UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Diseño e Implementación de un Sistema Embebido de Reconocimiento Facial para el Control de Acceso usando Deep Learning

Gustavo David Orna Villalta

Ingeniería Electrónica

Trabajo de integración curricular presentado como requisito
para la obtención del título de
Ingeniero Electrónico

Quito, 19 de diciembre de 2019

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

COLEGIO DE CIENCIAS E INGENIERÍAS

HOJA DE CALIFICACIÓN
DE TRABAJO DE INTEGRACIÓN CURRICULAR

Diseño e Implementación de un Sistema Embebido de Reconocimiento Facial para el Control de Acceso usando Deep Learning

Gustavo David Orna Villalta

Calificación:

Nombre del profesor, Título académico **Diego Benítez, Ph.D.**

Firma del profesor: _____

Quito, 19 de diciembre de 2019

I. Derechos de Autor

Por medio del presente documento certifico que he leído todas las Políticas y Manuales de la Universidad San Francisco de Quito USFQ, incluyendo la Política de Propiedad Intelectual USFQ, y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo quedan sujetos a lo dispuesto en esas Políticas.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Firma del estudiante: _____

Nombres y apellidos: Gustavo David Orna Villalta

Código: 00122532

Cédula de identidad: 0923643134

Lugar y fecha: Quito, 19 de diciembre de 2019

RESUMEN

Este proyecto presenta la forma en la que se utilizó el procesador Neural Compute Stick 2 para generar un sistema de control de accesos con reconocimiento facial de tipo embebido. El sistema está basado en un código de reconocimiento facial desarrollado en el lenguaje de programación Python, mismo que, a través de una cámara web, obtiene las imágenes en tiempo real para efectuar una comparación con los rostros guardados en una base de datos. Para la implementación se empleó una placa Raspberry Pi 3 B+, que brinda el sistema operativo base para la programación y tiene la capacidad de proporcionar una señal analógica en respuesta al sistema. Finalmente se implementó un circuito en base a la señal de salida, para realizar la apertura de una cerradura electromagnética que facilite el acceso físico.

Palabras clave: Neural Compute Stick 2, control de accesos, reconocimiento facial, embebido, Python, cerradura electromagnética.

ABSTRACT

This project presents the way in which the Neural Compute Stick 2 processor was used to generate an access control system with embedded facial recognition. The system is based on a facial recognition code developed in Python, which through a webcam, obtains the images in real time to make a comparison with the faces stored in a database. A Raspberry Pi 3 B+ was used for the implementation, which provides the base operating system for programming and has the capability to generate an analog signal in response to the system. Finally, a circuit was implemented based on the output signal, to perform the opening of an electromagnetic lock that facilitates physical access.

Key words: Neural Compute Stick 2, access control, facial recognition, embedded, Python, electromagnetic lock.

TABLA DE CONTENIDO

Diseño e Implementación de un Sistema Embebido de Reconocimiento Facial para el Control de Acceso usando Deep Learning

Gustavo Orna, Diego S. Benítez

Universidad San Francisco de Quito USFQ
Colegio de Ciencias e Ingenierías "El Politécnico"
Campus Cumbayá, Casilla Postal 17-1200-841, Quito, Ecuador
Email: gustavo.orna@estud.usfq.edu.ec, dbenitez@usfq.edu.ec

***Resumen*- Este proyecto presenta la forma en la que se utilizó el procesador Neural Compute Stick 2 para generar un sistema de control de accesos con reconocimiento facial de tipo embebido. El sistema está basado en un código de reconocimiento facial desarrollado en el lenguaje de programación Python, mismo que, a través de una cámara web, obtiene las imágenes en tiempo real para efectuar una comparación con los rostros guardados en una base de datos. Para la implementación se empleó una placa Raspberry Pi 3 B+, que brinda el sistema operativo base para la programación y tiene la capacidad de proporcionar una señal analógica en respuesta al sistema. Finalmente se implementó un circuito en base a la señal de salida, para realizar la apertura de una cerradura electromagnética que facilite el acceso físico.**

II. Introducción

En los últimos años, se ha incrementado el uso de sistemas de seguridad que requieren identificadores biométricos por el nivel de seguridad que ofrecen. Los sistemas biométricos permiten identificar a una determinada persona, mediante el análisis de distintas partes de su cuerpo, por ejemplo, la huella dactilar o el rostro. Los sistemas biométricos basados en reconocimiento facial son los más aceptados por los usuarios, al ser menos intrusivos que los otros sistemas, además de ser completamente inclusivos, puesto que no presenta limitaciones al momento de autenticar a personas con ausencia de extremidades o con dificultad de acceso a los medios basados en contacto o aproximación. El reconocimiento facial es el enfoque principal de estudio del Reconocimiento de Patrones y de la Visión Artificial, por las posibles aplicaciones que puede tener en el ámbito de la seguridad como la vigilancia, identificación personal y control de acceso [1].

