UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

SISTEMA DOMÓTICO DE VIDEOVIGILANCIA

POR

Matías Ignacio Delgado Salazar

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título profesional de Ingeniero Civil Electrónico

Profesor Guía
Mario Rubén Medina Carrasco

Enero 2022
Concepción (Chile)

Dedicatoria

Esta tesis está dedicada a:

Mi familia que siempre confió en mis capacidades de perseverancia. Me entregaron todo el apoyo que necesité, no solo durante el proceso de implementación del proyecto de tesis, sino durante todo el proceso de formación como ingeniero civil electrónico. Les doy las gracias por estar siempre cuando necesité algún consejo y/o ayuda emocional para levantar los ánimos.

Mis amigos y amigas que me acompañaron en momentos difíciles durante mi ciclo universitario. Si bien se encuentran los amigos y amigas previo a la universidad, durante el transcurso de los estudios universitarios se generan fuertes lazos de amistad que perduran una vez terminada esta etapa. A todos ellos y todas ellas, les dedico este proyecto por cada instancia que me entregaron en los momentos más difíciles del ciclo universitario.

Mis profesores y compañeros que me brindaron de su tiempo para enseñarme de buena manera la teoría que se me acomplejaba más. Sin sus tardes y noches de estudio hubiese sido mucho más complicado el proceso de estudio como electrónico.

A mi persona. Todas las recompensas que obtuve durante el ciclo universitario igualmente es gracias a mi propio esfuerzo y forma pensar. Siempre optimista. Dedico esta tesis a todas mis horas invertidas de estudio y aprendizaje.

Gracias.

Agradecimientos

Mis sinceros agradecimientos a mi profesor guía que estuvo desde un comienzo ayudando con ideas para implementar el sistema domótico de videovigilancia. Por una parte, la ayuda a nivel teórico y hardware fueron excelente, pero la que más destaco es la ayuda psicológica que me brindó. Cada problema que existió durante el proceso de implementación buscó alguna solución óptima y rápida. Esta acción me generaba un alivio antiestrés, por esto, quiero dar las gracias por ayudar de manera teórica, práctica y mental.

Por último, agradecer a la ayuda teórica de mi hermano mayor con relación a las redes de datos. Algunos problemas que surgieron durante la implementación del sistema domótico se los detallaba con el fin de buscar y aclarar el origen del error que se estaba cometiendo. Gracias a esto se pudieron solucionar varios problemas relacionados a las IP del router.

Agradecido a ellos dos por toda la ayuda que me brindaron durante la implementación del sistema domótico de videovigilancia.

Sumario

Los robos a las casas hoy en día es un problema evitable gracias al avance de la tecnología. Una alternativa que ha crecido mucho en este último tiempo es la domótica. Esta se centra en implementar un hogar inteligente y controlable, brindando principalmente seguridad y comodidad al dueño o a la dueña del hogar. En este caso, se detalla la implementación de un sistema domótico de videovigilancia que, a través de una cámara se permita reconocer un rostro a tiempo real, y si este no lo reconoce envía un correo de alerta al dueño del hogar.

Para desarrollar este sistema domótico de videovigilancia se usan los dispositivos electrónicos Orange Pi Lite y ESP32-CAM. Estos corresponden a un controlador y a una cámara IP, respectivamente. El flujo de datos del sistema implementado es el siguiente: cuando el módulo ESP32-CAM no detecte un rostro envía una señal de alerta a un segundo ESP32-CAM, con la finalidad de enviar el correo de alerta al dueño de la casa. Una vez recibido el correo, el interesado puede ingresar de manera remota al servidor web domótico levantado por el Orange Pi Lite. Una vez dentro el usuario ingresa a la sección de videovigilancia para poder ver el flujo de video que transmite en tiempo real el módulo ESP32-CAM.

Para lograr el reconocimiento facial y el correo de alerta se utiliza el software Arduino IDE y las bibliotecas ESP-WHO, ESP-FACE y ESP Mail Client. Por otra parte, para ingresar remotamente al servidor web se utiliza Apache y el dominio DDNS No-IP.

El sistema implementado trabaja bajo ciertas condiciones ambientales. Para el caso del ESP32-CAM debe existir una luminosidad mínima de 900 lux para que la tasa de aciertos sea de un 90%. En cambio, cuando hay más de un rostro en la captura, la tasa de fallos aumenta, independiente de las condiciones de luminosidad del ambiente. El sistema implementado tiene una alta tasa de fiabilidad cuando trabaja con plena luz del día, es decir, entre las 8 y 18 horas.

Summary

Home burglaries nowadays are a preventable problem thanks to the advance of technology. An alternative that has grown a lot in recent times is home automation or domotics. This focuses on implementing an intelligent and controllable home, mainly providing security and comfort to the homeowner. In this case, we detail the implementation of a video surveillance home automation system that, through a camera, allows to recognize a face in real time, and if it does not recognize it sends an email alert to the owner of the home.

The Orange Pi Lite and ESP32-CAM electronic devices are used to develop this video surveillance home automation system. These correspond to a controller and an IP camera, respectively. The data flow of the implemented system is as follows: when the ESP32-CAM module does not detect a face, it sends an alert signal to a second ESP32-CAM, in order to send the alert email to the owner of the house. Once the mail is received, the interested party can remotely access the home automation web server set up by the Orange Pi Lite. Once inside, the user enters the video surveillance section to view the video stream transmitted in real time by the ESP32-CAM module.

To achieve facial recognition and mail alerts, the Arduino IDE software and the ESP-WHO, ESP-FACE and ESP Mail Client libraries are used. On the other hand, Apache and the DDNS No-IP domain are used to remotely access the web server.

The implemented system works under certain environmental conditions. In the case of ESP32-CAM there must be a minimum luminosity of 900 lux for the hit rate to be 90%. On the other hand, when there is more than one face in the capture, the failure rate increases, independent of the ambient light conditions. The implemented system has a high reliability rate when working in full daylight, i.e. between 8 and 18 hours.

Índice general

1 Introducción 1
1.1 Introducción a la domótica 1
1.2 Sistema domótico de videovigilancia 1
2 Marco teórico 3
2.1 Revisión bibliográfica 4
3 Desarrollo 6
3.1 Selección de hardware y software 6
3.1.1 Hardware 6
3.1.1.1 Orange Pi Lite 6
3.1.1.2 ESP32-CAM 8
3.1.2 Software 10
3.1.2.1 Orange Pi Lite 10
3.1.2.2 ESP32-CAM 13
3.2 Implementación 19
3.2.1 Orange Pi Lite 19
3.2.2 ESP32-CAM 23
3.2.2.1 Protocolo WebSocket 24
3.2.2.2 Reconocimiento facial 25
3.2.2.3 Correo de alerta 27
3.2.2.4 Dominio No-IP 31
4 Resultados 32
4.1 Reconocimiento Facial 32
4.2 Ingreso al correo remitente 33
4.3 Correo de alerta 34

4.4 Ingreso remoto al servidor domótico 36
4.5 Tasa de acierto al reconocer rostro 39
5 Conclusiones 41
6 Glosario 43
7 Referencias 44
8 Anexos 47
8.1 Datasheet Orange Pi Lite 47
8.2 Datasheet ESP32-CAM 54
8.3 Código ESP32-CAM maestro 58
8.4 Código ESP32-CAM esclavo 67

Lista de figuras

Figura 1.1: Sistema domótico de videovigilancia implementado 2

Figura 3.1: Dispositivos para utilizar un Orange Pi Lite 7

Figura 3.2: Principales componente del controlador Orange Pi Lite 7

Figura 3.3: Módulo ESP32-CAM 8

Figura 3.4: Conexión convertidor USB a TTL al ESP32-CAM 9

Figura 3.5: Software Arduino IDE 10

Figura 3.6: Sistema operativo Armbian 11

Figura 3.7: Diagrama de flujo del OPLite 12

Figura 3.8: Incorporar URL del ESP32 13

Figura 3.9: Integrar placa esp32 al Arduino IDE 14

Figura 3.10: Biblioteca WebSocket en Arduino IDE 14

Figura 3.11: Incorporar biblioteca Esp32 Mail Client en Arduino IDE 15

Figura 3.12: Incorporar biblioteca Easy DDNS para registrar dominio DDNS 16

Figura 3.13: Entorno ESP-IDF para detectar y reconocer rostros 17

Figura 3.14: Modelos MobileNetV2 y redes convolucionales en cascada multitareas 18

Figura 3.15: Diagrama flujo de petición 20

Figura 3.16: Dominio No-IP 21

Figura 3.17: Nombre dominio No-IP y sesión utilizada 22

Figura 3.18: Flujo del dominio No-IP 23

Figura 3.19: Función para indicar la tarea de agregar nuevo rostro y guardar el nombre del rostro . 24

Figura 3.20: Función encargada de guardar el nuevo rostro y responder al cliente 25

Figura 3.21: Función dedicada a guardar un nuevo rostro en la lista de memoria del ESP32-CAM 25

Figura 3.22: Función reconocer rostro 26

Figura 3.23: Respuesta de confirmación “DOOR OPEN FOR _____” 27

Figura 3.24:Cuerpo setup() para enviar señal de alerta al ESP32-CAM esclavo 28

Figura 3.25: Información de la sesión que enviará el correo de alerta 29

Figura 3.26: Datos del correo remitente 29

Figura 3.27: Cuerpo del correo de alerta 29

Figura 3.28: Estableciendo conexión del correo, iniciando ESP-NOW y registrando el callback 30

Figura 3.29: Código loop del ESP32-CAM esclavo 30

Figura 3.30: Inicialización servicio No-IP 31

Figura 4.1: Rostro reconocido por ESP32-CAM maestro 32

Figura 4.2: Acierto con dos rostros frente al ESP32-CAM 33

Figura 4.3: Ingreso del ESP32-CAM esclavo al correo remitente - Protocolo SMTP 34

Figura 4.4: Envío correo de alerta 35

Figura 4.5: Recepción correo de alerta 36

Figura 4.6: Ingreso remoto a servidor web creado por OPLite 37

Figura 4.7: Acceso remoto al flujo de video del ESP32-CAM maestro 38

Figura 4.8: Aciertos al reconocer rostro 39

Figura 4.9: Fallos al reconocer rostro 40

Figura 4.10: Comportamiento acierto/falla al reconocer rostro 40

1 INTRODUCCIÓN

1.1 INTRODUCCIÓN A LA DOMÓTICA

El desarrollo de la tecnología en los últimos años ha marcado la forma de vivir de las personas. Es habitual pensar que gran parte del tiempo se busca solucionar la problemática de comodidad y seguridad en un mismo sistema electrónico y que además sea económico en cuanto a precio, para que todas las personas puedan acceder y aprovechar los beneficios que entrega el producto.

Un sistema electrónico, en este documento, se refiere a un conjunto de dispositivos electrónicos que pueden interactuar entre sí, conectándose de manera alámbrica o inalámbrica. Un ejemplo de sistema alámbrico es una conexión a través de cables entre un notebook, un Smart TV y unos parlantes. Por otra parte, la conexión a través de Bluetooth entre un celular, unos audífonos y un reloj de pulsera inteligente es considerado un sistema inalámbrico.

En el caso de este proyecto, se desarrollará e implementará un sistema domótico de videovigilancia, en el que se enlazarán un controlador, una cámara IP y un celular y/o computadora a través de Wi-Fi. Pero ¿en qué consiste un sistema domótico?

1.2 SISTEMA DOMÓTICO DE VIDEOVIGILANCIA

La domótica se define como la tecnología para desarrollar e implementar la automatización de las instalaciones comunes de un hogar, brindando diversos servicios como: preservar un ambiente cómodo y agradable a las personas que la habitan; dar seguridad del hogar ante robos y/o accidentes; lograr un ahorro energético o eficiencia energética del hogar, racionalizando el consumo eléctrico a través de una mejor gestión; entre otros.

Dentro de los múltiples servicios que puede entregar un sistema domótico, el presente proyecto le brinda al usuario la seguridad de su hogar a través de un sistema domótico de videovigilancia. Este sistema captura un flujo de video del estado actual del hogar, y también es capaz de reconocer rostros humanos en tiempo real de dicho flujo. En el caso que se detecte un rostro, pero no se logre reconocer, el sistema enviará un correo electrónico al usuario, alertándolo de la presencia de una persona desconocida fuera de su hogar. Con este aviso, el usuario podrá acceder al flujo de video en cualquier lugar y momento del día.

A continuación, se muestra un esquema que representa el comportamiento del sistema domótico de videovigilancia desarrollado en este documento.

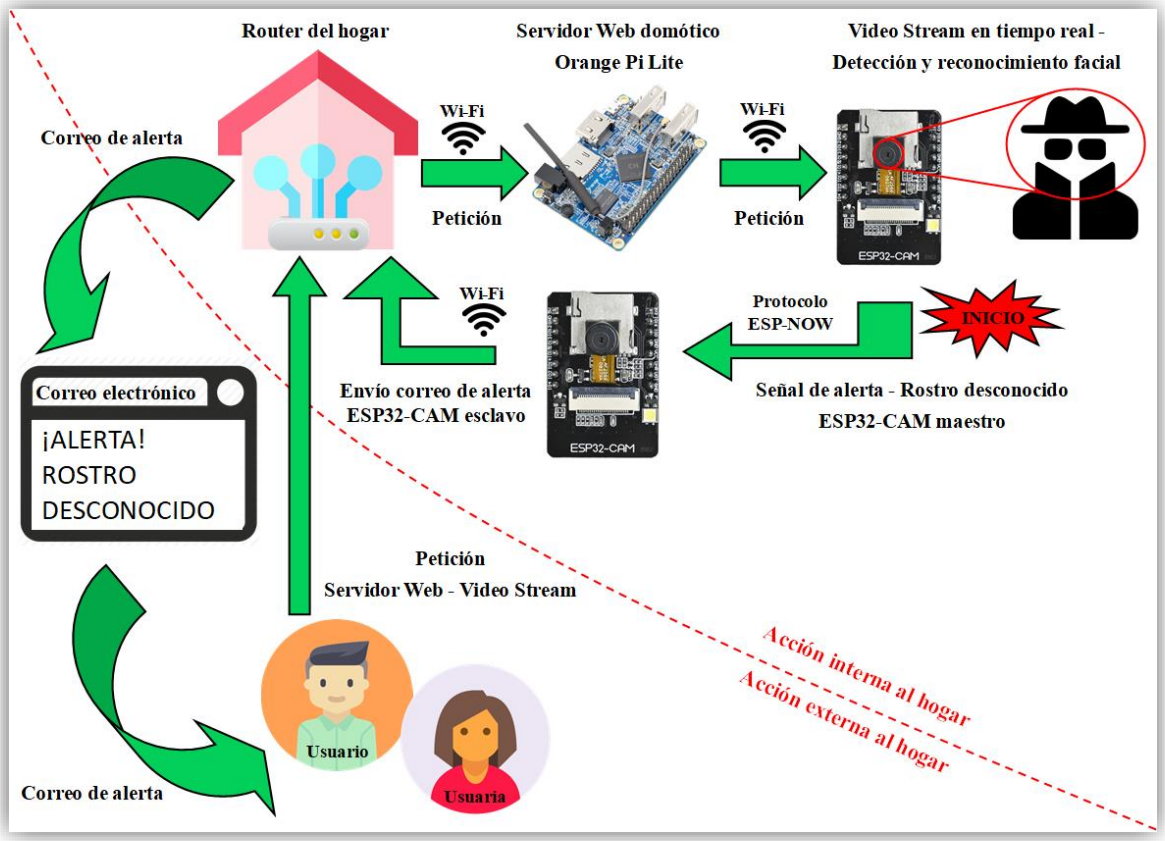


Figura 1.1: Sistema domótico de videovigilancia implementado

La interpretación de la figura 1.1 es la siguiente: una cámara IP modelo **ESP32-CAM** tiene la capacidad de reconocer rostros, capturar un flujo de video, transmitir en tiempo real y conectarse a Internet por medio de Wi-Fi. Si se posa una persona frente al ESP32-CAM y este no reconoce el rostro, le enviará a través del **protocolo ESP-NOW** una señal de alerta a otro dispositivo que necesariamente debe tener incluido un chip ESP32. Este nuevo dispositivo será el encargado de enviar un correo de alerta al dueño o dueña del hogar. Luego, al momento de recibir el correo de alerta el o la interesada inmediatamente podrá conectarse remotamente al servidor web domótico para ingresar a la sección de video vigilancia y acceder al flujo de video del ESP32-CAM, logrando ver en tiempo real quién se encuentra fuera de su hogar. Esta conexión remota se logra gracias al dominio DDNS **No-IP**, al NAT que se realiza sobre el router del hogar y al controlador **Orange Pi Lite (OPLite)**.

2 MARCO TEÓRICO

La historia de la domótica surge a mediados de los años 60. En 1966 se desarrolló e implementó el primer intento de casa inteligente, pero no hubo gran avance hasta el año 1998 cuando se expandió la Internet y dio paso a la nueva era del Internet de las Cosas (IoT). Desde el 1999 hasta el 2006 las empresas se dedicaron a adaptar los electrodomésticos existentes a un sistema domótico. Finalmente, en el 2007, cuando se lanzó el primer smartphone al comercio, comenzaron a surgir las primeras aplicaciones para controlar remotamente los electrodomésticos del hogar [20].

En la actualidad, cada vez es más habitual ver instalado un sistema domótico en la vía pública y/o en los lugares habitados y no habitados, principalmente casas particulares y locales comerciales, respectivamente. Uno de los dispositivos que más ha crecido en el mercado de la domótica ha sido la cámara de videovigilancia. La finalidad de implementar un sistema domótico de videovigilancia es disminuir la cifra de robos y hurtos en los lugares previamente mencionados.

Gracias a los resultados entregados el 2019 por la revista hispanoamericana TECNOSeguro, se puede asegurar un crecimiento significativo en las ventas de cámaras de videovigilancia. Los datos registrados indican un incremento por tercer año consecutivo en el mercado de equipos de videovigilancia [21].

En este trabajo, dicha alza de ventas se analizó desde tres perspectivas. La primera apunta al aumento de inseguridad o vulnerabilidad que sienten los dueños y las dueñas cuando su casa queda sola. La segunda se centra en la cantidad de robos que se producen en casas particulares y/o locales comerciales. La última se enfoca en el costo monetario que tiene el implementar un sistema domótico de videovigilancia.

Debido a este último análisis es por el cual se realizó el presente trabajo. En promedio, instalar un sistema domótico básico de videovigilancia cuesta alrededor de 150.000 pesos chilenos [22], por lo cual, para muchos lugares habitados, monetariamente no es fácil acceder a este servicio.

El presente trabajo detalla la implementación de un sistema domótico de videovigilancia con reconocimiento facial en tiempo real y correo de alerta, economizando gastos monetarios pero sin disminuir la calidad del servicio, manteniendo una conexión remota, robusta y un sistema confiable.

2.1 REVISIÓN BIBLIOGRÁFICA

Estudios y análisis de la Fundación Paz Ciudadana de Chile muestran que, desde el 2010 hasta el 2015, aproximadamente el 20% del total de la muestra encuestada afirma que tuvieron un robo de hogar dentro de los últimos 6 meses hasta la fecha que se realizó la encuesta. En cambio, desde el 2016 al 2019 se registra una disminución de casos al 15%, y en los años 2020 y 2021 baja en promedio al 9,1% [23]. Para los registros de estos últimos dos años se debe considerar la existencia de la pandemia y el aumento de estadía en las casas.

Otro análisis que muestra la Fundación Paz Ciudadana son las medidas que están tomando los dueños o las dueñas para incrementar la seguridad del hogar. Los robos a las casas están repercutiendo en el estilo de vida de las personas. Cada vez es más inseguro salir a realizar deberes y dejar sola la casa. Mentalmente, el dueño o la dueña queda con la incertidumbre si entrarán a robar o no a su hogar. Resultados del índice de Paz Ciudadana detalla que:

> El 70% de las personas declara haberse puesto de acuerdo con sus vecinos para protegerse de la delincuencia. Adicionalmente, el 90% de las personas tomó al menos una conducta que restringe sus libertades (dejar de salir a ciertas horas; reforzar la seguridad de la vivienda; dejar de ir a ciertos lugares; y dejar de usar artículos de valor en público). (Fundación Paz Ciudadana, 2021, p. 63)

Por otra parte, gracias a los resultados entregados en el presente año (2021) por el Centro de Estudios y Análisis de Datos (CEAD) se puede analizar el comportamiento histórico en Chile de los robos y hurtos en lugares habitados, principalmente en casas particulares. Del total de casos denunciados desde el 2010 hasta el 2021, en promedio, el 59% de los casos se centran desde el 2010 al 2015, el 33% se centran en los años 2016 al 2019, y sólo el 8% en los últimos dos años [24]. Nuevamente, al igual que los registros de la Fundación Paz Ciudadana, esta baja puede ser afectada por la pandemia COVID-19.

Considerando estos análisis, por el comportamiento que han tenido los casos de robos en lugares habitados y/o no habitados en los últimos 10 años, es decir, desde el 2010 hasta el 2020, y el aumento de las ventas del mercado de equipos de videovigilancia en los mismos años, se puede deducir que, en gran parte el factor que ha influido en la baja de robos en casas particulares y locales comerciales se debe a los servicios de seguridad que entrega un sistema domótico de videovigilancia.

Por otra parte, desarrollar e implementar proyectos domóticos desde una Orange Pi para asegurar un sistema robusto y estable es bastante viable. Gracias a las pruebas realizadas por la comunidad de Orange Pi en español, es posible confirmar un levantamiento de servidor web exitoso. En este caso, se utilizará el sistema operativo Armbian, el controlador Orange Pi Lite y el software Apache para levantar dicho servidor web domótico [10].

Para el caso del ESP32-CAM se ha utilizado en bastantes proyectos de reconocimiento facial, obteniéndose resultados con alta tasa de confiabilidad. En los proyectos implementados se utilizan principalmente las bibliotecas ESP-WHO y ESP-FACE para realizar el trabajo de detectar y reconocer rostros en tiempo real [26] [27] [28].

> "WebSockets son buenos para situaciones en las que necesita comunicaciones casi en tiempo real, como los siguientes escenarios y aplicaciones: Juegos multijugador online, Aplicaciones de chat, Internet de las cosas, Aplicaciones en tiempo real, Aplicaciones en tiempo real." (Lee, 2020, p. 1).

Un protocolo dedicado al Internet de las cosas es el WebSocket, debido a su comportamiento en la comunicación dúplex completa y continua entre cliente y servidor [29]. Ya que el sistema domótico de videovigilancia a implementar necesita reconocer rostros en tiempo real, el protocolo WebSocket es ideal para trabajar con bajos niveles de latencia en las transferencias de datos.

3 DESARROLLO

3.1 SELECCIÓN DE HARDWARE Y SOFTWARE

El sistema domótico de videovigilancia desarrollado en el presente documento busca el equilibrio entre lo confiable, robusto y económico. La gama de equipos que tienen estas características y son capaces de implementar dicho sistema son los controladores y las cámaras IP.

Para encontrar un modelo de controlador que se adapte a los objetivos del sistema domótico de videovigilancia se consideraron las siguientes características [1]: conexión a Internet vía Wi-Fi para facilitar los puntos de conexión al resto de dispositivos del sistema domótico; velocidad de CPU mínima de 1.2GHz para evitar una sobrecarga de trabajo en un servidor web; ser ligero en peso; y que tenga un volumen aceptable para ubicar en cualquier sitio. Todas estas características las entrega un Orange Pi Lite (OPLite), el cual se utilizará para implementar este sistema domótico.

Por otro lado, es necesario implementar el flujo de video de vigilancia a tiempo real que va a reconocer rostros y enviar un correo de alerta cada vez que no lo reconozca. Para llevar a cabo esta etapa, se recurre a una cámara IP que contenga las siguientes características: flujo de video a tiempo real; conexión a Internet vía Wi-Fi; tecnología que tenga la capacidad de detectar y reconocer rostros; sea capaz de enviar un correo de alerta; y, por último, un volumen adecuado para poder situar el dispositivo como una cámara frontal del hogar. Estas características las posee el módulo ESP32-CAM.

A continuación, se detallará más a fondo las funcionalidades de cada dispositivo dentro del sistema domótico y las limitantes que tiene cada uno.

