UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

GESTIÓN DE VIDEO STREAMING MEDIANTE DISPOSITIVOS IOT EN PLATAFORMAS DOMÓTICAS

CUESTA VERA JUAN BRYAN
INGENIERO DE SISTEMAS

MACHALA
2021

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

GESTIÓN DE VIDEO STREAMING MEDIANTE DISPOSITIVOS IOT EN PLATAFORMAS DOMÓTICAS

CUESTA VERA JUAN BRYAN
INGENIERO DE SISTEMAS

MACHALA
2021

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

TRABAJO TITULACIÓN
PROPUESTAS TECNOLÓGICAS

GESTIÓN DE VIDEO STREAMING MEDIANTE DISPOSITIVOS IOT EN PLATAFORMAS DOMÓTICAS

CUESTA VERA JUAN BRYAN
INGENIERO DE SISTEMAS

HERNANDEZ ROJAS DIXYS LEONARDO

MACHALA, 26 DE ABRIL DE 2021

MACHALA
2021

Tesis J

INFORME DE ORIGINALIDAD

0%	0%	0%	0%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	es.unionpedia.org Fuente de Internet	<1%

Excluir citas	Activo	Excluir coincidencias	< 30 words
Excluir bibliografía	Activo		

CLÁUSULA DE CESIÓN DE DERECHO DE PUBLICACIÓN EN EL REPOSITORIO DIGITAL INSTITUCIONAL

El que suscribe, CUESTA VERA JUAN BRYAN, en calidad de autor del siguiente trabajo escrito titulado GESTIÓN DE VIDEO STREAMING MEDIANTE DISPOSITIVOS IOT EN PLATAFORMAS DOMÓTICAS, otorga a la Universidad Técnica de Machala, de forma gratuita y no exclusiva, los derechos de reproducción, distribución y comunicación pública de la obra, que constituye un trabajo de autoría propia, sobre la cual tiene potestad para otorgar los derechos contenidos en esta licencia.

El autor declara que el contenido que se publicará es de carácter académico y se enmarca en las dispociones definidas por la Universidad Técnica de Machala.

Se autoriza a transformar la obra, únicamente cuando sea necesario, y a realizar las adaptaciones pertinentes para permitir su preservación, distribución y publicación en el Repositorio Digital Institucional de la Universidad Técnica de Machala.

El autor como garante de la autoría de la obra y en relación a la misma, declara que la universidad se encuentra libre de todo tipo de responsabilidad sobre el contenido de la obra y que asume la responsabilidad frente a cualquier reclamo o demanda por parte de terceros de manera exclusiva.

Aceptando esta licencia, se cede a la Universidad Técnica de Machala el derecho exclusivo de archivar, reproducir, convertir, comunicar y/o distribuir la obra mundialmente en formato electrónico y digital a través de su Repositorio Digital Institucional, siempre y cuando no se lo haga para obtener beneficio económico.

Machala, 26 de abril de 2021

CUESTA VERA JUAN BRYAN
0705570679

DEDICATORIA

El presente trabajo lo dedico especialmente a mis padres quienes son fuente de inspiración, además de estar presentes en cada momento en que necesité consejos de superación. También a aquellas personas que aportaron un granito de arena de su conocimiento para ayudarme a cumplir con cada uno de los requisitos indispensables para culminar exitosamente la carrera de Ingeniería de Sistemas.

AGRADECIMIENTO

Agradezco en primer lugar a Dios por darme las fuerzas para luchar y no darme por vencido ante las adversidades.

A mis padres por sus sabios consejos que me motivaron día a día a cumplir cada uno de mis objetivos no sólo en lo académico sino también en la vida cotidiana.

A los docentes de la carrera de Ingeniería de Sistemas, quienes humildemente compartieron su conocimiento y experiencias para formar estudiantes con la capacidad de aportar mejoras a la sociedad.

Agradezco a mi tutor, el Ing. Dixys Leonardo Hernández Rojas por su constante apoyo en el desarrollo de este trabajo de titulación, pues gracias a él he adquirido nuevos conocimientos que me van a servir en la etapa profesional.

RESUMEN

En los últimos años se ha visto una creciente en la industria de la automatización del hogar a medida que aumenta la fabricación de dispositivos inteligentes y la dependencia hacia los teléfonos inteligentes.

A pesar de esto persiste el problema en la interoperabilidad debido a la falta de un protocolo de comunicación estándar por lo que el usuario debe operar con múltiples sistemas para la gestión de los dispositivos inteligentes, lo cual se convierte en una tarea tediosa pues a mayor cantidad de dispositivos más difícil será su gestión.

En este trabajo se implementó un sistema de domótica utilizando la plataforma Home Assistant con enfoque en la integración de diferentes tipos de cámaras y de varios fabricantes. Se consideraron cuatro tipos de dispositivos, en primer lugar, está la cámara TAPO C200 1080p con función PTZ, la cámara analógica 1080p con su respectivo DVR de la marca Hikvision, la placa ESP32-CAM con su módulo de cámara OV2640 y un smartphone Android con el software IP Webcam que lo convierte en un dispositivo de cámara de vigilancia preparado para su integración en Home Assistant. Con base en esto se proporcionaron distintas alternativas de integración como las nativas por medio de la API de Home Assistant para los dispositivos basados en el firmware ESPHome como la placa ESP32-CAM y para smartphones con el software IP Webcam, también mediante la tecnología WebRTC e incluso con MotionEye para soluciones en las que no sea posible implementar las anteriores lo que garantiza una baja latencia en la recepción de los flujos de video. Para la captura de imágenes se proporcionan alternativas como el uso del comando FFMPEG, la llamada al servicio de camera.snapshot de Home Assistant y el uso de APIs desarrolladas por los fabricantes de las cámaras. El sistema fue instalado en una Raspberry PI 4 con 4GB de RAM, considerada como la más potente de su tipo en la actualidad a la cual se le agregó un SSD para mejorar su rendimiento, debido a que, la carga de trabajo planteada en esta investigación no puede desarrollarse eficientemente por el método tradicional que consiste en iniciar el sistema operativo desde una tarjeta microSD de alta velocidad.

Al final se proporciona un análisis comparativo sobre la eficiencia en la ejecución de un escenario de automatización considerando la latencia en cada uno de los procesos que lo conforman permitiendo de esta manera conocer qué alternativa es la más óptima acorde a las necesidades de cada persona. Como resultado se obtuvo que las implementaciones con la tecnología WebRTC garantizan tiempos de latencia más bajos con la ventaja de que permiten el acceso al audio de la transmisión a diferencia de las otras opciones. Para la captura de imágenes se comprobó que el acceso por medio de la API proporcionada por el fabricante de las cámaras ofrece los tiempos más bajos en la adquisición de la misma con el beneficio del aprovechamiento total de la resolución de la cámara al no depender de software intermediario como FFMPEG o MotionEye que influyen en la velocidad y calidad de las imágenes.

Palabras claves:

IoT, Domótica, Sistemas domóticos, Home Assistant, Raspberry Pi, WebRTC, FFMPEG, MotionEye, API, ESPHome

ABSTRACT

In recent years there has been a growing in the home automation industry as smart device manufacturing and smartphone dependence increases.

Despite this, the problem persists in interoperability due to the lack of a standard communication protocol so the user must operate with multiple systems for the management of smart devices, which becomes a tedious task because the more difficult their management will be.

In this work, a home automation system was implemented using the Home Assistant platform with a focus on integrating different types of cameras and from various manufacturers. Four types of devices were considered, first of all, there is the TAPO C200 1080p camera with PTZ function, the 1080p analog camera with its respective Hikvision brand DVR, the ESP32-CAM board with its OV2640 camera module and an Android smartphone with IP Webcam software that makes it a surveillance camera device ready for integration into Home

Assistant. Based on this, different integration alternatives such as native ones were provided through the Home Assistant API for ESPHome firmware-based devices such as the ESP32-CAM board and for smartphones with IP Webcam software, also through WebRTC technology and even MotionEye for solutions where it is not possible to implement the above which ensures low latency in receiving video streams. For image capture alternatives such as using the FFMPEG command, calling the Home Assistant camera.snapshot service, and using APIs developed by camera manufacturers are provided. The system was installed on a Raspberry PI 4 with 4GB of RAM, considered to be the most powerful of its kind today to which an SSD was added to improve its performance, because the workload posed in this research cannot be efficiently developed by the traditional method of starting the operating system from a high-speed microSD card.

In the end, a comparative analysis is provided on the efficiency in the execution of an automation scenario considering the latency in each of the processes that make up it allowing to know which alternative is the most optimal according to the needs of each person. As a result, WebRTC-powered deployments ensure lower latency times with the advantage that they allow access to streaming audio unlike other options. Image capture found that access through the API provided by the camera manufacturer offers the lowest times in the acquisition of the camera with the benefit of the full use of camera resolution by not relying on intermediary software such as FFMPEG or MotionEye that influence the speed and quality of the images.

Keywords:

IoT, Home automation, Home automation systems, Home Assistant, Raspberry Pi, WebRTC, FFMPEG, MotionEye, API, ESPHome

CONTENIDO

DEDICATORIA 1
AGRADECIMIENTO 2
RESUMEN 3
ABSTRACT 4
INTRODUCCIÓN 11
1. CAPÍTULO I. DIAGNÓSTICO DE NECESIDADES Y REQUERIMIENTOS 13
1.1. Ámbito de aplicación: descripción del contexto y hechos de interés 13
1.2. Establecimiento de requerimientos 14
1.3. Justificación del requerimiento a satisfacer 14
2. CAPÍTULO II. DESARROLLO DEL PROYECTO 15
2.1. Definición del prototipo tecnológico 15
2.2. Fundamentación teórica del prototipo 16
2.2.1. IoT (Internet Of Things) 16
2.2.2. Domótica 16
2.2.2.1. Plataformas domóticas 16
2.2.2.1.1. Home Assistant 17
2.2.2.1.2. OpenHab 17
2.2.3. Raspberry Pi 17
2.2.4. IOTstack 17
2.2.5. ZeroTier 18
2.2.6. Streaming de video 18
2.2.6.1. Técnicas de compresión de video 18
2.2.6.1.1. MPEG-4 18
2.2.6.1.2. H.264 / MPEG-4 AVC 19
2.2.6.1.3. H.265 / High Efficiency Video Coding (HEVC) 19
2.2.6.2. Métodos de transmisión de video streaming 19
2.2.6.2.1. Streaming tradicional 19
2.2.6.2.2. Live Streaming 19
2.2.6.2.3. HTTP Pseudo Streaming 19
2.2.6.3. Protocolos de video streaming 20
2.2.6.3.1. Tecnología Push 20
2.2.6.3.1.1. Pila de protocolos RTP 20
2.2.6.3.2. Tecnología Pull 20
2.2.6.3.2.1. RTMP 20

2.2.6.3.2.2. HLS 20
2.2.6.3.2.3. DASH 21
2.3. OBJETIVOS DEL PROTOTIPO 22
2.3.1. Objetivo general 22
2.3.2. Objetivos específicos 22
2.4. DISEÑO DEL PROTOTIPO 22
2.5. EJECUCIÓN Y/O ENSAMBLAJE DEL PROTOTIPO 24
2.5.1. Instalación de IOTstack 24
2.5.2. Instalación de Home Assistant 24
2.5.3. Instalación de add-ons esenciales en Home Assistant 27
2.5.3.1. SSH web client 27
2.5.3.2. File Editor 28
2.5.3.3. HACS 29
2.5.4. Integración de las cámaras en Home Assistant 30
2.5.4.1. Integración de la cámara TAPO C200 30
2.5.4.2. Integración de la cámara analógica con su DVR Hikvision 34
2.5.4.3. Integración de la placa ESP32Cam 37
2.5.4.4. Integración del smartphone Android IP Webcam 39
2.5.5. Integración de servicios en Home Assistant 42
2.5.5.1. Facebox 43
2.5.5.2. Mozilla TTS 44
2.5.5.2.1. Local VLC 45
2.5.6. Acceso desde Internet mediante Zerotier 47
2.5.7. Construcción del escenario de automatización en Node-Red con API Home Assistant. 50
2.5.7.1. Instalación de Node-Red 50
2.5.7.2. Desarrollo del escenario 51
3. CAPÍTULO III. EVALUACIÓN DEL PROTOTIPO 53
3.1. PLAN DE EVALUACIÓN 53
3.1.1. Node-Red para la adquisición de los tiempos de procesamiento 55
3.1.2. Wireshark y PCAP Remote para la adquisición de los tiempos de comunicación 55
3.2. RESULTADOS DE LA EVALUACIÓN 56
3.2.1. Tiempo de procesamiento de imágenes con Facebox 56
3.2.2. Tiempo de generación del contenido de la notificación 58
3.2.3. Latencia de notificaciones en local 59

3.2.3.1. Latencia en envío desde Home Assistant hasta cliente Smartphone. 59
3.2.3.2. Latencia en envío desde cliente Smartphone hasta Home Assistant. 60
3.2.3.3. Comparación de latencia de notificaciones en local 60
3.2.4. Tiempo de generación y reproducción de notificación de voz mediante Mozilla TTS. 61
3.2.5. Latencia de video streaming en local 62
3.2.6. Reconocimiento de rostros con Facebox 63
3.2.7. Análisis de resultados 64
3.3. CONCLUSIONES 64
3.4. RECOMENDACIONES 65
BIBLIOGRAFÍA 66

ÍNDICE DE TABLAS

Tabla 1: Características de las cámaras integradas en Home Assistant. 53
Tabla 2: Características de los servidores. 53
Tabla 3: Características de los clientes. 54
Tabla 4: Resumen de tiempo de procesamiento de imágenes con Facebox. 56
Tabla 5: Resumen de tiempo de generación del contenido de notificación. 59
Tabla 6: Resumen de latencia en comunicación HA – cliente, por medio de notificaciones. 60
Tabla 7: Resumen de latencia en comunicación cliente - HA, por medio de notificaciones. 60
Tabla 8: Resumen de tiempo de generación y reproducción de notificación de voz mediante Mozilla TTS. 61
Tabla 9: Resumen de latencia de video streaming en local. 62
Tabla 10:Grado de acierto de reconocimiento de rostros con Facebox. 63
Tabla 11: Comparación de tiempos promedios de ejecución del escenario clasificados por cámaras. 64

ÍNDICE DE FIGURAS

Figura 1: Arquitectura de la propuesta. 15
Figura 2: Integraciones en Home Assistant. 23
Figura 3: Escenario de automatización. 24
Figura 4: Comando de instalación de IOTstack. 24
Figura 5: Dependencias para la instalación de Home Assistant. 25
Figura 6: Gestión del servicio ModemManager. 25
Figura 7: Instalación de Home Assistant 25
Figura 8: Creación de cuenta personal en Home Assistant. 26
Figura 9: Configuración inicial en Home Assistant. 26
Figura 10: Instalación del add-on SSH web client 27
Figura 11: Configuración del add-on SSH web client 28
Figura 12: Add-on SSH web client 28
Figura 13: Add-on File Editor 29
Figura 14: Comando para la instalación del add-on HACS 29
Figura 15: Integración de HACS en Home Assistant 30
Figura 16: HACS 30
Figura 17: Generación de credenciales en la aplicación TAPO. 31
Figura 18: Ingreso de credenciales en Integración de la cámara TAPO. 31
Figura 19: Opciones en la configuración de la cámara TAPO. 32
Figura 20: Definición del área que cubre la cámara TAPO. 32
Figura 21: Integración de la cámara TAPO. 32
Figura 22: Tarjeta Lovelace para la cámara TAPO. 34
Figura 23: Habilitar protocolos de integración en DVR. 35
Figura 24: Creación de usuario multimedia en DVR. 35

Figura 25: Habilitar protocolo RTSP en DVR. 35
Figura 26: Dispositivo ONVIF detectado. 36
Figura 27: Tarjeta Lovelace DVR Onvif. 36
Figura 28: Integración de DVR mediante MotionEye. 37
Figura 29: Módulo ESP32Cam mejorado. 39
Figura 30: Aplicación móvil IP Webcam. 39
Figura 31: Aplicación móvil IP Webcam, en funcionamiento. 40
Figura 32: Tarjeta Lovelace para Android IP Webcam. 41
Figura 33: Tarjeta Lovelace con sensores del dispositivo Android IP Webcam integrados en Home Assistant. 42
Figura 34: Servicio de reconocimiento facial, Facebox. 43
Figura 35: Servicio Mozilla TTS. 45
Figura 36: Configuración de Local VLC. 45
Figura 37: Interfaz web del complemento Local VLC. 46
Figura 38: Tarjeta Lovelace del add-on Local VLC. 46
Figura 39: Interacción Local VLC con Mozilla TTS. 47
Figura 40: Agregar rutas en Zerotier. 48
Figura 41: Instalación de cliente Zerotier en Raspberry 48
Figura 42: Agregar Raspberry a la red Zerotier. 48
Figura 43: Raspberry Pi dado de alta en la red Zerotier. 48
Figura 44: Cliente Android Zerotier. 49
Figura 45: Acceso remoto a Home Assistant. 50
Figura 46: Instalación de Node-Red. 50
Figura 47: Node-Red. 51
Figura 48: Aplicación Node-Red. Detección de movimiento. 51
Figura 49: Aplicación Node-Red. Envío de notificación al propietario. 52
Figura 50: Aplicación Node-Red. Respuesta de notificación del propietario. 52
Figura 51: Tiempos que intervienen en el escenario de automatización. 54
Figura 52: Tiempo máximo de procesamiento con Facebox. 57
Figura 53: Tiempo mínimo de procesamiento de imágenes con Facebox. 57
Figura 54: Tiempo promedio de procesamiento de imágenes con Facebox. 58
Figura 55: Resumen de tiempo de generación del contenido de notificación. 59
Figura 56: Resumen latencia de notificaciones en local 60
Figura 57: Resumen de tiempo de generación y reproducción de notificación de voz mediante Mozilla TTS. 61
Figura 58: Resumen de latencia de video streaming en local. 62
Figura 59: Grado de acierto en reconocimiento de rostros con Facebox. 63

INTRODUCCIÓN

En los últimos años se ha visto una creciente en la industria de la automatización del hogar a medida que aumenta la fabricación de dispositivos inteligentes.