Un aspecto importante para considerar de este trabajo es el bajo costo de inversión necesario en este sistema para lograr un reconocimiento confiable y efectivo, en comparación con otros sistemas que se encuentran en el mercado. Se puede encontrar distintos tipos de cerradura inteligente en el mercado, los sistemas oscilan desde los $260 en modelos básicos, hasta los $10.000 en modelos sofisticados que cuentan con más funciones. En este caso en particular, el sistema embebido implementado tuvo un costo aproximado de $200, sin la necesidad de un segundo paso externo de autenticación que aumente la confiabilidad.

Motivo por el cual, en este trabajo se presenta un prototipo de un sistema de seguridad basado en reconocimiento facial para controlar el acceso a un área restringida, mediante el uso de redes neuronales profundas (Deep Learning). El Deep Learning trata de descubrir diferentes niveles de representación, esperando que características de alto nivel puedan tener un significado más abstracto de los datos [2]. Es por esta razón que se utiliza el Deep Learning para extraer características de una base de datos, mismas que son esenciales para la identificación de rostros en este sistema.

Uno de los mayores problemas en los sistemas de reconocimiento facial es superar las limitaciones de los recursos disponibles como la capacidad del computador a utilizar para procesar la información de forma rápida [3]. Por este motivo se utiliza el procesador de Intel conocido como Neural Compute Stick 2 (NCS2). El NCS2 cuenta con la unidad de procesamiento de visión Intel Movidius Myriad X, el cual contiene 16 núcleos de procesamiento y un acelerador de hardware de redes neuronales profundas [4].

Fig. 1: Neural Compute Stick [4]

Con el procesador de Myriad X, el NCS2 es capaz de lograr un rendimiento comparable con el de un procesador de escritorio, por este motivo se utiliza el NCS2 para este sistema, pues los cálculos que vaya a realizar el programa para resolver las redes neuronales los realizará el NCS2, de esta manera no ocupa un gran porcentaje del procesamiento en el computador donde se ejecute. Este dispositivo es compatible con el kit de desarrollo OpenVINO, el cual es gratis y permite el desarrollo de aplicaciones que aplican la visión artificial. El kit de desarrollo OpenVINO permite inferencia de Deep Learning, soporta ejecución heterogénea en los aceleradores de visión artificial de Intel como el NCS2 y provee llamadas optimizadas para estándares de visión artificial como el OpenCV [5].

El sistema embebido que se plantea implementar utiliza el microcontrolador Raspberry Pi 3 B+, el cual cuenta con el

sistema operativo Raspbian Stretch basado en Debian. Es posible programar en este sistema operativo con varios lenguajes, pero el lenguaje de programación por el que se opto es el de Python. Se escogió Python por ser fácil de entender para los usuarios y por tener varias librerías destinadas al procesamiento de imágenes y datos. Teniendo en cuenta también que Python es compatible con el kit de desarrollo de OpenVINO para trabajar con el NCS2, se tiene que este es el lenguaje indicado para manipular todos los elementos del sistema que se implementó.

El Raspberry Pi 3 B+ cuenta con varios elementos que lo vuelven una herramienta útil para diversas aplicaciones como un procesador de cuatro núcleos de 64 bits a 1.4 GHz, un puerto HDMI, 4 puertos USB 2.0, 1 puerto microSD para cargar el sistema operativo y una cabecera extendida de 40 pines GPIO [6]. Para la implementación de este sistema de reconocimiento facial se planteó utilizar la distribución de pines GPIO del Raspberry, donde se pudo enviar una señal de respuesta en base a si la identificación facial tuvo éxito o no. Con esta señal de respuesta analógica se pudo también plantear un circuito para que cuando la identificación facial haya detectado un rostro de la base de datos, se abriera una cerradura electromagnética. El Raspberry Pi 3 B+ usado para la implementación se puede observar en la Figura 2 con sus pines GPIO y el resto de los elementos que componen este minicomputador.

Fig. 2: Microcontrolador Raspberry Pi 3 B+ [6]

Por medio del uso de las propiedades del NCS2 se busca generar un sistema de reconocimiento facial, el cual a través de una cámara web, pueda realizar comparaciones en tiempo real, con una base de datos previamente generada. Se pretende utilizar un programa extraído de la web, como base para establecer un reconocimiento facial funcional y modificarlo para que utilice el NCS2, de esta manera se puede lograr un reconocimiento facial confiable y obtener como respuesta a este sistema una señal de salida de los pines GPIO del Raspberry a utilizar. Con esta señal se pretende armar un circuito que tenga por objetivo manipular el acceso de personal a un área restringida, mediante la implementación de una cerradura electromagnética.

III. Metodología

A. *Desarrollo basado en pruebas (TDD)*

Se comenzó buscando una metodología apropiada para el desarrollo de software necesario para este proyecto, pero que, a su vez, se acople a la implementación del circuito electrónico complementario. Se requería probar el funcionamiento de las porciones código que cumplan con los requisitos que permitan cumplir el objetivo de lograr una autenticación facial confiable. Se eligió trabajar con la metodología de desarrollo basado en pruebas, o mejor conocido como Test-Driven Development (TDD). El método TDD es una forma de escribir código mediante la elaboración de pruebas antes de terminar de desarrollar un código funcional, para lo cual se establecen repeticiones pequeñas y rápidas para evaluar dichas pruebas [7]. Este método permite probar los módulos de código que se vayan construyendo con pruebas previamente diseñadas y se desarrolla el código de modo que vaya cumpliendo con los objetivos planteados para este trabajo. A medida que se va modificando el código y elaborando nuevas pruebas, las funciones previamente alcanzadas no se ven afectadas, puesto que el código nuevo debe satisfacer todas las pruebas anteriores.