3.1.1 Hardware

3.1.1.1 Orange Pi Lite

El OPLite cumple la función de ser el servidor web del sistema domótico. Está encargado de levantar el servidor a la Internet y de redireccionar las peticiones del cliente al flujo de video de vigilancia en tiempo real. Para energizar al controlador se necesita un transformador de 5[V] / 3[A]. Existe un requerimiento importante que se debe considerar al momento de encender el OPLite y es tener incorporado en el puerto microSD (tamaño mínimo 4Gb) una tarjeta con un sistema operativo (SO) instalado. En este proyecto se utiliza el SO Armbian para asegurar una estabilidad al servidor

web domótico. Su conexión a Internet a través de Wi-Fi facilita la ubicación del dispositivo y la interacción con el ESP32-CAM.

Por otra parte, una desventaja que posee este controlador es su alta temperatura de operación, se sobrecalienta con facilidad desde su encendido. Para solucionar este problema se recomienda utilizar un ventilador al momento de trabajar con él, así se evitará estropear el OPLite.

A continuación, se aprecian los distintos dispositivos que se necesitan para utilizar el controlador Orange Pi Lite:

Figura 3.1: Dispositivos para utilizar un Orange Pi Lite

Ahora bien, los principales componentes del OPLite son: los puertos de entrada, su procesador, su antena Wi-Fi y la memoria RAM. La siguiente imagen detalla la ubicación de cada componente:

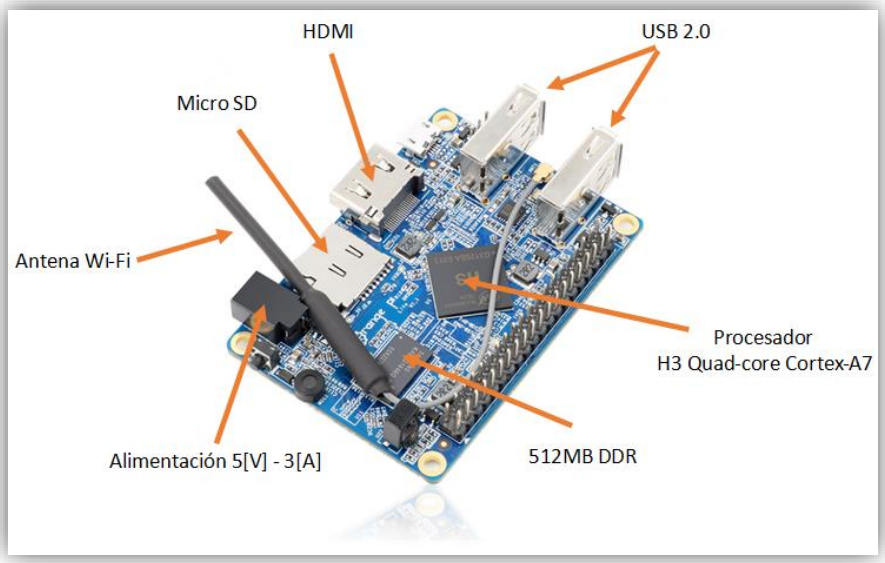


Figura 3.2: Principales componente del controlador Orange Pi Lite

El funcionamiento de este controlador se logra a través de la tecnología que tiene el procesador Cortex-A7 [2], desarrollado por ARM. Algunas de las características que ayudan a realizar el sistema domótico implementado son: la tecnología NEÓN, encargada de acelerar los algoritmos de procesamiento de señales y multimedia; las extensiones DSP y SIMD para optimizar operaciones a muy alta velocidad; y los cachés de nivel 1 y 2 para optimizar el rendimiento y el consumo de energía.

3.1.1.2 ESP32-CAM

El ESP32-CAM [3] es uno de los módulos más funcionales y completos para utilizar como cámara de video vigilancia debido a las capacidades que posee. Las que más resaltan para el desarrollo del sistema domótico son:

- Conexión a Internet a través de Wi-Fi
- Detección y reconocimiento de rostros
- Trabajar los formatos JPEG, BMP y GRAYSCALE
- Dimensión volumétrica pequeña
- Soportar un rango ambiental entre -40[°C] ~ 90[°C], con 90% de humedad relativa
- Envío de correo electrónico.

El ESP32-CAM puede ser alimentado con 3,3 o 5[V]. En este sistema se recomienda utilizar 5[V] debido a su trabajo continuo para detectar y reconocer los rostros a tiempo real. Al igual que el OPLite, se recomienda utilizar el dispositivo con un ventilador para evitar un sobrecalentamiento. A continuación, se puede apreciar el módulo ESP32-CAM con sus principales componentes:

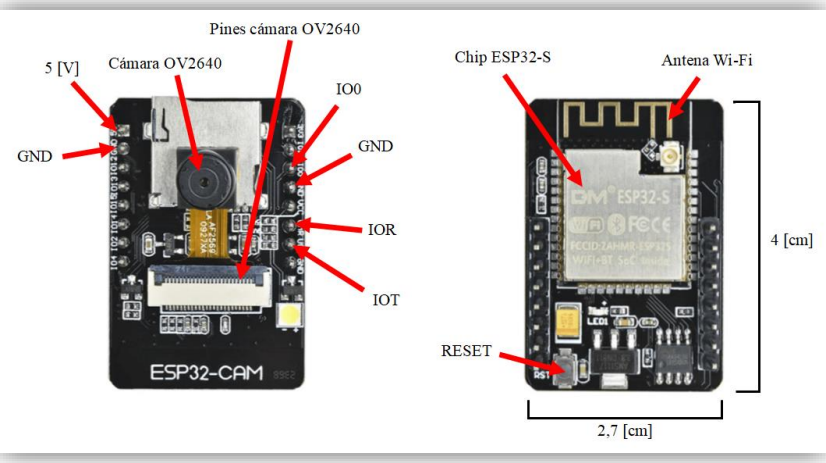


Figura 3.3: Módulo ESP32-CAM

El módulo se compone de un chip ESP32-S2 [25] desarrollado por Espressif; una antena Wi-Fi; un botón RESET; un puerto de entrada para incorporar una cámara OV2640; y 16 pines, 10 GPIO, 3 de alimentación y 3 tierra.

En este proyecto se utilizan sólo 6 pines para lograr implementar el sistema domótico de videovigilancia, estos corresponden a 3 GPIO, 1 alimentación y 2 tierra. Los pines utilizados poseen las siguientes características:

- GND: Tierra
- 5 [V]: Alimentación
- Pines cámara OV2640: Pines de conexión entre la cámara OV2640 y el chip ESP32-S2. Se encargan de programar y configurar las características de la cámara
- Pin IO0: Se conecta a GND para dejar el módulo en modo programación
- Pin IOR: Pin GPIO para recibir los datos de programación
- Pin IOT: Pin GPIO para transmitir los datos de programación

Por último, para poder cargar el código desde una computadora al ESP32-CAM se necesita un convertidor USB a TTL y conectar los pines como se detalla a continuación:

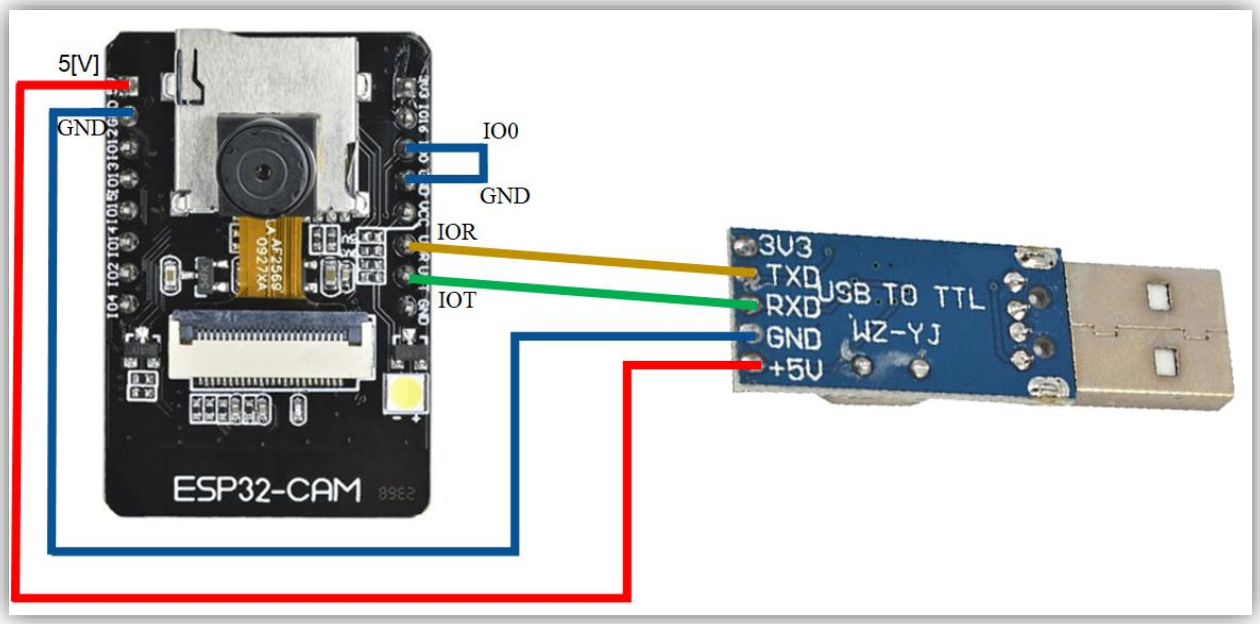


Figura 3.4: Conexión convertidor USB a TTL al ESP32-CAM

3.1.2 Software

Para llevar a cabo el sistema domótico de videovigilancia se utilizan softwares específicos para asistir la programación del controlador OPLite y del módulo ESP32-CAM.

Primeramente, se encuentra el programa Arduino IDE [12]. Es el software ideal para desarrollar la etapa de detección y reconocimiento facial, debido a que el módulo ESP32-CAM corresponde a la gama de dispositivos electrónicos Arduino. En este proyecto se utilizan dos lenguajes de programación para implementar el código que controla el ESP32-CAM, estos son C++ y PHP. Para utilizar PHP se acude al lenguaje de marcas de hipertexto HTML.

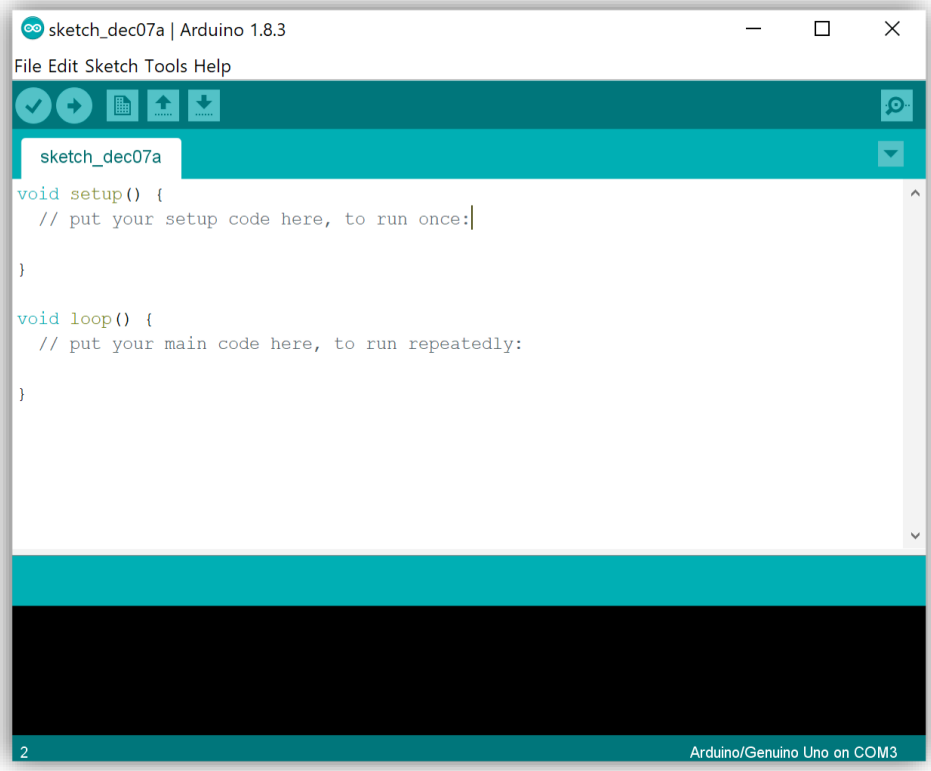


Figura 3.5: Software Arduino IDE

3.1.2.1 Orange Pi Lite

3.1.2.1.1 Sistema operativo

El sistema operativo (SO) más utilizado para trabajar un OPLite es Armbian [2], por su robustez y desarrollo dedicado a este controlador. Su sistema de compilación está basado en la de Debian. Para poder utilizar Armbian se debe descargar desde su página web oficial. También, necesita

el software Win32 Disk Imager para poder instalar Armbian en la tarjeta Micro SD. Hay que tener cuidado en esta etapa debido a que, si falla la instalación o se daña la tarjeta, el OPLite no encenderá.

Existen dos formas de utilizar Armbian. La primera es la más amigable para el manejo del SO, y corresponde a la interfaz de escritorio. Al OPLite se le debe conectar un teclado y un mouse para obtener el control. Además, es necesario conectarlo a una pantalla a través de su salida HDMI.

La segunda forma de utilizar el controlador es por medio de la interfaz de línea de comandos. Esta es más específica y se necesitan conocimientos previos de los códigos básicos para trabajar en Linux.

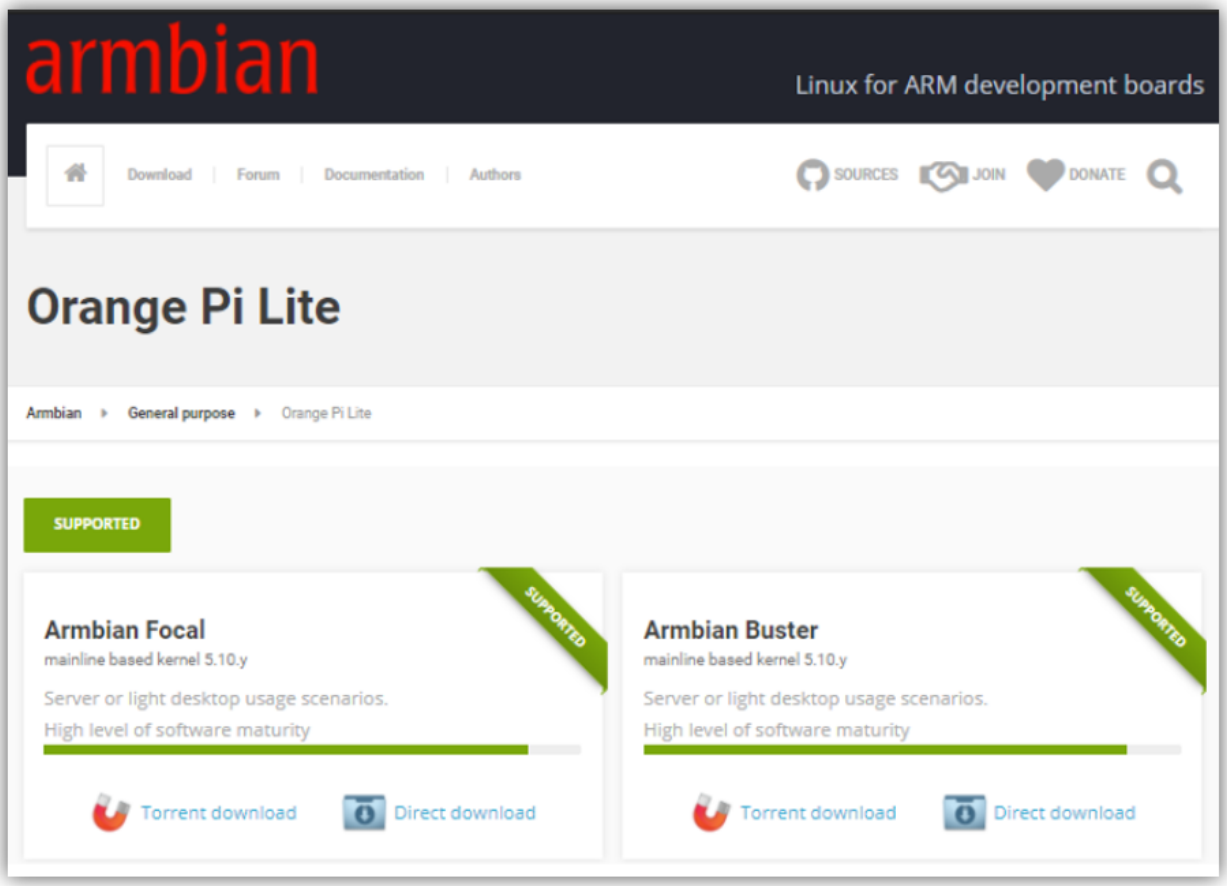


Figura 3.6: Sistema operativo Armbian

En este caso se utiliza el OPLite a través de la interfaz de línea de comandos. Se descarga e instala Armbian Buster de la Figura 4.7. Para ser uso de este medio (remoto) se utiliza el software Putty.

3.1.2.1.2 Putty

Esta aplicación tiene como objetivo conectarse por Wi-Fi al controlador OPLite, a través de un enlace remoto desde la computadora personal. La conexión se realiza a través de la IP que el router del hogar le asigna al OPLite.

Este software es usualmente utilizado en Windows para tener control remoto de diferentes dispositivos electrónicos, ya que ofrece una licencia gratuita. Para ingresar al OPLite se requiere sólo de la IP y el puerto, que usualmente es el 22 para utilizar el protocolo de comunicación de red remota SSH.

A continuación, se puede observar un diagrama de flujo para entender mejor el enlace entre todos los softwares utilizados en el control del OPLite:

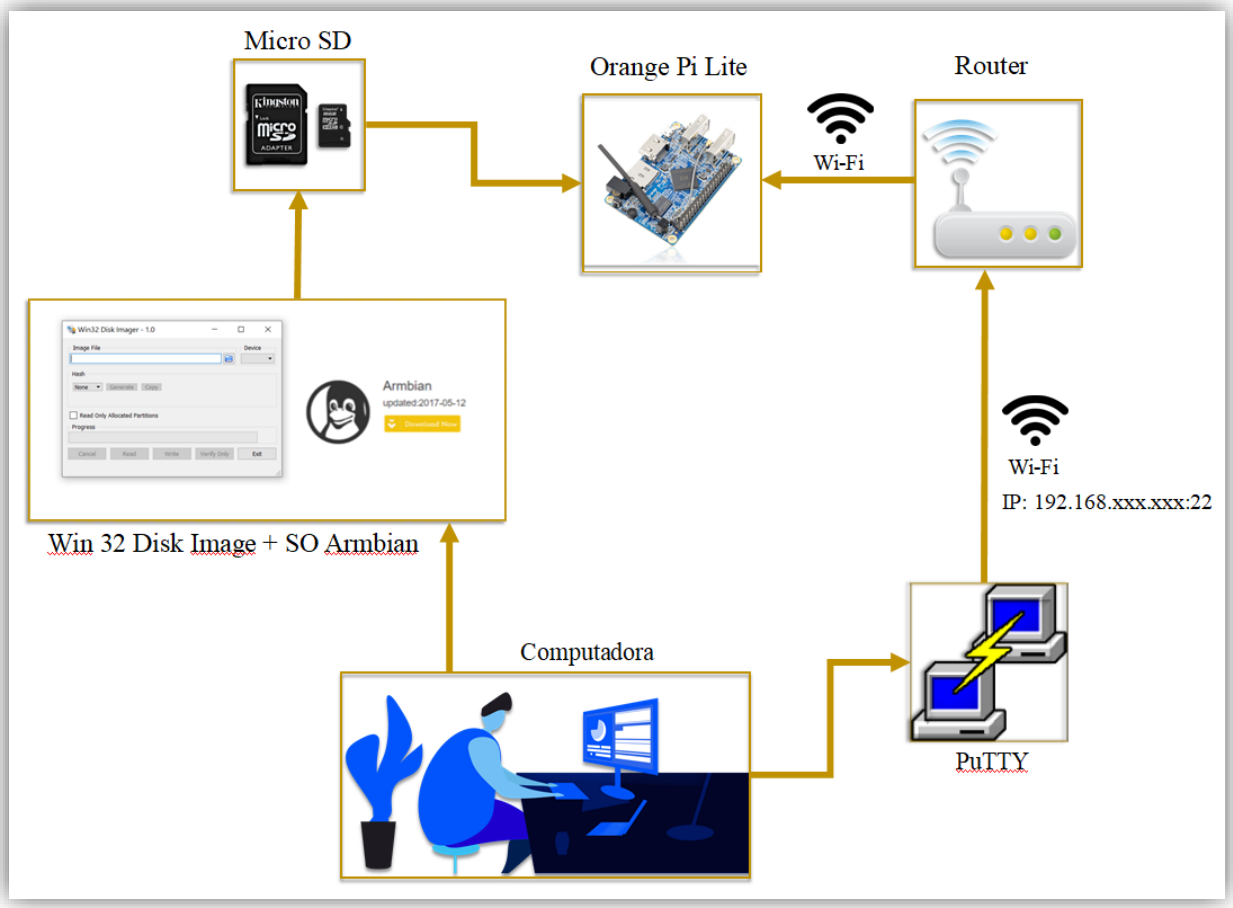


Figura 3.7: Diagrama de flujo del OPLite

3.1.2.2 ESP32-CAM

3.1.2.2.1 Arduino IDE

Arduino IDE es un software gratuito dedicado a trabajar toda la gama de dispositivos electrónicos relacionados a Arduino. En este caso, el módulo ESP32-CAM se encuentra dentro de esta gama de productos.

Para trabajar el módulo se necesita instalar las siguientes herramientas y bibliotecas dentro de Arduino IDE:

- **Incorporar URL para el soporte de la placa esp32.** Dentro de Arduino IDE ingresar a: Archivo > Preferencias > Gestor de URLs Adicionales de Tarjeta, aquí se debe ingresar el siguiente enlace: https://dl.espressif.com/dl/package_esp32_index.json

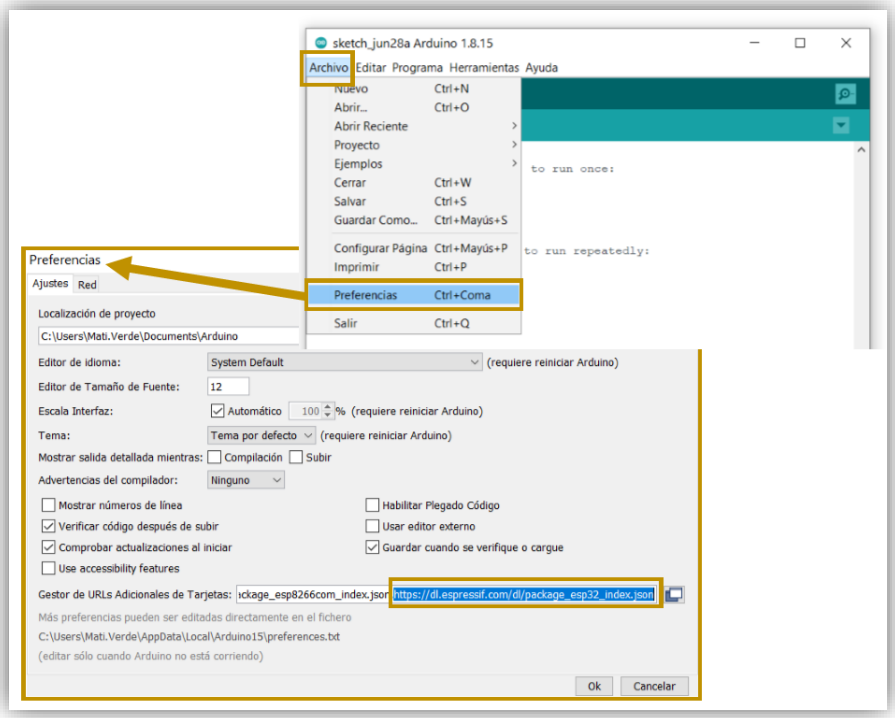


Figura 3.8: Incorporar URL del ESP32

- **Integrar la placa ESP32.** Dentro de Arduino IDE entrar a Herramientas > Placa: > Gestor de tarjetas. Aquí se busca e integra la placa ESP32.