A pesar de esto persiste el problema en la interoperabilidad debido a la falta de un protocolo de comunicación estándar por lo que el usuario debe operar con múltiples sistemas para la gestión de los dispositivos inteligentes.

Existen soluciones que permiten la integración de dispositivos de diferentes fabricantes como Smartthings de Samsung, Google Home, Alexa, entre otros [1], [2] pero todos tienen algo en común y es que dependen de la nube, un aspecto importante que se debe considerar si lo que se pretende es mantener la privacidad de la información personal. Además de que no admiten cualquier dispositivo, sino que tienen que estar etiquetados por el fabricante como dispositivos compatibles.

En este trabajo se implementó un sistema de domótica utilizando la plataforma Home Assistant con enfoque en la integración de diferentes tipos de cámaras y de varios fabricantes. Se proporcionan distintas alternativas de integración como las nativas por medio de la API de Home Assistant para los dispositivos basados en ESPHome y para smartphones con el software IP Webcam, también mediante la tecnología WebRTC e incluso con MotionEye para soluciones en las que no sea posible implementar las anteriores lo que garantiza una baja latencia en la recepción de los flujos de video. Para la captura de imágenes se proporcionan alternativas como el uso de FFMPEG, llamadas al servicio de camera.snapshot de Home Assistant y el uso de APIs proporcionadas por los fabricantes de las cámaras.

El sistema domótico fue instalado en una Raspberry PI 4 con un SSD para mejorar el rendimiento, debido a los recursos que demanda la gestión de video.

Además, se implementó un escenario de automatización basado en reconocimiento facial utilizando Facebox y notificaciones dictadas por voz a través de Mozilla TTS. Tanto los servicios como el sistema domótico funcionan de manera local lo que garantiza la privacidad de la información.

Al final se proporciona un análisis comparativo sobre la eficiencia en la ejecución de un escenario de automatización considerando la latencia en cada uno de los procesos que lo conforman permitiendo de esta manera conocer qué alternativa es la más óptima acorde a las necesidades de cada persona.

La estructura del presente informe se muestra a continuación.

En el **Capítulo 1,** describe la necesidad de implementar el sistema domótico y sus requerimientos.

El **Capítulo 2,** describe la definición del prototipo, la fundamentación teórica, objetivos, diseño y ejecución del prototipo.

El **Capítulo 3,** se define el plan de evaluación y los resultados de evaluación, conclusiones acordes a los resultados y las recomendaciones para futuros trabajos.

1. CAPÍTULO I. DIAGNÓSTICO DE NECESIDADES Y REQUERIMIENTOS

1.1. Ámbito de aplicación: descripción del contexto y hechos de interés

El término IoT está bastante relacionado con el control de dispositivos inteligentes con la capacidad de conectarse a internet para intercomunicarse entre sí. El término se asocia comúnmente a la implementación de sensores para conocer el estado actual de algún determinado escenario.

Dentro de los dispositivos IoT podemos considerar dos tipos: por un lado, los más acogidos por los usuarios finales como televisores inteligentes, refrigeradoras, lavadoras, etc y por otro lado dispositivos utilizados por investigadores tales como Raspberry Pi, Arduino, sensores, entre otros [3].

Los dispositivos con cámaras integradas son bastante utilizados en la vida cotidiana principalmente para capturar fotos y vídeos o incluso implementados en servicios de video vigilancia. Una función particular que destaca de estos dispositivos es la capacidad de transmitir en tiempo real las grabaciones mediante el uso de protocolos para la transmisión de video streaming, esto resulta muy útil desde el punto de vista del IoT pues permiten mediante el procesamiento y análisis del flujo de vídeo poder automatizar tareas dependiendo de cada situación en particular que sea capturado por las cámaras.

Debido a la gran variedad de dispositivos disponibles en el mercado y de distintos fabricantes surge la necesidad de un sistema que permita su adecuada integración. Los sistemas domóticos resuelven aquel inconveniente, pues su propósito es integrar cualquier dispositivo IOT. De esta manera se pueden reutilizar sistemas de video vigilancia ya implementados, porteros (doorbell) o incluso reciclar teléfonos inteligentes y evitar su obsolescencia programada.

La presente propuesta tecnológica consiste en la integración de cámaras para proveer de visión al sistema domótico Home Assistant, logrando de esta manera el aprovechamiento de la captura de imágenes y vídeos para su posterior procesamiento con algoritmos de reconocimiento artificial para automatizar tareas con base en reglas previamente establecidas.

1.2. Establecimiento de requerimientos

Actualmente existen multitud de dispositivos inteligentes aplicados en el ámbito de la domótica. El inconveniente surge cuando se trata con dispositivos de diferentes fabricantes en donde no existe un estándar que permita gestionarlos mediante un único sistema [4] y en el peor de los casos tener un sistema distinto para cada dispositivo, lo cual complica aún más la situación pues a mayor cantidad de dispositivos más complicada será su gestión. Las cámaras no se escapan de este problema pues existen del tipo analógica, digital, con funciones especiales como PTZ que le otorgan movilidad en distintos ángulos e incluso las hay con sensores incluidos, entre muchas otras funciones.

Se propone la implementación de un sistema domótico que permita la integración de las distintas cámaras mediante el uso de protocolos de integración como ONVIF e ISAPI con el objetivo de aprovechar el flujo de vídeo proporcionado por las mismas y realizar automatismos mediante el análisis de cada caso que faciliten la vida del usuario final.

1.3. Justificación del requerimiento a satisfacer

Existe la necesidad de integrar dispositivos de distintos fabricantes debido a la complejidad de su gestión al no existir un único sistema que pueda manejarlos.

A pesar de que existen sistemas que permiten integrar los flujos de video provenientes de las cámaras, estos se limitan al servicio de video vigilancia desaprovechando las bondades que ofrecen las cámaras.

La propuesta de este proyecto no sólo permite la integración de las cámaras considerando su heterogeneidad, sino que agrega mejoras a su funcionamiento otorgándole al sistema domótico visión para poder realizar automatismos de acuerdo a escenarios que las cámaras graben. Lo más interesante de todo esto es que se pueden reutilizar sistemas de video vigilancia ya implementados asimismo se pueden reutilizar smartphones que han pasado por su etapa de obsolescencia programada y ayudar al medio ambiente. Al aprovechar las imágenes y los vídeos proporcionado es posible procesar esas entradas y aplicar por ejemplo algoritmos de reconocimiento facial para definir reglas de

acuerdo a cada caso logrando de esta manera automatismos particulares para cada usuario del sistema.

2. CAPÍTULO II. DESARROLLO DEL PROYECTO

2.1. Definición del prototipo tecnológico

La arquitectura propuesta se muestra en la Figura 1. Consiste básicamente en la instalación del asistente domótico dentro de la red local en donde se integran todos los dispositivos inteligentes.

Además, se integran servicios de reconocimiento facial y TTS para otorgarle al asistente domótico una manera de poder comunicar sus acciones por medio de un sintetizador de texto a voz.

Se puede acceder al asistente domótico desde la misma red local mediante un smartphone o una computadora e incluso se puede acceder por medio de Internet a través del servicio de VPN seguro de la empresa ZeroTier.

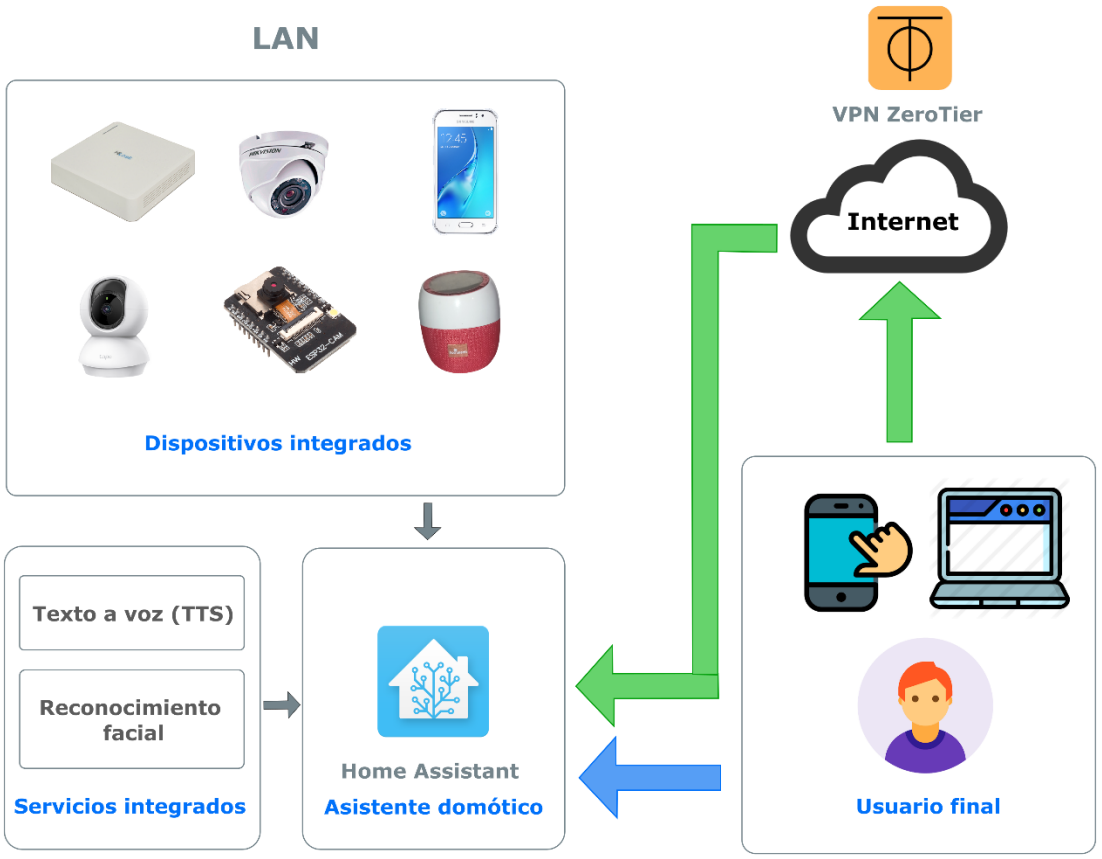

Figura 1: Arquitectura de la propuesta.

Fuente: Elaboración propia.

2.2. Fundamentación teórica del prototipo

2.2.1. IoT (Internet Of Things)

El término Internet Of Things o Internet de las Cosas se refiere a la interconexión de múltiples dispositivos con la capacidad de comunicarse entre sí a través de Internet con el objetivo de mejorar la calidad de vida. Basa su funcionamiento en la recopilación, procesamiento y análisis de datos del mundo real, captados por sensores y, con base en ellos actuar y mejorar la toma de decisiones en IoT [3] [5].

Un hecho importante al tratar con dispositivos IOT es que tienen recursos limitados y generalmente implica una variedad de tecnologías heterogéneas [6].

2.2.2. Domótica

Con énfasis en la comodidad surge el concepto de domótica o también conocido como casa inteligente que involucra la combinación de una red de sensores con objetos inteligentes conectados a Internet [7].

Este concepto desde el punto de vista de la sociedad se suele asociar a edificios modernos que se centran en ofrecer servicios para garantizar el confort y bienestar de sus residentes [8]. Sin embargo, también puede ser aplicado a hogares comunes [9] debido al aporte que hacen los fabricantes de dispositivos inteligentes para que su implementación sea sencilla para el usuario común.

En resumen, un hogar inteligente consta de los siguientes elementos: tecnologías de comunicación, dispositivos inteligentes. Además se considera el ahorro de energía, seguridad, cuidado de la salud y la comodidad [8].

2.2.2.1. Plataformas domóticas

Existe un auge en cuanto a la fabricación de dispositivos inteligentes cuya instalación y uso se convierte en una tarea sumamente sencilla gracias a las aplicaciones móviles que los acompañan. Sin embargo, cuando se trata con dispositivos de diferentes fabricantes, no existe un protocolo de comunicación abierto que permita integrarlos por lo que su gestión se ve claramente limitada [10]. Desde este punto de vista surgen los sistemas domóticos o también

conocidos como plataformas domóticas que facilitan la comunicación de los dispositivos de distintas marcas permitiendo no sólo su integración sino la ejecución de automatismos de acuerdo a las necesidades de cada usuario.

2.2.2.1.1. Home Assistant

Se trata de un software de automatización de open source desarrollado en Python cuyo objetivo es ser el sistema de control central para el monitoreo y control de sensores, dispositivos inteligentes [11].

Puede ser ejecutado en una Raspberry Pi o un servidor. Además, garantiza la privacidad debido a su gestión local por medio de una base de datos en SQLite [11], [12].

2.2.2.1.2. OpenHab

Con un propósito similar al de Home Assistant se enfoca en ofrecer un medio para controlar varios dispositivos de manera remota y servicios sin importar el protocolo de comunicación [1]. Al estar desarrollado en Java puede ser ejecutado en sistemas como Windows, Linux y macOS e incluso Raspberry Pi [13].

2.2.3. Raspberry Pi

Es una computadora compacta muy utilizada en proyectos de IOT debido a su estructura y eficiencia energética. El sistema operativo oficial está basado en Debian, una distribución GNU/Linux, denominada Raspbian [14]

Existen muchos proyectos en donde es aplicada por ejemplo en aspectos como la detección y visión por computadora gracias a periféricos de soporte para propósitos específicos como la captura de video para su gestión y transmisión por la red [15].

2.2.4. IOTstack

Es un constructor desarrollado en Python que utiliza docker-compose para la gestión de pilas de IOT en Raspberry Pi [16].

2.2.5. ZeroTier

Es una solución que permite el acceso remoto a dispositivos en una red local por medio de su integración en redes seguras creadas en los servidores de ZeroTier [17].

2.2.6. Streaming de video

Una secuencia de video es básicamente un conjunto de imágenes que se actualizan periódicamente lo que da la sensación de movimiento. Está conformado por: la frecuencia de actualización de los frames que indica la cantidad de imágenes que se procesarán por segundo, la tasa de bits que establece el número de bits utilizados para transmitir y procesar los datos expresados en bits por segundo o bps y derivados y, el alto y ancho del frame que indica la cantidad de pixeles que debe ocupar cada frame [18] . Cada uno de los componentes son configurables por lo que a mayor valor mejor será su calidad considerando que el tamaño del archivo de video aumenta.

El video streaming se trata del proceso de entrega de contenido multimedia a través de la red donde el ancho de banda y los códecs son dos factores muy importantes para garantizar su transmisión [19]. Es ampliamente utilizado en múltiples aplicaciones como la transmisión de deportes, en videoconferencias e incluso en sistemas de video vigilancia [20].

2.2.6.1. Técnicas de compresión de video

La compresión de los archivos de video es un factor muy importante, debido a que, reduce considerablemente su tamaño permitiendo de esta manera que su transmisión pueda efectuarse aún en redes con bajo nivel de ancho de banda [18]. El elemento encargado de esta labor es denominado códec que consiste en dos componentes: el codificador que comprime y el decodificador que hace lo contrario para su visualización [19].