B. *Investigación del código*

Para elaborar el programa se inició con la búsqueda de ejemplos de código que se pudieran usar de base para este proyecto. El primer objetivo fue demostrar la eficacia del dispositivo el NCS2 para la identificación de personas, para lo cual se tomó como base, un código que detecta objetos en tiempo real utilizando una red neuronal y un modelo pre-entrenado con los objetos que puede identificar (modelo Caffe) [8]. Para poder trabajar con el modelo pre-entrenado que se encontraba con el código, se utiliza el kit de herramientas de despliegue para Deep Learning. Este kit lo provee el kit de OpenVINO y es necesario para trabajar con el NCS2, está formado por dos componentes, un Optimizador de Modelo y una Maquina de Inferencia. El Optimizador de Modelo permite importar modelos entrenados como Caffe y Tensorflow, y convierte un modelo a una representación inmediata (IR) para usarlo en la aplicación. La Máquina de Inferencia utiliza una interfaz de programación de aplicaciones (API) basado en Python para trabajar con los archivos IR, y permite optimizar las cargas de trabajo para sistemas embebidos [9]. El proceso que realiza este kit para poder trabajar internamente con modelos pre-entrenados se puede apreciar en la Figura 3.

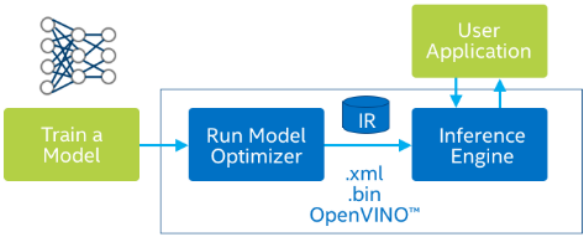

Fig. 3: Modelo del kit de herramientas de despliegue de Deep Learning [9]

Todo el proceso anterior tiene lugar para que el NCS2 sea el encargado de manejar los cálculos de las predicciones en la red neuronal utilizada. Una red neuronal es capaz de deducir información complicada en un patrón significativo. Puede procesar un gran número de entradas con la neurona entrenada para producir una salida, que corresponde al patrón encontrado [10]. Cabe recalcar que con este código únicamente se probó la implementación del NCS2 en un sistema de reconocimiento de objetos en tiempo real, que muestra en un recuadro el porcentaje de confiabilidad de que el objeto detectado

mediante la cámara sea el de la predicción. Aún se debía implementar el reconocimiento facial dentro del sistema.

Para la parte de reconocimiento facial se utilizó un código que comparaba las imágenes de video en tiempo real con los rostros guardados en una base de datos codificados utilizando un vector de 128 características (valores reales) por cada rostro [11]. Estos vectores se generan de una red pre-entrenada en un archivo Pickle. Para la detección de rostros en los cuadros de video usa el archivo entrenado de OpenCV Haar Cascade. Para el ingreso de rostros a la base de datos, se realiza un proceso manual, en el cual se efectúa la creación de carpetas con imágenes de las personas que se desea agregar al sistema. Cada carpeta debe tener el nombre de la persona a la cual pertenecen las imágenes que contienen. Esto se puede observar en la Figura 4, donde la carpeta "dataset" es la que contiene las carpetas de las personas que han sido agregadas a la base de datos.

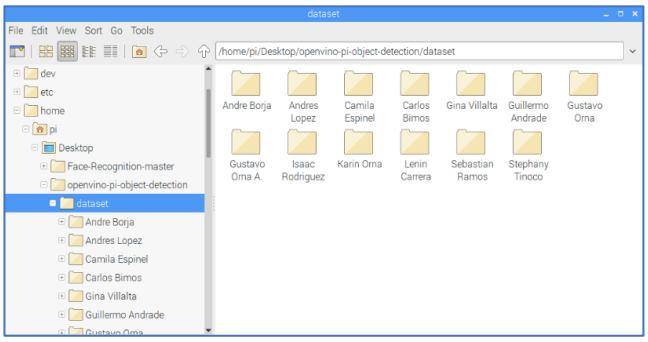

Figura 4: Carpeta de rostros "dataset"

Una vez determinadas las personas que pueden ser agregadas al sistema, se debe generar el archivo Pickle con los rostros codificados. Por este motivo se utiliza un código adicional que genera el archivo de rostros, que alimenta al módulo de reconocimiento facial. Este código es capaz de codificar todos los rostros ingresados en las carpetas, para comparar en tiempo de ejecución, estos vectores con los obtenidos por las imágenes obtenidas por la cámara, y así poder realizar un reconocimiento facial efectivo.