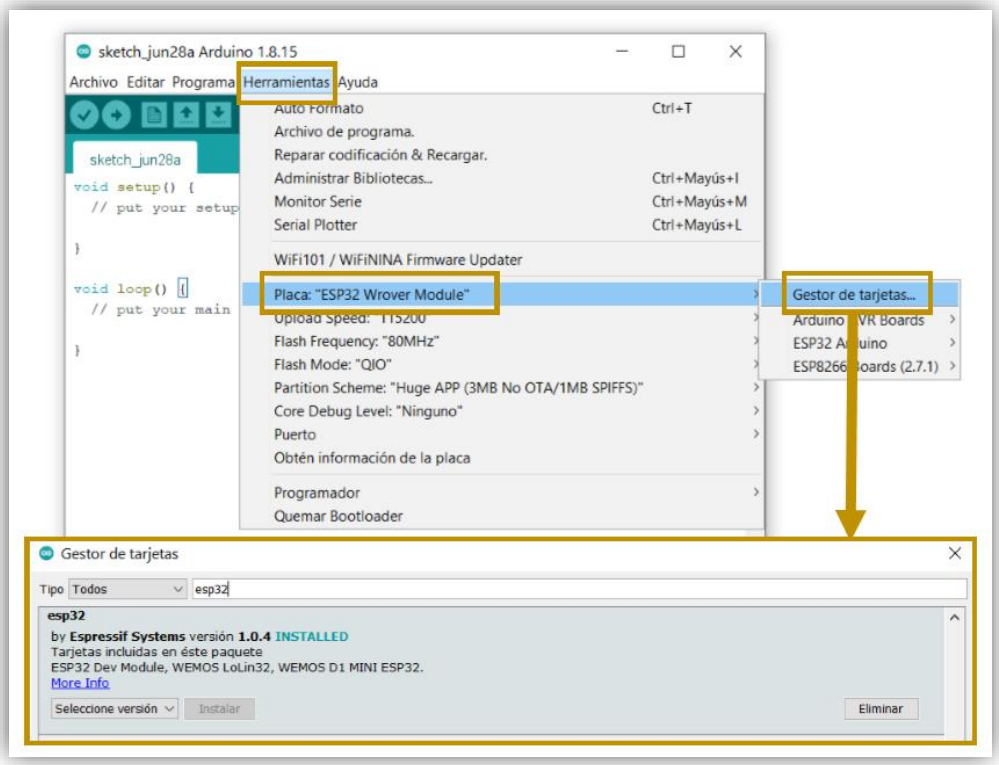


Figura 3.9: Integrar placa esp32 al Arduino IDE

- **Integrar protocolo de comunicación WebSocket.** Dentro de Arduino IDE entrar a Herramientas > Administrar Bibliotecas. Luego buscar e instalar *"arduinowebsockets"*.

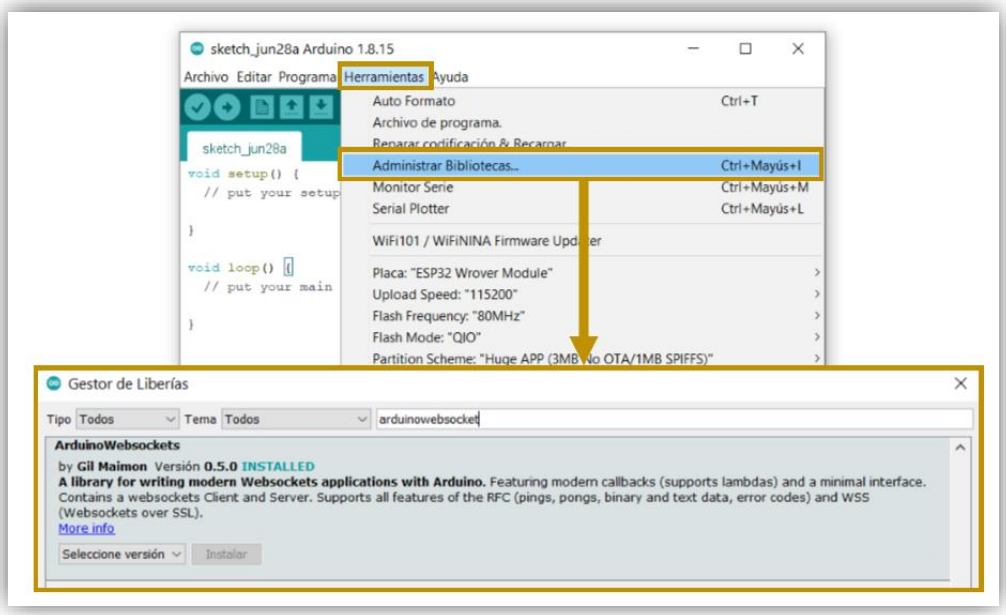


Figura 3.10: Biblioteca WebSocket en Arduino IDE

- **Integrar biblioteca ESP Mail Client [17] para enviar correo electrónico de alerta.** Descargar biblioteca[1] oficial en formato ".zip". Dentro de Arduino IDE entrar a Programa > Incluir Librería > Añadir biblioteca .ZIP > Instalar ".zip" *"Esp Mail Client"*.

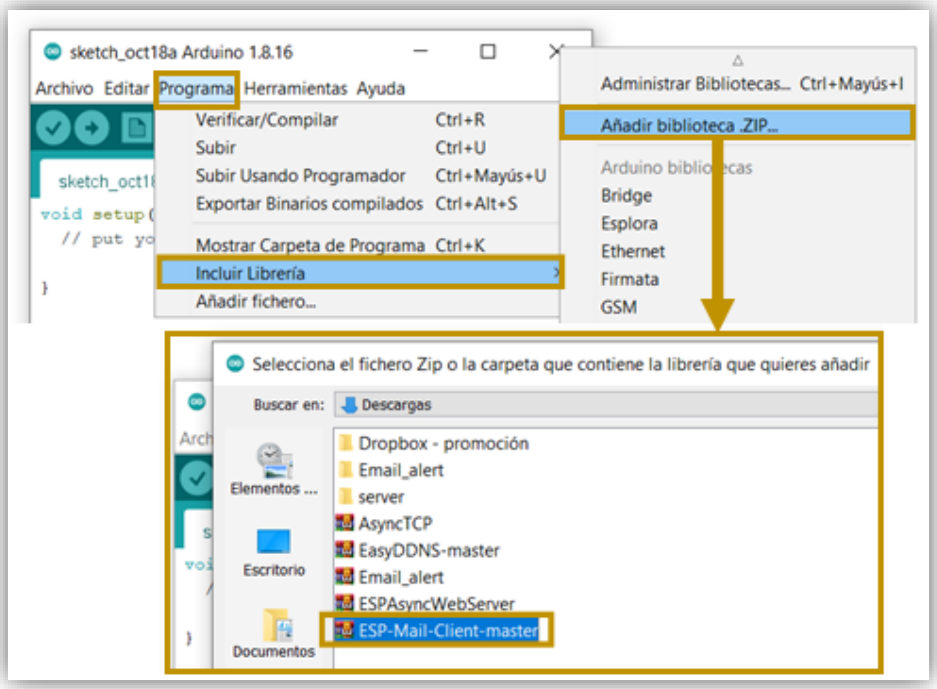


Figura 3.11: Incorporar biblioteca Esp32 Mail Client en Arduino IDE

- **Integrar biblioteca Easy DDNS para registrar el ESP32-CAM con un dominio DDNS.** Descargar biblioteca[2] oficial en formato ".zip". Dentro de Arduino IDE entrar a Programa > Incluir Librería > Añadir biblioteca .ZIP. Buscar e instalar *"EasyDDNS"*.

[1] https://github.com/mobizt/ESP-Mail-Client/archive/refs/heads/master.zip
[2] https://github.com/ayushsharma82/EasyDDNS.git

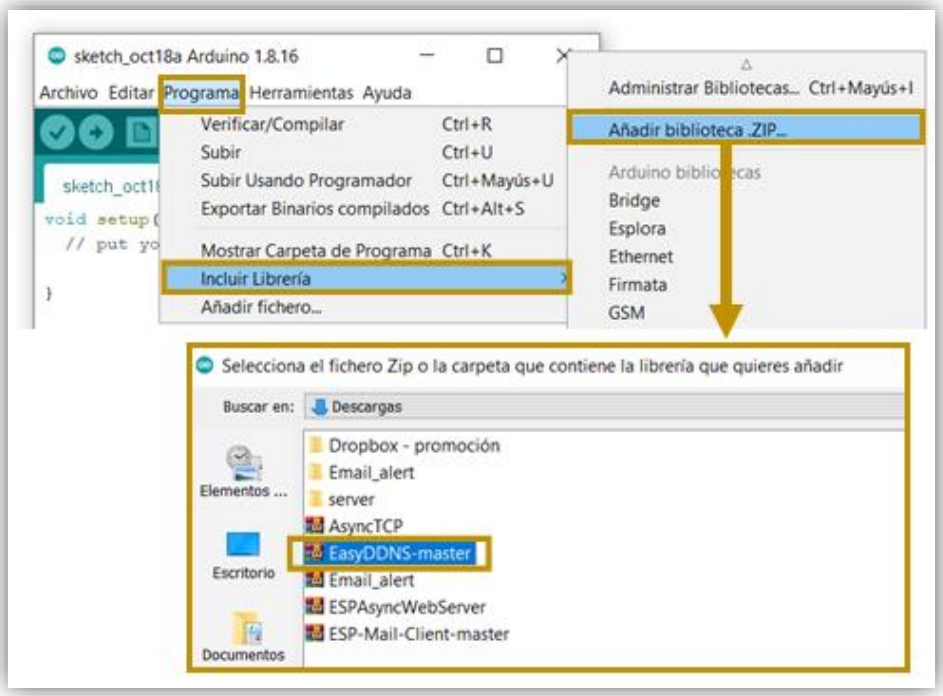


Figura 3.12: Incorporar biblioteca Easy DDNS para registrar dominio DDNS

- **Integrar biblioteca ESP-NOW para vincular múltiples dispositivos ESP32 entre sí.** En este caso no es necesario instalar la biblioteca desde "Administrar Bibliotecas", ya que viene incorporada en el Arduino IDE para el uso específico de los chips ESP32. Para utilizar esta librería, en el código se debe agregar la línea "*#include <esp_now.h>*".

3.1.2.2.2 ESP-IDF

Además de incorporar las bibliotecas vistas en la sección anterior, es necesario instalar el entorno de desarrollo de software ESP-IDF [4] [13], desarrollado por Espressif. Este entorno utiliza bibliotecas dedicadas a la detección y reconocimiento facial para poder aplicarlas en un módulo ESP32-CAM. El entorno ESP-IDF se obtiene gracias a un instalador de herramientas[3] que facilita la misma comunidad.

El objetivo de utilizar este entorno es poder instalar y trabajar con dos bibliotecas que se dedican al desarrollo de la detección y reconocimiento facial en los chips ESP32. La primera biblioteca corresponde a la red neuronal del proceso, llamada ESP-WHO [6]. Tiene la capacidad de crear el

[3] Instalación de herramientas ESP-IDF: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/windows-setup.html

proceso de detectar y reconocer los rostros de las personas; la segunda corresponde a la implementación de este proceso, llamada ESP-FACE [7]. Esta hace un llamado a la lógica de detección y reconocimiento facial que utiliza ESP-WHO.

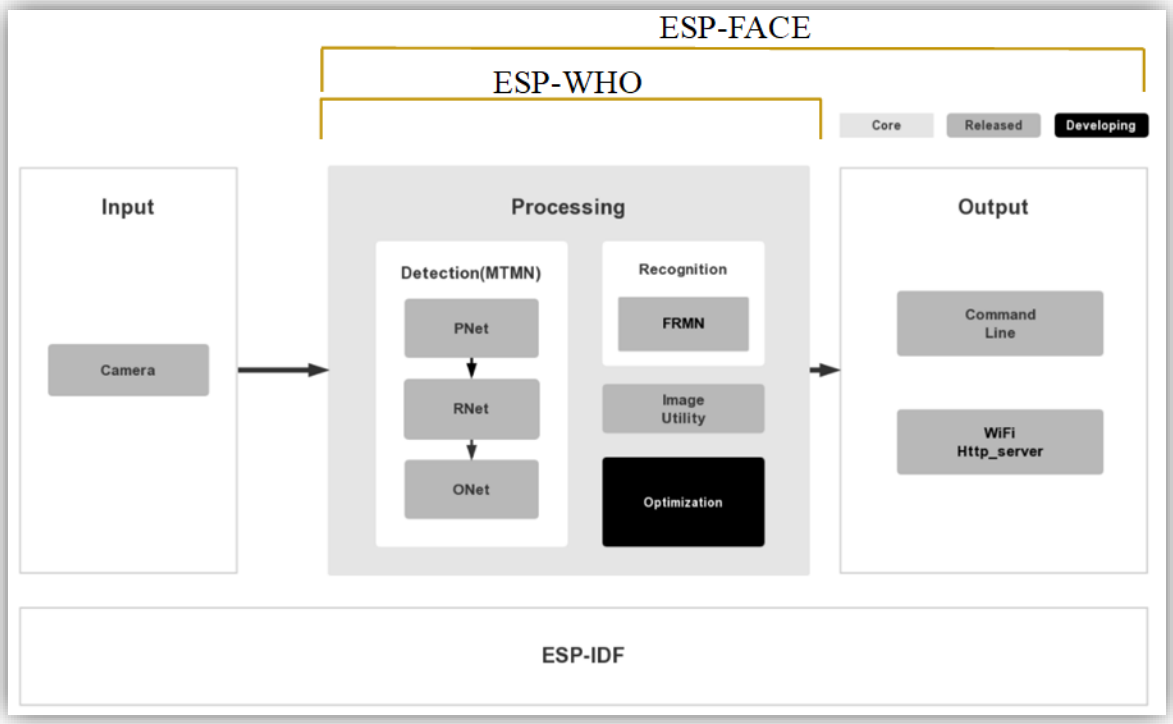


Figura 3.13: Entorno ESP-IDF para detectar y reconocer rostros

ESP-WHO [6] es una biblioteca que se dedica a desarrollar una red neuronal de detección y reconocimiento facial en un chip Espressif ESP32. Para utilizar esta función en un módulo ESP32-CAM es necesario utilizar la biblioteca ESP-FACE, debido a que esta contiene las API de las redes neuronales del ESP-WHO.

ESP-FACE [7] es una biblioteca desarrollada por Espressif, brinda las funciones de detectar y reconocer rostros en un chip ESP32. Para llevar a cabo este trabajo se utilizan dos modelos: MTMN para detectar y FRMN para reconocer.

MTMN se basa en la arquitectura MobileNetV2 [8] y en las redes convolucionales en cascada multitareas [9]. MobileNetV2 es una nueva arquitectura utilizada en dispositivos móviles que tiene como finalidad optimizar frames a través de una convolución más eficiente. Su trabajo es redimensionar la imagen a una más pequeña para así trabajar una menor cantidad de cuadros por segundo.

Por otra parte, las redes convolucionales en cascada multitareas se dividen en 3 filtros que van refinando cada vez más las muestras del rostro. La primera etapa consta de tomar varias muestras candidatas del rostro redimensionado, esta etapa se llama Proposal Network o red de propuestas rápida (P-Net). La segunda etapa es llamada Refine Network o red de refinamiento (R-Net) y corresponde a una refinación de la etapa anterior. Por último, se encuentra la tercera etapa llamada Output Network o red de salida (O-Net), entrega el cuadro delimitador final de todas las muestras y la posición de los puntos de referencia del rostro (cabeza, ojos, nariz y boca).

Para refinar aún más las muestras tomadas durante estas tres etapas, se implementa la técnica Non-Maximum Suppression (NMS). Esta se encarga de seleccionar una de todas las muestras candidatas que se encuentran demasiadas superpuestas. A continuación, se muestra como funciona cada etapa en conjunto a la técnica NMS.

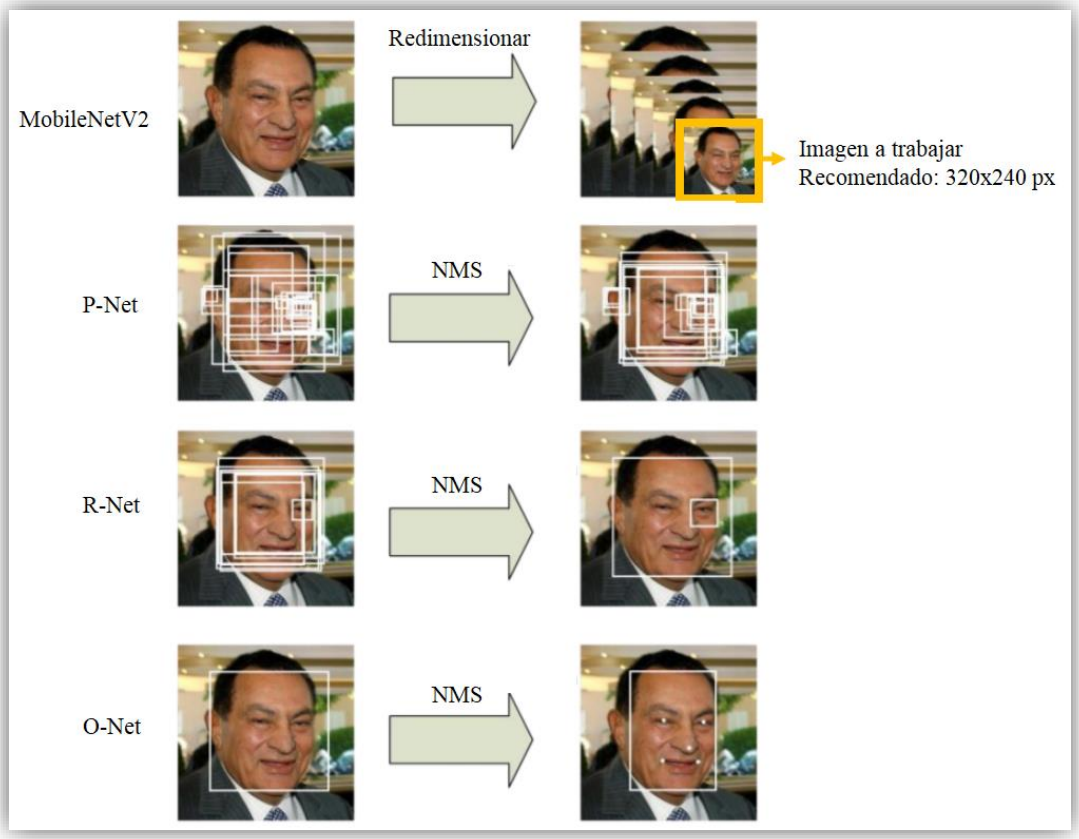


Figura 3.14: Modelos MobileNetV2 y redes convolucionales en cascada multitareas

3.2 IMPLEMENTACIÓN

La siguiente sección detalla la implementación del servidor domótico de videovigilancia en dos etapas. La primera explica los pasos para levantar un servidor web desde un OPLite y la segunda detalla la codificación para utilizar un módulo ESP32-CAM como mecanismo de seguridad de reconocimiento facial.

Para llevar a cabo la primera etapa se debe trabajar a nivel de redes de datos. A grandes rasgos, la primera mitad de esta sección se centrará en la configuración de direcciones IP (pública y privada) del OPLite y en incorporar los servicios que brindan los dominios DDNS [11]. El objetivo de esta primera implementación es habilitar el acceso remoto al servidor web del OPLite recurriendo a los dominios DDNS que entrega la comunidad de manera gratuita.

Luego, la segunda etapa de esta implementación detalla la codificación (desarrollada en C++) sobre el módulo ESP32-CAM para que logre la detección y reconocimiento facial a tiempo real, incluyendo un correo de aviso como sistema de alerta. Esto se realiza gracias al apoyo de las bibliotecas ESP-WHO, ESP-FACE y el entorno de desarrollo ESP-IDF.

De modo general, en esta sección se detalla el levantamiento del servidor web domótico para habilitar la conexión remota vía Wi-Fi al OPLite, y la codificación para detectar y reconocer rostros a tiempo real con sistema de alerta incorporado. Este último punto se realiza gracias a las capacidades que posee el chip ESP32 en el módulo ESP32-CAM.

3.2.1 Orange Pi Lite

Luego de configurar el OPLite como se detalló en el apartado 4.1.2.1, se debe programar internamente para que funcione como un servidor web [10]. Primeramente, para implementar este servidor web se debe modificar su IP privada dinámica a una IP privada estática o fija [5], debido a que el router del hogar cada cierto tiempo puede asignar más de una IP al controlador. La finalidad de esta etapa es lograr enlazar la IP pública con la IP privada, para tener un servidor web con acceso remoto.

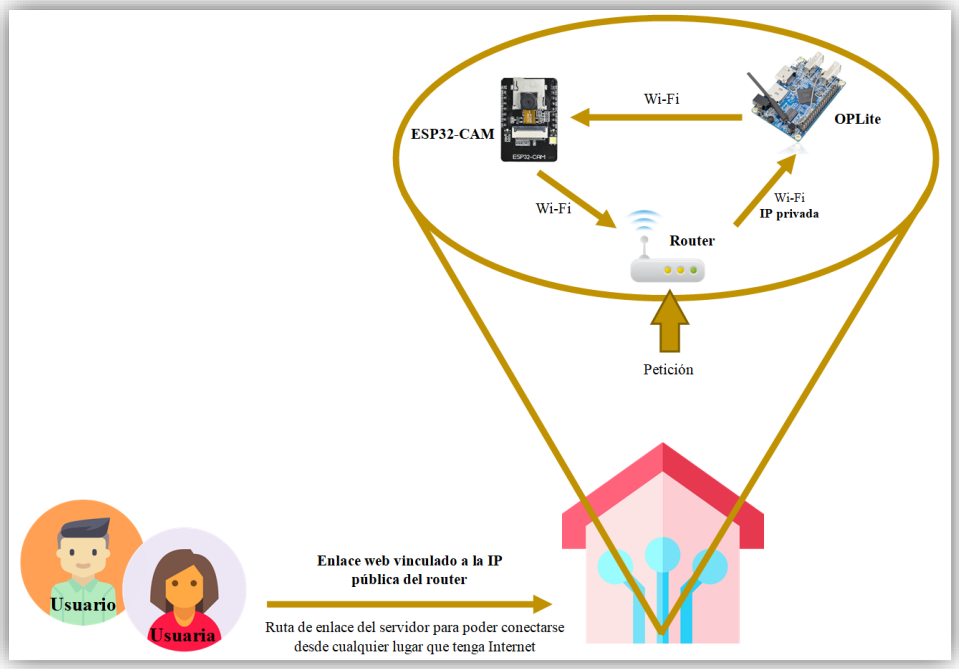


Figura 3.15: Diagrama flujo de petición

Para realizar esta modificación sobre la IP privada se tiene que alterar la carpeta "interfaz de red". Desde la terminal del OPLite se debe ejecuta el comando *"sudo nano /etc/network/interfaces"* para abrir la carpeta previamente mencionada. Luego, dentro de esta se debe detallar la IP privada fija con la siguiente especificación:

auto *nombre_interfaz* [4]
allow-hotplug *nombre_interfaz*
iface *nombre_interfaz* inet static
    address *dirección_IP* [5]
    gateway *dirección_puerta_de_enlace* [6]
    netmask *dirección_máscara* [7]
    broadcast *dirección_difusión* [8]

Una vez definida la IP privada fija del OPLite se debe enlazar con la IP pública del router. Para realizar esto hay que contactarse con la compañía de Internet del hogar, para que el técnico deje

[4] El nombre de la interfaz se conoce con el comando *"sudo ifconfig"*. En la OPLite utilizada aparece como *"wlan0"*
[5] Es la dirección IP que se quiere definir como estática
[6] La dirección de la puerta de enlace se conoce con el comando *"route"* o *"route -n"*
[7] La dirección de la máscara se conoce con el comando *"sudo ifconfig"*
[8] La dirección de la red de difusión se conoce con el comando *"sudo ifconfig"*

habilitada y vinculada la IP pública con la IP privada que se detalló en el procedimiento anterior, en conjunto al puerto que se utilizará. En este caso se trabaja con el puerto 443.

Hasta este punto se han logrado enlazar ambas IP, pero existe otro problema que se abordará a continuación. Las primeras veces que se quiera ingresar al servidor del OPLite se deberá buscar en el navegador web la IP pública[9] del router, pero como las compañías de Internet trabajan con IP públicas dinámicas, esta IP irá cambiando en un tiempo determinado por la compañía y la siguiente vez que cambie la IP no se tendrá acceso al servidor, porque no existirá conocimiento de este cambio de IP. Para solucionar este problema se utiliza un servidor DDNS.

Los servidores DDNS (Dynamic Domain Name System) trabajan con nombres de dominios fijos en la Internet. Se encargan de traducir el nombre de una dirección web a una dirección IP pública de un router. Su proceso es el siguiente: el nombre del dominio se enlaza con la IP para detectar su momento de cambio, cuando esto ocurra, actualizará automáticamente su enlace (esta actualización automática depende directamente del dominio que se utiliza) y la dirección web mantendrá su nombre. En este caso se utilizó un dominio llamado No-IP.