2.2.6.1.1. MPEG-4

Se trata de un estándar de codificación de video con baja tasa de bits y alta relación de compresión lo que da como resultado archivos relativamente pequeños que ocupan poco espacio de almacenamiento. [21]

2.2.6.1.2. H.264 / MPEG-4 AVC

Es el estándar de codificación de video ampliamente utilizado en los dispositivos actuales. Fue un desarrollo en conjunto entre ITU-T Video Coding Experts Group y el ISO/IEC Moving Picture Experts Group [21]. Por su alto nivel de compresión requiere de mayor potencia de procesamiento por parte del reproductor [18].

2.2.6.1.3. H.265 / High Efficiency Video Coding (HEVC)

La codificación de vídeo de alta eficiencia o HEVC es la mejora a H.264 debido a que ofrece calidades de vídeo 4K [22]. Ofrece una ganancia del 50% en la calidad con respecto a su antecesor H.264 [23]. A pesar de no ser relativamente nuevo aún no es implementado ampliamente en los dispositivos a diferencia de H.264 [24].

2.2.6.2. Métodos de transmisión de video streaming

2.2.6.2.1. Streaming tradicional

Se basa en descargar completamente los archivos multimedia para su posterior visualización. Esto tiene la ventaja de que no existen retardos durante la reproducción aunque por contra se debe esperar a que el archivo esté completamente descargado [19].

2.2.6.2.2. Live Streaming

Permite realizar transmisiones en vivo a partir de una fuente como una cámara hacia el Internet mediante protocolos de streaming que controlan la transferencia de información [25].

2.2.6.2.3. HTTP Pseudo Streaming

También conocido como descarga progresiva se basa en reproducir el contenido conforme se va descargando lo que produce una percepción de streaming real. Aunque con el inconveniente de que si no se cuenta con un buen ancho de banda en el cliente la experiencia de usuario puede no ser tan agradable [26].

2.2.6.3. Protocolos de video streaming

2.2.6.3.1. Tecnología Push

Una vez que se establece la conexión entre el cliente y el servidor, el servidor comienza a enviar segmentos de video continuamente sin que el cliente las solicite. El cliente decide en qué momento desea finalizar la comunicación [27].

2.2.6.3.1.1. Pila de protocolos RTP

El Protocolo de Transporte en Tiempo Real o RTP es utilizado para la transferencia de audio y video en streaming en tiempo real. Muy utilizado en aplicaciones como videoconferencia [28]. El protocolo por sí sólo no proporciona control del flujo multimedia ni de la congestión por lo que ahí entra en acción RTCP para encargarse de ello [29].

Real Time Streaming Protocol (RTSP) es otro protocolo perteneciente a la pila RTP. Rastrea el estado del cliente antes de proceder a realizar la transmisión. No admite calidad de imagen adaptativa [30].

Los navegadores no tienen la capacidad de reproducir el flujo de video por lo que se requiere de reproductores como VLC para visualizar la transmisión.

2.2.6.3.2. Tecnología Pull

En este tipo de conexiones el cliente es el encargado de realizar las solicitudes al servidor por lo que la calidad de la transmisión depende enteramente del mismo y de su calidad de red. Se utiliza en protocolos basados en HTTP sobre TCP [31].

2.2.6.3.2.1. RTMP

Desarrollado por Adobe, el Protocolo de Mensajes en Tiempo Real (RTMP) está enfocado a la transmisión entre productos de la familia Adobe Flash. Opera sobre TCP por lo que garantiza que durante la transmisión no existan pérdidas [30].

2.2.6.3.2.2. HLS

Fue desarrollado por Apple Inc. HTTP Live Streaming o HLS es un protocolo de transmisión con velocidad de bits adaptable que funciona sobre HTTP. Basa su funcionamiento en la división del flujo de vídeo en secuencias de pequeños archivos codificados a diferentes velocidades de bits con la idea de que el cliente pueda elegir el archivo con la mayor tasa de bits acorde al ancho de banda disponible [30].

2.2.6.3.2.3. DASH

Fue emitido en 2012 por el grupo de expertos de imágenes en movimiento MPEG. Popularmente conocido como DASH permite la transmisión a través de HTTP [32]. Proporciona una capacidad de transmisión adaptativa que permite adaptar la velocidad de video acorde a las condiciones en la red. Esto permite minimizar el reajuste en buffer en los clientes. Debido a que las redes soportan el protocolo TCP, DASH se ha convertido en una opción muy popular en la transición de medios garantizando que la transmisión no tenga pérdidas [33]

2.3. OBJETIVOS DEL PROTOTIPO

2.3.1. Objetivo general

Gestionar el video streaming proporcionado por dispositivos IOT comerciales y de desarrollo mediante su integración en el sistema domótico Home Assistant para la ejecución de automatismos basados en reconocimiento facial.

2.3.2. Objetivos específicos

- Realizar el estado del arte sobre IOT, domótica y video streaming.
- Instalar el sistema domótico Home Assistant en la placa Raspberry Pi 4.
- Integrar los dispositivos comerciales en Home Assistant.
- Integrar el dispositivo de desarrollo ESP32-CAM en Home Assistant.
- Integrar servicios de reconocimiento facial y TTS en Home Assistant.
- Realizar pruebas de latencia basadas en el acceso a nivel local y remoto para el aseguramiento del nivel de satisfacción en la experiencia de usuario.

2.4. DISEÑO DEL PROTOTIPO

El sistema domótico implementado en esta investigación fue Home Assistant, debido a su auge en el área de la domótica y la automatización, el mismo que fue instalado en el hardware Raspberry Pi 4 por su popularidad en los sistemas IoT. Con el objetivo de facilitar la instalación de las múltiples herramientas se decidió hacer uso del proyecto IOTstack, que basa su funcionamiento en contenedores; con esto se logró la instalación de Home Assistant Supervised y Portainer-CE para gestionar los contenedores.

Home Assistant permite la integración de dispositivos de múltiples marcas así como servicios externos para ampliar su funcionamiento, tal es el caso de Facebox para reconocimiento facial y Mozilla TTS para dotar de voz al sistema domótico.

Se integraron cuatro tipos de dispositivos, en primer lugar está la cámara TAPO C200 1080p con función PTZ, la cámara analógica 1080p con su respectivo DVR de la marca Hikvision, la placa ESP32-CAM con su módulo de cámara OV2640 y un smartphone Android con el software IP Webcam que lo convierte

en un dispositivo de cámara de vigilancia preparado para su integración en Home Assistant.

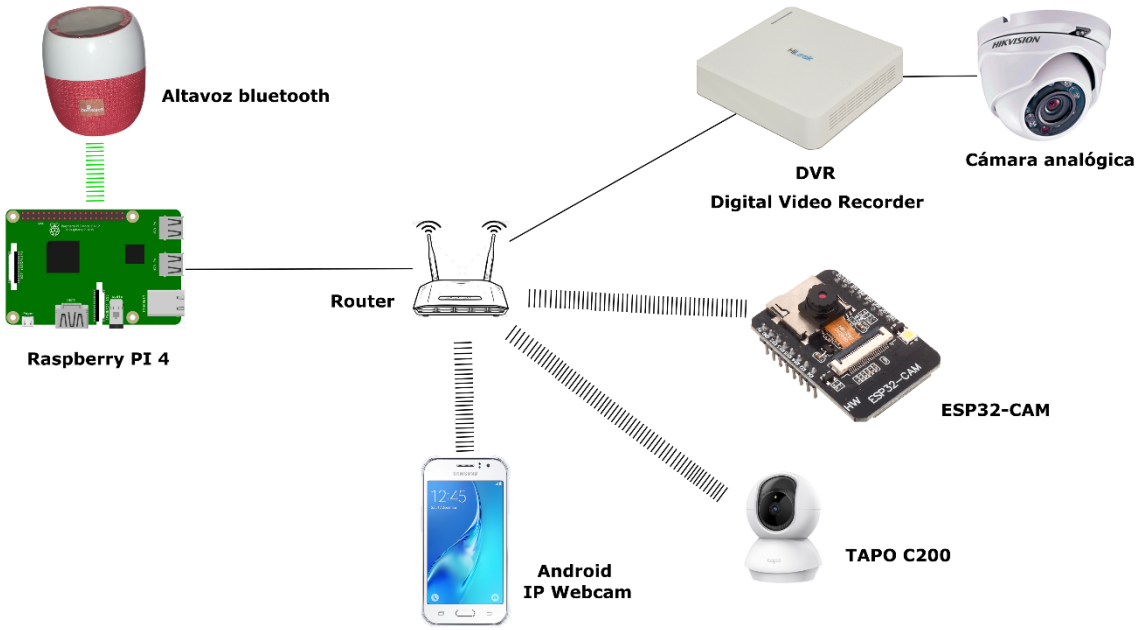

Figura 2: Integraciones en Home Assistant.

Fuente: Elaboración propia.

Además se implementó un escenario de automatización que consiste en ejecutar tareas a partir de una señal de detección de movimiento emitido por las integraciones donde el propietario tiene la opción de confirmar acciones con base en la respuesta a notificaciones.

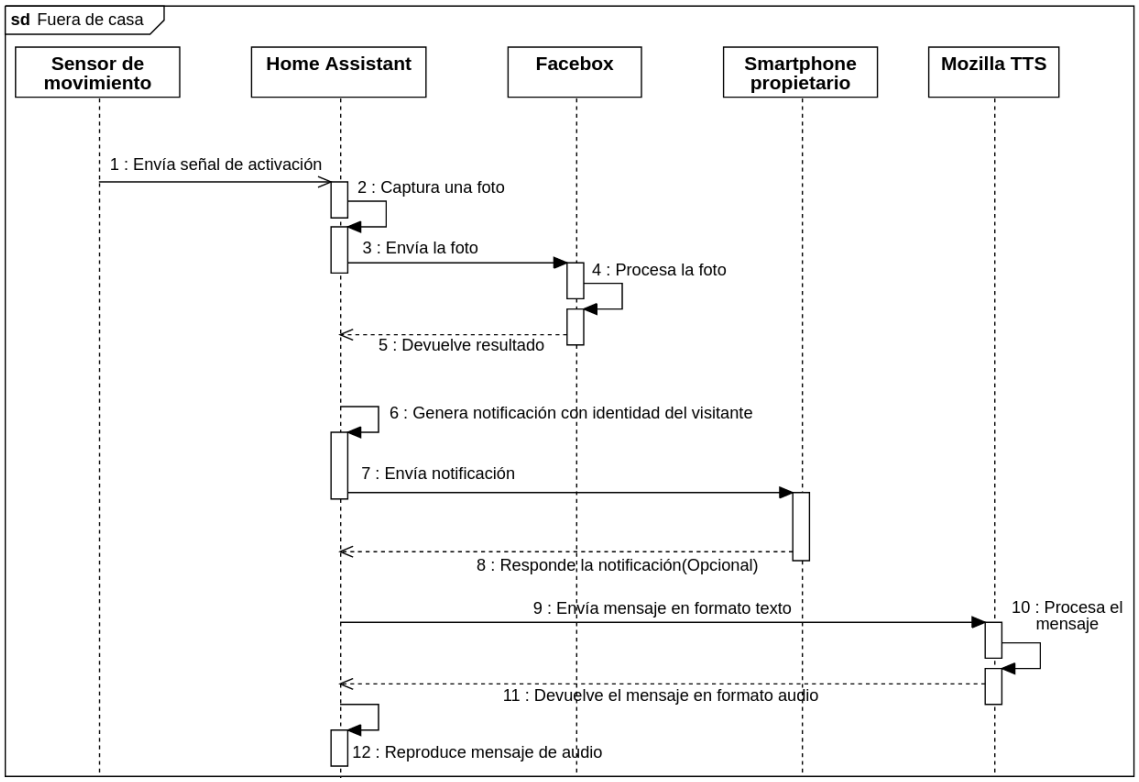

Figura 3: Escenario de automatización.

Fuente: Elaboración propia.

2.5. EJECUCIÓN Y/O ENSAMBLAJE DEL PROTOTIPO

2.5.1. Instalación de IOTstack

Para la instalación del proyecto IOTstack se ejecuta el siguiente comando.

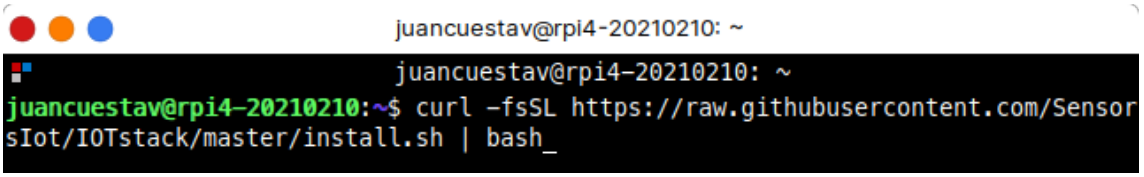

Figura 4: Comando de instalación de IOTstack.

Fuente: Elaboración propia.

Aparecen dos diálogos, uno para confirmar la instalación de Python3 y otro para Docker y Docker-Compose. Al finalizar se reinicia el sistema.

2.5.2. Instalación de Home Assistant

Como se mencionó, se emplea IOTstack para la instalación del sistema domótico, pero antes se deben instalar las dependencias que se muestran en la Figura 5.

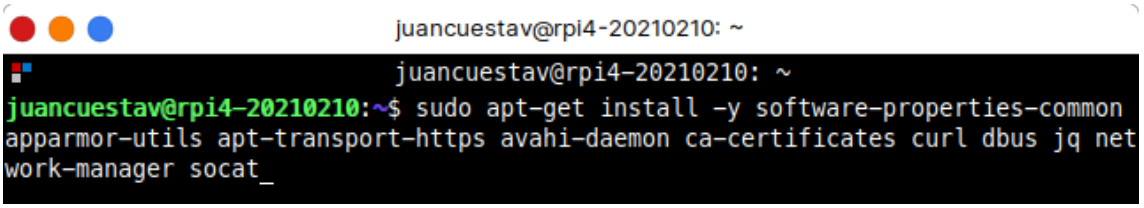

Figura 5: Dependencias para la instalación de Home Assistant.
Fuente: Elaboración propia.

Luego se procede a detener y deshabilitar el servicio de ModemManager.

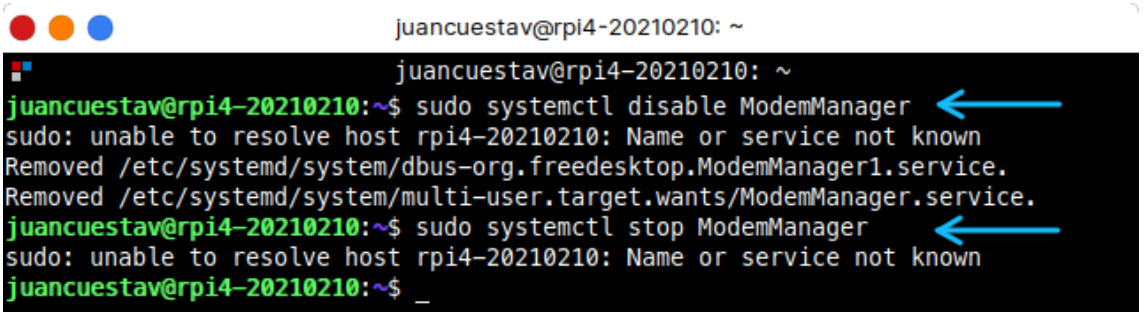

Figura 6: Gestión del servicio ModemManager.
Fuente: Elaboración propia.

A continuación, se procede a instalar Home Assistant. Para ello se ingresa al directorio en donde se instaló IOTstack y se ejecuta el archivo menú.sh en donde se escoge la opción Native Installs, luego Hass.io (Supervisor), se confirma la instalación y se ingresa la contraseña del usuario actual. Después se procede a seleccionar la arquitectura (raspberrypi-64). Una vez realizado lo anterior comenzará la instalación del sistema, esto toma alrededor de veinte minutos. Se puede visualizar el proceso mediante un navegador web, en la dirección IP de la placa Raspberry indicando el puerto 8123.

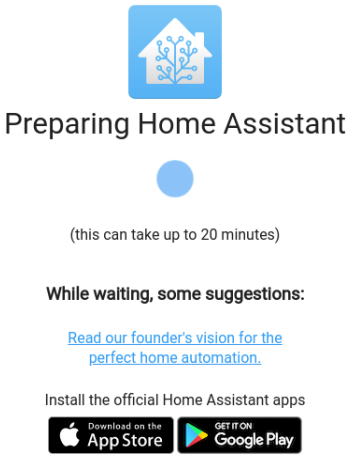

Figura 7: Instalación de Home Assistant
Fuente: Elaboración propia.

Una vez instalado se crea una cuenta personal, se establece la ubicación, la zona horaria y el sistema de unidades; el sistema tiene la opción de auto detectar estos valores.