Se pudo concluir que este código era capaz de lograr un reconocimiento facial de modo eficiente, sin la necesidad de utilizar la capacidad de procesamiento provista por el NCS2. Sin embargo, se detectó una vulnerabilidad en el módulo de reconocimiento facial, ya que era capaz de otorgar acceso a rostros que se presentaban delante de la cámara en forma impresa o con el uso de una pantalla. Para mitigar esta vulnerabilidad se optó por acoplar los dos códigos, el de reconocimiento facial que realice la comparación de rostros, y el de detección de objetos que garantice que la imagen presentada delante de la cámara sea de una persona. De este modo se agrega una segunda capa de seguridad al sistema, sin añadir un paso de autenticación adicional, como sucede en varios sistemas comerciales actuales. En el caso de que una persona intente suplantar a otra que, si se encuentre en la base de datos del sistema con una foto u otro medio de replicación de rostros, sería posible si solo se ejecutara el código de reconocimiento facial. Por ende, se busca utilizar el código de detección de objetos antes descrito para que, con un cierto porcentaje de confiabilidad alto, se pueda asegurar que es una persona la que está frente a la cámara.

C. Pruebas de prerrequisitos

Una vez encontrados los códigos base de este trabajo, fue imperativo descargar y preparar los requisitos necesarios para poder ejecutar estos códigos en la plataforma Raspbian. La preparación de este sistema operativo implica, entre los requisitos más importantes, asegurarse que exista suficiente memoria para almacenar los paquetes necesarios y garantizar acceso a internet para descargar las librerías y herramientas necesarias.

El requisito indispensable para utilizar el NCS2, es el kit de desarrollo OpenVINO. Para su ejecución es necesario haber instalado algunas librerías como GTK y CMake. También se debe establecer Python 3 como el lenguaje por defecto. Después, se debe descargar el paquete de OpenVINO que se encuentra en la página de Intel, donde existe un paquete específico para Raspberry Pi. Una vez descargado el archivo se debe establecer las variables de entorno de OpenVINO, para inicializar el ambiente y poder trabajar en él. Una vez hecho esto, se establecen las reglas USB del NCS2, para poder utilizar la Maquina de Inferencia en los ejemplos que vienen con el paquete de OpenVINO, o utilizarlo en otras aplicaciones. Con estos prerrequisitos ya se puede conectar el NCS2 al Raspberry Pi. Se creó también un ambiente virtual de Python para poder trabajar en él con todas las librerías necesarias para este trabajo, incluyendo el paquete de OpenVINO. Una vez terminada la instalación de este paquete, se puede vincular la versión de OpenCV del OpenVINO, al ambiente virtual previamente creado para poder usarlo en el desarrollo de esta aplicación.

Como se trata de un periférico esencial para este sistema, se probó la cámara web que se utilizaría en el sistema final. Para esto la cámara debía ser reconocida por el Raspberry Pi, por lo que se tuvo que habilitar en la configuración raspi-config de la plataforma. Habiendo hecho esta configuración se reinició el sistema operativo y se tomó una foto con la cámara para asegurarse que ya fuera reconocida en la plataforma.

Con todo el proceso antes realizado, se hizo una prueba para comprobar el funcionamiento del NCS2 en la plataforma Raspberry Pi. Se puede observar en la Figura 5, una ventana autónoma donde se aprecia la ejecución del programa de detección de objetos [8], cada objeto encontrado se encierra en un recuadro, presentando el nombre del objeto y su porcentaje de confiabilidad. El porcentaje se calcula en base a la comparación de los objetos encontrados en la imagen con los objetos de la red neuronal pre-entrenada, y solo se muestra el porcentaje en la ventana autónoma de "Frame" cuando supera un mínimo porcentaje de confiabilidad fijado, en este caso es el 20% [8].

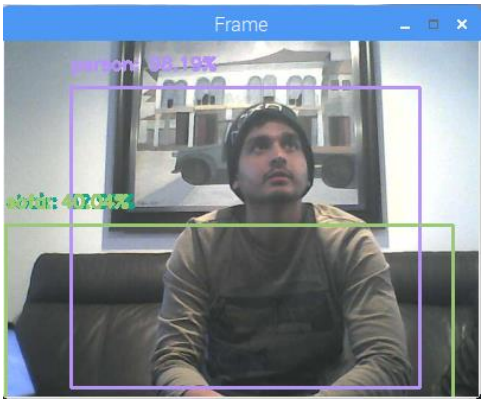

Figura 5: Prueba de detección de objetos con NCS2

Una vez asegurado el correcto funcionamiento del NCS2 con la cámara web, se procede a construir el código requerido para este sistema.

D. Construcción del código

Con los prerrequisitos y códigos de detección de objetos [8] y reconocimiento facial [11], se pudo empezar a crear el código final para la implementación. Como ya se habían escogido los códigos de detección de objetos y reconocimiento facial, lo esencial ahora es acoplar los dos códigos de tal manera que se puedan efectuar las acciones de ambos al mismo tiempo sin interferir el uno con el otro. Por este motivo se estructuró el código final de la manera como se muestra en el diagrama de bloques de la figura 4.