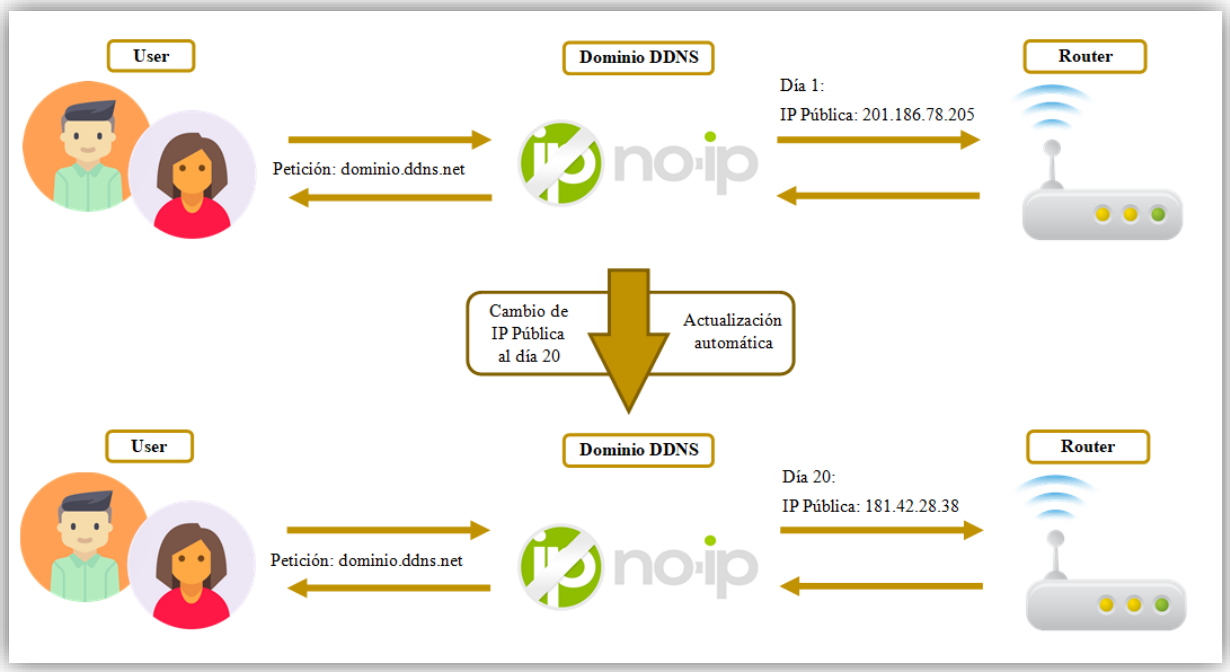


Figura 3.16: Dominio No-IP

[9] Página web utilizada para encontrar la IP pública del router: *"https://www.cual-es-mi-ip.net"*

Para utilizar No-IP se deben seguir los siguientes pasos:

1- Ingresar desde la computadora a la página: *"https://www.noip.com/es-MX"*
2- Crear una cuenta en el dominio No-IP
3- Crear un nombre de dominio para el servidor web y anotar datos de inicio de sesión

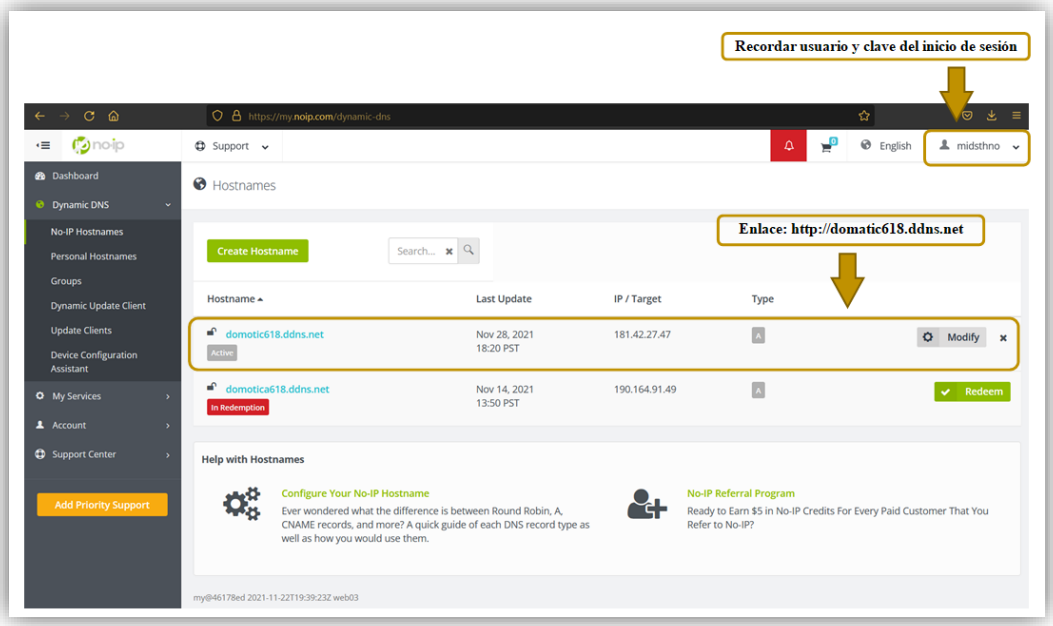


Figura 3.17: Nombre dominio No-IP y sesión utilizada

Luego de crear y vincular el enlace entre el dominio DDNS con la IP pública del router, se debe configurar el OPLite para levantar el dominio creado en No-IP. Para realizar esto es necesario instalar el software No-IP DUC en el OPLite.

No-IP DUC es una herramienta que permite actualizar automáticamente el vínculo del dominio DDNS con la IP pública cada vez que esta cambia. A continuación, se detallan los pasos para instalar No-IP desde la terminal del OPLite:

1- Se crea una carpeta dedicada a No-IP con *"sudo mkdir /home/pi/noip"*
2- Ingresar a la carpeta creada: *"sudo cd /home/pi/noip"*
3- Descargar No-IP DUC: *"sudo wget https://www.noip.com/client/linux/noip-duc-linux.tar.gz"*, luego, *"sudo tar vzxf noip-duc-linux.tar.gz"*

Una vez descargado el software No-IP DUC, desde el directorio que se creó se deben ubicar los archivos descargados, para realizar esto se utiliza *"sudo cd noip-22.1.9-1"*. Luego, para instalar el programa se deben ejecutar los siguientes códigos:

sudo make; *sudo make install*

Para terminar de configurar el software y dejarlo habilitado en el controlador, se debe iniciar sesión y especificar la frecuencia de verificación de IP pública.

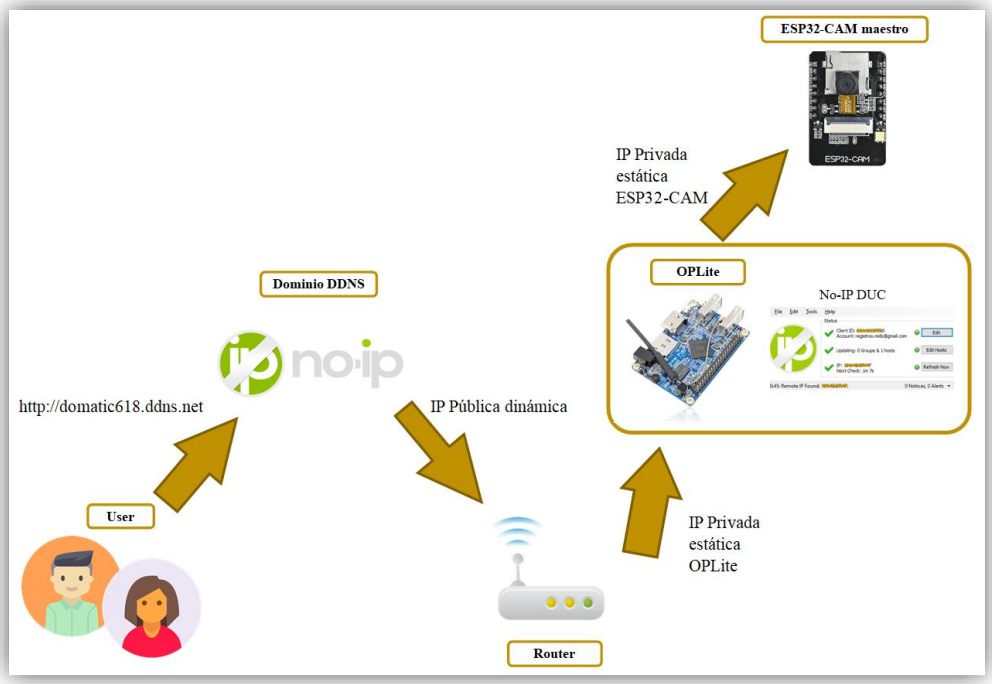


Figura 3.18: Flujo del dominio No-IP

3.2.2 ESP32-CAM

Una vez preparado el entorno de trabajo Arduino IDE (apartado 3.1.2.2) se debe desarrollar el código que pueda reconocer rostros y enviar un correo de alerta cada vez que no lo reconozca. Primeramente, se explicará el comportamiento del protocolo de comunicación WebSocket [14] frente a las solicitudes del cliente. Luego, se detallarán las principales funciones para detectar y reconocer rostros a tiempo real. Esta acción la realiza el ESP32-CAM que trabaje como dispositivo maestro. Luego, a este mismo código se le incorporará el dominio No-IP creado en el apartado 3.2.1. Finalmente, para enviar el correo de alerta se detallarán dos códigos. El primero corresponde al que

envía la señal de alerta, esta se desarrolla en la codificación del ESP32-CAM maestro. El segundo se debe implementar en el ESP32-CAM esclavo, dispositivo que va a recibir la señal de alerta.

Por otra parte, el ESP32-CAM maestro además de cumplir la función de reconocer rostros va a crear su propio servidor web, con la finalidad de poder entregar el video a tiempo real. Para realizar este trabajo se utiliza el lenguaje de etiqueta HTML pero transformada en hexadecimal.

3.2.2.1 Protocolo WebSocket

El protocolo WebSocket proporciona una comunicación bidireccional y full-duplex entre cliente y servidor, es decir, ambos pueden transmitir y recibir datos al mismo tiempo. En este caso, el cliente corresponde al usuario y el servidor al ESP32-CAM.

Primeramente, para utilizar este protocolo se debe incluir la biblioteca **#include <ArduinoWebsockets.h>**. Luego, se define el puerto 82 para realizar la comunicación entre cliente y servidor. Este puerto se fija en el apartado *void setup()* del código, con la expresión **socket_server.listen(82)**, donde "socket_server" es un objeto de clase WebSocketServer dedicado a recibir solicitudes de los clientes. Por último, para recibir la acción de solicitud del cliente, en el *void loop()* se debe utilizar la expresión **auto client = socket_server.accept()**. Al momento de recibir la solicitud del cliente, el websocket server deberá realizar la tarea que se le especificó.

En este sistema domótico de videovigilancia existen 2 tareas: agregar nuevo rostro y reconocer rostro. Si se quiere agregar un nuevo rostro a la base de datos primero se debe agregar el nombre con la siguiente función:

```
if (msg.data().substring(0, 8) == "capture:") {
  g_state = START_ENROLL;
  char person[FACE_ID_SAVE_NUMBER * ENROLL_NAME_LEN] = {0,};
  msg.data().substring(8).toCharArray(person, sizeof(person));
  memcpy(st_name.enroll_name, person, strlen(person) + 1);
  client.send("CAPTURING");
}
```

Figura 3.19: Función para indicar la tarea de agregar nuevo rostro y guardar el nombre del rostro

Donde *g_state* corresponde a una variable global del código que se encarga de declarar si la tarea específica es agregar (START_ENROLL) o reconocer (START_RECOGNITION). Luego, la expresión *msg_data()* se encarga de almacenar el fragmento del string *person* que contiene el nombre

del rostro. *Memcpy* se encarga de guardar dicho fragmento en la variable *st_name*. Finalmente, se envía la respuesta al cliente con la expresión *client.send("CAPTURING")*.

Luego, para guardar el rostro con su respectivo nombre en la memoria del chip ESP32 se utiliza la siguiente función:

```
if (g_state == START_ENROLL)
{
  int left_sample_face = do_enrollment(&st_face_list, out_res.face_id);
  char enrolling_message[64];
  sprintf(enrolling_message, "SAMPLE NUMBER %d FOR %s", ENROLL_CONFIRM_TIMES - left_sample_face, st_name.enroll_name);
  client.send(enrolling_message);
  if (left_sample_face == 0)
  {
    ESP_LOGI(TAG, "Enrolled Face ID: %s", st_face_list.tail->id_name);
    g_state = START_RECOGNITION;
    char captured_message[64];
    sprintf(captured_message, "FACE CAPTURED FOR %s", st_face_list.tail->id_name);
    client.send(captured_message);
    send_face_list(client);
  }
```

Figura 3.20: Función encargada de guardar el nuevo rostro y responder al cliente

Donde la subfunción principal *do_enrollment()* realiza todo el trabajo de guardar el nuevo rostro en la memoria del ESP32-CAM. A continuación, se puede observar esta subfunción:

```
static inline int do_enrollment(face_id_name_list *face_list, dl_matrix3d_t *new_id)
{
  ESP_LOGD(TAG, "START ENROLLING");
  int left_sample_face = enroll_face_id_to_flash_with_name(face_list, new_id, st_name.enroll_name);
  ESP_LOGD(TAG, "Face ID %s Enrollment: Sample %d", st_name.enroll_name, ENROLL_CONFIRM_TIMES - left_sample_face);
  return left_sample_face;
}
```

Figura 3.21: Función dedicada a guardar un nuevo rostro en la lista de memoria del ESP32-CAM

La función *enroll_face_id_to_flash_with_name* se encarga de guardar en la lista *face_list -> st_face_list (Figura 4.21)* ubicada en la memoria interna del ESP32-CAM, el nombre del nuevo rostro *(st_name)* con su respectivo id *new_id -> out_res (Figura.4.21)* para alinear el rostro al momento de querer reconocerlo.

3.2.2.2 Reconocimiento facial

Una vez guardado el nuevo rostro, el chip ESP32 debe ser capaz de detectarlo y reconocerlo en tiempo real. Las funciones que se utilizan para detectar y reconocer rostros en tiempo real están detalladas en las bibliotecas ESP-WHO [6] y ESP-FACE [7]. Para hacer uso de estas se deben incluir las librerías:

- **#include "fd_forward.h":** ESP-FACE detecta rostros utilizando el modelo MTMN.
- **#include "fr_forward.h" y #include "fr_flash.h":** ESP-FACE reconoce rostros utilizando el model FRMN.

La función principal para reconocer rostros es la que se muestra a continuación:

```
if (g_state == START_RECOGNITION  && (st_face_list.count > 0))
{
  face_id_node *f = recognize_face_with_name(&st_face_list, out_res.face_id);
  if (f)
  {
    char recognised_message[64];
    sprintf(recognised_message, "DOOR OPEN FOR %s", f->id_name);
    client.send(recognised_message);
  }
  else
  {
    // Send message via ESP-NOW
    envioData.AlertaSend = 1;
    esp_err_t result = esp_now_send(slaveAddress, (uint8_t *) &envioData, sizeof(envioData));

    if (result == ESP_OK) {
      Serial.println("Sent with success");
    } else {
      Serial.println("Error sending the data");
    }
    delay(3000);
    client.send("FACE NOT RECOGNISED");
  }
}
```

Figura 3.22: Función reconocer rostro

El objetivo de utilizar el entorno ESP-IDF es poder facilitar la tarea de detectar y reconocer rostros en tiempo real, en este caso, utilizar ESP-FACE ayuda a utilizar sólo una función para esta tarea, que es la función *recognize_face_with_name()*. El trabajo de esta función es alinear el rostro detectado en tiempo real (variable *out_res.face_id*) con los rostros guardados en memoria (variable *st_face_list*). Si el ESP32-CAM reconoce el rostro detectado, este le enviará inmediatamente una respuesta de confirmación al cliente.

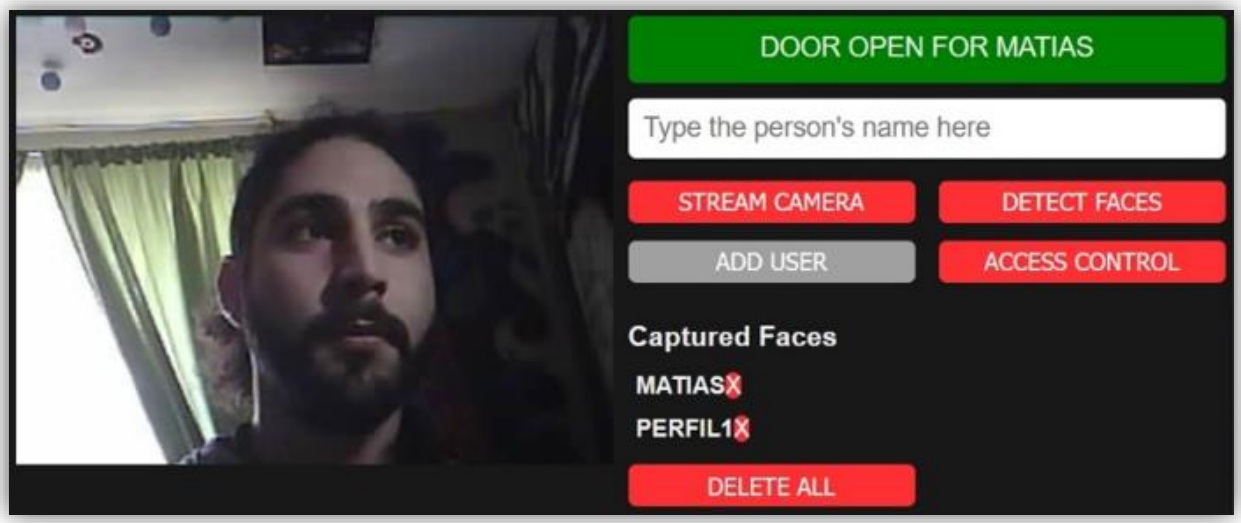


Figura 3.23: Respuesta de confirmación "DOOR OPEN FOR ____"

En cambio, si el ESP32-CAM no reconoce el rostro detectado procederá a enviar un correo de alerta al cliente.

3.2.2.3 Correo de alerta

Las funciones que hacen posible enviar el correo de alerta se obtienen a partir de la biblioteca **#include <esp_now.h>**. ESP-NOW [15] [16] es un protocolo de comunicación que utilizan los dispositivos con chip ESP32 para poder comunicarse entre sí en un corto alcance y a baja potencia, sin la necesidad de utilizar Wi-Fi como medio de conexión.

Para comprender el trabajo del protocolo ESP-NOW se explicará el código utilizado en el ESP32-CAM maestro, dispositivo encargado de enviar la señal de alerta al dispositivo esclavo, y el código utilizado en el ESP32-CAM esclavo, módulo encargado de enviar el correo de alerta al cliente.

Primeramente, se deben crear las variables globales que se utilizarán en el código del dispositivo maestro, en este caso:

- slaveAddress: guarda la dirección MAC del dispositivo esclavo.
- envioData: variable que define el estado on/off de la señal de alerta al dispositivo esclavo.

Luego, en el apartado *void setup ()* se debe inicializar el protocolo ESP-NOW. Para realizar este trabajo se recurre a la función *esp_now_init ()*. Lo siguiente es registrar la función callback encargada de enviar las alertas al ESP32-CAM esclavo. Para inicializar esta tarea se utiliza la función

esp_now_register_send_cb que, a la vez llama a la función *OnSent*. Esta última se utiliza para declarar una variable que guarde la dirección MAC del ESP32-CAM esclavo y para verificar el estado del envío de la alerta con *esp_now_send_status_t*.

Una vez verificada la inicialización de la alerta al ESP32-CAM esclavo, se debe declarar una variable que guarde la información del dispositivo esclavo. Esta tarea se realiza utilizando la variable de tipo *esp_now_peer_info_t*, puntualmente, se debe declarar la dirección MAC del dispositivo esclavo utilizando la función *memcpy ()*. Por último, para agregar el dispositivo esclavo se debe declarar la variable recién creada a la función *esp_now_add_peer ()*.

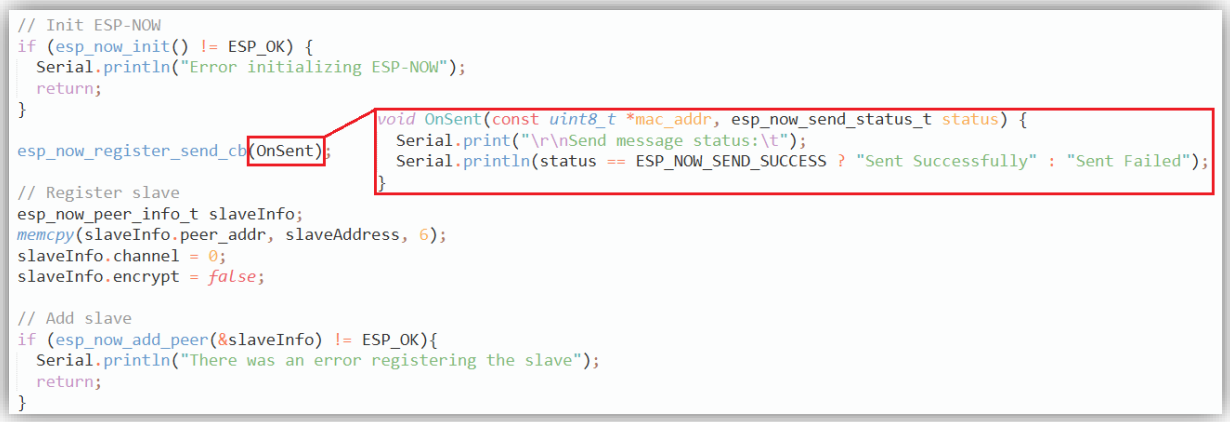


Figura 3.24:Cuerpo setup() para enviar señal de alerta al ESP32-CAM esclavo

En cambio, en el cuerpo *void loop ()* se trabaja con la variable global *envioData*. Como se puede apreciar en la Figura 3.23, cuando el ESP32-CAM maestro no reconozca el rostro detectado, la función *esp_now_send ()* enviará la señal al dispositivo esclavo con *envioData* igual a 1. Esto activará en el dispositivo esclavo el envío del correo de alerta al cliente.

Por otra parte, en el ESP32-CAM esclavo se define sólo una variable global. En este caso, la variable *recibirData* recibirá cada petición que envíe la variable *envioData* del ESP32-CAM maestro. Luego, se especifica el dominio que utiliza ESP-NOW, el puerto de red y el correo remitente junto a su contraseña. Por último, se deben declarar: el objeto (una instancia de sesión) que enviará el correo; la estructura del mensaje; y los datos de configuración de la sesión.

```
#define SMTP_HOST "smtp.gmail.com"
#define SMTP_PORT 465
#define AUTHOR_EMAIL "*****@gmail.com"
#define AUTHOR_PASSWORD "*****"
SMTPSession smtp;
SMTP_Message message;
ESP_Mail_Session session;
```

Figura 3.25: Información de la sesión que enviará el correo de alerta

Luego, en el *void setup ()* se detallan los datos del correo remitente, del correo receptor del mensaje de alerta y el cuerpo del mensaje. Además, se establece la conexión a la sesión, se inicializa nuevamente el ESP-NOW con la función *esp_now_init ()* y, por último, se registra la función callback con *esp_now_register_recv_cb*, encargada de enviar una respuesta cada vez que reciba la alerta del dispositivo maestro. Con la ayuda de la función *OnRecv*, el registro puede detectar la dirección MAC del ESP32-CAM maestro.

```
session.server.host_name = SMTP_HOST;
session.server.port = SMTP_PORT;
session.login.email = AUTHOR_EMAIL;
session.login.password = AUTHOR_PASSWORD;
session.login.user_domain = "";
```

Figura 3.26: Datos del correo remitente

```
message.sender.name = "ESP32-CAM-FACE";
message.sender.email = AUTHOR_EMAIL;
message.subject = "ROSTRO DESCONOCIDO FUERA DE TU HOGAR";
message.addRecipient("CASA618", "registros.mids@gmail.com");

String textMsg = "ROSTRO DESCONOCIDO FUERA DE TU HOGAR";
message.text.content = textMsg.c_str();
message.text.charSet = "utf-8";
message.text.transfer_encoding = Content_Transfer_Encoding::enc_7bit;
message.priority = esp_mail_smtp_priority::esp_mail_smtp_priority_high;
```

Figura 3.27: Cuerpo del correo de alerta

Figura 3.28: Estableciendo conexión del correo, iniciando ESP-NOW y registrando el callback

Por otra parte, en el *void loop ()* del dispositivo esclavo existe sólo un condicional if para enviar el correo de alerta. Cuando la variable *recibirData* se active, la función *MailClient.sendMail* se encarga de ingresar al correo remitente y enviar el cuerpo del mensaje de alerta. Finalmente, la variable *recibirData* se hace 0 para evitar un loop infinito de envío de correo.

```
void loop(){
  Serial.println(recibirData.AlertaSend);
  if (recibirData.AlertaSend = 1){
    if (!MailClient.sendMail(&smtp, &message))
    Serial.println("Error sending Email, " + smtp.errorReason());

    //to clear sending result log
    smtp.sendingResult.clear();

    ESP_MAIL_PRINTF("Free Heap: %d\n", MailClient.getFreeHeap());
    delay(3000);
  }
  recibirData.AlertaSend = 0;
}
```

Figura 3.29: Código loop del ESP32-CAM esclavo

3.2.2.4 Dominio No-IP

La implementación del dominio No-IP es específica y se utiliza sólo en la codificación del ESP32-CAM maestro, ya que es el dispositivo encargado de realizar un servidor web con un flujo de video en tiempo real. Primeramente, se debe incluir la biblioteca **#include <EasyDDNS.h>** para poder utilizar el dominio No-IP.