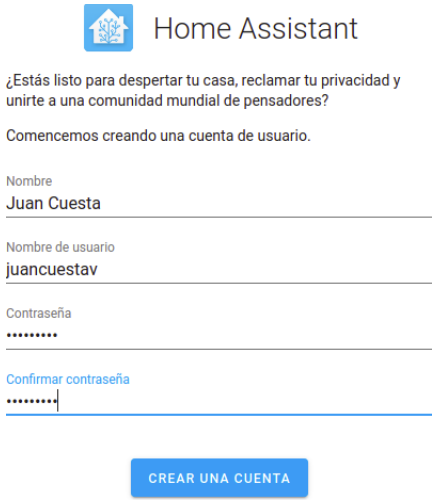

Figura 8: Creación de cuenta personal en Home Assistant.
Fuente: Elaboración propia.

Figura 9: Configuración inicial en Home Assistant.
Fuente: Elaboración propia.

2.5.3. Instalación de add-ons esenciales en Home Assistant

Los siguientes son los tres complementos básicos para trabajar con las integraciones y su gestión.

2.5.3.1. SSH web client

Ofrece una sesión SSH mediante un emulador de terminal integrado al cual se puede acceder por medio del cliente de Home Assistant, ya sea desde el navegador web o por la aplicación móvil. Para su instalación es necesario habilitar el *modo avanzado* en el Perfil de Usuario. Luego, con la ayuda del supervisor en la Tienda de complementos se busca SSH & Web Terminal y se instala.

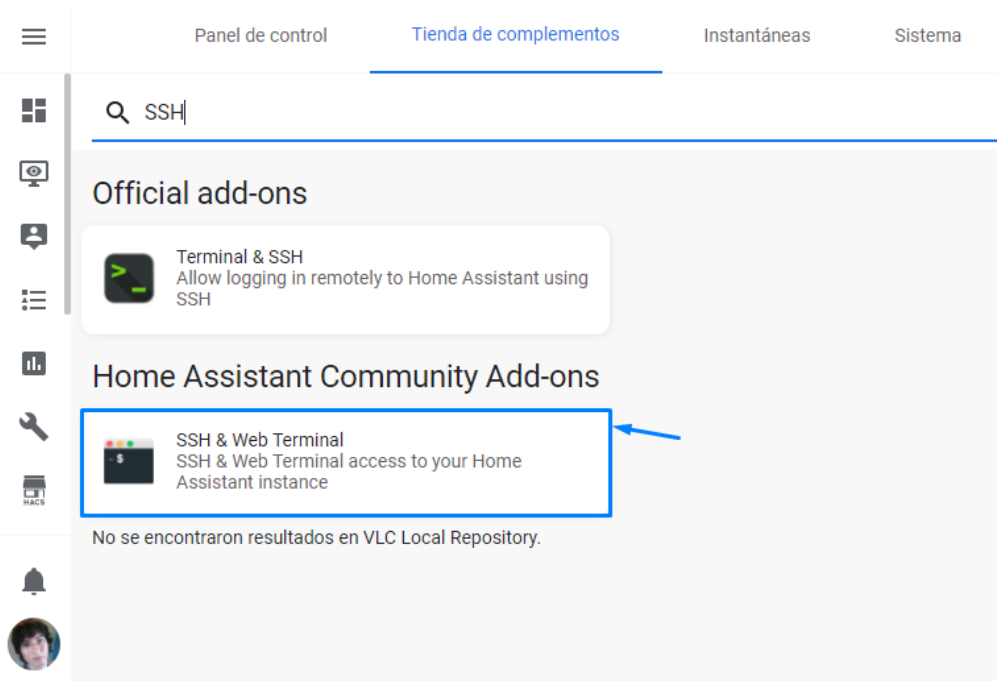

Figura 10: Instalación del add-on SSH web client
Fuente: Elaboración propia.

Al finalizar, en la pestaña Configuración se agregan las credenciales para tener acceso a una sesión SSH. Es probable que deba cambiar el puerto a 23 en caso de que existan conflictos. Una vez configurado se inicia el servicio.

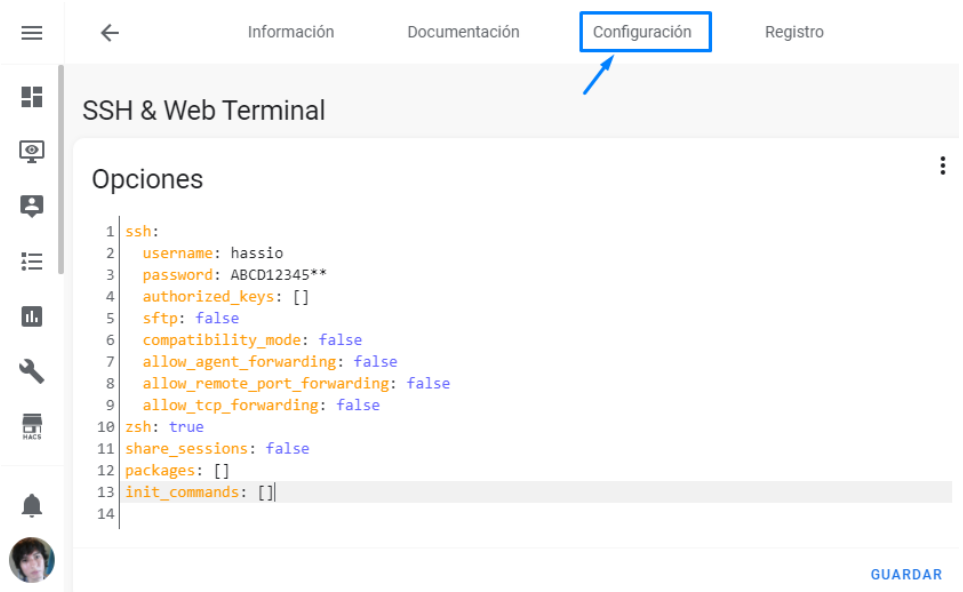

Figura 11: Configuración del add-on SSH web client
Fuente: Elaboración propia.

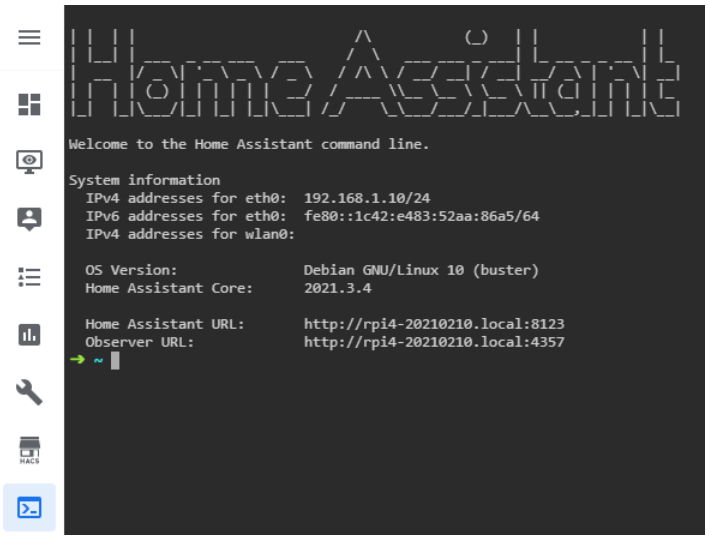

Figura 12: Add-on SSH web client
Fuente: Elaboración propia.

2.5.3.2. File Editor

Este complemento permite la edición de los archivos de configuración del sistema domótico. Su instalación es similar a SSH & Web Terminal. Se realiza mediante el supervisor y se busca mediante el nombre File Editor. Al finalizar la instalación se inicia el servicio y tendrá acceso a una interfaz gráfica de edición de archivos de texto.

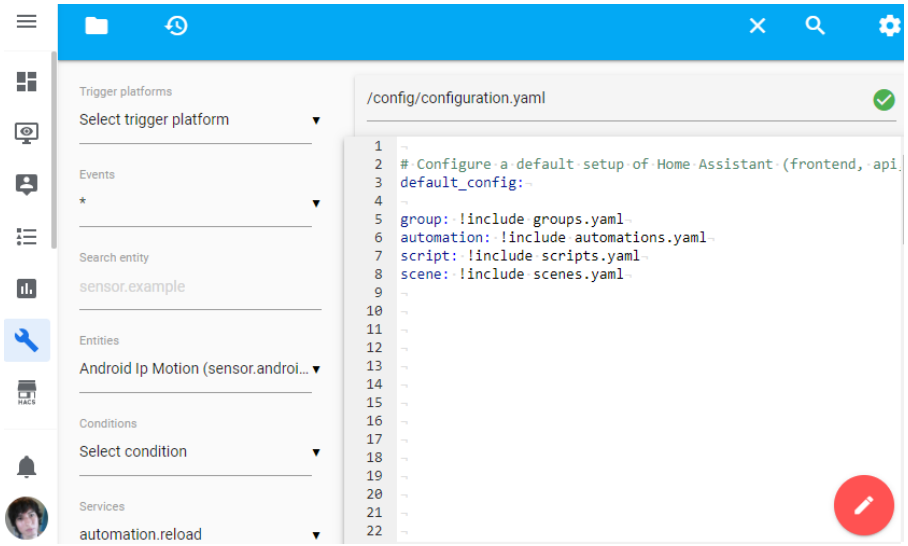

Figura 13: Add-on File Editor
Fuente: Elaboración propia.

2.5.3.3. HACS

Como su sitio oficial lo anuncia es una poderosa interfaz de usuario que facilita la gestión de integraciones que no se encuentran en el Supervisor de Home Assistant, además proporciona otras funcionalidades, tales como: temas, plugins, scripts en python y otras cosas más que lo convierten en un complemento perfecto para mejorar el sistema domótico.

Existen distintas maneras para instalarlo, sin embargo, se optó por la más sencilla, debido a su simplicidad, esto es mediante el complemento SSH & Web Terminal. Al adquirir una sesión se debe dirigir al directorio /config y ejecutar el comando propuesto en la Figura 14.

```
➜ ~ cd config
➜ config wget -q -O - https://hacs.xyz/install | bash -
```

Figura 14: Comando para la instalación del add-on HACS
Fuente: Elaboración propia.

Una particularidad de HACS es que se instala como una integración, por lo tanto, luego de finalizada su instalación se debe dirigir a la Configuración de Home Assistant, Integraciones y Añadir Integración, buscar por el nombre HACS agregarlo. Entonces el asistente solicita que se inicie sesión en una cuenta de GitHub para que se agregue el token de acceso. Concluido lo anterior se debe esperar unos minutos antes de poder usar HACS, puesto que,

el mismo se encuentra actualizando en segundo plano la lista de repositorios que utiliza para realizar su trabajo.

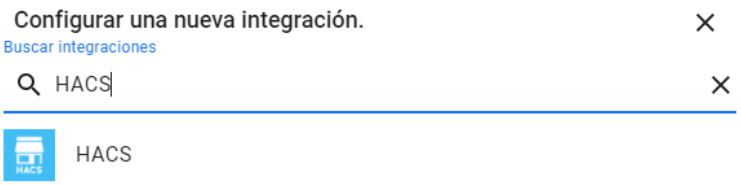

Figura 15: Integración de HACS en Home Assistant

Fuente: Elaboración propia.

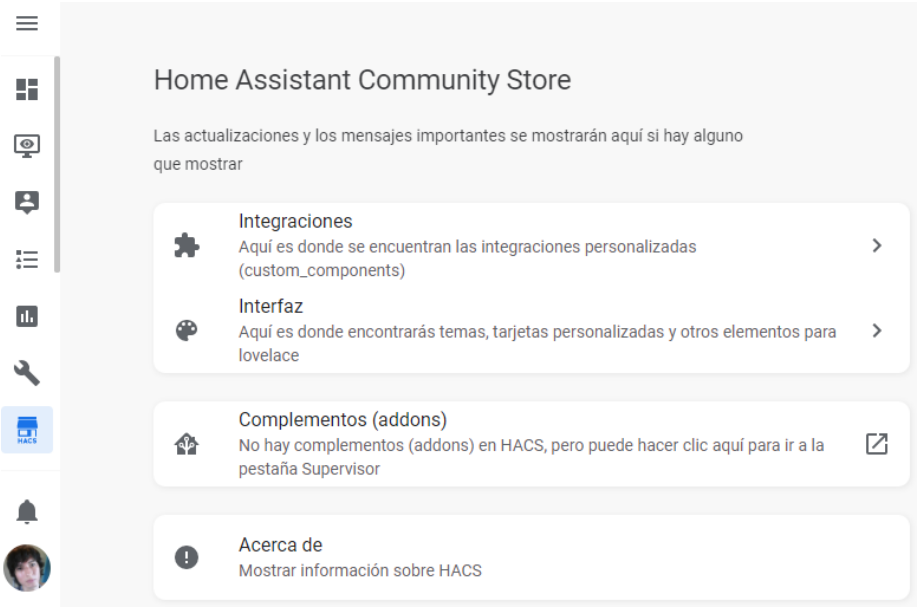

Figura 16: HACS

Fuente: Elaboración propia.

2.5.4. Integración de las cámaras en Home Assistant

Debido a la heterogeneidad de los dispositivos, no existe una integración genérica para todos ellos, es decir, cada uno a pesar de poseer la misma funcionalidad se integran de distintas maneras.

2.5.4.1. Integración de la cámara TAPO C200

Se inicia usando el complemento HACS en Home Assistant. En su sección de Integraciones, Explore & Add Repositories, se busca por el nombre TAPO y se procede a instalar.

Luego en la Configuración de Home Assistant, en Integraciones, Añadir Integración, se busca nuevamente por el nombre TAPO. El asistente solicita la IP de la cámara además de las credenciales de acceso a la misma. Para obtener estas credenciales es necesario crear una cuenta mediante la aplicación oficial de TAPO, en ajustes avanzados, tal como se muestra en la Figura 17.

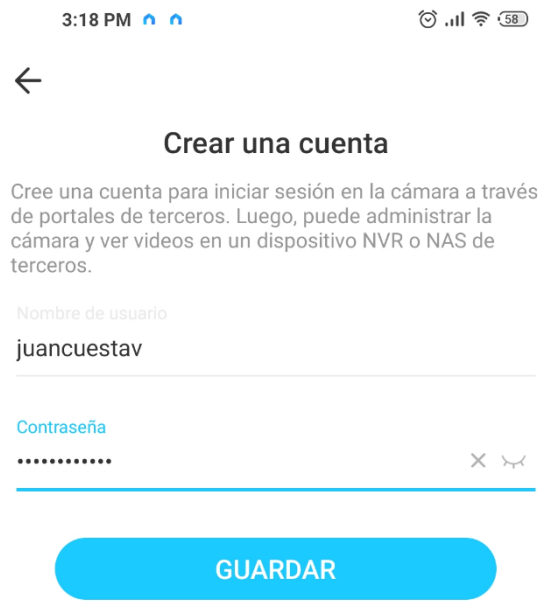

Figura 17: Generación de credenciales en la aplicación TAPO.

Fuente: Elaboración propia.

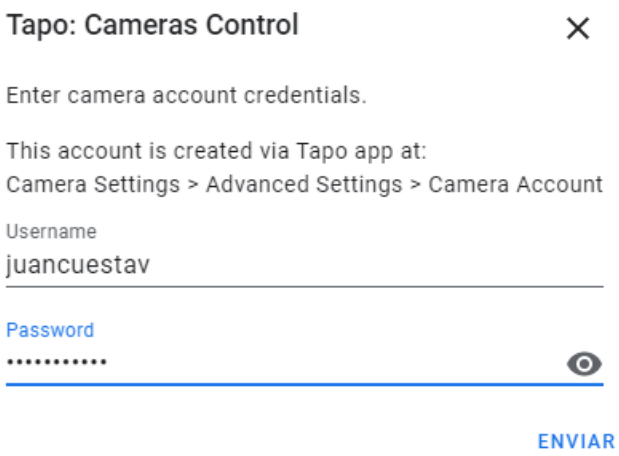

Figura 18: Ingreso de credenciales en Integración de la cámara TAPO.

Fuente: Elaboración propia.

En Home Assistant habilite las opciones Enable motion sensor y Automatically synchronise time. La opción Use Stream from Home Assistant no se habilita,

pues agrega una latencia de aproximadamente dos segundos, afectando la experiencia de usuario.

