Proyecto Timbre Inteligente

1st Sergio Pereyra ,2nd Carlos E. Warzyca
sergio_pereira_89@hotmail.com, carlos.e.warzyca@gmail.com

***Abstract*—Este trabajo presenta el desarrollo de un sistema de control de acceso inteligente basado en reconocimiento facial y de voz. El sistema integra librerías de visión artificial (OpenCV), procesamiento de audio (PyAudio), almacenamiento en la nube (Google Drive API), mensajería instantánea (WhatsApp API) y gestión de registros en base de datos MySQL. El objetivo es automatizar el acceso residencial, garantizando seguridad y trazabilidad mediante la combinación de tecnologías de identificación biométrica y comunicación en tiempo real.**

***Index Terms*—OpenCV, Pyaudio, Google API, WhatsApp API, MySQL, seguridad, acceso automático.**

I. Introducción

La seguridad en accesos residenciales constituye un desafío creciente en el ámbito de la domótica y la automatización. El presente proyecto propone un timbre inteligente capaz de identificar usuarios mediante reconocimiento facial y de voz, otorgando acceso únicamente a personas autorizadas. En caso de no reconocimiento, el sistema envía los datos capturados a los administradores para su validación manual.

II. Desarrollo

El sistema se activa al presionar la tecla de simulación (”Q”), iniciando la captura de imagen y audio. Los datos se procesan mediante algoritmos de reconocimiento y se almacenan en la nube. Según el resultado, se habilita o deniega el acceso, registrando cada evento en la base de datos.

A. Diagrama de flujo:

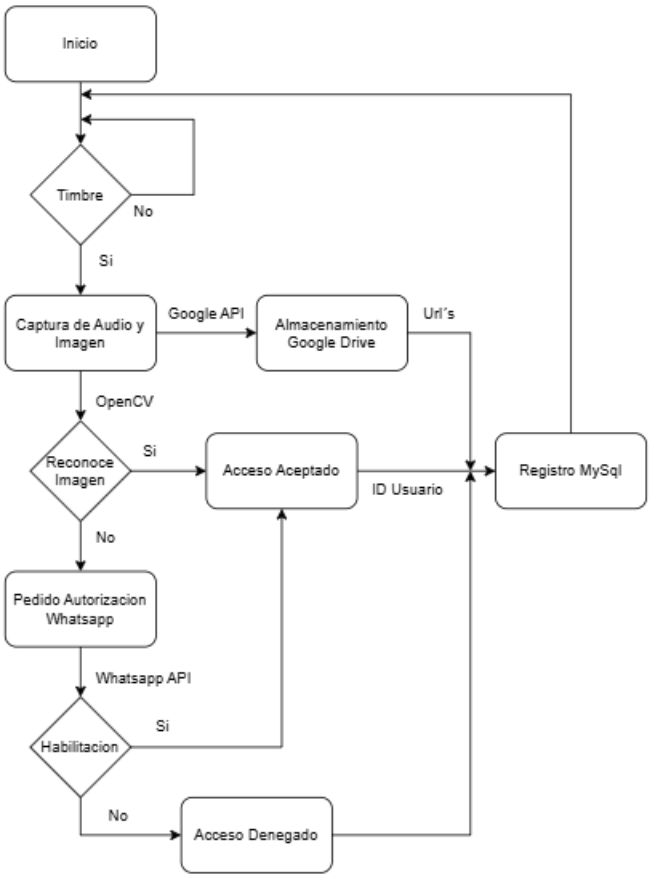

Fig. 1. Diagrama de flujo

B. Paquetes instalados en Python

TABLE I
PIP LIST:

Package	Version
pip	25.3
pathlib	1.0.1
opencv-python	4.12.0.88
face-recognition	1.3.0
face_recognition_models	0.3.0
google-api-core	2.25.1
google-api-python-client	2.182.0
google-auth	2.40.3
google-auth-httplib2	0.2.0
google-auth-oauthlib	1.2.2
googleapis-common-protos	1.70.0
PyAudio	0.2.14
mysql-connector-python	9.5.0
asyncio	4.0.0
python-telegram-bot	22.5