Luego, en el *void setup ()* se deben declarar: los servicios del dominio, en este caso se utiliza No-IP; la dirección creada en el dominio; el username y password para ingresar a la sesión No-IP (apartado 3.2.1). También, se implementa la función *onUpdate ()* para notificar a través del puerto Serial cuándo la IP pública es actualizada.

```
EasyDDNS.service("noip");
EasyDDNS.client("domotic618.ddns.net", "USERNAME:XXXXXXX@gmail.com", "PASSWORD:XXXX");
EasyDDNS.onUpdate([&](const char* oldIP, const char* newIP){
  Serial.print("EasyDDNS - IP Change Detected: ");
  Serial.println(newIP);
});
```

Figura 3.30: Inicialización servicio No-IP

Por último, para chequear el cambio de la IP pública del router en el cuerpo *void loop ()* se utiliza la función *EasyDDNS.update ()*. En este caso, se define el chequeo cada 5 minutos, es decir, la función *update* queda como *EasyDDNS.update (300000)*.

4 RESULTADOS

El sistema domótico de videovigilancia desarrollado en este documento tiene principalmente dos estados, cuando reconoce o cuando desconoce un rostro detectado. A continuación, se presentan resultados visuales de dichos estados, en el cual el ESP32-CAM maestro enviará o no la señal de alerta para que el ESP32-CAM esclavo inicie el envío del correo de alerta.

4.1 RECONOCIMIENTO FACIAL

En la siguiente imagen se puede apreciar cuando el ESP32-CAM maestro logra reconocer un rostro. Este, al momento de reconocer dicho rostro enviará una confirmación del estado al cliente web ("DOOR OPEN FOR _____"). Paralelamente, en el terminal del Arduino IDE es posible visualizar la confirmación del rostro reconocido con un Serial.println().

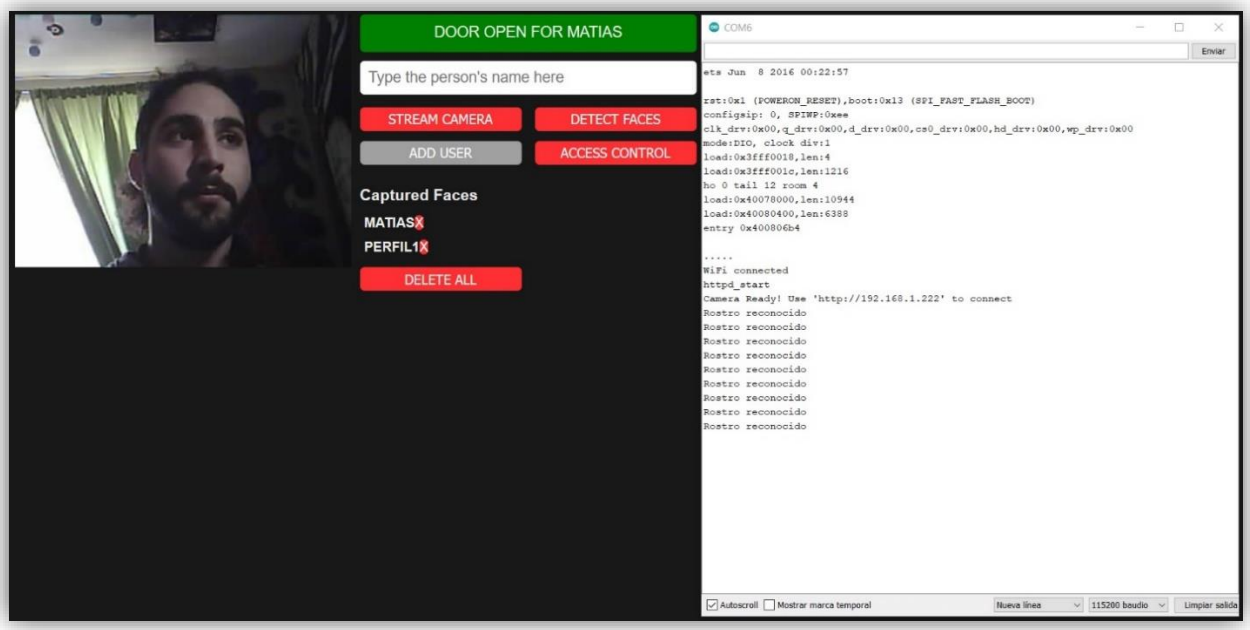


Figura 4.1: Rostro reconocido por ESP32-CAM maestro

Por otra parte, existe la posibilidad de que haya dos rostros frente al ESP32-CAM maestro. En este caso la cantidad de aciertos disminuye. Esto puede suceder debido a la presencia del rostro desconocido que acompaña al que se encuentra guardado en memoria, pero igualmente el módulo ESP32-CAM logra reconocer el rostro guardado, como se puede mostrar a continuación:

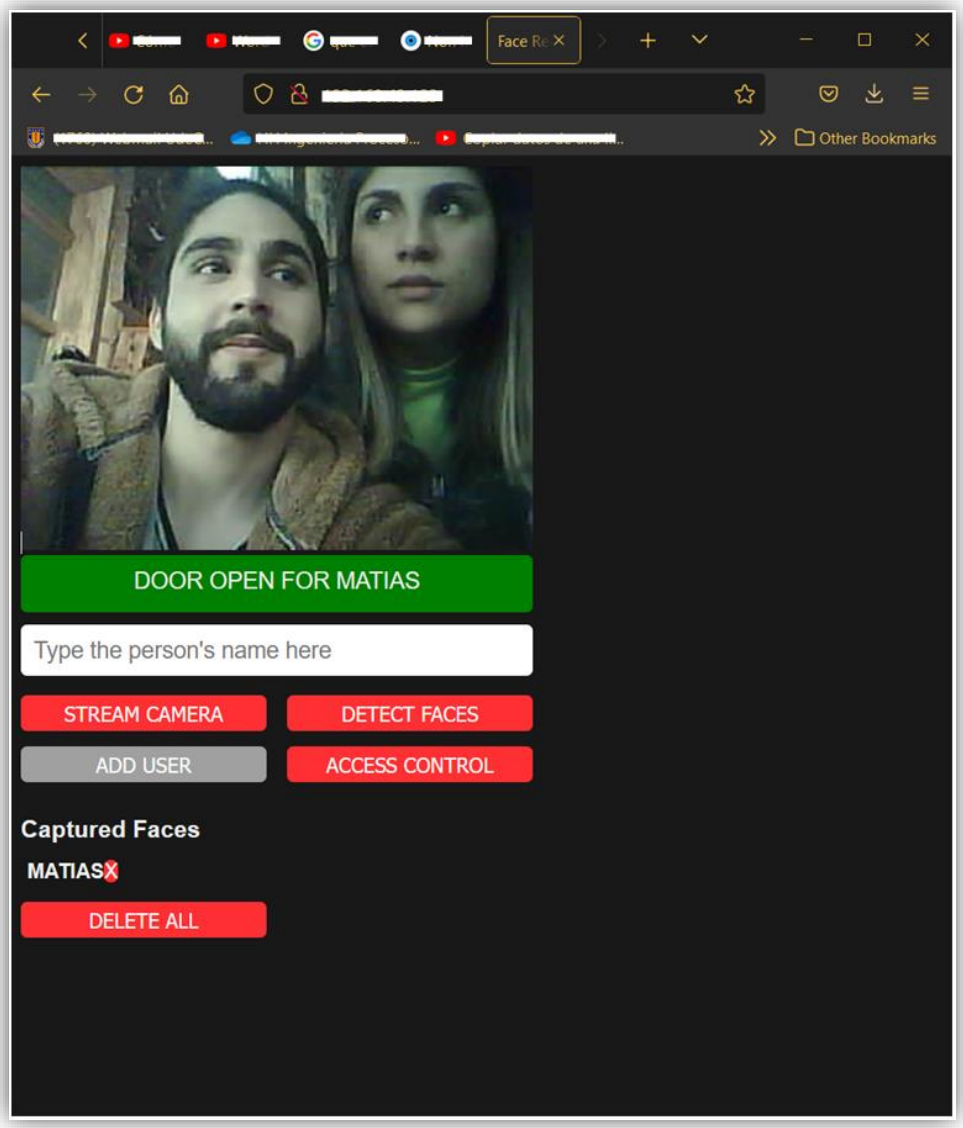


Figura 4.2: Acierto con dos rostros frente al ESP32-CAM

Gracias a la prueba realizada se infiere que el ESP32-CAM trabaja el reconocimiento facial de un rostro a la vez, lo cual, si él o la dueña del hogar viene acompañada hay que evitar que la cámara detecte dos o más rostros a la vez. Asegurando esto se evitará una falsa alarma, si es que existiesen más de un destinatario incluido en el correo de alerta.

4.2 INGRESO AL CORREO REMITENTE

A continuación, desde el terminar serial del entorno de desarrollo Arduino IDE se logra observar el ingreso del ESP32-CAM esclavo al correo remitente a través del protocolo SMTP [18].

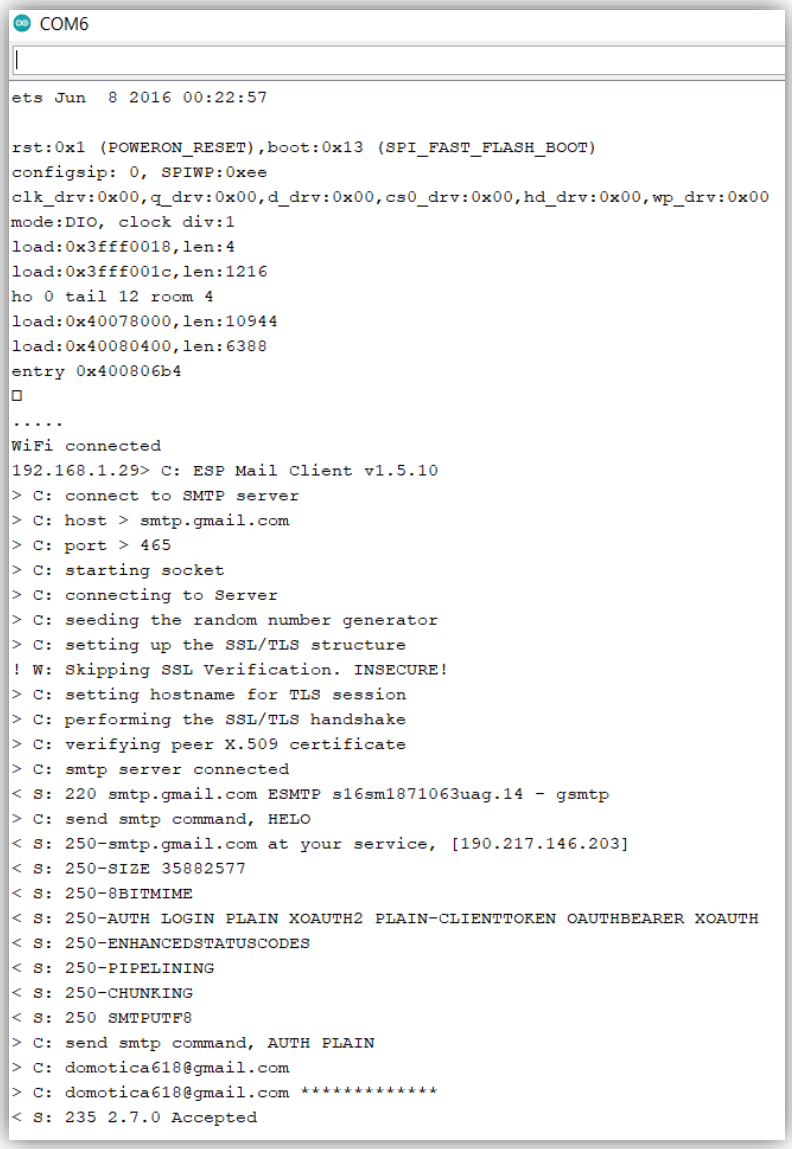


Figura 4.3: Ingreso del ESP32-CAM esclavo al correo remitente - Protocolo SMTP

4.3 CORREO DE ALERTA

Como se mencionó al inicio de este capítulo, el segundo estado del sistema implementado es cuando no es posible reconocer un rostro. En esta ocasión es importante observar desde el terminal serial del entorno Arduino IDE cómo trabajan los módulos ESP32-CAM maestro y esclavo cuando se comunican a través del protocolo ESP-NOW.

A continuación, se puede observar el momento exacto cuando el ESP32-CAM maestro envía dos señales de alerta al ESP32-CAM esclavo, con el fin de enviar el correo de alerta al destinatario interesado.

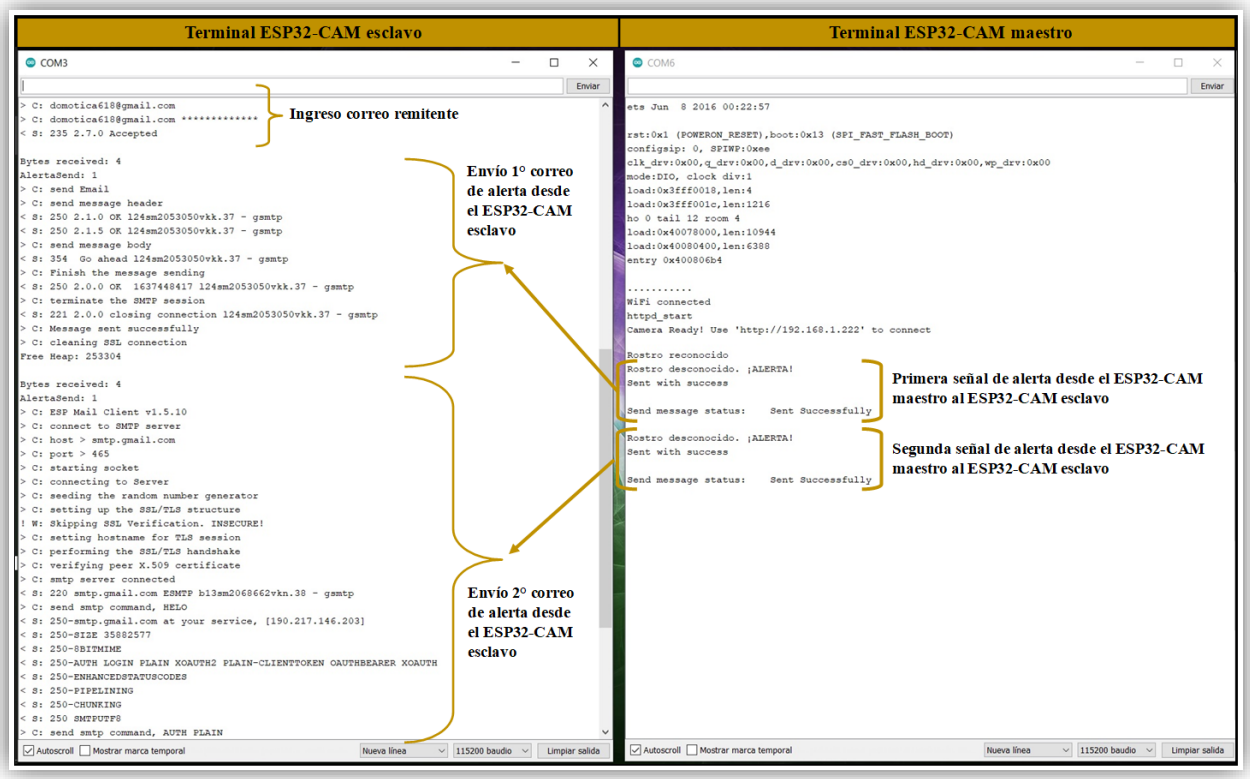


Figura 4.4: Envío correo de alerta

En la imagen previamente mostrada se observó desde el terminal serial del ESP32-CAM maestro y esclavo el lado remitente del correo de alerta. Finalmente, desde el correo electrónico del destinatario (detallado en la Figura 3.27) se puede observar la recepción del correo de alerta.

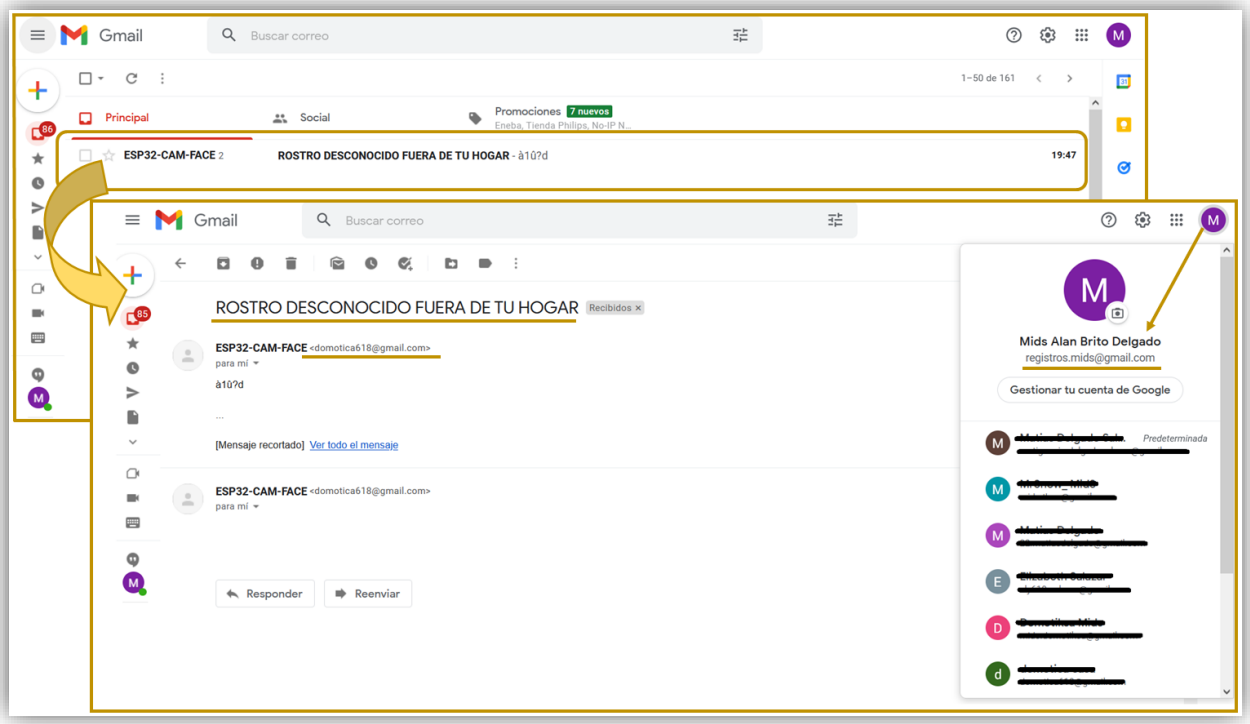


Figura 4.5: Recepción correo de alerta

En la Figura previamente mostrada se observa que el remitente efectivamente es el que se detalla en la Figura 3.26, el título del correo está bien desplegado y además, las dos pruebas realizadas tuvieron éxito. Con esto, el trabajo de enviar el correo de alerta resultó exitoso.

4.4 INGRESO REMOTO AL SERVIDOR DOMÓTICO

En la Figura 4.1 se ingresó al flujo de video del ESP32-CAM maestro de manera local, es decir, el ingreso al servidor domótico fue cuando el computador estaba conectado a la misma señal Wi-Fi del servidor domótico. En este caso, desde el buscador web de un navegador de Internet se escribe la dirección IP del Orange Pi Lite luego, en la sección videovigilancia se redirecciona al servidor del dispositivo maestro.

En cambio, en las siguientes imágenes se observa el ingreso al servidor domótico de manera remota, es decir, desde cualquier lugar externo al hogar. Para ingresar al servidor del OPLite, en el buscador web del navegador del celular se debe ingresar el nombre del dominio DDNS creado, agregando el puerto 443, detallado en la codificación del ESP32-CAM maestro. Una vez dentro, en la sección videovigilancia se redireccionará al dominio No-IP con puerto 80.

Figura 4.6: Ingreso remoto a servidor web creado por OPLite

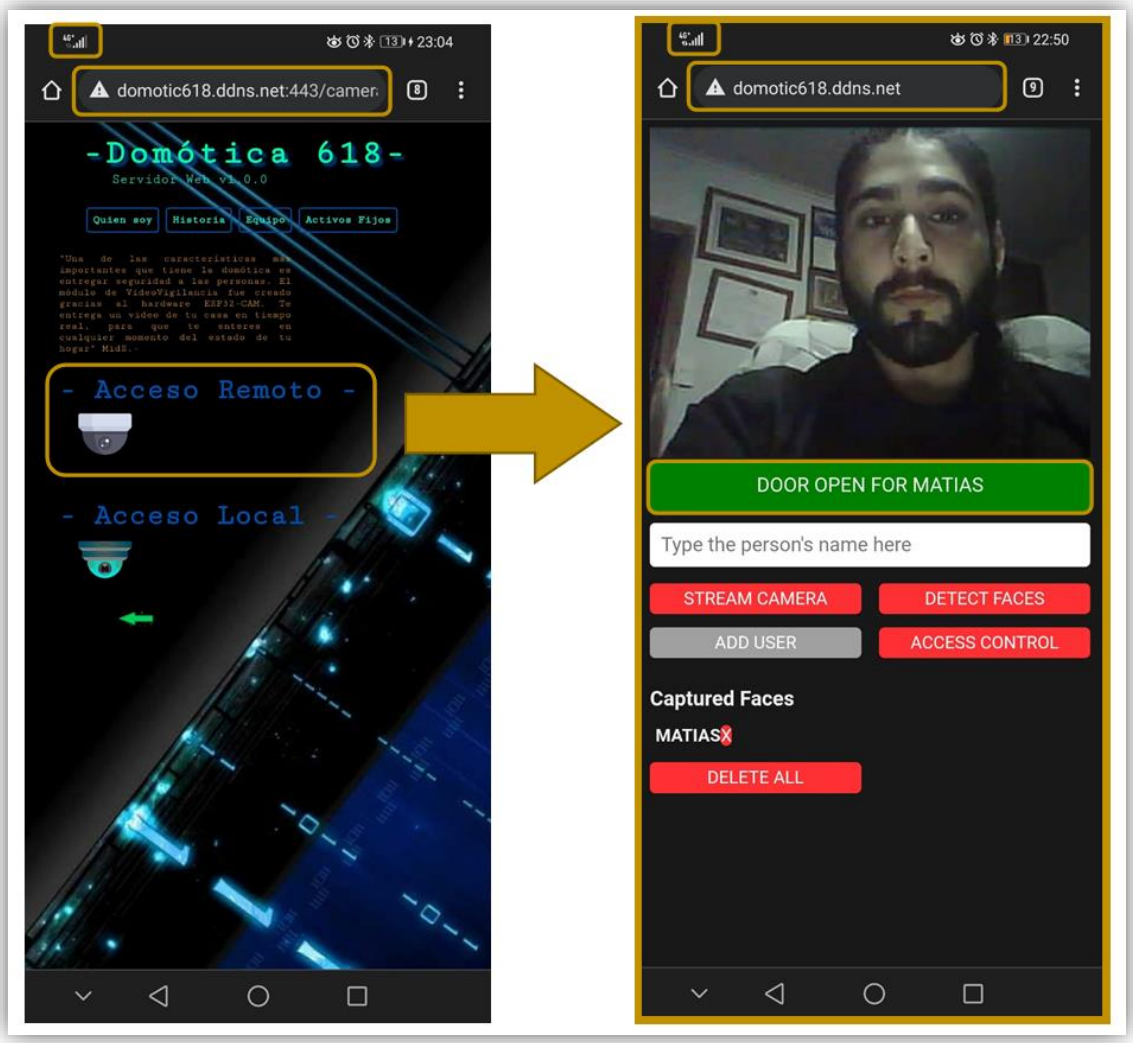


Figura 4.7: Acceso remoto al flujo de video del ESP32-CAM maestro

4.5 TASA DE ACIERTO AL RECONOCER ROSTRO

La tasa de acierto y fallos al momento de reconocer un rostro es un dato de suma importancia para poder validar la confiabilidad del ESP32-CAM y analizar bajo qué condiciones puede llegar a trabajar el sistema de reconocimiento facial asertivo. Este porcentaje de acierto es alterado directamente por al ambiente en el que se esté trabajando el reconocimiento de rostro.