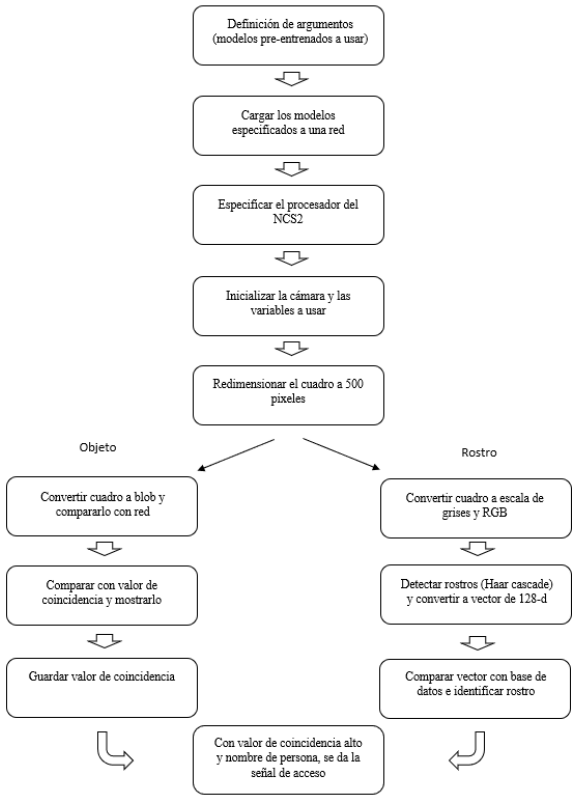

Figura 6: Diagrama de bloques del código implementado.

Teniendo en cuenta que los dos códigos son similares en algunos aspectos, se puede apreciar en el diagrama de bloques anterior que desde el inicio necesitan de una misma estructura. Lo primero que se definen son los argumentos para ejecutar el programa, en esta parte es necesario recalcar que cada código tiene sus propios argumentos con los cuales trabajar. Cada uno tiene su propio modelo pre-entrenado el cual necesita para funcionar. Por esta razón simplemente se usan todos los argumentos de los dos códigos. Una vez definidos los tipos de archivos en los argumentos, se cargan los modelos pre-entrenados para poder acceder a ellos en el código.

Para lograr que el procesador Myriad X del NCS2 sea el que se encargue de las operaciones con la red creada a partir del modelo Caffe para la identificación de objetos es necesario especificar este proceso en el código.

Antes de empezar el proceso de reconocimiento en tiempo real se inicializa la cámara web a usar y las variables a usar para la identificación como el porcentaje de confiabilidad de una persona. Además de eso se utiliza un redimensionamiento de los pixeles de los cuadros de video para usar un máximo de 500 pixeles por cuadro.

El procesamiento de los cuadros se realiza con los dos métodos de los códigos al mismo tiempo, la detección de objeto con la identificación de rostros, cada uno con su propio método de procesar los cuadros. En el método de detección de objetos, se empieza tomando el cuadro de video y convirtiendo a un archivo binario conocido como un blob, el cual después es ingresado a la red del modelo pre-entrenado Caffe para poder comparar con los objetos guardados en la red, con esto se obtienen las detecciones de los objetos. Una vez encontrado los objetos en el cuadro, se hace el cálculo de confiabilidad para saber el porcentaje de qué tan parecido es el objeto que aparece en el cuadro con el objeto en la red. El valor de confiabilidad se muestra en una ventana aparte junto con un cuadro que encierra el objeto encontrado. El valor de confiabilidad que se va generando se guarda en otra variable para compararlo con un valor escogido que determine el porcentaje que debe cumplir, en este caso la persona, para asegurar que si es una persona la que se encuentra enfrente de la cámara. En este caso en particular se propuso un valor de 98% de confiabilidad que debe alcanzar para poder considerarlo como una persona.

Para la identificación de rostro, se toma el mismo cuadro y se convierte a escala de gris para la detección de rostro y a RGB para el reconocimiento. Se detectan los rostros en la escala de grises usando el modelo de Haar-cascade del código de reconocimiento facial [11]. Se dibujan también los recuadros que encierran los rostros después de haberlos detectado. Una vez hecho esto, se utiliza el formato de RGB junto con las delimitaciones de los recuadros antes dibujados, para codificar los rostros a vectores de 128 valores usando una función de una librería especifica. Usando la misma librería, se compara el vector encontrado con los vectores de los rostros guardados en el archivo Pickle. Mientras la comparación de un valor booleano “Falso”, se guarda en la variable del nombre de la persona el valor de “Intruso”. Cuando se genera el valor de “True”, se cuenta el número de veces que dicho rostro fue encontrado en la base de datos, con el máximo número de veces que se logró reconocer con uno de los rostros de la base de datos, se genera el nombre de la carpeta de dicho rostro para mostrar en la ventana externa de “Frame”.