C. Timbre:

Condicional de timbre presionado (en aplicación de PC se emula con una tecla), si ocurre este hecho, se procede a realizar la captura de datos

D. Captura de Audio y Video:

- Se utiliza OpenCV junto con clasificadores Haar para detectar rostros. Posteriormente, se aplican librerías de reconocimiento facial para comparar con la base de datos de usuarios registrados.

```
faceClassif=cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")

cap=cv2.VideoCapture(0)
fps=cap.get(cv2.CAP_PROP_FPS)
size=(int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)),int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)))

while(True):
    ret,frame = cap.read()
    if(ret==True):
        frame=cv2.flip(frame,1)
        orig=frame.copy()
        faces=faceClassif.detectMultiScale(frame,
                                           scaleFactor=1.1,
                                           minNeighbors=5)

        for (x,y,w,h) in faces:
            face=orig[y:y+h,x:x+w]
            face=cv2.cvtColor(face,cv2.COLOR_BGR2RGB)
            actual_face_encoding=face_recognition.face_encodings(face, known_face_locations=[(0, w,h,0)])[0]
            result=face_recognition.compare_faces(facesEncodings, actual_face_encoding)
            if True in result:
                index=result.index(True)
                name =facesNames[index]
                color=(125,220,0)
            else:
                name="Desconocido"
                color=(50,50,255)

            cv2.rectangle(frame, (x,y+h),(x+w,y+h+30),color,-1)
            cv2.rectangle(frame, (x,y),(x+w,y+h),color,2)
            cv2.putText(frame,name,(x,y+h+25),2,1,(255,255,255),2,cv2.LINE_AA)
            cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)
        cv2.imshow("Frame",frame)
        if (cv2.waitKey(1)&0xFF==ord("q")):
            cv2.imwrite('portero.jpg',frame)
            break
cap.release()
cv2.destroyAllWindows()
```

Fig. 2. captura del rostro

- El audio se graba en formato WAV mediante PyAudio, con una duración de 3 segundos, y se convierte a MP3 para su almacenamiento y transmisión.

```
duracion=3
archivo="puerta.wav"

audio=pyaudio.PyAudio()

stream=audio.open(format=pyaudio.paInt16,channels=1,
                  rate=44100,input=True,
                  frames_per_buffer=1024)

print("Grabando ...")
frames=[]

for i in range(0,int(44100/1024*duracion)):
    data=stream.read(1024)
    frames.append(data)

print("Grabacion a terminado ")

stream.stop_stream()
stream.close()
audio.terminate()

waveFile=wave.open(archivo,'wb')
waveFile.setnchannels(1)
waveFile.setsampwidth(audio.get_sample_size(pyaudio.paInt16))
waveFile.setframerate(44100)
waveFile.writeframes(b''.join(frames))
waveFile.close()
```

Fig. 3. captura de audio

E. Almacenamiento:

Los archivos de imagen y audio se suben automáticamente a Google Drive, generando enlaces compartidos para su posterior consulta.

```
creds = None

if os.path.exists("token.json"):
  creds = Credentials.from_authorized_user_file("token.json", SCOPES)

if not creds or not creds.valid:
  if creds and creds.expired and creds.refresh_token:
    creds.refresh(Request())
  else:
    flow = InstalledAppFlow.from_client_secrets_file(
        "credentials.json", SCOPES
    )
    creds = flow.run_local_server(port=0)

  with open("token.json", "w") as token:
    token.write(creds.to_json())
try:
  service = build("drive", "v3", credentials=creds)

  file_metadata = {"name": "portero.jpg", "parents": [folder_id]}
  media = MediaFileUpload(
      "portero.jpg", mimetype="image/jpeg", resumable=True
  )

  file = (
      service.files()
      .create(body=file_metadata, media_body=media, fields="id")
      .execute()
  )
  print(f'File ID: "{file.get("id")}".')
  permission = {
      'type': 'anyone',
      'role': 'reader',
  }
```