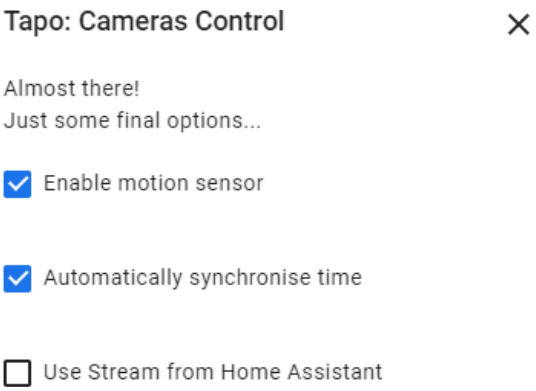

Figura 19: Opciones en la configuración de la cámara TAPO.
Fuente: Elaboración propia.

Se indica el área que cubre este dispositivo.

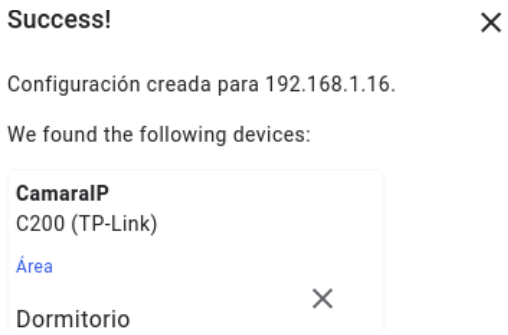

Figura 20: Definición del área que cubre la cámara TAPO.
Fuente: Elaboración propia.

La integración completa debe verse similar a la Figura 21, con dos entidades: una para el flujo de video en alta definición identificado por camera.camara_ip_hd y otro para el de baja calidad con el nombre camera.camara_ip_sd. La entidad binary_sensor.camara_ip_motion se activa una vez ha sido detectado un movimiento en el campo de visión del dispositivo.

Figura 21: Integración de la cámara TAPO.
Fuente: Elaboración propia.

Para el caso de visualización de video en tiempo real se usó una tarjeta personalizada de Lovelace cuyo código es proporcionado a continuación.

```
type: picture-glance
title: Dormitorio
camera_image: camera.camara_ip_hd
camera_view: live
entities:
  - entity: camera.camara_ip_hd
    icon: 'mdi:arrow-left-drop-circle-outline'
    tap_action:
      action: call-service
      service: tapo_control.ptz
      service_data:
        entity_id: camera.camara_ip_hd
        pan: LEFT
  - entity: camera.camara_ip_hd
    icon: 'mdi:arrow-up-drop-circle-outline'
    tap_action:
      action: call-service
      service: tapo_control.ptz
      service_data:
        entity_id: camera.camara_ip_hd
        tilt: UP
  - entity: camera.camara_ip_hd
    icon: 'mdi:arrow-down-drop-circle-outline'
    tap_action:
      action: call-service
      service: tapo_control.ptz
      service_data:
        entity_id: camera.camara_ip_hd
        tilt: DOWN
  - entity: camera.camara_ip_hd
    icon: 'mdi:arrow-right-drop-circle-outline'
    tap_action:
      action: call-service
      service: tapo_control.ptz
      service_data:
        entity_id: camera.camara_ip_hd
        pan: RIGHT
  - entity: camera.camara_ip_hd
    icon: 'mdi:eye-outline'
    tap_action:
      action: call-service
      service: tapo_control.set_privacy_mode
      service_data:
        entity_id: camera.camara_ip_hd
        privacy_mode: 'off'
  - entity: camera.camara_ip_hd
    icon: 'mdi:eye-off-outline'
    tap_action:
      action: call-service
      service: tapo_control.set_privacy_mode
      service_data:
        entity_id: camera.camara_ip_hd
        privacy_mode: 'on'
  - entity: camera.camara_ip_hd
```

```
    icon: 'mdi:power'
    tap_action:
      action: call-service
      service: tapo_control.reboot
      service_data:
        entity_id: camera.camara_ip_hd
hold_action:
  action: more-info
```

Es importante recordar que este dispositivo tiene función PTZ por lo que la integración la aprovecha muy bien.

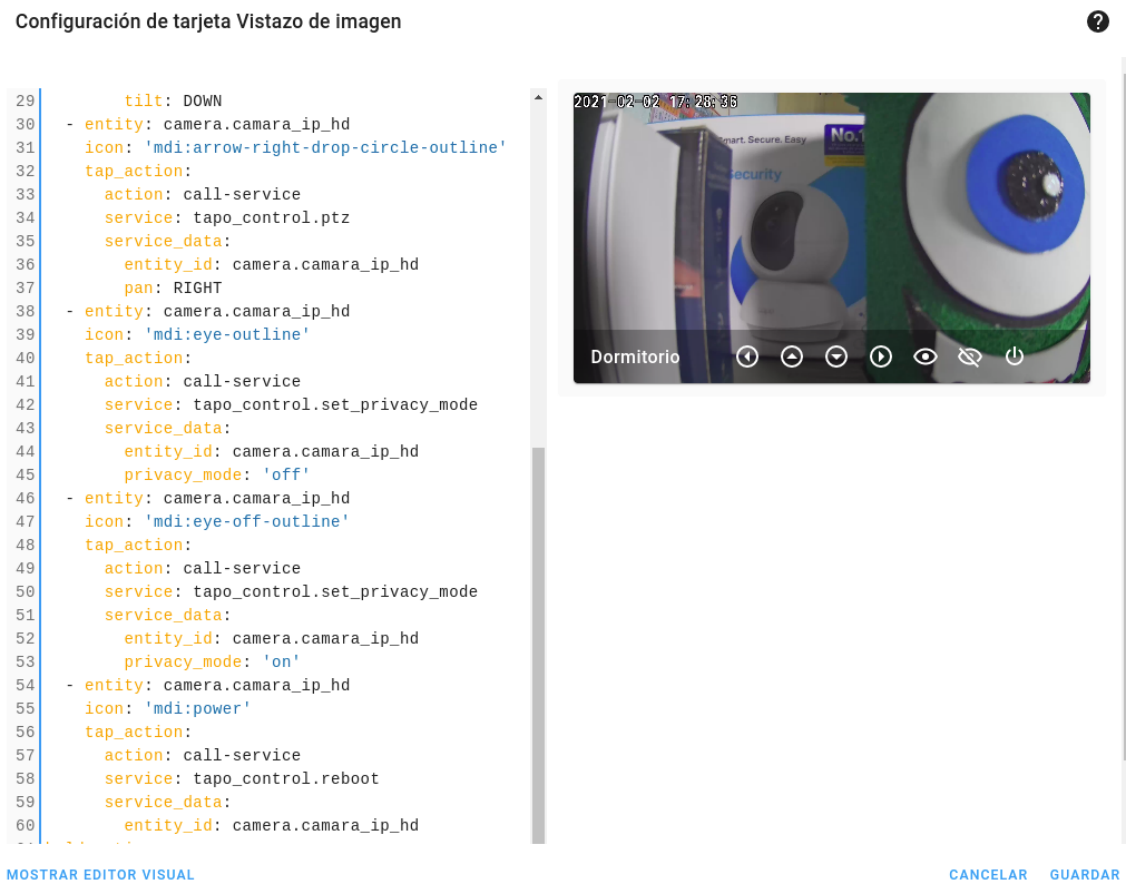

Figura 22: Tarjeta Lovelace para la cámara TAPO.

Fuente: Elaboración propia.

Además del complemento TAPO existe una solución que reduce significativamente la latencia por medio del módulo RTSP2WebRTC con la ventaja de acceso al audio pero sin la interacción PTZ de la cámara.

2.5.4.2. Integración de la cámara analógica con su DVR Hikvision

Los pasos descritos funcionan para cualquier DVR con soporte ONVIF. Lo primero es habilitar los protocolos de integración a utilizar, en este caso el dispositivo cuenta con dos: ONVIF e ISAPI.

Figura 23: Habilitar protocolos de integración en DVR.

Fuente: Elaboración propia.

Después se procede a crear un usuario multimedia por medio del cual se procede a realizar la integración en Home Assistant.

Figura 24: Creación de usuario multimedia en DVR.

Fuente: Elaboración propia.

Finalmente se habilita el protocolo RTSP para visualizar el streaming de video de las cámaras conectadas al DVR.

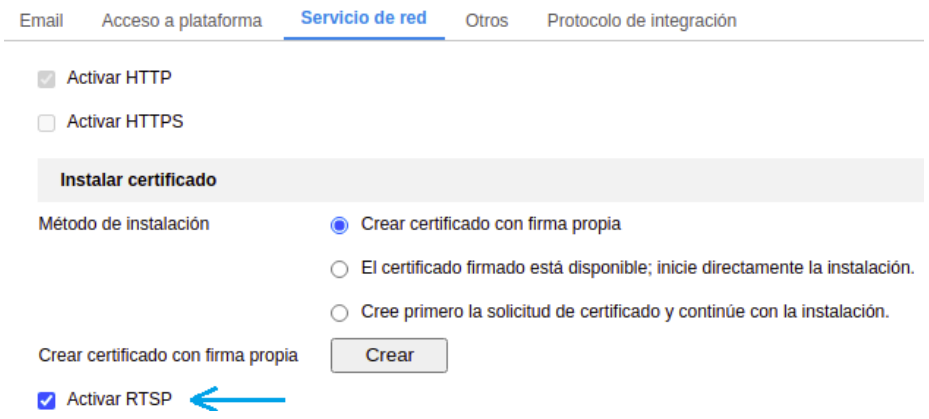

Figura 25: Habilitar protocolo RTSP en DVR.

Fuente: Elaboración propia.

En la Configuración de Home Assistant, en Integraciones se procede a añadir una nueva integración del tipo ONVIF. El asistente busca en la red los dispositivos y, una vez detectado, se solicitan las credenciales de acceso que corresponden al usuario multimedia previamente creado.

Figura 26: Dispositivo ONVIF detectado.

Fuente: Elaboración propia.

La manera más sencilla y adecuada cuando se utilizan las integraciones por ONVIF es mediante la entidad de cámara proporcionada por la integración del DVR, de ésta manera se logra aprovecha el reproductor que facilita la tarea de pausar, avanzar o retroceder la transmisión de video. La tarjeta Lovelace se presenta a continuación.

Figura 27: Tarjeta Lovelace DVR Onvif.

Fuente: Elaboración propia.

Para la obtención del evento de detección de movimiento se agregan las siguientes líneas en el archivo configuration.yaml especificando la dirección IP del DVR y las credenciales utilizadas para iniciar sesión en su interfaz web.

```
binary_sensor:
  - platform: hikvision
    host: 192.168.1.40
    username: user
    password: pass
```

Si bien es cierto la integración nativa está funcional se comprobó que tiene una latencia aproximada de 2 segundos por lo que se presenta una alternativa con MotionEye que lo reduce significativamente. Se debe configurar la resolución y los cuadros por segundos de cada cámara.

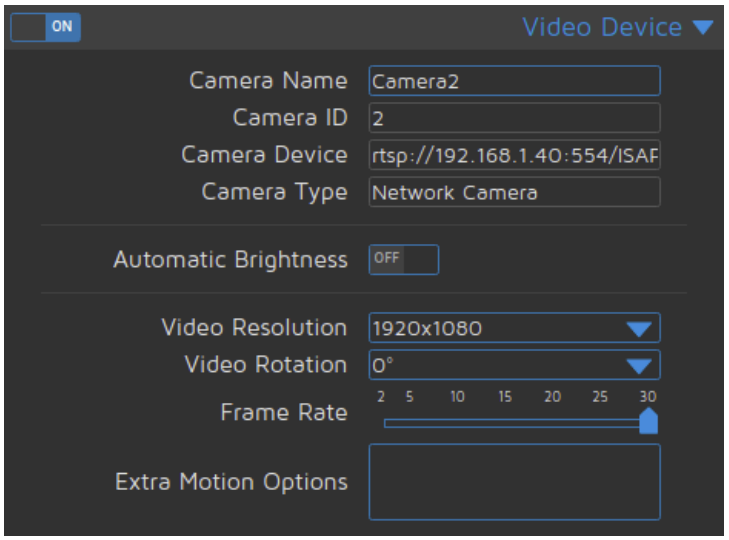

Figura 28: Integración de DVR mediante MotionEye.

Fuente: Elaboración propia.

2.5.4.3. Integración de la placa ESP32Cam

Se empleó el firmware desarrollado en ESPHome, puesto que, proporciona una perfecta compatibilidad con el sistema domótico. Este dispositivo cuenta únicamente con el módulo de cámara OV2640, por lo tanto, para que esté al nivel de las demás cámaras se le agregó un sensor PIR para la detección de movimiento, un pulsador y un buzzer, los mismos que fueron integrados en Home Assistant.

El código de ESPHome utilizado se proporciona a continuación.

```
substitutions:
  hostname: 'integracion_ha'
  ssid: 'RED_Wifi'
  password: '12345'
esphome:
  name: $hostname
  platform: ESP32
  board: esp32dev
wifi:
  ssid: $ssid
  password: $password
  fast_connect: True
```

```
api:
  reboot_timeout: 0s
ota:
logger:

# ESP32-CAM
esp32_camera:
  external_clock:
    pin: GPIO0
    frequency: 20MHz
  i2c_pins:
    sda: GPIO26
    scl: GPIO27
  data_pins: [GPIO5, GPIO18, GPIO19, GPIO21, GPIO36, GPIO39,
GPIO34, GPIO35]
  vsync_pin: GPIO25
  href_pin: GPIO23
  pixel_clock_pin: GPIO22
  power_down_pin: GPIO32
  idle_framerate: 0.05 fps
  vertical_flip: False
  horizontal_mirror: True
  name: esphome_esp32cam

web_server:
  port: 80

# Luz LED
output:
  - platform: gpio
    pin: GPIO4
    id: gpio_4
light:
  - platform: binary
    output: gpio_4
    name: $hostname light

# Sensores
binary_sensor:
  - platform: gpio
    pin: GPIO13
    name: "PIR Sensor"
    device_class: motion
  - platform: gpio
    pin:
      number: GPIO15
      inverted: True
    name: "Timbre Sensor"
    device_class: sound
```

El prototipo se muestra en la siguiente figura.

Figura 29: Módulo ESP32Cam mejorado.
Fuente: Elaboración propia.

2.5.4.4. Integración del smartphone Android IP Webcam

Para este dispositivo en particular fue necesario instalarle una aplicación llamada IP Webcam que puede ser localizada en la Play Store. Se optó por esta aplicación, debido a que, está respaldada por los desarrolladores de Home Assistant quienes le proporcionaron su propia etiqueta de configuración, denominada android_ip_webcam, para su respectiva integración.

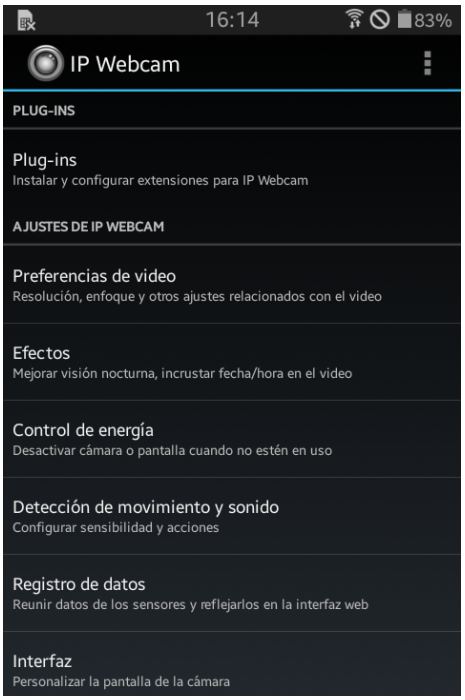

Figura 30: Aplicación móvil IP Webcam.
Fuente: Elaboración propia.

Esta aplicación genera un pequeño servidor web que permite administrar la configuración de la cámara, además de la misma aplicación. Además, ofrece ONVIF como protocolo de integración, RTSP para flujo de video en tiempo real, detección de movimiento, y muchas otras funciones que pueden, gracias a la integración ser gestionadas por Home Assistant. La Figura 31 muestra la aplicación en funcionamiento.

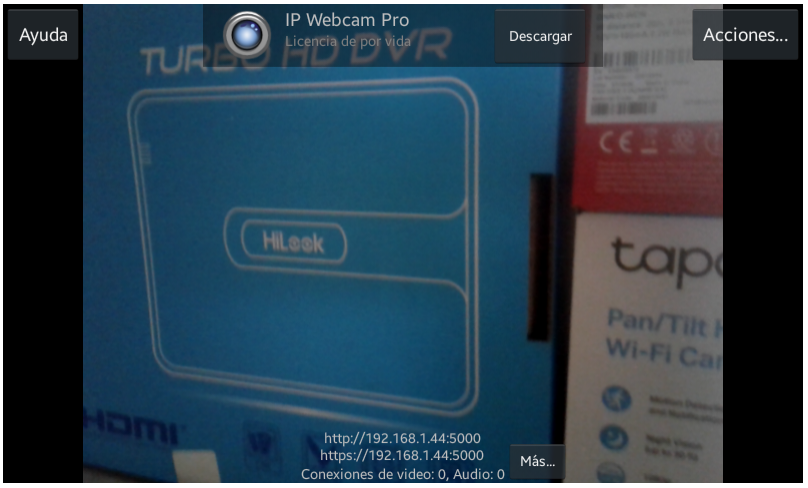

Figura 31: Aplicación móvil IP Webcam, en funcionamiento.
Fuente: Elaboración propia.