Se realizaron varias pruebas de reconocimiento facial para reconocer el/los principal/es factor/es que altera/n la tasa de aciertos y fallos, y el principal factor que disminuye el porcentaje de acierto es cuando no hay presencia de luz sobre el rostro. Este factor es reiterativo y suele ser el que más afecta al momento de reconocer un rostro. Gracias a la aplicación Android "Light Meter" se consiguió medir la intensidad de luz mínima que se necesita para reconocer bien un rostro y este valor corresponde a 900 lux (lx).

A continuación, se muestra el comportamiento de aciertos y fallas cuando el ESP32-CAM comienza el trabajo de reconocer rostros en un ambiente favorable:

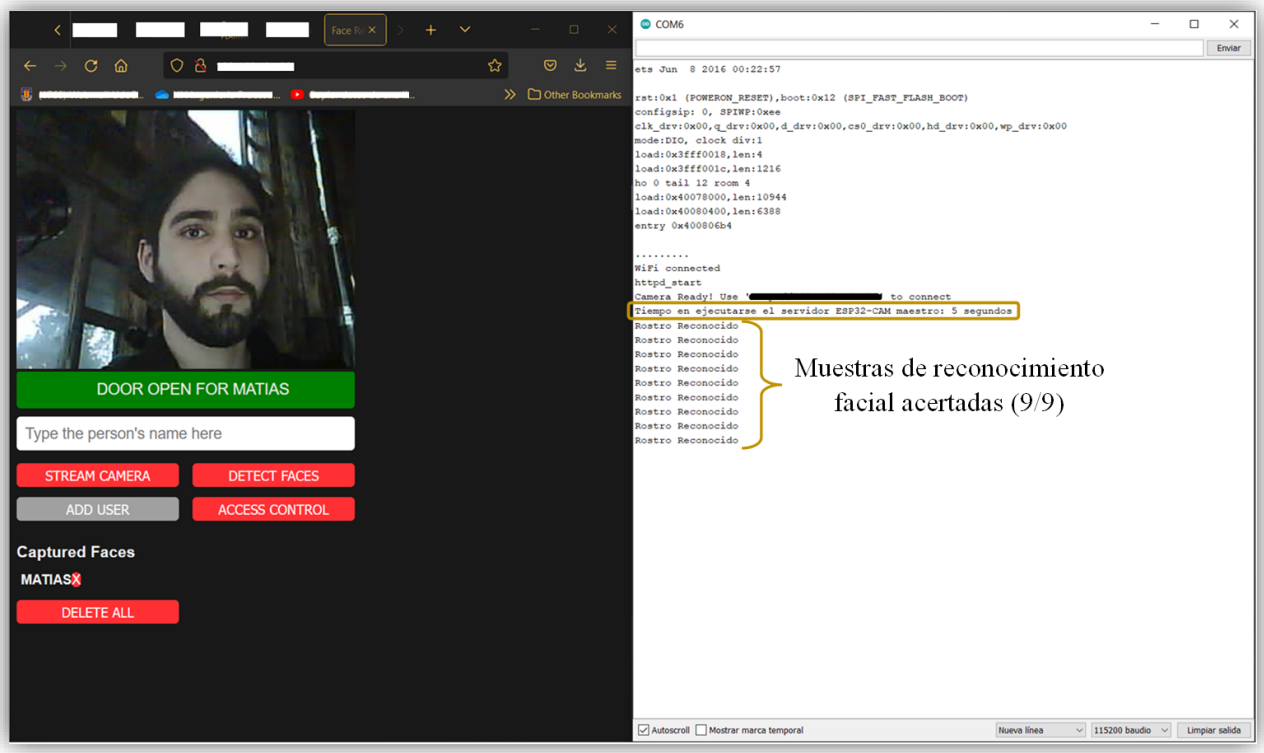


Figura 4.8: Aciertos al reconocer rostro

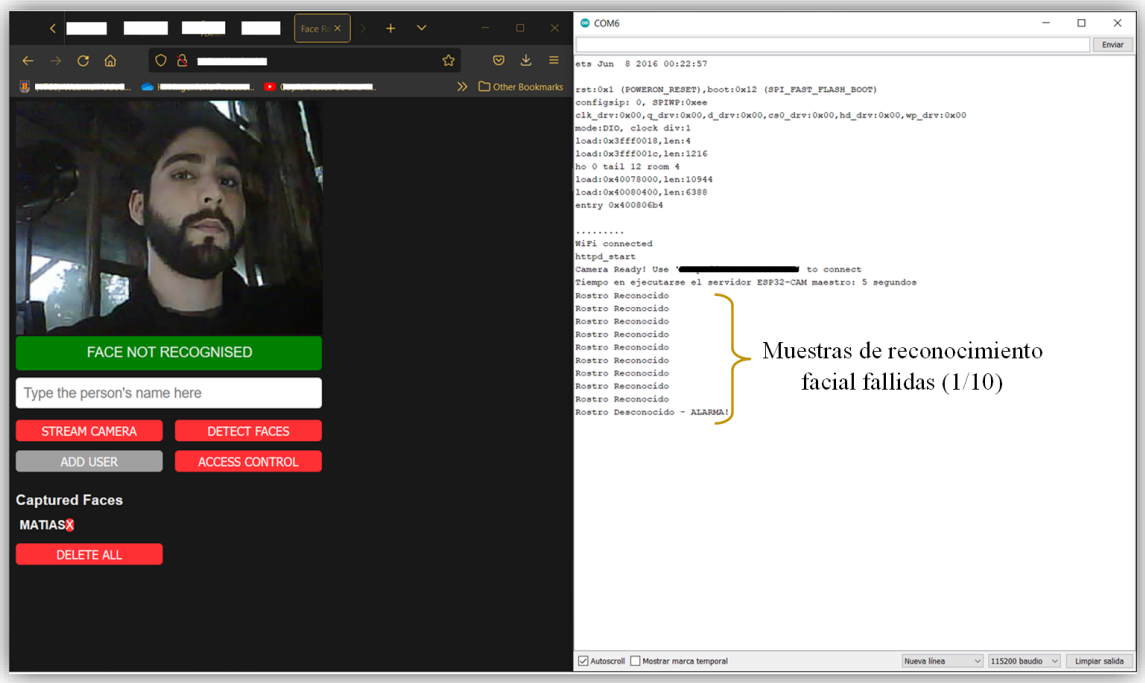


Figura 4.9: Fallos al reconocer rostro

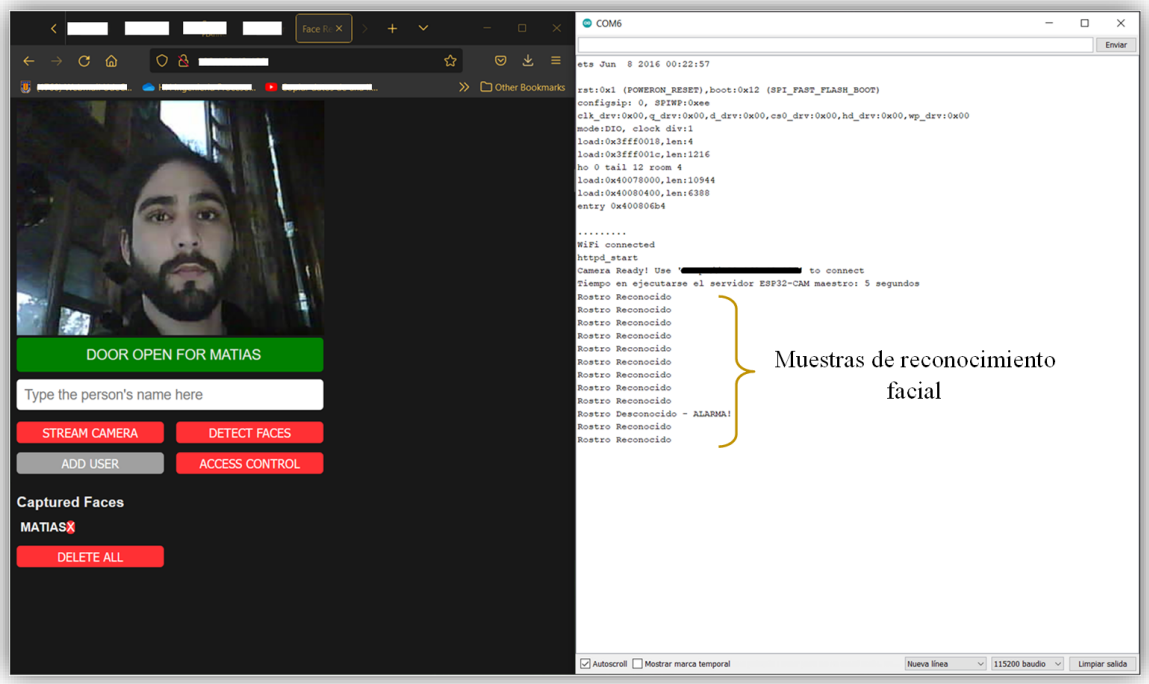


Figura 4.10: Comportamiento acierto/falla al reconocer rostro

En las figuras recién mostradas se puede observar que el porcentaje de acierto en un ambiente favorable alcanza un acierto del 90%. Esto podría mejorar con la cámara ubicada fuera de la casa, así el rostro podrá ser iluminado directamente con la luz del día (aproximadamente de 20k a 100k lx).

5 CONCLUSIONES

En la actualidad está siendo más frecuente implementar en los hogares un sistema domótico para solucionar algunas debilidades que la casa podría tener, como la seguridad de ingreso, la temperatura ambiental, constantes filtraciones de gases tóxicos, entre otros. Este trabajo se centró en la seguridad del hogar, desarrollando e implementando un sistema domótico de videovigilancia con reconocimiento facial a tiempo real, con el fin de brindar una mayor seguridad al dueño o dueña del hogar.

El sistema domótico de videovigilancia implementado tiene varias ventajas con relación a la estabilidad y fiabilidad en su conexión remota. Primeramente, se obtuvo un buen desempeño por parte del controlador Orange Pi Lite al momento de trabajar de forma continua para levantar el servidor web. Los alcances de sus componentes e incorporación del ventilador para controlar y mantener la temperatura de la placa fueron fundamental para lograr un sistema rápido y sin sobrecalentamiento al momento de trabajar continuamente.

Una segunda ventaja es la compatibilidad entre hardware y software utilizado. Los componentes del controlador OPLite se dedican a trabajar de forma estable con los programas instalados. Por una parte se encuentra Armbian, un sistema operativo dedicado a trabajar con un Orange Pi Lite, y por otro lado se tiene el trabajo de Apache en conjunto al dominio No-Ip para levantar un servidor web estable a la Internet, asegurando un acceso remoto fiable.

Por otra parte, utilizar el ESP32-CAM nos asegura aproximadamente un 90% de acierto al momento de reconocer un rostro en un ambiente favorable. Existen dos parámetros que afectan directamente este porcentaje de acierto. El primer parámetro corresponde a la cantidad de rostros que se detectan a la vez. Para evitar falsas alarmas es recomendable que el módulo ESP32-CAM vaya detectando de a un rostro a la vez. El segundo parámetro corresponde a la cantidad de luz en el ambiente. Se midió la mínima luminosidad que se necesita en un ambiente favorable para que el ESP32-CAM reconozca con facilidad el rostro y esta corresponde a 900 lx. Entonces, para asegurar este porcentaje de acierto se recomienda ubicar la cámara cerca del timbre del hogar y trabajar directamente con la luz del día, entre 8 y 18 horas, para tener un mínimo de 20.000 lx en el ambiente.

Un posible efecto y alcance que podría generar a futuro utilizar las bibliotecas ESP-WHO, ESP-FACE, ESP_Mail_Client, EasyDDNS, ESP-NOW y WebSocket son las constantes actualizaciones

que realizan sus creadores en GitHub. Estas podrían modificar su forma de operar dentro del flujo de datos, afectando la estructura del código. La forma de trabajar en el caso del ESP32-CAM obliga al programador intervenir a nivel de hardware y software.

Por último, como trabajo a futuro se podría migrar el sistema a otros modelos de dispositivos electrónicos para poder comparar la calidad de ambiente de trabajo. Como ejemplo, se tiene un modelo de cámara dedicado al reconocimiento facial y de voz, este corresponde al ESP-EYE, igualmente desarrollada por Espressif. Este modelo de cámara tiene un valor más elevado que el ESP32-CAM, es dimensionalmente más pequeño y además, duplica los MB de la PSRAM, esto afecta directamente a la velocidad de trabajo al momento de reconocer los rostros. También, este modelo podría aumentar la tasa de aciertos al momento de detectar dos rostros a la vez y/o cuando exista una luminosidad ambiental menor a 900 lx. Otro trabajo a futuro correspondería incorporar sensores de gases, control de luces y/o cambiar el controlador por uno más robusto si comienza a afectar el aumento de dispositivos a controlar.

6 GLOSARIO

Hardware: Conjunto de elementos físicos o materiales que constituyen una computadora o un sistema informático.

Controlador: Circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales que cumplen una tarea específica.

Software: Programa que permite realizar distintas tareas en un sistema informático.

Entorno de desarrollo integrado (IDE): Se utiliza para desarrollar software (programas digitales) relacionados con la codificación.

Biblioteca Arduino IDE: Son códigos elaborados por la comunidad para facilitar la programación de tareas específicas.

Internet de las Cosas: Conexión de dispositivos electrónicos (hardware) con la Internet.

7 REFERENCIAS

[1] Xunlong Software CO. (2016). *Orange Pi Lite*. Obtenido de Orange Pi: http://www.orangepi.org/orangepilite/

[2] Baidu. (2021). *Baidu*. Obtenido de Cortex-A7: https://baike.baidu.com/item/Cortex-A7

[3] Shenzhen B&T Technology Co., Ltd. (2020). *Ai-Thinker*. Obtenido de ESP32-CAM: http://www.ai-thinker.com/pro_view-24.html

[4] Espressif Systems (Shanghai) Co., Ltd. (2021). *Espressif*. Obtenido de ESP-IDF: https://www.espressif.com/en/products/sdks/esp-idf

[5] Ríos, J. (18 de Febrero de 2017). *Descubriendo la Orange Pi*. Obtenido de Establecer IP Estática en Armbian: https://descubriendolaorangepi.wordpress.com/2017/02/18/establecer-ip-estatica-en-armbian/

[6] Jiong, Y. (2018). *GitHub*. Obtenido de ESP-WHO: https://github.com/espressif/esp-who

[7] Jiong, Y. (2021). *GitHub*. Obtenido de ESP-FACE: https://github.com/espressif/esp-dl

[8] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4510-4520.

[9] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters*, 1499-1503.

[10] Pérez, J. (Mayo de 2020). *Orange Pi en español*. Obtenido de Servidor Web con Orange Pi: https://orangepiweb.es/servidor-web-orangepi.php

[11] No-IP. (2020). *Knowledge Base*. Obtenido de How to Install the No-IP DUC on Raspberry Pi: https://www.noip.com/support/knowledgebase/install-ip-duc-onto-raspberry-pi/

[12] Ingeniería MCI Ltda. (2021). *Arduino*. Obtenido de Arduino IDE 1.8.16: https://www.arduino.cc/en/software

[13] Espressif Systems (Shanghai) Co., Ltd. (2021). *Espressif*. Obtenido de ESP-IDF: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/

[14] Maimon, G. (2018). *GitHub*. Obtenido de Arduino Websockets: https://github.com/gilmaimon/ArduinoWebsockets

[15] Espressif. (2021). *GitHub*. Obtenido de ESP-NOW: https://github.com/espressif/esp-now

[16] Espressif Systems (Shanghai) Co., Ltd. (2021). *Espressif*. Obtenido de ESP-NOW: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html

[17] Suwatchai, K. (2020). *GitHub*. Obtenido de ESP-Mail-Client: https://github.com/mobizt/ESP-Mail-Client

[18] Ramadass, S., Al Bazar, H., Abouabdalla, O., & Manasrah, A. (2009). Active E-mail system protocols monitoring algorithm. *TENCON 2009 - 2009 IEEE Region 10 Conference* (págs. 1-6). Singapore : IEEE.

[19] Sharma, A. (2020). *GitHub*. Obtenido de EasyDDNS: https://github.com/ayushsharma82/EasyDDNS

[20] Prosegur. (2021). *Movistar Prosegur Alarmas*. Obtenido de Evolución de la domótica: cómo nace y cómo es ahora: https://blog.prosegur.es/evolucion-de-la-domotica-historia/#la-domotica-en-la-actualidad

[21] TECNOSeguro. (09 de Agosto de 2019). *TECNOSeguro Magazín Digital*. Obtenido de El mercado mundial de equipos de videovigilancia llega a su tercer año de crecimiento sostenido : https://www.tecnoseguro.com/analisis/cctv/mercado-mundial-equipos-videovigilancia-crecimiento-ihs

[22] Sodimac S.A. (2021). Sodimac . Obtenido de Servicio Instalación de Cámaras de Seguridad: https://www.sodimac.cl/sodimac-cl/content/a2620003/Instalacion-de-Camaras-de-Seguridad

[23] Fundación Paz Ciudadana. (14 de Octubre de 2021). *Paz Ciudadana*. Obtenido de Índice Paz Ciudadana 2021: https://pazciudadana.cl/proyectos/documentos/indice-paz-ciudadana-2021/

[24] Centro de Estudio y Análisis del Delito. (2021). CEAD. Obtenido de Estadísticas Delictuales: http://cead.spd.gov.cl/estadisticas-delictuales/

[25] Espressif Systems (Shanghai) Co., Ltd. (2021). Espressif. Obtenido de ESP32-S2 Series: https://www.espressif.com/en/products/socs/esp32-s2 Chip ESP32-S2

[26] RobotZeroOne. (11 de Septiembre de 2019). RobotZero. Obtenido de Access Control with Face Recognition: https://robotzero.one/access-control-with-face-recognition/

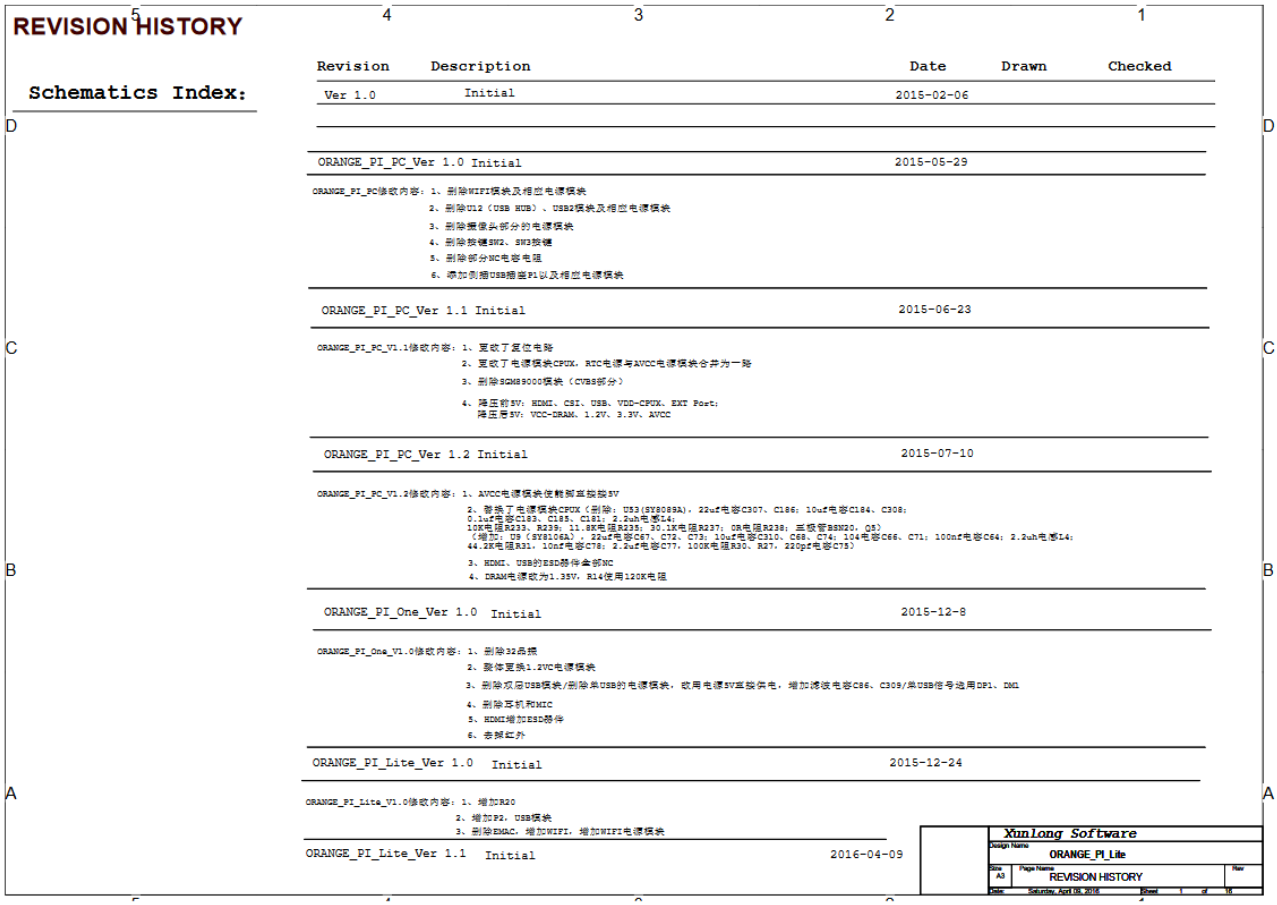
[27] RobotZeroOne. (13 de Abril de 2019). RobotZero. Obtenido de ESP32-CAM Face Recognition for Home Automation: https://robotzero.one/esp32-face-door-entry/

[28] RobotZeroOne. (26 de Mayo de 2019). RobotZero. Obtenido de ESP-WHO Face Recognition with WebSocket Communication: https://robotzero.one/esp-who-recognition-with-names/

[29] Dotcom-Monitor. (16 de Octubre de 2020). LoadView. Obtenido de Aplicaciones WebSocket de pruebas de carga: https://www.loadview-testing.com/es/blog/pruebas-de-carga-de-aplicaciones-basadas-en-websocket/

8 ANEXOS

8.1 DATASHEET ORANGE PI LITE



REVISION HISTORY

Schematics Index:

Revision	Description	Date	Drawn	Checked
Ver 1.0	Initial	2015-02-06		
ORANGE_PI_PC_Ver 1.0	Initial	2015-05-29		

ORANGE_PI_PC修改内容：1、删除WIFI模块及相应电源模块
2、删除U12（USB HUB）、USB2模块及相应电源模块
3、删除摄像头部分的电源模块
4、删除按键SW2、SW3按键
5、删除部分NC电容电阻
6、添加侧插USB插座P1以及相应电源模块

ORANGE_PI_PC_Ver 1.1	Initial	2015-06-23

ORANGE_PI_PC_V1.1修改内容：1、更改了复位电路
2、更改了电源模块CPUX，RTC电源与AVCC电源模块合并为一路
3、删除SGM8900模块（CVBS部分）
4、降压前5V：HDMI、CSI、USB、VDD-CPUX、EXT Port;
降压后5V：VCC-DRAM、1.2V、3.3V、AVCC

