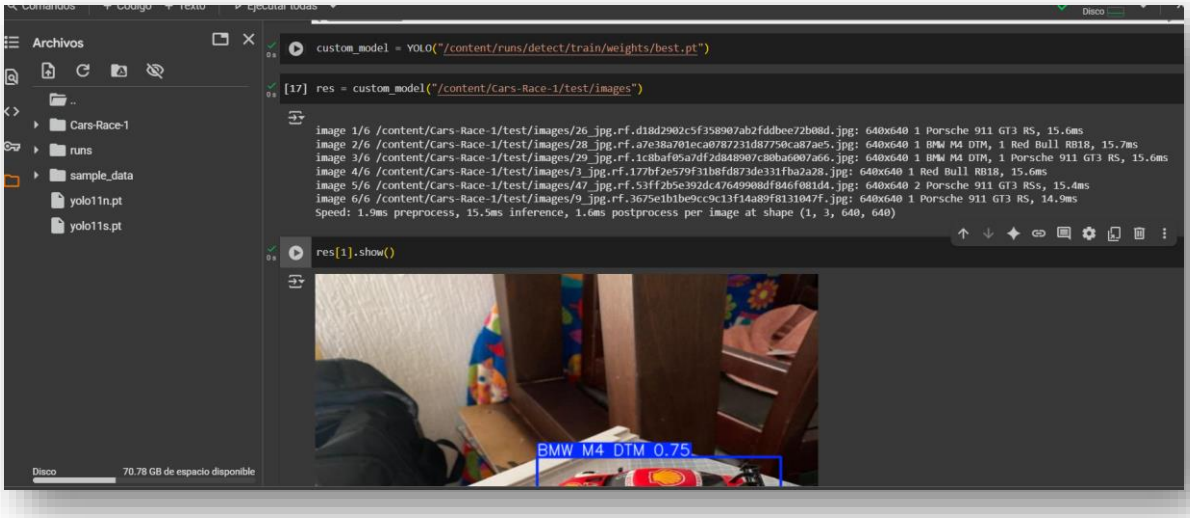
Ahora ya tuve que meter lo que fue el dataset y me generó unos archivos ya del programa con lo que lo entrené con yolov11



Despues ya compile las demás librerías de lo ultralytics y el roboflow, junto al archivo previo que me generó que era el data.yaml

```
[ ] from ultralytics import YOLO
    model = YOLO("yolo11s.pt")

Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/yolo11s.pt to 'yolo11s.pt'...
100%|██████████| 18.4M/18.4M [00:00<00:00, 143MB/s]

[ ] data_path = "/content/Cars-Race-1/data.yaml"
    results = model.train(data=data_path,
                          epochs=15,
                          imgsz=640)

Ultralytics 8.3.155 🚀 Python-3.11.13 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
engine/trainer: agnostic_nms=False, amp=True, augment=False, auto_augment=randaugment, batch=16, bgr=0.0, box=7.5, cache=False, cfg=None, clas
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
100%|██████████| 755k/755k [00:00<00:00, 14.5MB/s]Overriding model.yaml nc=80 with nc=3

                   from  n    params  module                                       arguments
  0                  -1  1       928  ultralytics.nn.modules.conv.Conv             [3, 32, 3, 2]
  1                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  2                  -1  1     26080  ultralytics.nn.modules.block.C3k2            [64, 128, 1, False, 0.25]
  3                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
  4                  -1  1    103360  ultralytics.nn.modules.block.C3k2            [128, 256, 1, False, 0.25]
  5                  -1  1    590336  ultralytics.nn.modules.conv.Conv             [256, 256, 3, 2]
  6                  -1  1    346112  ultralytics.nn.modules.block.C3k2            [256, 256, 1, True]
  7                  -1  1   1180672  ultralytics.nn.modules.conv.Conv             [256, 512, 3, 2]
  8                  -1  1   1380352  ultralytics.nn.modules.block.C3k2            [512, 512, 1, True]
  9                  -1  1    656896  ultralytics.nn.modules.block.SPPF            [512, 512, 5]
 10                  -1  1    990976  ultralytics.nn.modules.block.C2PSA           [512, 512, 1]
 11                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
```