El código utilizado para la integración y que debe ser agregado en el archivo configuration.yaml, se muestra a continuación.

```
android_ip_webcam:
  - host: 192.168.1.44
    port: 5000
    username: juancuestav
    password: ABCD12345**
    sensors:
      - audio_connections
      - battery_level
      - battery_temp
      - battery_voltage
      - light
      - motion
      - pressure
      - proximity
      - sound
      - video_connections
    switches:
      - exposure_lock
      - ffc
      - focus
      - gps_active
      - motion_detect
      - night_vision
      - overlay
```

```
    - torch
    - whitebalance_lock
    - video_recording
```

La representación Lovelace de la cámara se muestra en la Figura 32 que además de proporcionar el video en tiempo real también se pueden definir controles, tales como el intercambio entre la cámara frontal o trasera o, el encendido y apagado del flash. El espacio de la tarjeta es limitado por lo que se sugiere definir los sensores más relevantes.

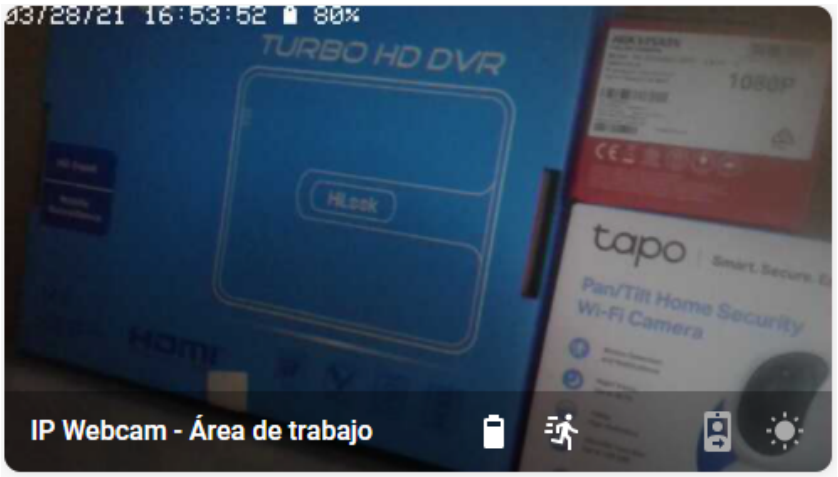

Figura 32: Tarjeta Lovelace para Android IP Webcam.
Fuente: Elaboración propia.

El código para crear la tarjeta de la Figura 32 se presenta a continuación.

```
type: picture-glance
title: IP Webcam - Área de trabajo
entities:
  - entity: sensor.ip_webcam_battery_level
  - entity: sensor.android_motion
  - entity: switch.ip_webcam_front_facing_camera
  - entity: switch.ip_webcam_torch
camera_view: live
aspect_ratio: 1920x1080
camera_image: camera.ip_webcam
```

La Figura 33 muestra algunos de los múltiples sensores detectados en el Smartphone, los mismos que pueden ser manipulados desde la interfaz del asistente domótico.

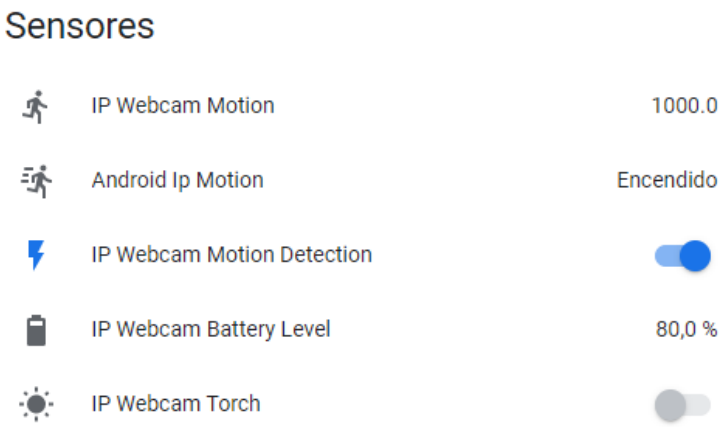

Figura 33: Tarjeta Lovelace con sensores del dispositivo Android IP Webcam integrados en Home Assistant.
Fuente: Elaboración propia.

El código para crear la tarjeta de la Figura 33 se presenta a continuación.

```
type: entities
entities:
  - entity: sensor.ip_webcam_motion
  - entity: sensor.android_motion
  - entity: switch.ip_webcam_motion_detection
  - entity: sensor.ip_webcam_battery_level
  - entity: switch.ip_webcam_torch
  - entity: sensor.ip_webcam_audio_connections
  - entity: sensor.ip_webcam_battery_temperature
  - entity: sensor.ip_webcam_battery_voltage
  - entity: switch.ip_webcam_exposure_lock
  - entity: switch.ip_webcam_focus
  - entity: switch.ip_webcam_front_facing_camera
  - entity: switch.ip_webcam_gps_active
  - entity: switch.ip_webcam_night_vision
  - entity: switch.ip_webcam_overlay
  - entity: sensor.ip_webcam_proximity
  - entity: sensor.ip_webcam_video_connections
  - entity: switch.ip_webcam_video_recording
  - entity: switch.ip_webcam_white_balance_lock
state_color: true
title: Sensores
show_header_toggle: false
```

2.5.5. Integración de servicios en Home Assistant

Con el propósito de aplicar el caso de estudio planteado en la sección 2.4, se integraron dos servicios. Es importante mencionar que ambos se encuentran funcionando dentro de la red de área local en un equipo adicional a la Raspberry Pi 4, por lo que prescindimos de la nube aportando seguridad a los datos. El equipo propuesto tiene las siguientes características: sistema operativo Debian 10, procesador Core i5 8va Gen, 4GB RAM y 50GB de almacenamiento.

La placa Raspberry Pi 4, a pesar de ser la más potente de su tipo a la fecha, aún no cuenta con la capacidad de realizar la tarea de reconocimiento facial y TTS a una velocidad que no afecte la experiencia de usuario por lo que se planteó el uso de un equipo adicional para que se encargue de su procesamiento.

Ambos servicios requieren de Docker.

2.5.5.1. Facebox

Aplicar algoritmos de reconocimiento facial, capaces de aprovechar los dispositivos con cámaras integrados en el asistente domótico, brinda la posibilidad de establecer infinidad de escenarios y aplicar reglas de acuerdo a cada caso. Para cumplir con ese propósito se implementó Facebox.

Lo primero que se debe hacer para utilizarlo es crear una cuenta en el sitio oficial de Machinebox.io y, una vez iniciada la sesión dirigirse a la sección MachineBox, ClasificationBox. Ahí se presenta una llave, la cual, debe ser utilizada cada vez que inicia el servicio de Facebox.

El comando para ejecutarlo se muestra a continuación.

```
docker run -p 8081:8080 -e "MB_KEY=$MB_KEY" machinebox/facebox
```

Puede acceder mediante el puerto 8081. La Figura 34 muestra la interfaz web del servicio en ejecución.

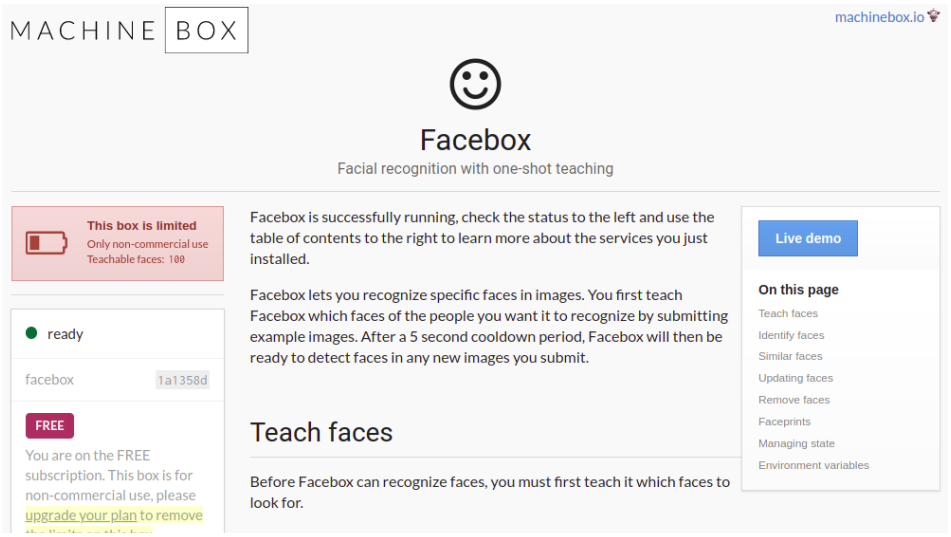

Figura 34: Servicio de reconocimiento facial, Facebox.
Fuente: Elaboración propia.

2.5.5.2. Mozilla TTS

Desde el punto de vista de la domótica es común la interacción entre los habitantes de la vivienda y el sistema. Sobre esta premisa se planteó el uso de un conversor de texto a voz para que de acuerdo a las reglas programadas notifique por medio de un altavoz su situación actual.

En el archivo configuration.yaml existe una sección para definir el TTS a usar, por defecto viene con el de Google, pero éste requiere de acceso a Internet, lo cual, se pretende prescindir por motivos de seguridad de los datos. Entonces se reemplaza por el servicio marytts que trabaja en el puerto 59125 y se especifica la dirección IP del host que ejecuta el servicio en la red de área local.

```
tts:
  - platform: marytts
    host: 192.168.1.34
```

En el mismo archivo se define la ejecución de un script que es el encargado de generar el archivo de audio a partir de texto, al mismo que se le asigna el pseudónimo génesis.

```
shell_command:
  genesis: bash genesis_generate_voice.sh {{text}}
```

Este archivo está ubicado en la ruta /config de Home Assistant con el nombre genesis_generate_voice.sh y se le debe asignar permisos de ejecución con chmod +x.

```
curl -G --output www/genesis.wav --data-urlencode "text=$1"
"http://192.168.1.34:59125/api/tts"
```

Para ejecutarlo se usa el comando.

```
docker run -it -p 59125:5002 synesthesiam/mozillatts:es
```

Y se accede a la interfaz web mediante el puerto 59125.

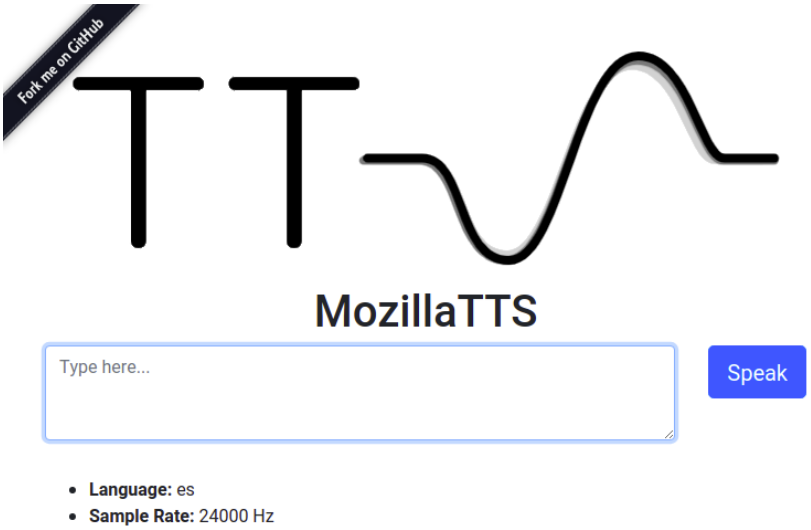

Figura 35: Servicio Mozilla TTS.
Fuente: Elaboración propia.

2.5.5.2.1. Local VLC

Mozilla TTS requiere de una entidad que sea capaz de reconocer los dispositivos de audio para que reproduzca los archivos generados. Local VLC es el candidato perfecto para esa labor pues basa su funcionamiento en el reproductor VLC con el beneficio de que se agrega como una entidad disponible para ser gestionada desde Home Assistant. Para su instalación se usa el Supervisor, Tienda de complementos. Una vez instalado el complemento se debe realizar la configuración de las credenciales de acceso. En la pestaña configuración tal como lo muestra la Figura 36.

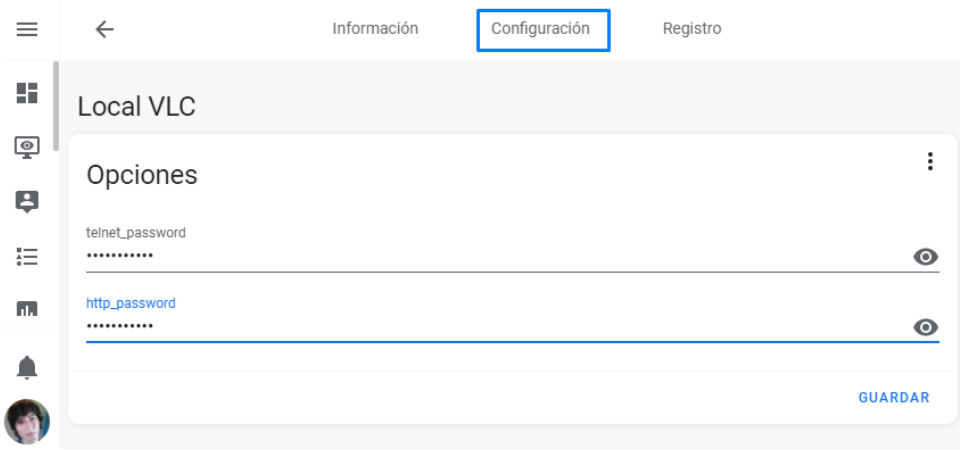

Figura 36: Configuración de Local VLC.
Fuente: Elaboración propia.

Se proporciona una interfaz web del reproductor al que se puede acceder por medio del puerto 9892, esto para verificar que su instalación y configuración son correctas, puesto que, su verdadera gestión será desde Home Assistant.

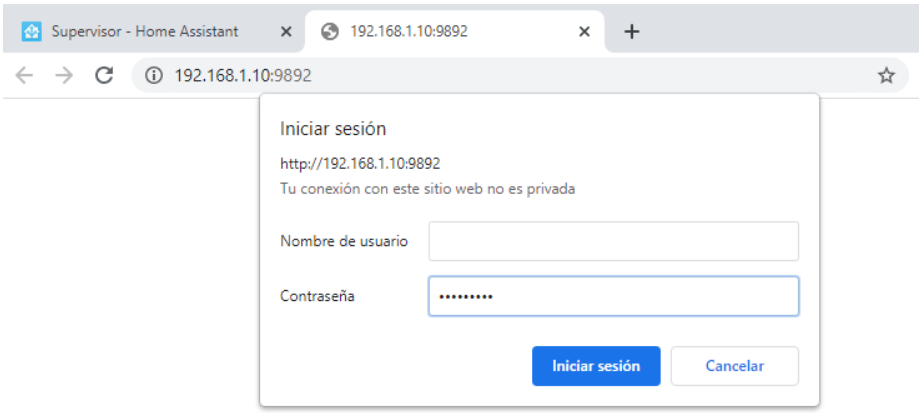

Figura 37: Interfaz web del complemento Local VLC.
Fuente: Elaboración propia.

En el archivo configuration.yaml se agregan las siguientes líneas para que Home Assistant pueda gestionarlo como entidad. Debido a que el audio es reproducido por la placa Raspberry Pi 4 se especifica la IP como localhost.

```
- platform: vlc_telnet
  name: Remote Player
  host: 'localhost'
  port: 4212
  password: ABCD12345**
```

El código para la tarjeta Lovelace de Local VLC se presenta a continuación.

```
type: media-control
entity: media_player.remote_player
```

Figura 38: Tarjeta Lovelace del add-on Local VLC.
Fuente: Elaboración propia.

En la Figura 39 se muestra la tarjeta Lovelace en funcionamiento. Se logra observar en la parte inferior un campo para escribir texto que será convertido a voz por Mozilla TTS.