ORANGE_PI_PC_Ver 1.2	Initial	2015-07-10

ORANGE_PI_PC_V1.2修改内容：1、AVCC电源模块使能脚直接拉5V
2、替换了电源模块CPUX（删除：U53(SY8089A)，22uf电容C307、C186；10uf电容C184、C308；
0.1uf电容C183、C185、C181；2.2uh电感L4；
10K电阻R233、R239；11.8K电阻R235；30.1K电阻R237；OR电阻R238；三极管BSM20、Q5）
（增加：U9（SY8106A），22uf电容C67、C72、C73，10uf电容C310、C68、C74；104电容C66、C71；100nf电容C64；2.2uh电感L4；
44.2K电阻R31，10nf电容C78；2.2uf电容C77，100K电阻R30、R27，220pf电容C75）
3、HDMI、USB的ESD器件全部NC
4、DRAM电源改为1.35V，R14使用120K电阻

ORANGE_PI_One_Ver 1.0	Initial	2015-12-8

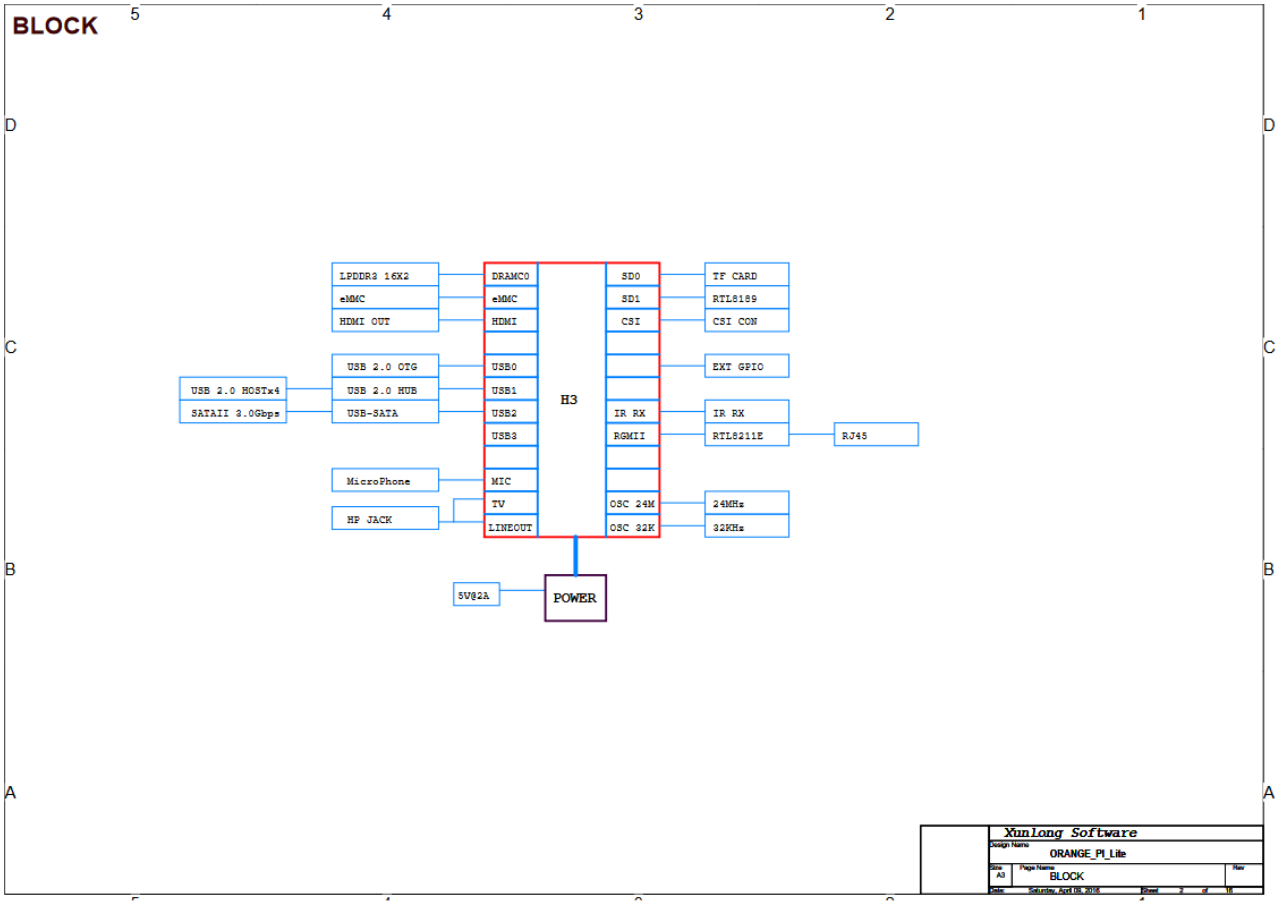
ORANGE_PI_One_V1.0修改内容：1、删除32晶振
2、整体更换1.2VC电源模块
3、删除双层USB模块/删除单USB的电源模块，改用电源5V直接供电，增加滤波电容C86、C309/单USB信号选用DP1、DM1
4、删除耳机和MIC
5、HDMI增加ESD器件
6、去掉红外

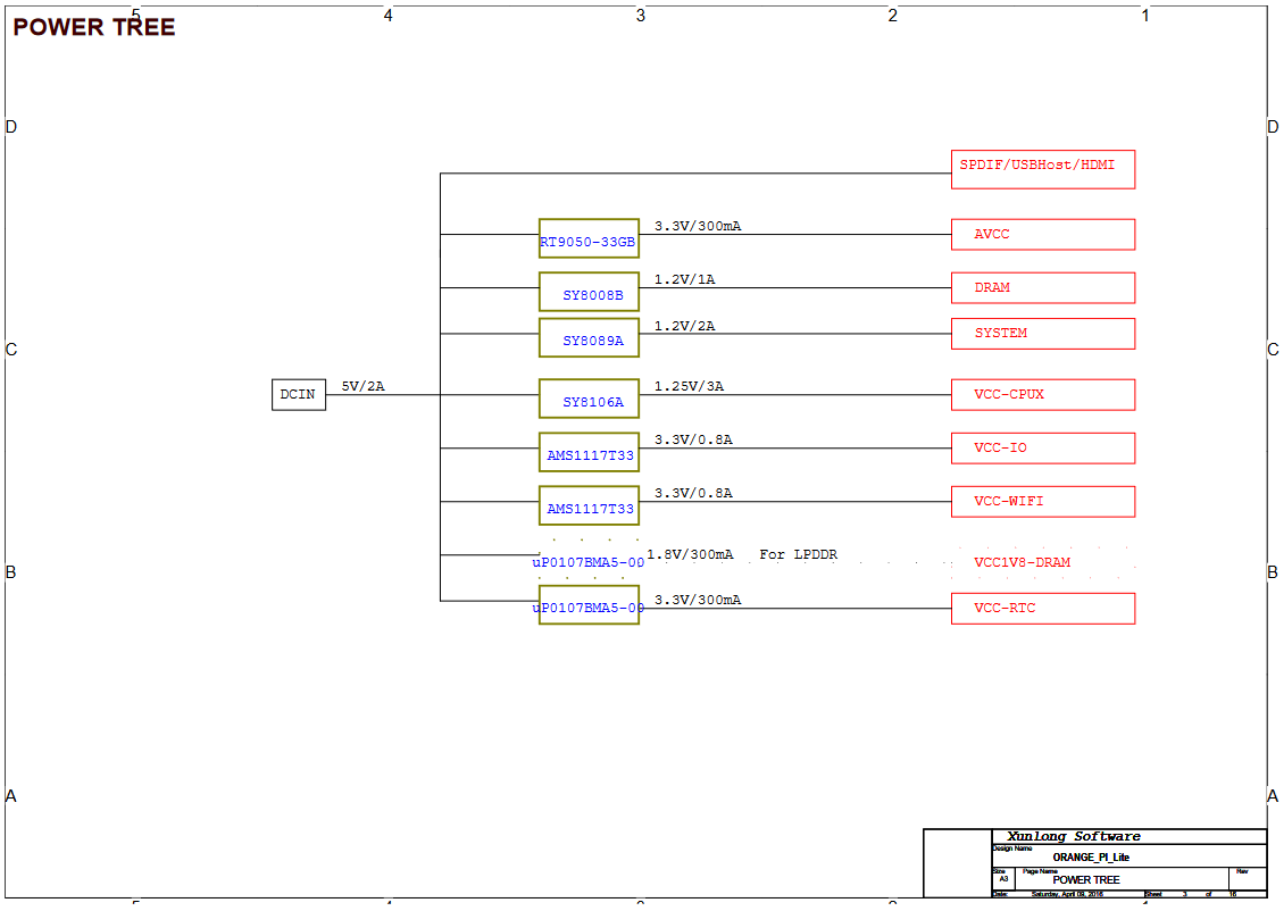
ORANGE_PI_Lite_Ver 1.0	Initial	2015-12-24

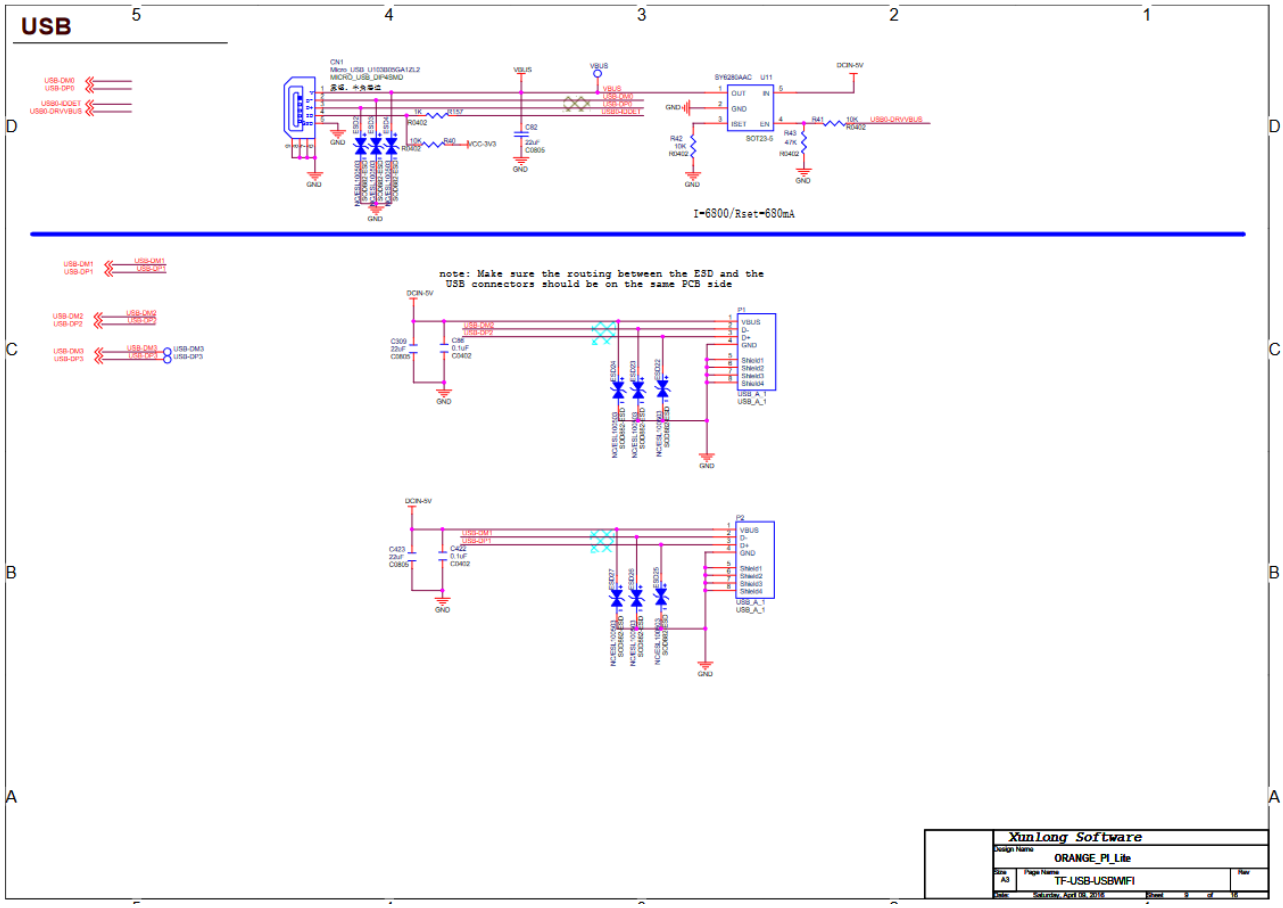
ORANGE_PI_Lite_V1.0修改内容：1、增加CR20
2、增加P2、USB模块
3、删除EMAC，增加WIFI，增加WIFI电源模块

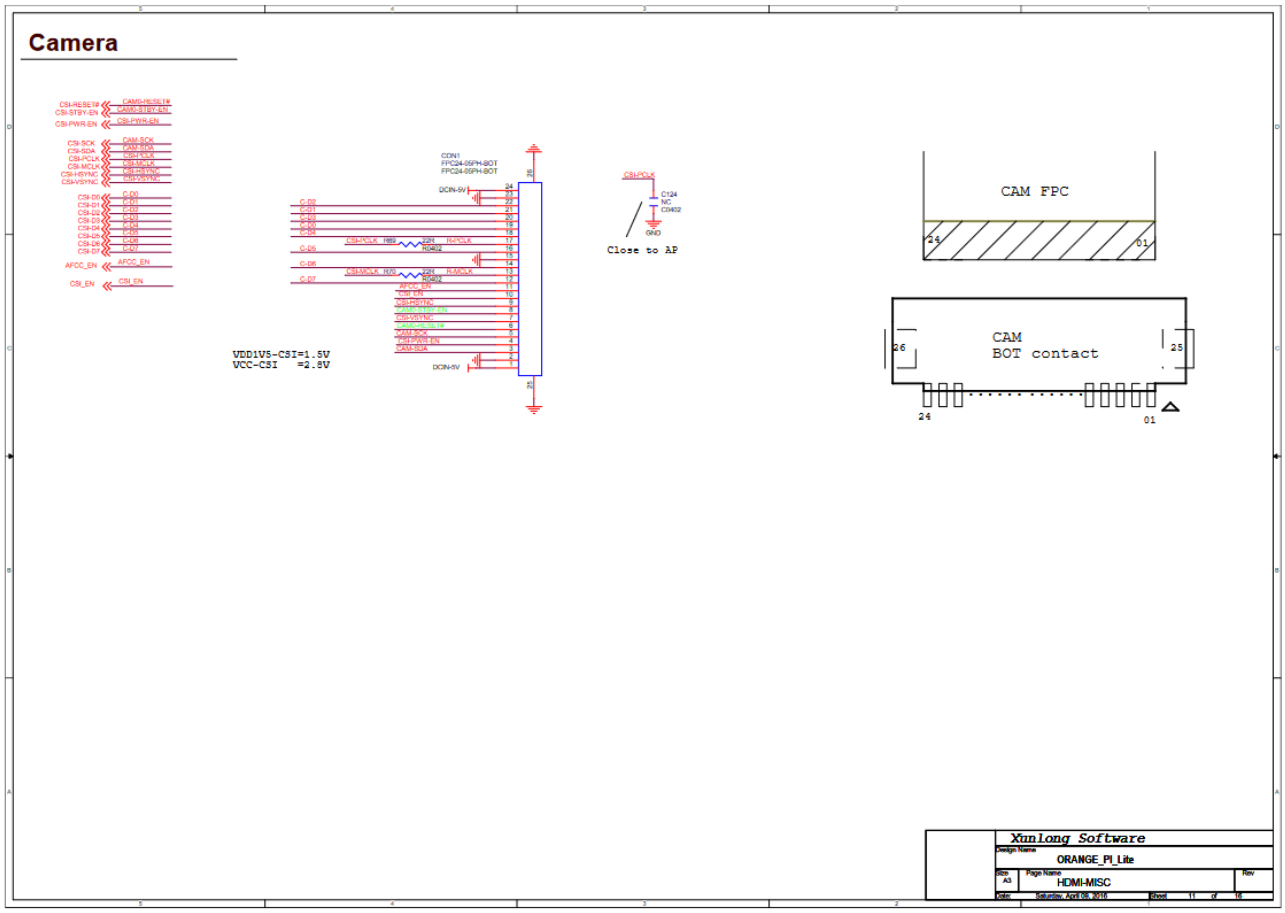
ORANGE_PI_Lite_Ver 1.1	Initial	2016-04-09

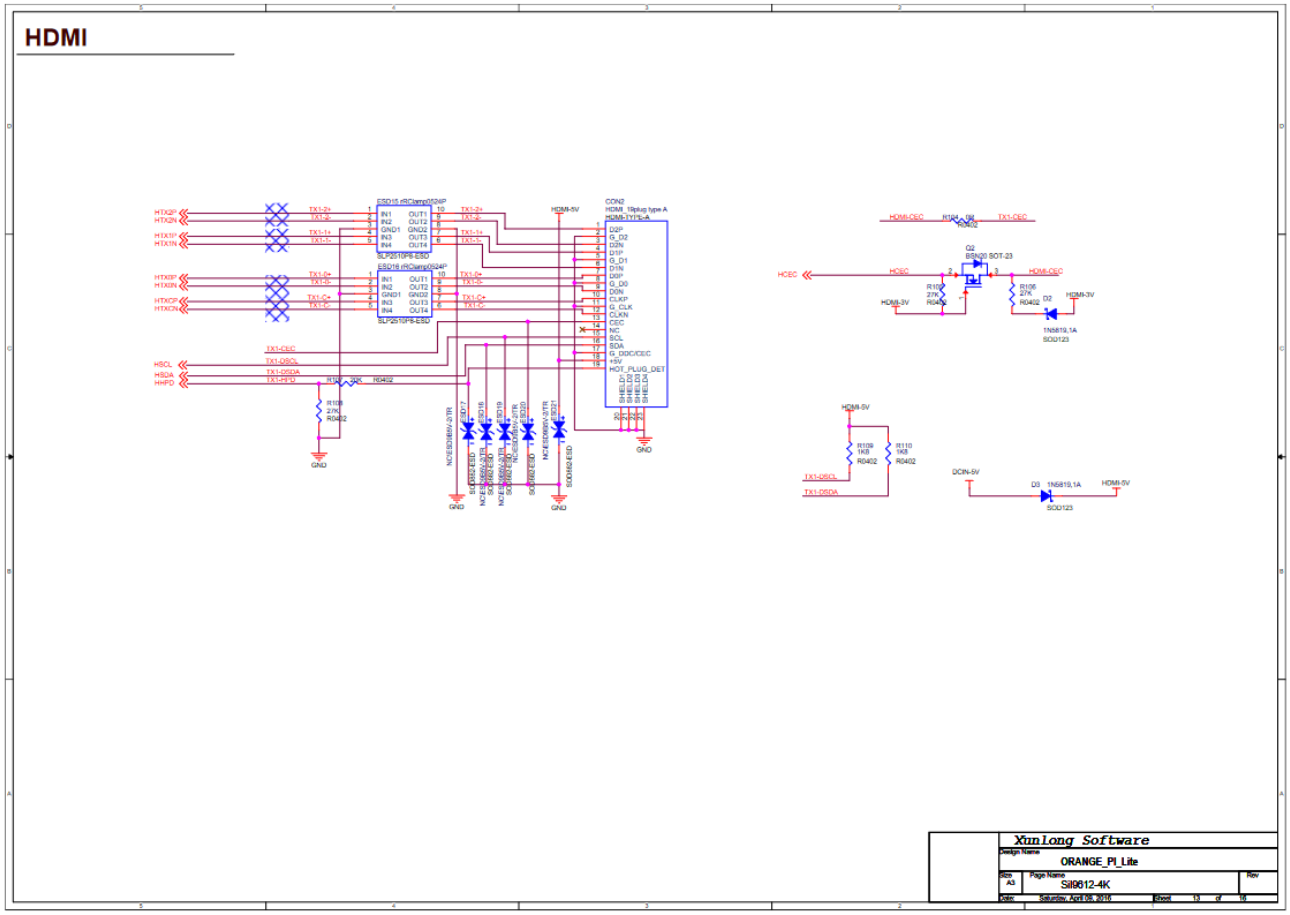
Xunlong Software
Design Name: ORANGE_PI_Lite
Size: A3 | Page Name: REVISION HISTORY | Rev
Date: Saturday, April 09, 2016 | Sheet 1 of 16

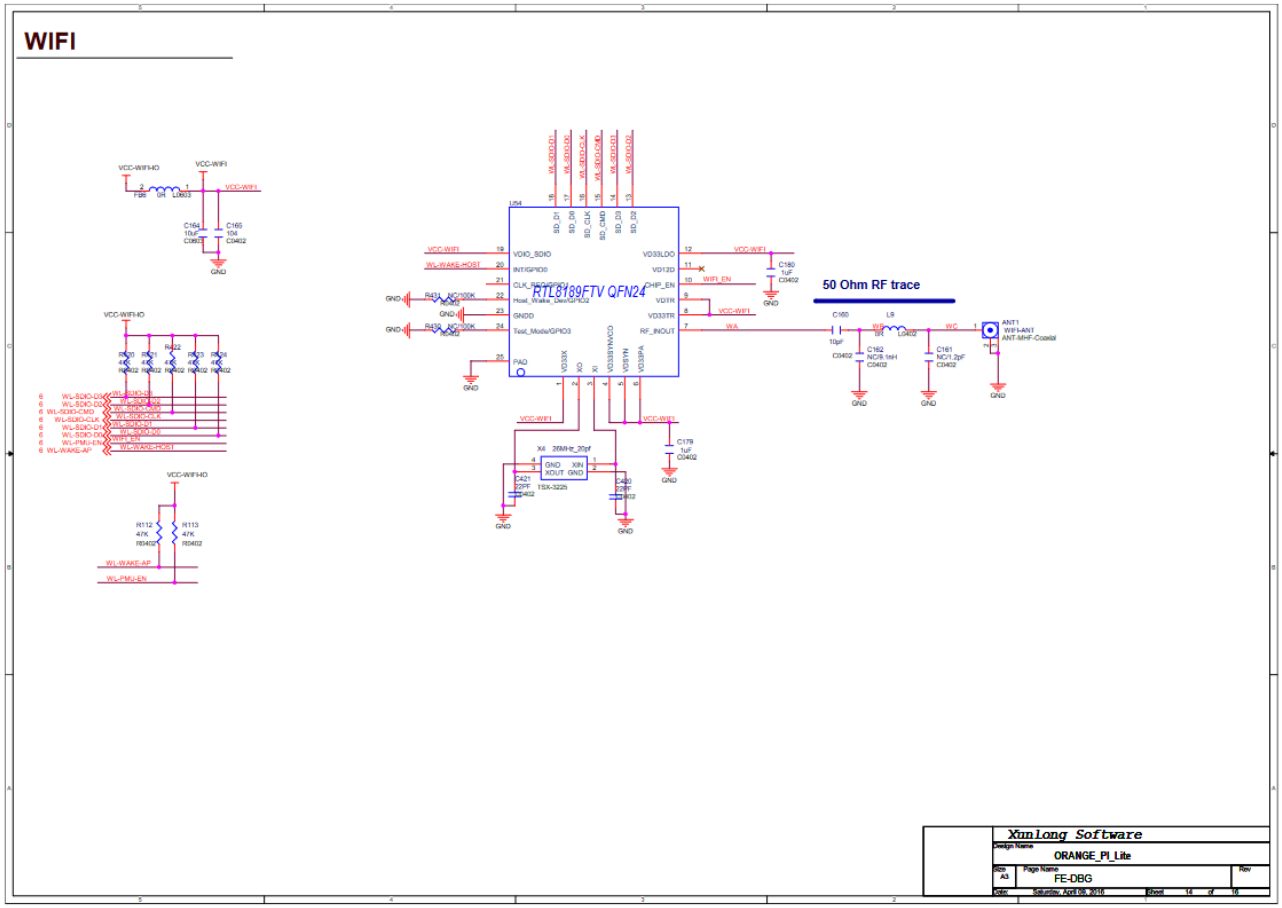

BLOCK
LPDDR3 16X2
eMMC
HDMI OUT
USB 2.0 OTG
USB 2.0 HOSTx4
USB 2.0 HUB
SATAII 3.0Gbps
USB-SATA
MicroPhone
HP JACK
DRAMC0
eMMC
HDMI
USB0
USB1
USB2
USB3
MIC
TV
LINEOUT
H3
SD0
SD1
CSI
IR RX
RGMII
OSC 24M
OSC 32K
TF CARD
RTL8189
CSI CON
EXT GPIO
IR RX
RTL8211E
RJ45
24MHz
32KHz
5V@2A
POWER
Xunlong Software
ORANGE_PI_Lite
BLOCK


POWER TREE
SPDIF/USBHost/HDMI
RT9050-33GB
3.3V/300mA
AVCC
SY8008B
1.2V/1A
DRAM
SY8089A
1.2V/2A
SYSTEM
DCIN
5V/2A
SY8106A
1.25V/3A
VCC-CPUX
AMS1117T33
3.3V/0.8A
VCC-IO
AMS1117T33
3.3V/0.8A
VCC-WIFI
uP0107BMA5-00
1.8V/300mA
For LPDDR
VCC1V8-DRAM
uP0107BMA5-00
3.3V/300mA
VCC-RTC
Xunlong Software
ORANGE_PI_Lite
POWER TREE


USB
I-6800/Rset-680mA
note: Make sure the routing between the ESD and the
USB connectors should be on the same PCB side
Xunlong Software
ORANGE_PI_Lite
TF-USB-USBWIFI

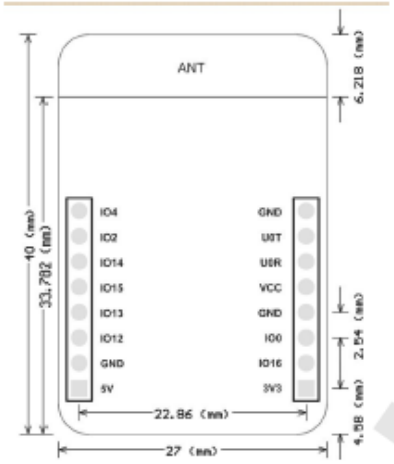

Camera
CDN1
FPC24-05FH-BOT
FPC24-05FH-BOT
DCIN-5V
C124
NC
CD402
GND
Close to AP
CAM FPC
24
01
CAM
BOT contact
26
25
24
01
VDD1V5-CSI=1.5V
VCC-CSI =2.8V
Xunlong Software
ORANGE_PI_Lite
HDMI-MISC
Saturday, April 08, 2016

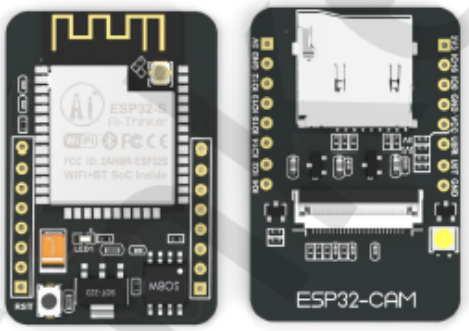

HDMI
Xunlong Software
ORANGE_PI_Lite
SII9612-4K
Saturday, April 05, 2014


WIFI
RTL8189FTV QFN24
50 Ohm RF trace
Xunlong Software
ORANGE_PI_Lite
FE-DBG

8.2 DATASHEET ESP32-CAM

ESP32-CAM Module





Features

- The smallest 802.11b/g/n Wi-Fi BT SoC Module
- Low power 32-bit CPU,can also serve the application processor
- Up to 160MHz clock speed， Summary computing power up to 600 DMIPS
- Built-in 520 KB SRAM, external 4MPSRAM
- Supports UART/SPI/I2C/PWM/ADC/DAC
- Support OV2640 and OV7670 cameras,Built-in Flash lamp.
- Support image WiFI upload
- Support TF card
- Supports multiple sleep modes.
- Embedded Lwip and FreeRTOS
- Supports STA/AP/STA+AP operation mode
- Support Smart Config/AirKiss technology
- Support for serial port local and remote firmware upgrades (FOTA)

Overview

The ESP32-CAM has a very competitive small-size camera module that can operate independently as a minimum system with a footprint of only 27*40.5*4.5mm and a deep sleep current of up to 6mA.

