Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas
Arquitectura de Computadores y Ensambladores 2
Primer Semestre 2024

Computer Vision & Arduino

Juan Pedro Valle Lema, Mauricio Antonio Castro Guerra , Diego Facundo Pérez Nicolau, Diana Estefanía Berducido Domingo, Marcos Arnoldo Itzep Ixmay
***202101648, 202100299, 202106538, 202000277,* 201907156**

Introducción

OpenCV es una librería que nos permite realizar múltiples actividades relacionadas con el escaneo de objetos, rostros o lo que nosotros deseemos que pueda observar la cámara. En esta ocasión veremos la implementación de esta junto a Python, Arduino y Processing para poder escanear rostros, teniendo en arduino una representación física de cuando hay y no hay rostros y en Processing una representación gráfica de cuándo no hay rostros. En esta práctica se desarrollará un dispositivo de detección de rostros humanos utilizando una cámara integrada y la biblioteca OpenCV de Python.

Objetivos

Generales

- Desarrollar una aplicación de reconocimiento facial utilizando OpenCV, Python y Processing que permita identificar rostros en tiempo real.

Específicos

- Entender los fundamentos del reconocimiento facial.
- Aprender a utilizar OpenCV y Python para trabajarlos en conjunto en el proceso de imágenes y tareas de visión por computadora.

Desarrollo de la práctica Stack Design Framework

En el mundo actual, la tecnología juega un papel crucial en casi todos los aspectos de nuestra vida diaria. Sin embargo, con el creciente uso de la tecnología, también surgen nuevos desafíos y problemas. Uno de estos desafíos es el reconocimiento facial, una tecnología que, aunque prometedora, todavía tiene muchos obstáculos que superar.

El reconocimiento facial es una forma de biometría que utiliza medidas y características faciales para identificar individuos. Aunque esta tecnología ha existido durante décadas, su uso se ha vuelto más prevalente con el advenimiento de las redes neuronales y el aprendizaje profundo. Sin embargo, a pesar de sus avances, el reconocimiento facial todavía enfrenta varios problemas, como la precisión y la privacidad.

En este informe, nos proponemos abordar estos problemas mediante el desarrollo de una aplicación de reconocimiento facial utilizando OpenCV, Python y Processing. Nuestro objetivo es crear un sistema que no solo sea preciso, sino que también respete la privacidad del usuario. A través de este trabajo, esperamos contribuir a la mejora de la tecnología de reconocimiento facial y, en última instancia, a la sociedad en general.

Detector de Rostros

Infraestructura del Producto

- **Hardware**
  - Arduino
  - Jumpers/Cables y LED’s
  - Case del Encapsulamiento
  - Cartón
- **Software**
  - Código de Arduino
  - Código de Python
  - Código de Processing