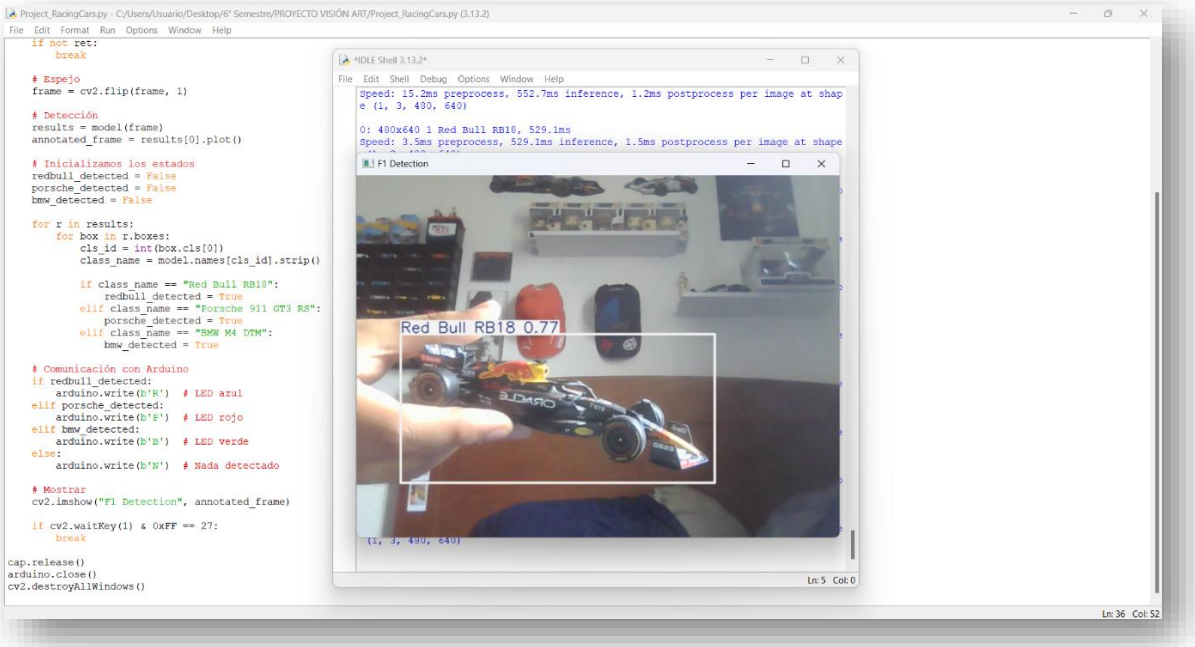
Siguiendo con el proceso del entrenamiento ya tuve que hacer otros comando los cuales en los archivos me generó un modelo de entrenamiento que se llama “best.pt” que como su nombre lo indica fue el que se hizo para lo que es la IA y ya lo tuve que testear

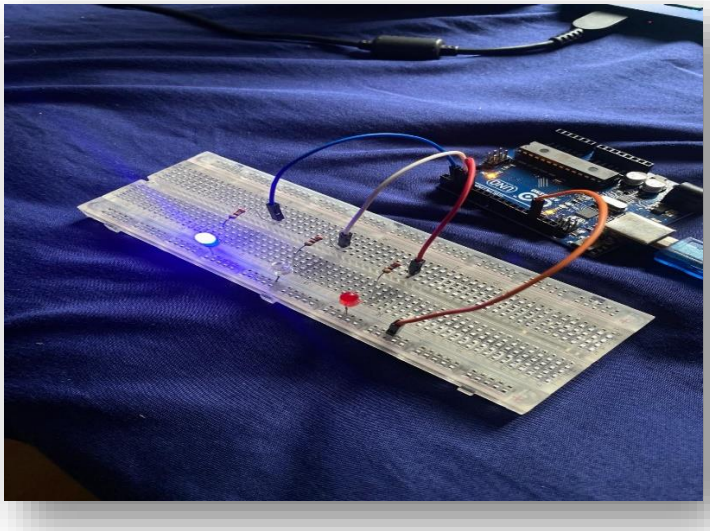


RESULTADOS:

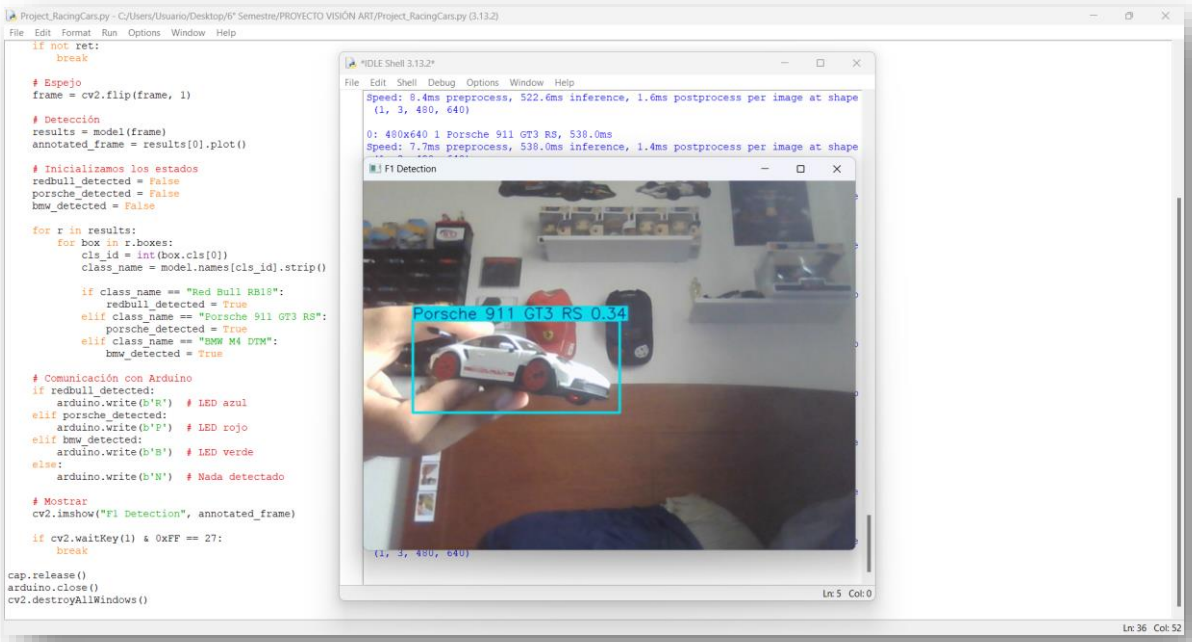
Ya con mi modelo de IA creado ahora lo que tocaba era probarlo ya en el Python y solo descargando el archivo best.pt y con mi código que tenía tanto para Python cómo para el Arduino, estas son las pruebas:

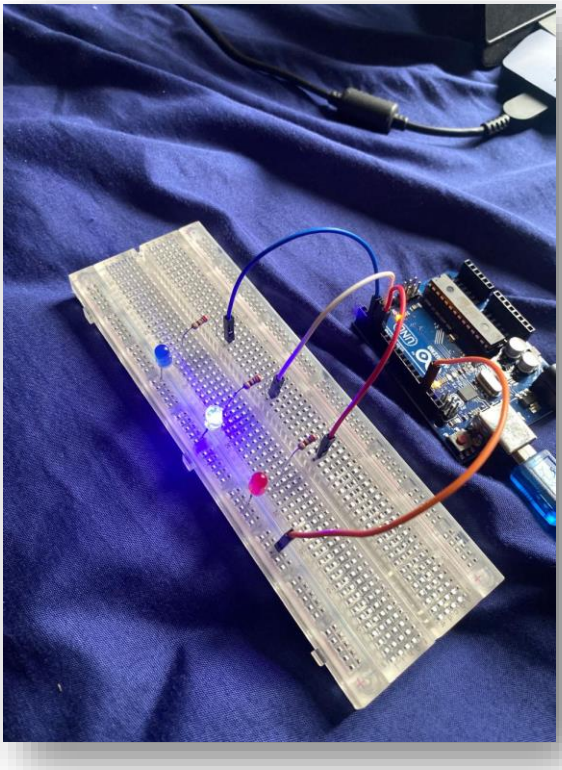
PRUEBA 1:



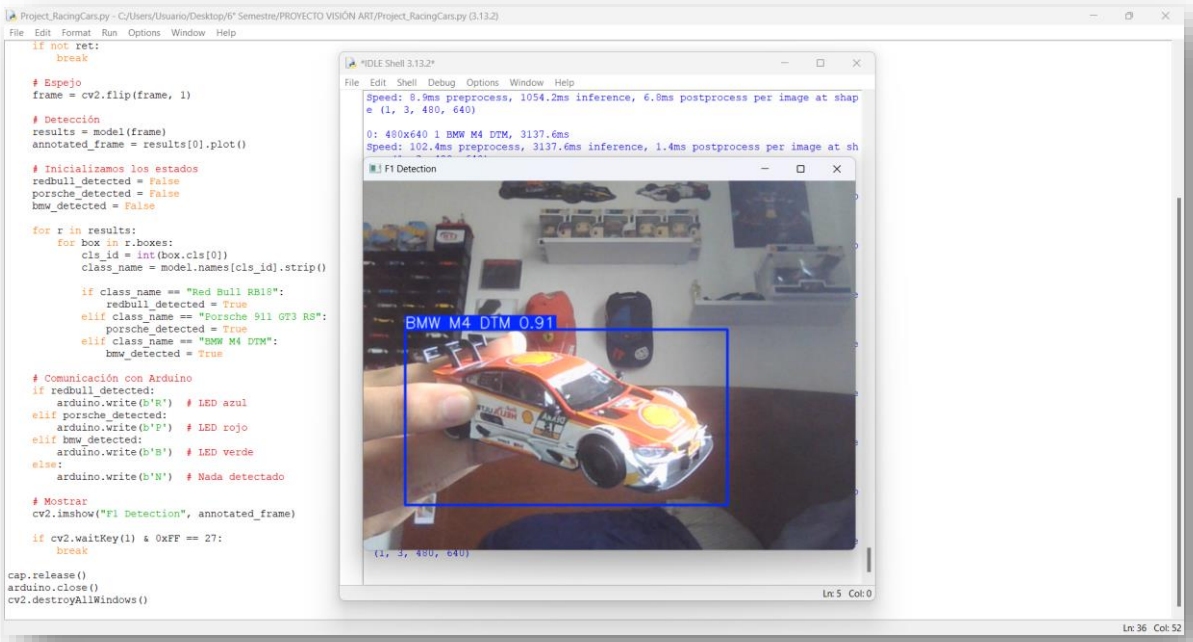


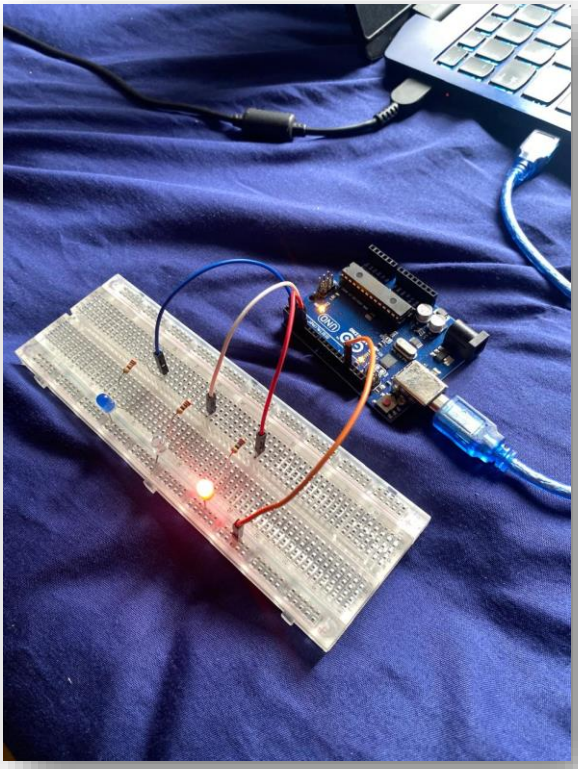
PRUEBA 2:


Project_RacingCars.py - C:/Users/Usuario/Desktop/6° Semestre/PROYECTO VISIÓN ART/Project_RacingCars.py (3.13.2)
File Edit Format Run Options Window Help
if not ret:
break
# Espejo
frame = cv2.flip(frame, 1)
# Detección
results = model(frame)
annotated_frame = results[0].plot()
# Inicializamos los estados
redbull_detected = False
porsche_detected = False
bmw_detected = False
for r in results:
for box in r.boxes:
cls_id = int(box.cls[0])
class_name = model.names[cls_id].strip()
if class_name == "Red Bull RB18":
redbull_detected = True
elif class_name == "Porsche 911 GT3 RS":
porsche_detected = True
elif class_name == "BMW M4 DTM":
bmw_detected = True
# Comunicación con Arduino
if redbull_detected:
arduino.write(b'R') # LED azul
elif porsche_detected:
arduino.write(b'P') # LED rojo
elif bmw_detected:
arduino.write(b'B') # LED verde
else:
arduino.write(b'N') # Nada detectado
# Mostrar
cv2.imshow("F1 Detection", annotated_frame)
if cv2.waitKey(1) & 0xFF == 27:
break
cap.release()
arduino.close()
cv2.destroyAllWindows()
IDLE Shell 3.13.2
File Edit Shell Debug Options Window Help
Speed: 8.4ms preprocess, 522.6ms inference, 1.6ms postprocess per image at shape
(1, 3, 480, 640)
0: 480x640 1 Porsche 911 GT3 RS, 538.0ms
Speed: 7.7ms preprocess, 538.0ms inference, 1.4ms postprocess per image at shape
F1 Detection
Porsche 911 GT3 RS 0.34
(1, 3, 480, 640)
Ln: 5 Col: 0
Ln: 36 Col: 52