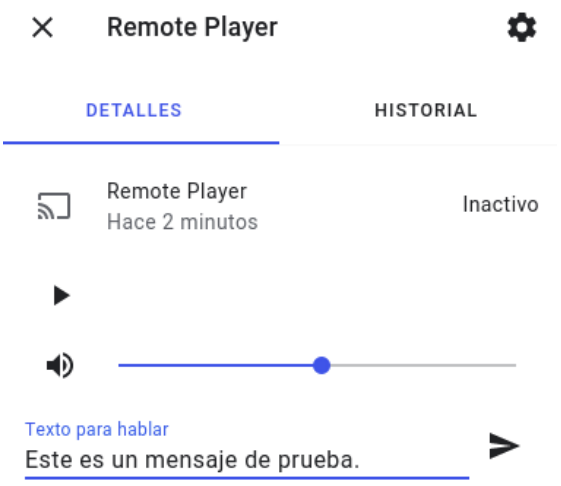

Figura 39: Interacción Local VLC con Mozilla TTS.
Fuente: Elaboración propia.

2.5.6. Acceso desde Internet mediante Zerotier

Esta herramienta ofrece acceso a la red local en donde se encuentra instalado el sistema domótico y sus integraciones de manera remota a través de Internet. Su funcionamiento es sencillo y consiste en crear, mediante su sitio web, una red a la cual se agregan todos los dispositivos a los que se desee tener acceso. Cada uno de estos dispositivos deben tener un cliente de Zerotier instalado, pues desde ahí se debe registrar un ID para ser agregado en la red. A continuación, se presentan los pasos de una manera más detallada.

Lo primero que se debe hacer es crear una cuenta en su sitio oficial y una vez se haya accedido se crea una red presionando el botón Create a Network. Luego se accede a la red y se agrega una nueva ruta; si existe otra se puede eliminar. Algo importante que se debe mencionar es que al tratarse de una red de clase C el tercer octeto debe ser diferente al de la red local, caso contrario no funcionará.

Figura 40: Agregar rutas en Zerotier.
Fuente: Elaboración propia.

Para agregar el primer dispositivo se usa el cliente Zerotier para Raspberry. El siguiente comando lo descarga y ejecuta automáticamente.

Figura 41: Instalación de cliente Zerotier en Raspberry
Fuente: Elaboración propia.

Se le otorga acceso a la red Zerotier mediante su Network ID, esto utilizando el software cliente llamado zerotier-cli.

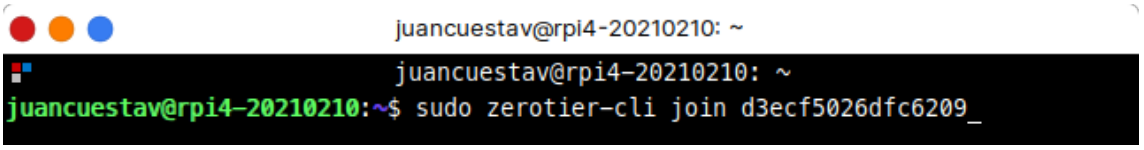

Figura 42: Agregar Raspberry a la red Zerotier.
Fuente: Elaboración propia.

Una vez ejecutado el comando anterior, en la cuenta de Zerotier aparece el nuevo dispositivo al que se le debe asignar un nombre para identificarlo, debido a que, únicamente se muestra una MAC ficticia en su representación.

Figura 43: Raspberry Pi dado de alta en la red Zerotier.
Fuente: Elaboración propia.

Otro dispositivo que se agregó a la red fue un Smartphone, el cual cumple el papel de cliente del servicio de Home Assistant. Los conceptos son los mismos con la diferencia de que ahora se debe instalar la aplicación Android correspondiente, la misma que está disponible en la Play Store. Para agregarlo se ingresa el Network ID en la aplicación, automáticamente aparece una llave en la barra de estado que indica que está conectado a la red.

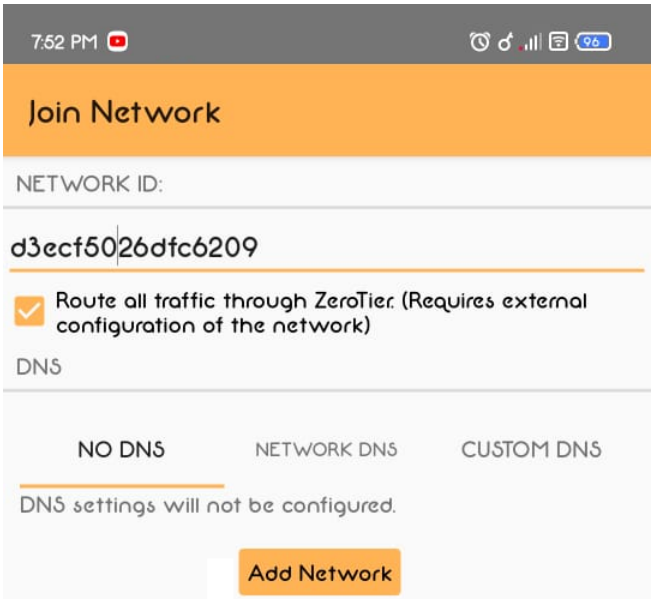

Figura 44: Cliente Android Zerotier.
Fuente: Elaboración propia.

De igual manera en la cuenta de Zerotier se le asigna un nombre identificativo.

Como punto final se accede al asistente domótico usando los datos móviles del Smartphone para verificar que efectivamente se puede acceder a través de Internet. La IP de acceso es la que le asignó Zerotier, es decir, 192.168.100.97 en el puerto 8123 de Home Assistant.

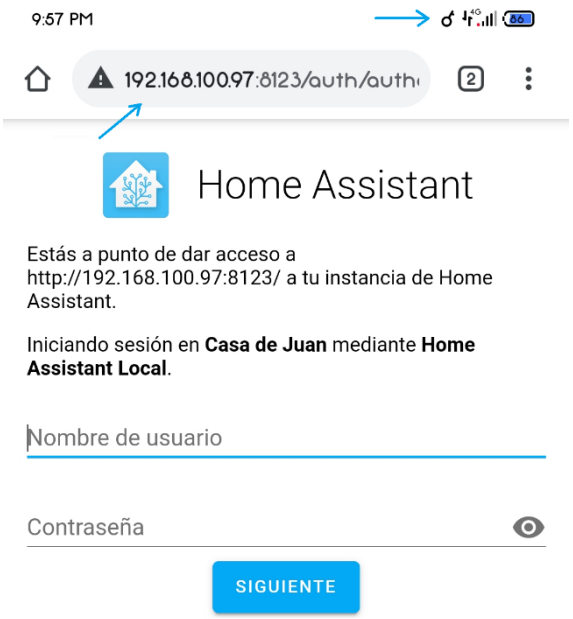

Figura 45: Acceso remoto a Home Assistant.
Fuente: Elaboración propia.

2.5.7. Construcción del escenario de automatización en Node-Red con API Home Assistant.

2.5.7.1. Instalación de Node-Red

Por simplicidad se realiza la instalación de Node-Red por medio del supervisor de Home Assistant. Para ello se dirige a Supervisor y Tienda de complementos en donde se lo busca y se instala.

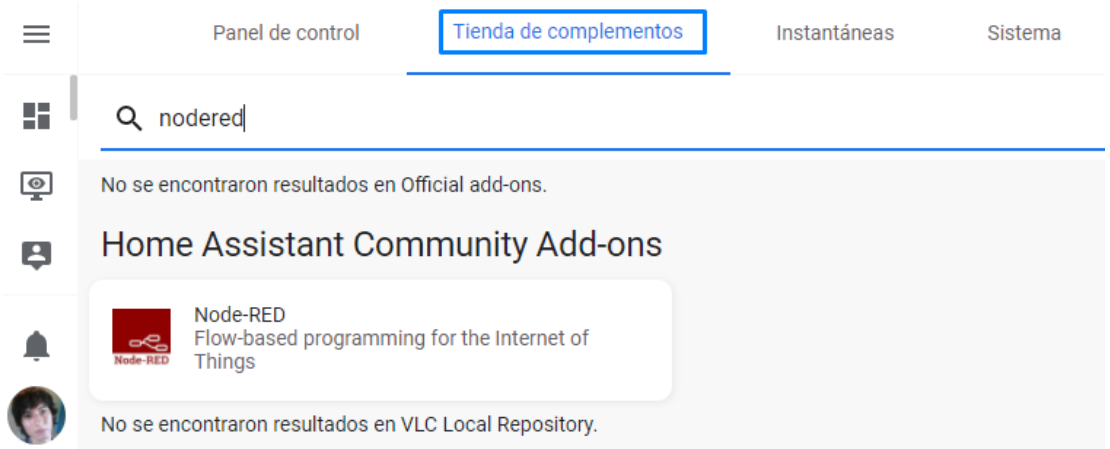

Figura 46: Instalación de Node-Red.
Fuente: Elaboración propia.

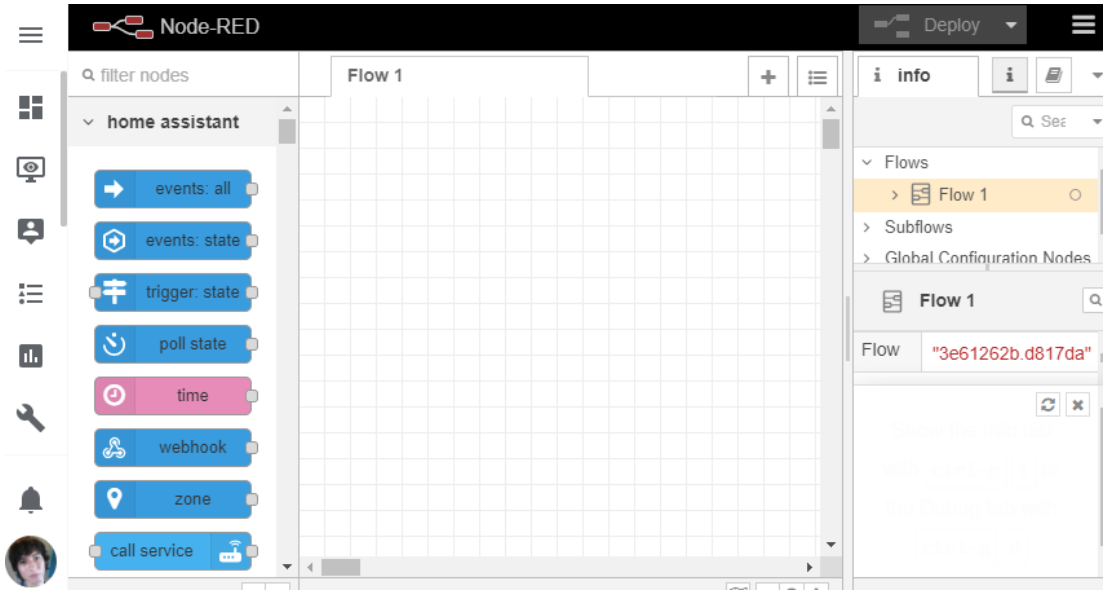

Figura 47: Node-Red.
Fuente: Elaboración propia.

2.5.7.2. Desarrollo del escenario

Se desarrolló una aplicación en Node-Red que permite la implementación del escenario de automatización por medio de flujos.

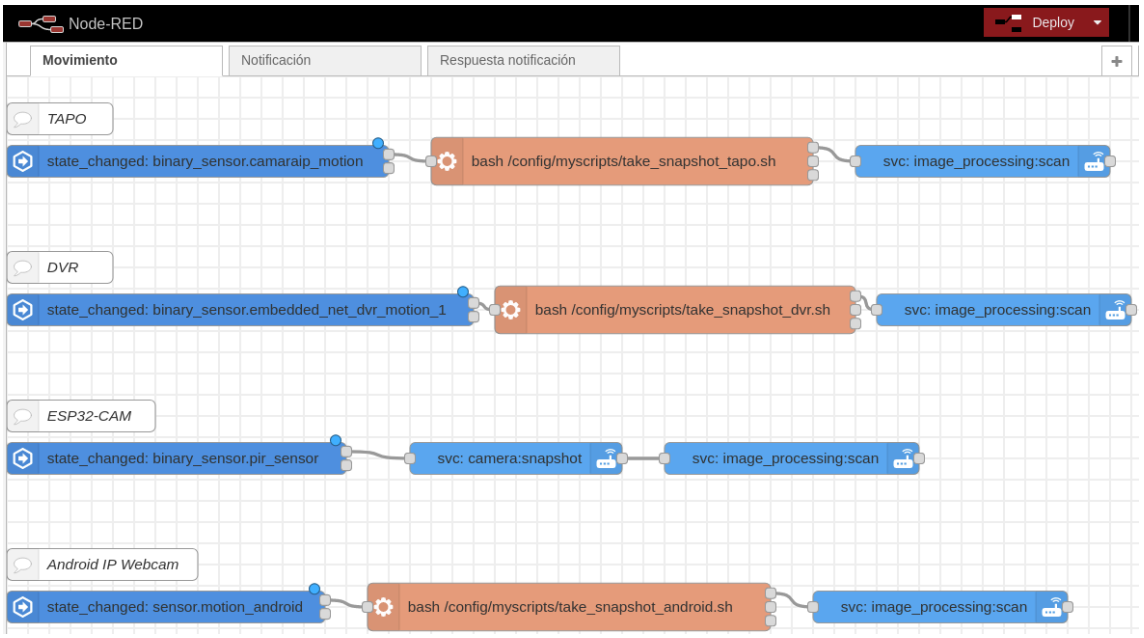

Figura 48: Aplicación Node-Red. Detección de movimiento.
Fuente: Elaboración propia.

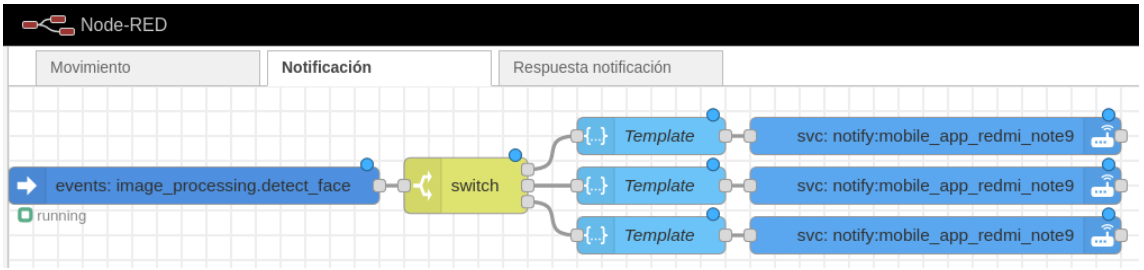

Figura 49: Aplicación Node-Red. Envío de notificación al propietario.

Fuente: Elaboración propia.

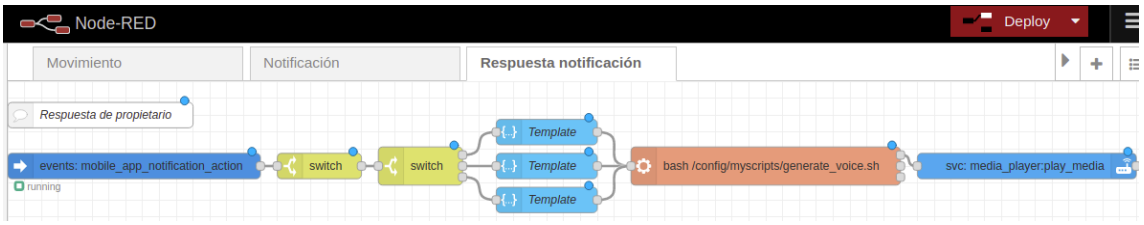

Figura 50: Aplicación Node-Red. Respuesta de notificación del propietario.

Fuente: Elaboración propia.

La aplicación se divide en tres secciones: la primera consiste en recibir la señal de detección de movimiento proporcionado por los sensores de las cámaras, luego se hace una captura de la imagen actual y se llama al servicio de image_processing.scan para que analice la imagen capturada. La segunda sección se activa cuando un rostro es detectado y, por medio de un nodo template se elabora una cadena de texto personalizada de acuerdo al rostro analizado. Sí se trata de uno desconocido se emite el mensaje “Una persona desconocida está en la entrada principal.” y en caso de que se trate de un rostro ya registrado se emite el mismo mensaje mencionando el nombre de la persona identificada. Luego de esto se comunica por medio de una notificación al smartphone personal del propietario con el mensaje ya mencionado anexando además la foto capturada y un botón de acción para decidir qué hacer en cada caso.

La tercera sección se activa una vez el propietario ejecute una acción desde la notificación recibida en su smartphone personal la cual indica que se puede abrir la puerta de la entrada principal. Si el propietario lo confirma se anuncia por medio del altavoz que la puerta principal ha sido abierta acto seguido el propietario recibe en su teléfono la confirmación de la acción.

3. CAPÍTULO III. EVALUACIÓN DEL PROTOTIPO

3.1. PLAN DE EVALUACIÓN

Antes de mencionar el plan de evaluación es necesario conocer algunos aspectos referentes a los recursos utilizados en esta investigación. Por ello se muestran a continuación las tablas con las características correspondientes.