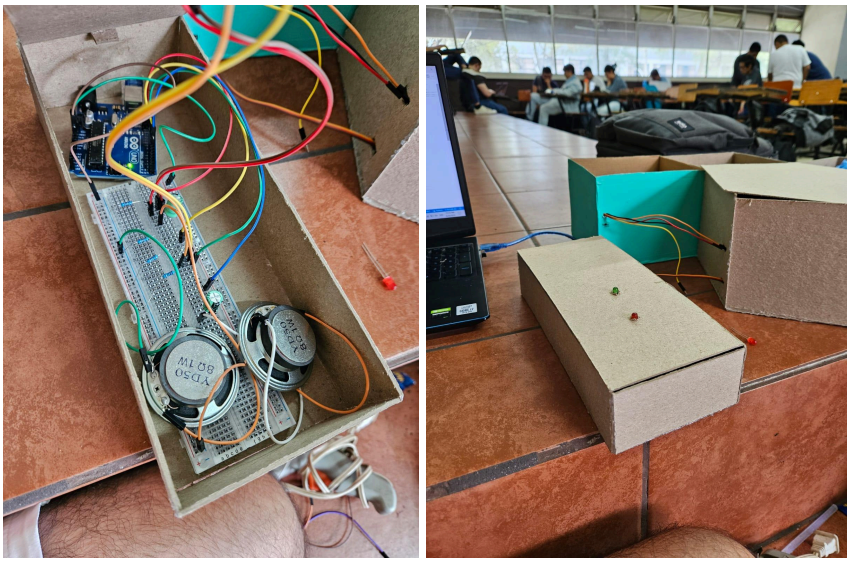

Figura 1: Prototipo del Producto

Sensores

- Cámara Web Integrada en Laptop

OPCIONES SECUNDARIAS

Conectividad

El dispositivo principal es un Arduino, el cual por medio de la cámara web recolecta imágenes y las procesa OpenCV para reconocer rostros a través de Python. Estos datos son enviados por medio de comunicación serial y recibidos por la aplicación que funciona con Python. Posteriormente los datos recibidos son guardados en una base de datos local (el conteo que hace mientras la cámara está reconociendo rostros), estos mismos pueden ser consultados en un txt.

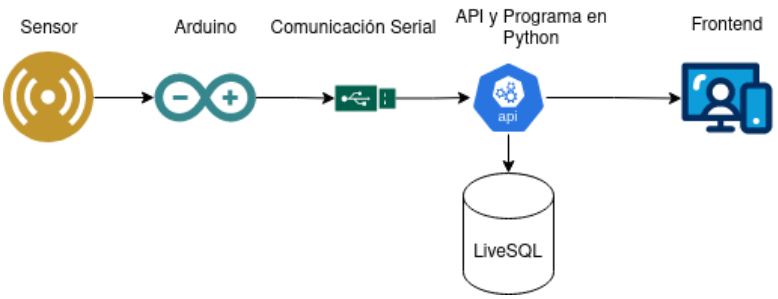

Figura 2: Conectividad del Producto

Tamaño del objeto

La maqueta tiene una altura de 10 cm por un ancho de 20 cm de largo.