Fig. 4. guardado en drive

```
  shareable_urlfoto = file_details.get('webViewLink')
  print(f'Shareable URL: {shareable_urlfoto}')
  file_metadata = {"name": "puerta.wav", "parents": [folder_id]}
  media = MediaFileUpload(
      "puerta.wav", mimetype="audio/wav", resumable=True
  )

  file = (
      service.files()
      .create(body=file_metadata, media_body=media, fields="id")
      .execute()
  )
  print(f'File ID: "{file.get("id")}".')
  permission = {
      'type': 'anyone',
      'role': 'reader',
  }
  service.permissions().create(fileId=file.get("id"), body=permission).execute()
  file_details = service.files().get(fileId=file.get("id"), fields='webViewLink').execute()
  shareable_urlaudio = file_details.get('webViewLink')
  print(f'Shareable URL: {shareable_urlaudio}')
except HttpError as error:

  print(f"An error occurred: {error}")
```

Fig. 5. finalización del guardado en drive

F. Notificación y Autorización:

Si el reconocimiento falla, los archivos se envían vía Telegram API al administrador, quien decide si habilitar o denegar el acceso.

```
#Funcion para enviar mensaje por telegram
async def envia_mensaje():
    bot=Bot(token=TOKEN)
    buttons=[[KeyboardButton("Aceptar")], [KeyboardButton("Rechazar")], [KeyboardButton("Esperar")]]
    await bot.send_photo(chat_id=chat_id, photo=open('portero.jpg', 'rb'))
    await bot.send_audio(chat_id=chat_id, audio=open('puerta.wav', 'rb'))
    await bot.send_message(chat_id=chat_id, text=message, reply_markup=ReplyKeyboardMarkup(buttons))
```

Fig. 6. Envió Mensaje Multimedia Telegram

G. Habilitación:

En caso de ser positiva se le asigna el estado de habilitado, en caso de ser negativa se le asigna el estado de rechazado, ambos con ID: N/N

H. Registro Mysql:

Cada evento se almacena en MySQL, incluyendo estado, usuario, fecha/hora y enlaces de los archivos capturados.

```
basedatos = mysql.connector.connect(user='root',
                                    password='',
                                    host='localhost',
                                    database='timbrepes',
                                    port='3306')
print(basedatos)
micursorbasedatos=basedatos.cursor()
"""
micursorbasedatos.execute("DESCRIBE accesousuario")
for x in micursorbasedatos:
    print(x)
"""
#Obtener fecha y hora actual
fechaactual=datetime.now()
formato_fechaactual=fechaactual.strftime('%Y-%m-%d %H:%M:%S')
#Insertar datos en la tabla accesousuario
micursorbasedatos.execute("INSERT INTO accesousuario (Estado, Usuario, Fecha, EnlaceFoto, EnlaceAudio) VALUES (%s, %s, %s, %s, %s)",
                          ("Aceptado",
                           name,
                           formato_fechaactual,
                           shareable_urlfoto,
                           shareable_urlaudio))
basedatos.commit()

if name=="Desconocido":
    print("Rostro no reconocido")

else:
    name = name[:-4]
    print("Rostro reconocido:", name)
```

Fig. 7. actualización base de datos

III. Conclusión

El Timbre Inteligente constituye una solución innovadora para la automatización del acceso residencial, integrando tecnologías de visión artificial, procesamiento de audio y servicios en la nube. El sistema demuestra la viabilidad de combinar herramientas de software libre con APIs comerciales para lograr un control de acceso seguro y eficiente. No obstante, para alcanzar un nivel profesional y comercial, es necesario fortalecer la seguridad, mejorar la precisión de los algoritmos biométricos y garantizar la escalabilidad del sistema. Con estas mejoras, el proyecto puede evolucionar hacia un producto competitivo en el ámbito de la domótica y la seguridad inteligente.

Fig. 8. Enlace al github del proyecto (click en la imagen)

A. Implementación

Se adjunta el siguiente vídeo donde se muestra la implementacion del sistema (click en la imagen)