ESP-32CAM can be widely used in various IoT applications. It is suitable for home smart devices, industrial wireless control, wireless monitoring, QR wireless identification, wireless positioning system signals and other IoT applications. It is an ideal solution for IoT applications.

ESP-32CAM adopts DIP package and can be directly inserted into the backplane to realize rapid production of products, providing customers with high-reliability connection mode, which is convenient for application in various IoT hardware terminals.

Product Specifications

Module Model	ESP32-CAM
Package	DIP-16
Size	27*40.5*4.5（±0.2）mm
SPI Flash	Default 32Mbit
RAM	520KB SRAM +4M PSRAM
Bluetooth	Bluetooth 4.2 BR/EDR and BLE standards
Wi-Fi	802.11 b/g/n/
Support interface	UART、SPI、I2C、PWM
Support TF card	Maximum support 4G
IO port	9
UART Baudrate	Default 115200 bps
Image Output Format	JPEG(OV2640 support only),BMP,GRAYSCALE
Spectrum Range	2412 ~2484MHz
Antenna	Onboard PCB antenna, gain 2dBi
Transmit Power	802.11b: 17±2 dBm (@11Mbps) 802.11g: 14±2 dBm (@54Mbps) 802.11n: 13±2 dBm (@MCS7)
Receiving Sensitivity	CCK, 1 Mbps : -90dBm CCK, 11 Mbps: -85dBm 6 Mbps (1/2 BPSK): -88dBm 54 Mbps (3/4 64-QAM): -70dBm MCS7 (65 Mbps, 72.2 Mbps): -67dBm
Power Dissipation	Turn off the flash lamp:180mA@5V Turn on the flash lamp and turn on the brightness to the maximum:310mA@5V Deep-sleep: Minimum power consumption can be achieved 6mA@5V Moderm-sleep: Minimum up to 20mA@5V Light-sleep: Minimum up to 6.7mA@5V
Security	WPA/WPA2/WPA2-Enterprise/WPS
Power Supply Range	5V
Operating Temperature	-20 ℃ ~ 85 ℃
Storage Environment	-40 ℃ ~ 90 ℃ , < 90%RH

Weight	10g

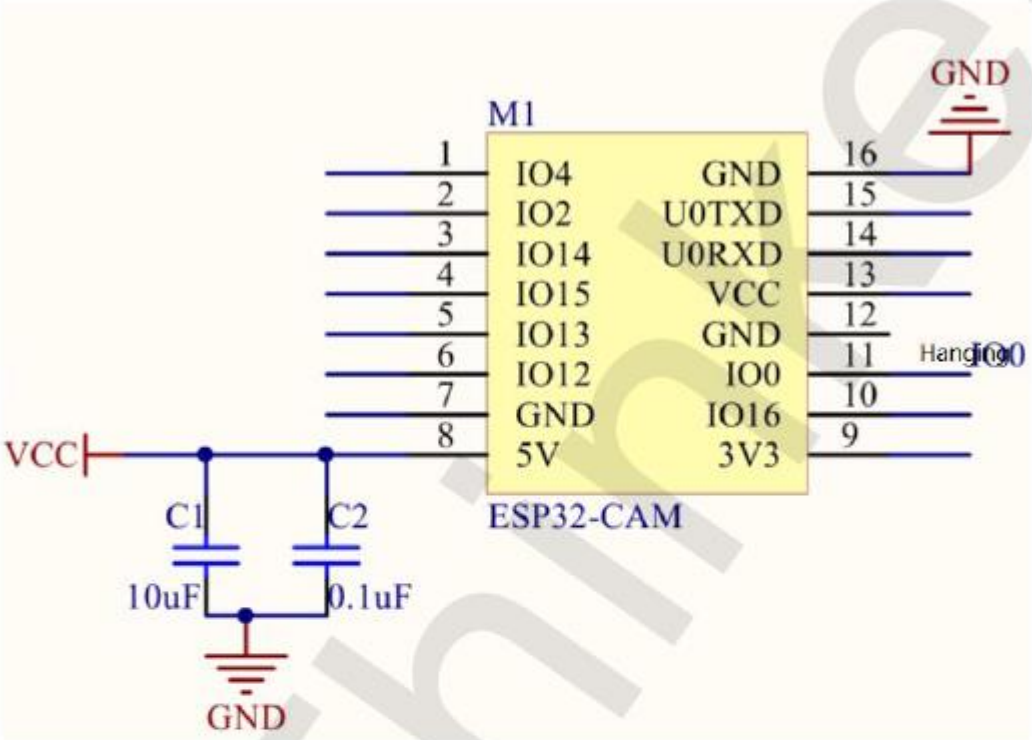
ESP32-CAM module picture output format rate

Format Size	QQVGA	QVGA	VGA	SVGA
JPEG	6	7	7	8
BMP	9	9	-	-
GRAYSCALE	9	8	-	-

Internal Pin Connect

CAM	ESP32	SD	ESP32
D0	PIN5	CLK	PIN14
D1	PIN18	CMD	PIN15
D2	PIN19	DATA0	PIN2
D3	PIN21	DATA1/Flash lamp	PIN4
D4	PIN36	DATA2	PIN12
D5	PIN39	DATA3	PIN13
D6	PIN34		
D7	PIN35		
XCLK	PIN0		
PCLK	PIN22		
VSYNC	PIN25		
HREF	PIN23		
SDA	PIN26		
SCL	PIN27		
POWER PIN	PIN32		

Minimum system diagram



Contact US

Shenzhen Ai-Thinker Technology Co., Ltd

Address: 7/F, Fengze Building B, Huafeng Industrial Park 2th, Hangkong street,Xixiang Raod, Baoan, Shenzhen China

Website:www.ai-thinker.com **Tel：0755-29162996** **E-mail:support@aithinker.com**

8.3 CÓDIGO ESP32-CAM MAESTRO

```
#include <esp_now.h>
#include <WiFi.h>
#include "HTTPClient.h"
#include <EasyDDNS.h>
#include <ArduinoWebsockets.h>
#include "esp_http_server.h"
#include "esp_timer.h"
#include "esp_camera.h"
#include "camera_index.h"
#include "Arduino.h"
#include "fd_forward.h"
#include "fr_forward.h"
#include "fr_flash.h"

const char* ssid = "XXXXXXX";
const char* password = "XXXXXXXX";

uint8_t slaveAddress[] = {0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX};

typedef struct struct_message {
    int AlertaSend;
} struct_message;

struct_message envioData;

WiFiServer server(80);

#define ENROLL_CONFIRM_TIMES 5
#define FACE_ID_SAVE_NUMBER 7

// Select camera model
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE
#define CAMERA_MODEL_AI_THINKER
#include "camera_pins.h"

using namespace websockets;

WebsocketsServer socket_server;

camera_fb_t * fb = NULL;

bool face_recognised = false;
```

```
void app_facenet_main();
void app_httpserver_init();

typedef struct
{
  uint8_t *image;
  box_array_t *net_boxes;
  dl_matrix3d_t *face_id;
} http_img_process_result;


static inline mtmn_config_t app_mtmn_config()
{
  mtmn_config_t mtmn_config = {0};
  mtmn_config.type = FAST;
  mtmn_config.min_face = 80;
  mtmn_config.pyramid = 0.707;
  mtmn_config.pyramid_times = 4;
  mtmn_config.p_threshold.score = 0.6;
  mtmn_config.p_threshold.nms = 0.7;
  mtmn_config.p_threshold.candidate_number = 20;
  mtmn_config.r_threshold.score = 0.7;
  mtmn_config.r_threshold.nms = 0.7;
  mtmn_config.r_threshold.candidate_number = 10;
  mtmn_config.o_threshold.score = 0.7;
  mtmn_config.o_threshold.nms = 0.7;
  mtmn_config.o_threshold.candidate_number = 1;
  return mtmn_config;
}

mtmn_config_t mtmn_config = app_mtmn_config();

face_id_name_list st_face_list;

static dl_matrix3du_t *aligned_face = NULL;

httpd_handle_t camera_httpd = NULL;

typedef enum
{
  START_STREAM,
  START_DETECT,
  SHOW_FACES,
  START_RECOGNITION,
  START_ENROLL,
  ENROLL_COMPLETE,
```

```
  DELETE_ALL,
} en_fsm_state;
en_fsm_state g_state;

typedef struct
{
  char enroll_name[ENROLL_NAME_LEN];
} httpd_resp_value;

httpd_resp_value st_name;

IPAddress local_IP(192, 168, XXX, XXX);
IPAddress gateway(192, 168, XXX, XXX);
IPAddress subnet(255, 255, 255, 0);

void OnSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nSend message status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Sent Successfully" : "Sent Failed");
}

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);

  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
```

```
  if (psramFound()) {
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t * s = esp_camera_sensor_get();
  s->set_framesize(s, FRAMESIZE_QVGA);

  if (!WiFi.config(local_IP, gateway, subnet)) {
    Serial.println("STA Failed to configure");
  }

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");

  app_httpserver_init();
  app_facenet_main();
  socket_server.listen(82);

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");

  server.begin();

  EasyDDNS.service("noip");
  EasyDDNS.client("domotic618.ddns.net", "USERNAME:XXXXXXX@gmail.com",
"PASSWORD:XXXX");
  EasyDDNS.onUpdate([&](const char* oldIP, const char* newIP){
    Serial.print("EasyDDNS - IP Change Detected: ");
```

```
    Serial.println(newIP);
  });

  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  esp_now_register_send_cb(OnSent);

  esp_now_peer_info_t slaveInfo;
  memcpy(slaveInfo.peer_addr, slaveAddress, 6);
  slaveInfo.channel = 0;
  slaveInfo.encrypt = false;

  if (esp_now_add_peer(&slaveInfo) != ESP_OK){
    Serial.println("There was an error registering the slave");
    return;
  }

}

static esp_err_t index_handler(httpd_req_t *req) {
  httpd_resp_set_type(req, "text/html");
  httpd_resp_set_hdr(req, "Content-Encoding", "gzip");
  return httpd_resp_send(req, (const char *)index_ov2640_html_gz, index_ov2640_html_gz_len);
}

httpd_uri_t index_uri = {
  .uri       = "/",
  .method    = HTTP_GET,
  .handler   = index_handler,
  .user_ctx  = NULL
};

void app_httpserver_init ()
{
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  if (httpd_start(&camera_httpd, &config) == ESP_OK)
    Serial.println("httpd_start");
  {
    httpd_register_uri_handler(camera_httpd, &index_uri);
  }
}

void app_facenet_main()
{
```

```
  face_id_name_init(&st_face_list, FACE_ID_SAVE_NUMBER, ENROLL_CONFIRM_TIMES);
  aligned_face = dl_matrix3du_alloc(1, FACE_WIDTH, FACE_HEIGHT, 3);
  read_face_id_from_flash_with_name(&st_face_list);
}

static inline int do_enrollment(face_id_name_list *face_list, dl_matrix3d_t *new_id)
{
  ESP_LOGD(TAG, "START ENROLLING");
  int left_sample_face = enroll_face_id_to_flash_with_name(face_list, new_id,
st_name.enroll_name);
  ESP_LOGD(TAG, "Face ID %s Enrollment: Sample %d", st_name.enroll_name,
ENROLL_CONFIRM_TIMES - left_sample_face);
  return left_sample_face;
}

static esp_err_t send_face_list(WebsocketsClient &client)
{
  client.send("delete_faces");
  face_id_node *head = st_face_list.head;
  char add_face[64];
  for (int i = 0; i < st_face_list.count; i++)
  {
    sprintf(add_face, "listface:%s", head->id_name);
    client.send(add_face);
    head = head->next;
  }
}

static esp_err_t delete_all_faces(WebsocketsClient &client)
{
  delete_face_all_in_flash_with_name(&st_face_list);
  client.send("delete_faces");
}

void handle_message(WebsocketsClient &client, WebsocketsMessage msg)
{
  if (msg.data() == "stream") {
    g_state = START_STREAM;
    client.send("STREAMING");
  }
  if (msg.data() == "detect") {
    g_state = START_DETECT;
    client.send("DETECTING");
  }
  if (msg.data().substring(0, 8) == "capture:") {
    g_state = START_ENROLL;
    char person[FACE_ID_SAVE_NUMBER * ENROLL_NAME_LEN] = {0,};
```

```
    msg.data().substring(8).toCharArray(person, sizeof(person));
    memcpy(st_name.enroll_name, person, strlen(person) + 1);
    client.send("CAPTURING");
   }
   if (msg.data() == "recognise") {
    g_state = START_RECOGNITION;
    client.send("RECOGNISING");
   }
   if (msg.data().substring(0, 7) == "remove:") {
    char person[ENROLL_NAME_LEN * FACE_ID_SAVE_NUMBER];
    msg.data().substring(7).toCharArray(person, sizeof(person));
    delete_face_id_in_flash_with_name(&st_face_list, person);
    send_face_list(client);
   }
   if (msg.data() == "delete_all") {
    delete_all_faces(client);
   }
}


void open_door(WebsocketsClient &client) {
    Serial.println("Door Unlocked");
    client.send("door_open");
}

void loop()
{
  EasyDDNS.update(300000);

  auto client = socket_server.accept();

  g_state = START_RECOGNITION;
  client.send("RECOGNISING");

  dl_matrix3du_t *image_matrix = dl_matrix3du_alloc(1, 320, 240, 3);
  http_img_process_result out_res = {0};
  out_res.image = image_matrix->item;
  send_face_list(client);

  while (client.available()) {
    client.poll();
    fb = esp_camera_fb_get();

    if (g_state == START_DETECT || g_state == START_ENROLL || g_state ==
START_RECOGNITION)
    {
      out_res.net_boxes = NULL;
```

```
    out_res.face_id = NULL;

    fmt2rgb888(fb->buf, fb->len, fb->format, out_res.image);
    out_res.net_boxes = face_detect(image_matrix, &mtmn_config);

    if (out_res.net_boxes){
      if (align_face(out_res.net_boxes, image_matrix, aligned_face) == ESP_OK)
      {

        out_res.face_id = get_face_id(aligned_face);

        if (g_state == START_ENROLL)
        {
          int left_sample_face = do_enrollment(&st_face_list, out_res.face_id);
          char enrolling_message[64];
          sprintf(enrolling_message, "SAMPLE NUMBER %d FOR %s",
ENROLL_CONFIRM_TIMES - left_sample_face, st_name.enroll_name);
          client.send(enrolling_message);
          if (left_sample_face == 0)
          {
            ESP_LOGI(TAG, "Enrolled Face ID: %s", st_face_list.tail->id_name);
            g_state = START_STREAM;
            char captured_message[64];
            sprintf(captured_message, "FACE CAPTURED FOR %s", st_face_list.tail->id_name);
            client.send(captured_message);
            send_face_list(client);

          }
        }

        if (g_state == START_RECOGNITION  && (st_face_list.count > 0))
        {
          face_id_node *f = recognize_face_with_name(&st_face_list, out_res.face_id);
          if (f)
          {
            char recognised_message[64];
            sprintf(recognised_message, "DOOR OPEN FOR %s", f->id_name);
            client.send(recognised_message);
            Serial.println("Rostro reconocido");
          }
          else
          {
            Serial.println("Rostro desconocido. ¡ALERTA!");
            envioData.AlertaSend = 1;
            esp_err_t result = esp_now_send(slaveAddress, (uint8_t *) &envioData,
sizeof(envioData));
```

```
            if (result == ESP_OK) {
              Serial.println("Sent with success");
            } else {
              Serial.println("Error sending the data");
            }
            delay(3000);
            client.send("FACE NOT RECOGNISED");
          }
        }
        dl_matrix3d_free(out_res.face_id);
      }

    }
    else
    {
      if (g_state != START_DETECT) {
        client.send("NO FACE DETECTED");
      }
    }
  }
  client.sendBinary((const char *)fb->buf, fb->len);

  esp_camera_fb_return(fb);
  fb = NULL;
 }
}
```

8.4 CÓDIGO ESP32-CAM ESCLAVO

```
#include <esp_now.h>
#include <WiFi.h>
#include <ESP_Mail_Client.h>

const char* ssid = "NOMBRE-RED-WIFI";
const char* password = "CLAVE-RED-WIFI";

typedef struct struct_message {
    int AlertaSend;
} struct_message;

struct_message envioData;

#define SMTP_HOST "smtp.gmail.com"
#define SMTP_PORT 465
#define AUTHOR_EMAIL "XXXXX@gmail.com"
#define AUTHOR_PASSWORD "XXXXX"
SMTPSession smtp;
SMTP_Message message;
ESP_Mail_Session session;

void smtpCallback(SMTP_Status status);

void OnRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&envioData, incomingData, sizeof(envioData));
  Serial.print("Bytes received: ");
  Serial.println(len);
  Serial.print("AlertaSend: ");
  Serial.println(envioData.AlertaSend);
}

void setup(){
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.print(WiFi.localIP());
```

```
  smtp.debug(1);

  session.server.host_name = SMTP_HOST;
  session.server.port = SMTP_PORT;
  session.login.email = AUTHOR_EMAIL;
  session.login.password = AUTHOR_PASSWORD;
  session.login.user_domain = "";

  message.sender.name = "ESP32-CAM-FACE";
  message.sender.email = AUTHOR_EMAIL;
  message.subject = "ROSTRO DESCONOCIDO FUERA DE TU HOGAR";
  message.addRecipient("CASA618", "registros.mids@gmail.com");

  String textMsg = "ROSTRO DESCONOCIDO FUERA DE TU HOGAR";
  message.text.content = textMsg.c_str();
  message.text.charSet = "utf-8";
  message.text.transfer_encoding = Content_Transfer_Encoding::enc_7bit;
  message.priority = esp_mail_smtp_priority::esp_mail_smtp_priority_high;

  if (!smtp.connect(&session))
    return;

  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  esp_now_register_recv_cb(OnRecv);
}

void loop(){
  if (envioData.AlertaSend == 1){
    if (!MailClient.sendMail(&smtp, &message))
    Serial.println("Error sending Email, " + smtp.errorReason());

    smtp.sendingResult.clear();

    ESP_MAIL_PRINTF("Free Heap: %d\n", MailClient.getFreeHeap());
    delay(3000);
  }
  envioData.AlertaSend = 0;
}

void smtpCallback(SMTP_Status status)
{
  Serial.println(status.info());
```

```
  if (status.success())
  {
    Serial.println("----------------");
    ESP_MAIL_PRINTF("Message sent success: %d\n", status.completedCount());
    ESP_MAIL_PRINTF("Message sent failled: %d\n", status.failedCount());
    Serial.println("----------------\n");
    struct tm dt;

    for (size_t i = 0; i < smtp.sendingResult.size(); i++)
    {
      SMTP_Result result = smtp.sendingResult.getItem(i);
      time_t ts = (time_t)result.timestamp;
      localtime_r(&ts, &dt);

      ESP_MAIL_PRINTF("Message No: %d\n", i + 1);
      ESP_MAIL_PRINTF("Status: %s\n", result.completed ? "success" : "failed");
      ESP_MAIL_PRINTF("Date/Time: %d/%d/%d %d:%d:%d\n", dt.tm_year + 1900, dt.tm_mon +
1, dt.tm_mday, dt.tm_hour, dt.tm_min, dt.tm_sec);
      ESP_MAIL_PRINTF("Recipient: %s\n", result.recipients);
      ESP_MAIL_PRINTF("Subject: %s\n", result.subject);
    }
    Serial.println("----------------\n");

    smtp.sendingResult.clear();
  }
}
```

UNIVERSIDAD DE CONCEPCIÓN – FACULTAD DE INGENIERÍA
RESUMEN DE MEMORIA DE TÍTULO

Departamento **:** Departamento de Ingeniería Eléctrica
Carrera **:** Ingeniería Civil Electrónica
Nombre del memorista **:** Matias Ignacio Delgado Salazar
Título de la memoria **:** Sistema domótico de videovigilancia
Fecha de la presentación oral: 01/04/2022

Profesor Guía **:** Mario Rubén Medina Carrasco
Profesores Revisores **:** Sergio Sobarzo G. – Jorge Pezoa N.
Concepto **:**
Calificación **:**

Resumen (máximo 200 palabras)

Se desarrolla e implementa un sistema domótico de videovigilancia que es capaz de detectar y reconocer rostros humanos en tiempo real, con el fin de brindar seguridad al hogar frente a posibles robos que puedan ocurrir. Dicha seguridad se logra gracias a un correo de alerta que envía automáticamente el sistema al momento de detectar y no reconocer un rostro.

El sistema desarrollado se logra implementar gracias a dos dispositivos electrónicos de bajo costo y, a la vez funcionales. Estos corresponden a la cámara ESP32-CAM y el controlador Orange Pi Lite.

Por último, las principales ventajas del sistema implementado son: alta tasa de acierto en la detección y reconocimiento facial en tiempo real a condiciones ideales en el ambiente de trabajo; señal de alerta vía correo electrónico; conexión remota estable al servidor web; acceso en tiempo real al video del ESP32-CAM; utilización de dispositivos de bajo costo y de fácil accesibilidad.