PRUEBA 3:


Project_RacingCars.py - C:/Users/Usuario/Desktop/6° Semestre/PROYECTO VISIÓN ART/Project_RacingCars.py (3.13.2)
File Edit Format Run Options Window Help
if not ret:
break
# Espejo
frame = cv2.flip(frame, 1)
# Detección
results = model(frame)
annotated_frame = results[0].plot()
# Inicializamos los estados
redbull_detected = False
porsche_detected = False
bmw_detected = False
for r in results:
for box in r.boxes:
cls_id = int(box.cls[0])
class_name = model.names[cls_id].strip()
if class_name == "Red Bull RB18":
redbull_detected = True
elif class_name == "Porsche 911 GT3 RS":
porsche_detected = True
elif class_name == "BMW M4 DTM":
bmw_detected = True
# Comunicación con Arduino
if redbull_detected:
arduino.write(b'R') # LED azul
elif porsche_detected:
arduino.write(b'P') # LED rojo
elif bmw_detected:
arduino.write(b'B') # LED verde
else:
arduino.write(b'N') # Nada detectado
# Mostrar
cv2.imshow("F1 Detection", annotated_frame)
if cv2.waitKey(1) & 0xFF == 27:
break
cap.release()
arduino.close()
cv2.destroyAllWindows()
IDLE Shell 3.13.2
File Edit Shell Debug Options Window Help
Speed: 8.9ms preprocess, 1054.2ms inference, 6.8ms postprocess per image at shape (1, 3, 480, 640)
0: 480x640 1 BMW M4 DTM, 3137.6ms
Speed: 102.4ms preprocess, 3137.6ms inference, 1.4ms postprocess per image at sh
F1 Detection
BMW M4 DTM 0.91
(1, 3, 480, 640)
Ln: 5 Col: 0
Ln: 36 Col: 52



MI CÓDIGO DE PYTHON:

```python
from ultralytics import YOLO
import cv2
import serial
import time

# Inicializar Arduino (ajusta COM si es diferente)
arduino = serial.Serial('COM3', 9600)
time.sleep(2)

# Cargar modelo YOLO
model = YOLO("best.pt")

# Cámara
cap = cv2.VideoCapture(0)

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    # Espejo
    frame = cv2.flip(frame, 1)

    # Detección
    results = model(frame)
    annotated_frame = results[0].plot()

    # Inicializamos los estados
    redbull_detected = False
    porsche_detected = False
    bmw_detected = False

    for r in results:
        for box in r.boxes:
            cls_id = int(box.cls[0])
            class_name = model.names[cls_id].strip()

            if class_name == "Red Bull RB18":
                redbull_detected = True
            elif class_name == "Porsche 911 GT3 RS":
                porsche_detected = True
            elif class_name == "BMW M4 DTM":
                bmw_detected = True
```

```
    # Comunicación con Arduino
    if redbull_detected:
        arduino.write(b'R')  # LED azul
    elif porsche_detected:
        arduino.write(b'P')  # LED rojo
    elif bmw_detected:
        arduino.write(b'B')  # LED verde
    else:
        arduino.write(b'N')  # Nada detectado

    # Mostrar
    cv2.imshow("F1 Detection", annotated_frame)

    if cv2.waitKey(1) & 0xFF == 27:
        break

cap.release()
arduino.close()
cv2.destroyAllWindows()
```

MI CÓDIGO DE ARDUINO:

```
void setup() {
  Serial.begin(9600);
  pinMode(2, OUTPUT); // LED Azul (Red Bull RB18)
  pinMode(3, OUTPUT); // LED Rojo (Porsche 911 GT3 RS)
  pinMode(4, OUTPUT); // LED Verde (BMW M4 DTM)
}

void loop() {
  if (Serial.available()) {
    char data = Serial.read();

    // Apagar todos primero
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);

    // Encender solo el que corresponde
    if (data == 'R') digitalWrite(2, HIGH);     // Red Bull
    else if (data == 'P') digitalWrite(3, HIGH); // Porsche
    else if (data == 'B') digitalWrite(4, HIGH); // BMW
    // 'N' ya los deja apagados
  }
}

```

ENLACE DE MI REPOSITORIO DE GITHUB:

https://github.com/Ing-OscarValencia/VisionArtificialOGV-2025.git