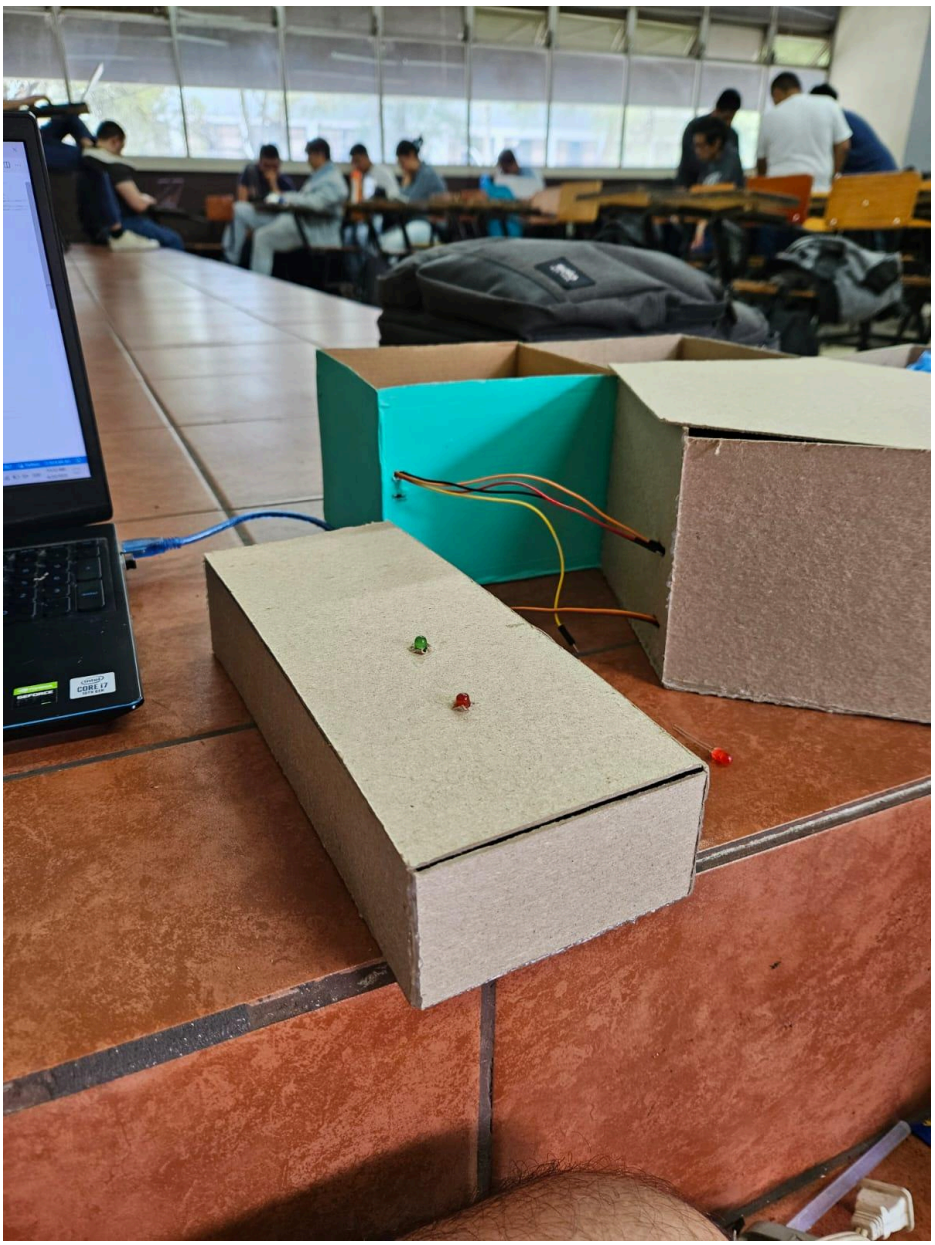

Figura 3: Imagen del Prototipo

Entorno del objeto
La idea es usarlo como un dispositivo móvil que podría ser conectado a cualquier computadora que tenga una conexión a una cámara web integrada o externa.

Consumo de Energía
De 46-90 mA

Analítica

La analítica parte de este dispositivo fue enfocada en el sector de datos, brindando un análisis descriptivo de ellos, esto con el fin de poder encontrar tendencias y patrones respecto a los rostros presentados a la cámara:

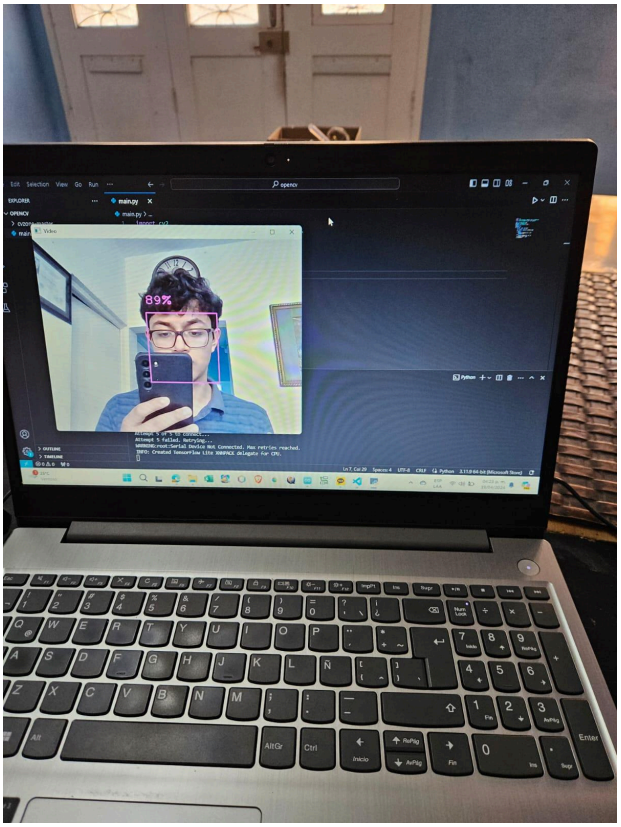

Figura 4: Visualización porcentaje facial

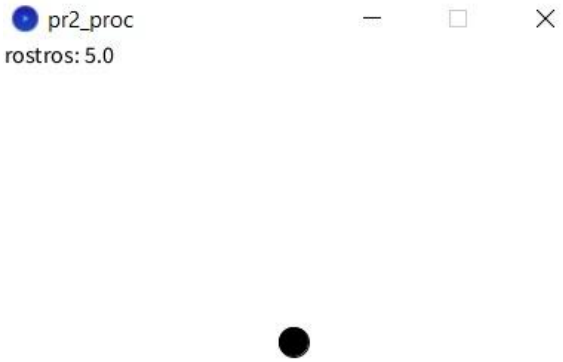

Figura 5: Interfaz por Processing

pr2 proc

```
int contador = 0;
String archivoCompartido = "contador.txt";
float r=0;

void setup() {
  size(400, 400);
}

void draw() {
  background(255);

  // Leer el valor del contador desde el archivo compartido
  String[] lines = loadStrings(archivoCompartido);
  if (lines.length > 0) {
    contador = int(lines[0]);
    println("Rostros de Python: " + contador);
  }

  // Dibujar el círculo con el valor actual del contador
  float circleSize = map(contador, 0, 100, 0, width);
  ellipse(width/2, height/2, circleSize, circleSize);
  r=contador;
  text("rostros: "+r,5,11);
  textSize(16);
  fill(0);


}
```

Figura 6: Código Processing

```python
import cv2
from cvzone.FaceDetectionModule import FaceDetector
from cvzone.SerialModule import SerialObject
import time

arduino = SerialObject('COM6')
cap = cv2.VideoCapture(0)
detector = FaceDetector()


caras = []
contador = 0
while True:
    success, img = cap.read()

    img, bBoxes = detector.findFaces(img)
    file = open("contador.txt","r")
    datos_previos = int(file.read().strip())

    if len(bBoxes) > 0:
        score = bBoxes[0]['score'][0]
        if score > 0.9:
            arduino.sendData([1,0, 1, 0])
            contador =datos_previos+1
            print(contador)
            face = "VERDADERO"
            cv2.putText(img, f'CARA DETECTADA: {(face)}', (20, 70), cv2.FONT_HERSHEY_PLAIN,1, (0, 255, 0), 2)
            file = open("contador.txt","w")
            file.write(str(contador))
            file.close()
            time.sleep(1)
        elif score <0.9 :
            arduino.sendData([0,1, 0, 1])
            face = "FALSO"
            cv2.putText(img, f'CARA DETECTADA: {(face)}', (20, 70), cv2.FONT_HERSHEY_PLAIN,1, (0, 255, 0), 2)
            time.sleep(4)
```

Figura 7: Código Python

Imagen prototipo

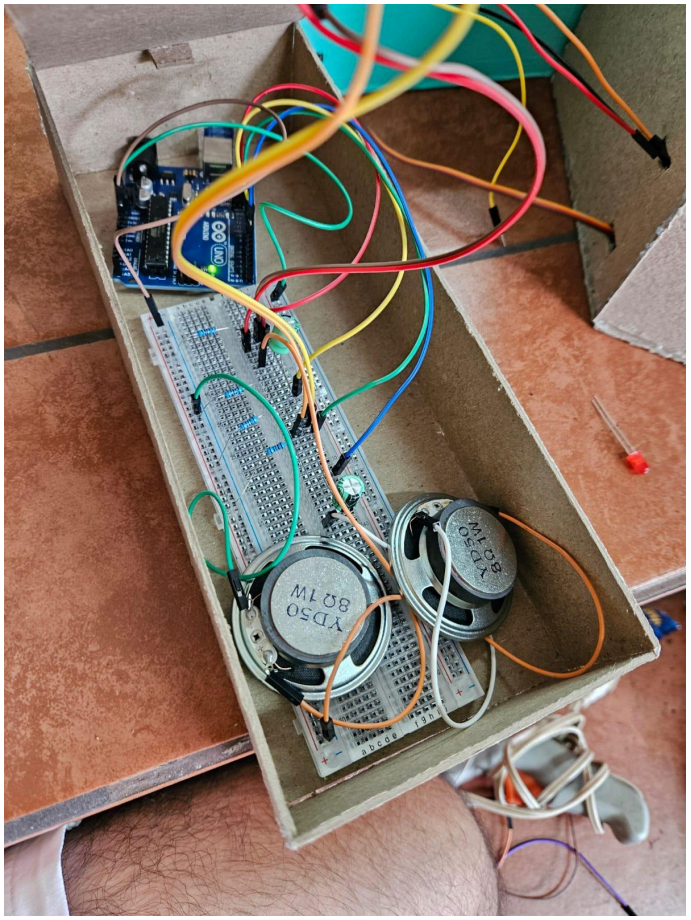

Figura 8: Imagen del Prototipo

Link repositorio: https://github.com/DFacundoPerezN/ACE2_1S24_G7.git