Finalmente, se generó una condición dentro del programa, si la confiabilidad de que sea una persona es mayor al 98% y el nombre en pantalla de la persona es diferente a “Intruso”, se muestra un mensaje en consola con el nombre de la persona. En esta condición también se genera un encendido de led de la interfaz GPIO del Raspberry Pi, teniendo en este caso la señal externa cuando se efectúa un reconocimiento de personal efectivo. Aparte también se genera un tiempo de “hold” en todo el programa para que la señal externa del programa se mantenga por un periodo de 10 segundos. De esta manera la señal no es tan fugaz y permite trabajar de manera eficiente con dicha señal para diseñar el circuito que apertura una cerradura electromagnética. Ejecutando entonces el programa anterior, se puede apreciar en consola y en la ventana externa el reconocimiento que realiza como se observa en la Figura 7.

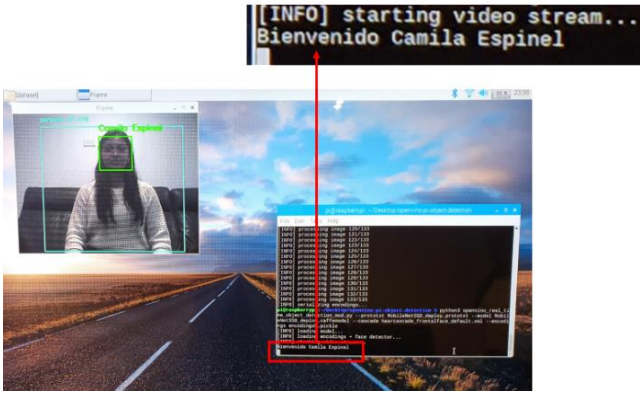

Figura 7: Reconocimiento de personal usando el código final.

E. Circuito de apertura de cerradura

Para el armado del circuito se utilizó la misma metodología TDD de realizar pruebas antes de seguir armando el circuito para asegurar que los elementos del circuito fueran los necesarios para este trabajo. Primero se necesitaba probar que la señal de salida del Raspberry Pi fuera la suficiente para encender un led. Por este motivo se conectó un led directamente al pin de salida especificado en el código anterior (Pin GPIO 17), junto con una resistencia conectada en serie y está a tierra. Se utilizo la tierra que proporcionaba el Raspberry Pi de otro de sus pines. Teniendo en cuenta que señal de salida tiene un voltaje de 3.3 V, para esta prueba se utilizó una resistencia de 100 Ω. Ejecutando el programa anterior con este circuito, se comprobó que la señal de salida si se proveía del Pin 17, encendiendo el led.

La siguiente parte era probar como con esta señal de salida se podía accionar una cerradura electromagnética. Para la apertura y cierre de la cerradura se optó por usar un relé de 5V. Este relé sería el que cerraría el circuito para suministrar el voltaje que necesita la cerradura electromagnética para realizar la apertura, en este caso el voltaje que se requería era de 12V. Para lograr esto se conectó el pin de COM a la fuente de 12V y el pin de normalmente abierto (NO) a la cerradura. El otro punto de la cerradura iba a la tierra de la fuente de 12V. Teniendo en cuenta que el Raspberry Pi no tiene una salida de 12V, se utilizó una fuente externa que es capaz de suministrar 12V DC a partir del voltaje de un enchufe que es 120V AC. Sin embargo, para poder activar el relé es necesario también energizar la bobina dentro del relé con 5V para conectar internamente el pin de COM con el pin NO de la cerradura. Para energizar entonces esta bobina, se conectó un extremo de la bobina con uno de los pines del Raspberry Pi que suministra 5V. El otro extremo de la bobina del relé se conectó a dos elementos, el primero un diodo de protección entre los extremos de la bobina para evitar que la corriente fluya de regreso al pin del Raspberry Pi. El otro elemento conectado al extremo de la bobina es el que permite la conexión a tierra para activar el relé, pero de manera controlada, en este caso se utilizó un transistor. El transistor es capaz de operar como un interruptor. El transistor se activa y se cierra como interruptor cuando se suministra suficiente corriente en la base y este entra en saturación. El transistor se mantiene abierto como interruptor cuando no existe corriente en la base de este [12]. El transistor entonces se conectó de tal manera que la base de este estuviera conectada a la resistencia en serie con el led que se había utilizado al principio para verificar el funcionamiento del código. Por lo tanto, el momento en que se dé la señal de salida del Raspberry Pi por el reconocimiento facial, esta señal llegara a la base del transistor, conectando la tierra conectada al colector con un extremo de la bobina en el emisor, energizando con los 5V el relé y finalmente energizando la cerradura para su apertura. Con toda esta conexión también se tuvo que cambiar esta resistencia entre el led y el transistor a una de 1kΩ. Un diagrama de todo el circuito previamente explicado se puede apreciar en la Figura 8, desde el pin del GPIO 17 del Raspberry hasta la cerradura electromagnética.

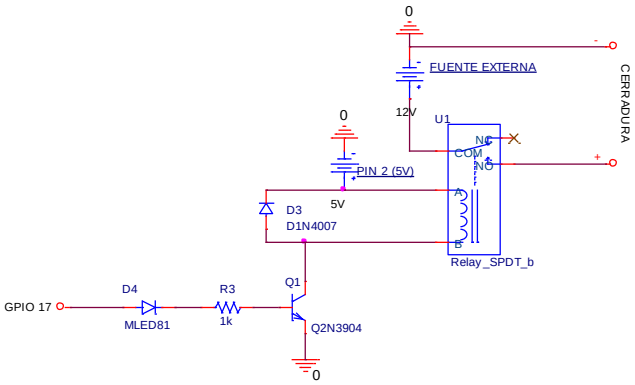

Figura 8: Diagrama del circuito controlador de la cerradura.

Realizando toda esta conexión en una placa se logró obtener el circuito capaz de controlar la apertura de la cerradura electromagnética a partir de la señal de reconocimiento de personal del código. Como en el código se especificó que cuando reconozca a una persona, se mantenga en "hold" el programa, esto se aprecia en todo el circuito pues la señal se mantiene constante por un periodo de 10 segundos. Esto se observa en el circuito por el led encendido y la cerradura abierta durante este intervalo de tiempo. El circuito ya conectado para probar el sistema se observa en la Figura 9.

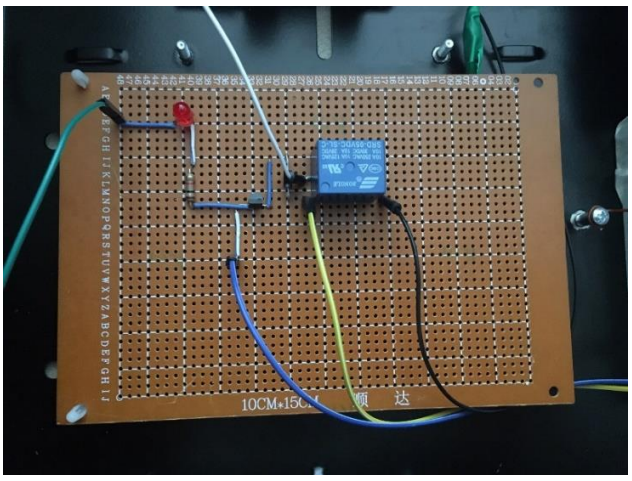

Figura 9: Circuito controlador de apertura de cerradura.

IV. Resultados y Experimentación

Para comprobar la efectividad del sistema implementado, se realizaron pruebas con 15 sujetos con los cuales se debía entrenar el sistema e ingresarlos manualmente a la base de datos, para posteriormente probar si el sistema es capaz de reconocerlos y abrir la cerradura electromagnética correctamente.

Las pruebas se realizaron de manera satisfactoria, se efectúa la apertura de la cerradura electromagnética, cada vez

que se reconoce a una persona cuyo rostro esté ingresado en la base de datos del sistema. Sin embargo, hubo casos en las pruebas en donde el sistema confundía un rostro con otro. La forma de verificar que una persona fue reconocida por el sistema, es el mensaje que aparece en consola con el nombre de la persona reconocida. Se realizo la prueba de identificación del sistema 5 veces por persona para determinar cuántas veces el sistema lograba reconocer efectivamente la persona que estaba en frente y cuantas veces la confundía con otra persona de la base de datos. Los resultados de las pruebas realizadas se pueden observar en la matriz de confusión de la Tabla I.

Como se aprecia en la Tabla I, los resultados muestran cuantas veces, de los 5 intentos por persona, el sistema se equivocó de persona con otra dentro de la base de datos.

		Predicciones														
		S1	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**	**S15**
Valores reales	**S1**	**3**	0	0	0	0	0	2	0	0	0	0	0	0	0	0
	S2	1	**4**	0	0	0	0	0	0	0	0	0	0	0	0	0
	S3	0	0	**5**	0	0	0	0	0	0	0	0	0	0	0	0
	S4	0	0	0	**4**	0	0	0	0	0	0	1	0	0	0	0
	S5	0	0	0	0	**5**	0	0	0	0	0	0	0	0	0	0
	S6	0	0	0	0	0	**5**	0	0	0	0	0	0	0	0	0
	S7	1	0	0	0	0	0	**4**	0	0	0	0	0	0	0	0
	S8	0	0	0	0	0	0	0	**5**	0	0	0	0	0	0	0
	S9	0	0	0	0	0	0	0	0	**5**	0	0	0	0	0	0
	S10	0	0	0	0	0	0	0	0	0	**5**	0	0	0	0	0
	S11	0	0	0	1	0	0	0	0	0	0	**4**	0	0	0	0
	S12	0	0	0	0	0	0	0	0	0	1	0	**4**	0	0	0
	S13	0	0	0	0	0	0	0	0	0	0	0	0	**5**	0	0
	S14	0	0	0	0	0	0	0	0	0	0	0	0	0	**5**	0
	S15	0	0	0	0	0	0	0	0	0	0	0	0	0	2	**3**

TABLE I: Matriz de confusión de pruebas de reconocimiento.

De la matriz anterior se puede calcular el porcentaje de precisión general del sistema, el cual se obtiene de la relación del total de aciertos, 66, con el total de pruebas, 75, obteniendo de esta manera un 88% de precisión.

En algunos casos la persona fue identificada los 5 intentos, mientras que en otros casos las identificó un número menor de veces hasta 3 de los 5 intentos. Se podría inferir que los casos en los que se obtuvo un número menor de reconocimientos y hubo confusión con otro sujeto de prueba fuera porque esas personas poseen características similares del rostro. La comparación de rostros se realiza en el sistema por comparación de vectores que representan la cuantificación de los rostros en las imágenes, de la base de datos y de las recibidas por la cámara de video. Cabe mencionar que los cuadros recibidos de la cámara de video se redujeron a un máximo de 500 pixeles para luego compararlos. Se podría inferir que quizás con un mayor número de pixeles en los cuadros del video, se podría tener un reconocimiento mejor con las imágenes guardadas y un mejor rendimiento del reconocimiento facial del sistema.

Una segunda prueba se realizó para determinar si los dos módulos de códigos acoplados podían mitigar la vulnerabilidad de que si una persona, fuera de la base de datos, intentaba suplantar a otra que si estaba en la base de datos con una fotografía digital o impresa. Esta prueba se realizó con las imágenes digitales de 5 sujetos previamente ingresados en la base de datos. Una misma persona acercó a la cámara de video del sistema embebido, un teléfono celular con las imágenes digitales de los 5 sujetos para probar la doble autenticación implementada en el sistema. Una muestra de cómo se realizó esta prueba se puede observar en la Figura 10.

Figura 10: Prueba de doble autenticación.

Los resultados de esta prueba reflejaron el correcto funcionamiento de la doble autenticación. En todos los casos de esta prueba, el sistema no permitió que la cerradura se abriera por no cumplir el requisito de alcanzar el 98% de seguridad de detección de persona frente a la cámara.

V. Conclusión

En este trabajo se demostró la eficacia de trabajar con el NCS2 y un Raspberry Pi para implementar un sistema embebido de reconocimiento facial, a un bajo costo, para el control de una cerradura electromagnética usando Deep Learning. Se implementó un código de Python capaz de realizar un reconocimiento facial confiable utilizando el NCS2 para poder distribuir la cantidad de procesamiento en el Raspberry Pi y ejecutar este tipo de código sin mayor complicación. En este trabajo se muestra la correcta implementación de la interfaz GPIO del Raspberry Pi para poder generar una señal externa capaz de ser utilizada para una gran variedad de aplicaciones, como en este caso de controlar una cerradura para el control de acceso. El código implementado puede ser mejorado no solo en función del propósito de este trabajo, como sería un reconocimiento facial aún más confiable y con un menor índice de error, sino también para un sistema más complejo de control de acceso como un sistema con mayor personal autorizado a revisar o un sistema que controle el acceso a varias áreas restringidas. Como trabajo futuro se pueden explorar otras aplicaciones que se podrían realizar utilizando el NCS2 con el Raspberry Pi para sistemas embebidos que requieran un alto nivel de procesamiento, dado que se trabaja con el lenguaje de Python para utilizar estos elementos se pueden realizar diversas aplicaciones que necesiten procesar datos *in situ* como la autenticación por voz u otro tipo de reconocimientos para generar una acción en físico.

References

[1] L. Liying and H. Yue, “Study on Access Control System Based on Face Recognition”, 2008 IEEE International Conference on Computer Science and Software (CSSE), Hubei, 2008, pp. 1-3.

[2] T. Guo, J. Dong, H. Li and Y. Gao, “Simple Convolutional Neural Network on Image Classification”, 2017 IEEE 2nd International Conference on Big Data Analysis, Beijing, 2017, pp. 1-4.

[3] E. Yanakova, T. Ishkova, A. Belyaev, V. Koldaev and M. Kolobanova, “Facial Recognition Technology on ELcore Semantic Processors for Smart Cameras”, 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), Saint Petersburg and Moscow, 2019, pp. 1-4.

[4] Intel. (2019). Recuperado de https://software.intel.com/es-es/neural-compute-stick
[5] Intel (2019). Recuperado de https://software.intel.com/es-es/articles/OpenVINO-RelNotes
[6] Raspberry Pi. (2018). Recuperado de https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/
[7] D. Janzen and H. Saiedian, "Test-driven development concepts, taxonomy, and future direction", Computer IEEE, 2005, pp. 1-8.
[8] A. Rosebrock. (2019, Abril 8). OpenVINO, OpenCV, and Movidius NCS on the Raspberry Pi [Online]. Available: https://www.pyimagesearch.com/2019/04/08/openvino-opencv-and-movidius-ncs-on-the-raspberry-pi/
[9] Intel. (2019,). Recuperado de https://software.intel.com/en-us/openvino-toolkit/deep-learning-cv
[10] P. Nithyakani, A. Shanthini and G. Ponsam, "Human Gait Recognition using Deep Convolutional Neural Network", 2019 3rd International Conference on Computing and Communications Technologies (ICCCT), Chennai, 2019, pp. 1-4.
[11] A. Rosebrock. (2018, Junio 25). Raspberry Pi Face Recognition [Online]. Available: https://www.pyimagesearch.com/2018/06/25/raspberry-pi-face-recognition/
[12] D. W. Hart, "Power Electronics", Valparaiso: McGraw-Hill, 2011.