La Tabla *1* muestra las características más relevantes de las cámaras integradas. Cabe recalcar que todas codifican la señal de video utilizando el códec H.264, el cual, es uno de los códecs soportados por la Raspberry Pi 4 que ejecuta el asistente domótico.

Tabla 1: Características de las cámaras integradas en Home Assistant.

Cámaras	Resolución	Frames	Protocolos utilizados
Cámara TAPO C200	1080p	15fps	RTSP (Foto y video).
Cámara analógica y DVR Hikvision	1080p	30fps	RTSP (Video), ISAPI (Foto).
Smartphone Android con software IP Webcam	5MP en fotos y 720p en video	30fps	RTSP (Foto), API de Home Assistant (Video).
ESP32-CAM	480p	15fps	API de Home Assistant (Foto y video).

Fuente: Elaboración propia.

La Tabla *2* muestra las características de los servidores. La placa Raspberry Pi 4 es la encargada de ejecutar Home Assistant mientras que la máquina virtual se centra en los servicios de Facebox y Mozilla TTS que requieren mayor capacidad de procesamiento.

Tabla 2: Características de los servidores.

Raspberry Pi 4	
Sistema Operativo:	Debian 10
Procesador:	ARM64
RAM:	4GB
Almacenamiento:	SSD 120GB
Máquina virtual	
Sistema Operativo:	Debian 10
Procesador:	Core i5 8va Gen
RAM:	4GB

Fuente: Elaboración propia.

La Tabla *3* muestra las características de los dispositivos clientes. Estos acceden a Home Assistant mediante la aplicación móvil o desde un navegador web.

Tabla 3: Características de los clientes.

Aplicación móvil	Navegador Web
Xiaomi Redmi Note 9 Android 10 3GB de RAM	Ubuntu 18.04 LTS

Fuente: Elaboración propia.

El plan de evaluación está basado en el escenario de automatización presentado en la Figura 3 que consiste en calcular el intervalo de tiempo (ms) ocupado en la ejecución de las tareas programadas para el cumplimiento del automatismo. Estas pruebas implican el uso de dos clientes: en local, dos dispositivos (Ubuntu 18.04 LTS y un Smartphone Android) y en remoto el mismo Smartphone, pero con acceso desde Internet.

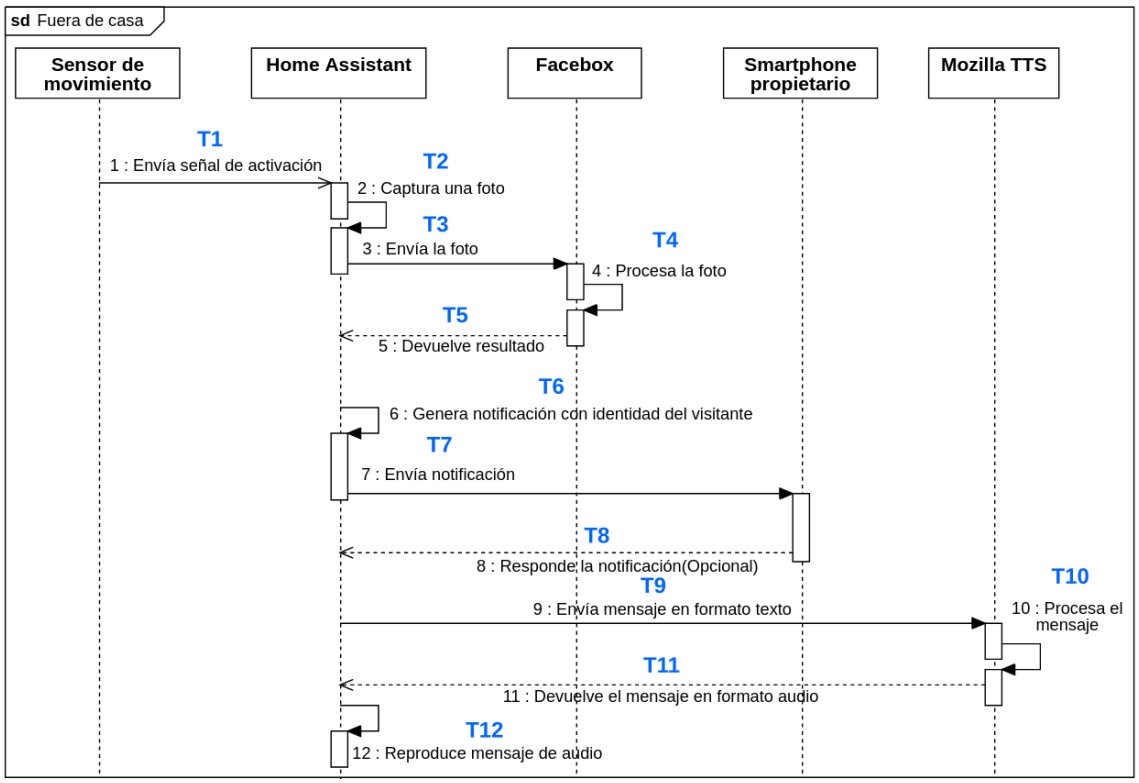

Figura 51: Tiempos que intervienen en el escenario de automatización.

Fuente: Elaboración propia.

Se identificaron cinco procesos.

1. Procesamiento de imágenes con Facebox que involucra los tiempos T2, T3, T4 y T5.
2. Generación del contenido de la notificación que involucra el tiempo T6.
3. Envío de notificación desde Home Assistant a Smartphone del propietario que involucra el tiempo T7.
4. Respuesta de notificación desde Smartphone del propietario a Home Assistant que involucra el tiempo T8.
5. Generación y reproducción de notificación de voz mediante Mozilla TTS que involucra los tiempos T9, T10, T11 y T12.

El tiempo T1 es omitido debido a que las cámaras no cuentan con la capacidad de ofrecer el tiempo de detección de la señal de movimiento (tiempo de origen) y únicamente se cuenta con el tiempo de recepción de la señal en el asistente domótico (tiempo de llegada).

3.1.1. Node-Red para la adquisición de los tiempos de procesamiento

Se toma como base el escenario de automatización desarrollado en Node-Red. El objetivo es poder cuantificar los tiempos que toma ejecutar cada proceso para elaborar tablas que permitan determinar la fiabilidad de las automatizaciones en el sistema domótico.

3.1.2. Wireshark y PCAP Remote para la adquisición de los tiempos de comunicación

Mediante la aplicación móvil PCAP Remote se logra capturar los paquetes e identificar el tiempo exacto en que es recibida la notificación en el Smartphone; este se considera el tiempo de llegada. El tiempo de origen se lo toma a partir de la aplicación en Node-Red, pues ahí se visualiza el tiempo en que la notificación es enviada. La diferencia de ambos da como resultado el tiempo ocupado en la comunicación. Es importante recalcar que ambos relojes están sincronizados.

Se utilizó Wireshark para visualizar los archivos emitidos por la aplicación PCAP Remote.

3.2. RESULTADOS DE LA EVALUACIÓN

Cada resultado surge a partir de una muestra de 100 valores capturados en cada uno de los procesos que forman parte del escenario.

3.2.1. Tiempo de procesamiento de imágenes con Facebox

Se toma como referencia el tiempo desde la captura de la imagen (T2), el envío de la misma al servidor que ejecuta Facebox (T3), el procesamiento (T4) y la devolución de la respuesta (T5).

Es importante mencionar que la forma en cómo cada dispositivo captura la imagen es diferente debido a las características de cada uno. Para la cámara Tapo se utilizó FFMPEG, para el DVR se utilizó su API proporcionada por el protocolo ISAPI, para el dispositivo Android IP se utilizó la API que ofrece la aplicación IP Webcam lo que da como resultado una imagen descargable y finalmente el ESP32-CAM obtiene la imagen por medio de la integración de Home Assistant en la llamada al servicio y camera.snapshot.

Tabla 4: Resumen de tiempo de procesamiento de imágenes con Facebox.

	Tiempo (ms)	T2	T3, T4, T5	Total (ms)
Tapo C200	**Máximo**	5,533	1,510	7,043
	Mínimo	2,253	1,418	3,671
	Promedio	3,450	1,442	4,891
DVR	**Máximo**	0,502	1,708	2,210
	Mínimo	0,318	1,397	1,715
	Promedio	0,379	1,457	1,836
Android IP Webcam	**Máximo**	7,410	2,771	10,181
	Mínimo	1,300	1,511	2,811
	Promedio	2,874	1,653	4,527
ESP32-CAM	**Máximo**	1,333	0,574	1,907
	Mínimo	0,019	0,249	0,268
	Promedio	0,182	0,313	0,495

Fuente: Elaboración propia.

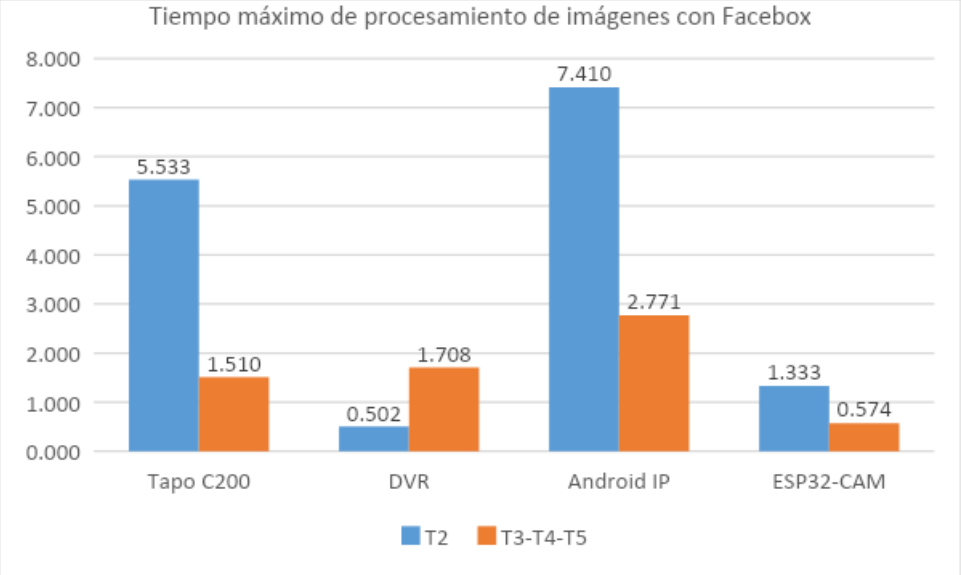

Figura 52: Tiempo máximo de procesamiento con Facebox.

Fuente: Elaboración propia.

El tiempo máximo de procesamiento de imágenes con Facebox da como resultado tiempos altos para la captura de la imagen siendo Android IP el dispositivo con el mayor tiempo igual a 7,410 ms, seguido por la cámara Tapo con 5,533 ms, luego el dispositivo ESP32-CAM con 1,333 ms y finalmente el DVR con 0,502 ms. Para el análisis, procesamiento y devolución de resultados se obtienen tiempos máximos de 2,771 ms para Android IP Webcam, 1,708 ms para el DVR, 1,510 ms para Tapo C200 y 0,574 ms para el ESP32-CAM.

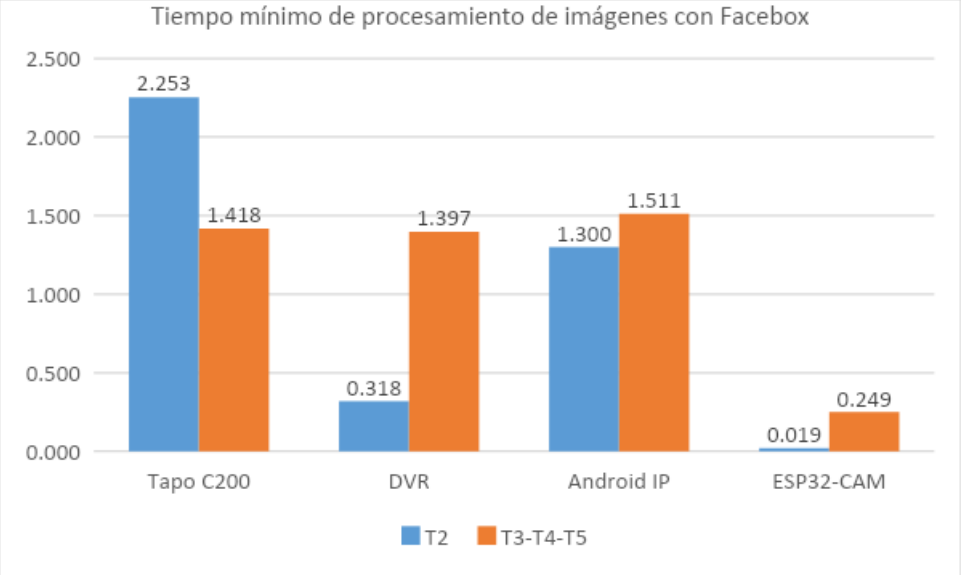

Figura 53: Tiempo mínimo de procesamiento de imágenes con Facebox.

Fuente: Elaboración propia.

El tiempo mínimo de procesamiento de imágenes con Facebox da como resultado tiempos bajos para la captura de la imagen siendo ESP32-CAM el dispositivo con el menor tiempo igual a 0,019 ms, seguido por el DVR con 0,318 ms, luego el dispositivo Android IP con 1,300 ms y finalmente la cámara Tapo con 2,253 ms. Para el análisis, procesamiento y devolución de resultados se obtienen tiempos mínimos de 0,249 ms para el ESP32-CAM, 1,397 ms para el DVR, 1,418 ms para Tapo C200 y 1,511 ms para Android IP Webcam.

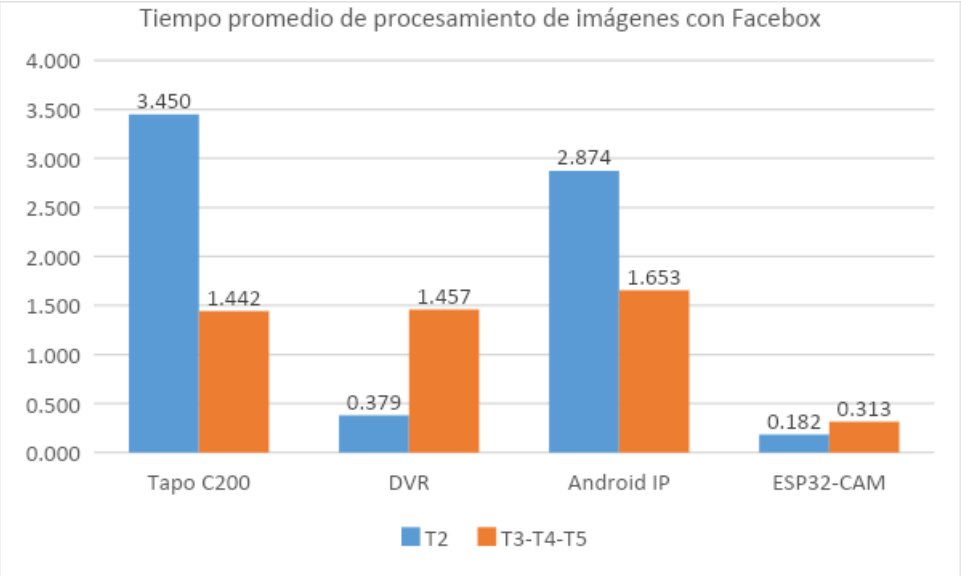

Figura 54: Tiempo promedio de procesamiento de imágenes con Facebox.

Fuente: Elaboración propia.

El tiempo promedio de procesamiento de imágenes con Facebox da como resultado las siguientes medias para la captura de la imagen siendo ESP32-CAM el dispositivo con el menor tiempo igual a 0,182 ms, seguido por el DVR con 0,379 ms, luego el dispositivo Android IP con 2,874 ms y finalmente la cámara Tapo con 3,450 ms. Para el análisis, procesamiento y devolución de resultados se obtienen los tiempos de 0,313 ms para el ESP32-CAM, 1,442 ms para Tapo C200, 1,457 ms para el DVR y 1,653 ms para Android IP Webcam.

3.2.2. Tiempo de generación del contenido de la notificación

Se considera el tiempo que ocupa en la generación del contenido de la notificación con el mensaje de texto personalizado (T6) el mismo que tiene la siguiente estructura: *“ ‘Invitado’. Lo espera en la entrada principal.”*, en donde *Invitado* es el nombre de la persona identificada.

Tabla 5: Resumen de tiempo de generación del contenido de notificación.

Tiempo	Total (ms)
Máximo	2,654
Mínimo	0,466
Promedio	854,04

Fuente: Elaboración propia.

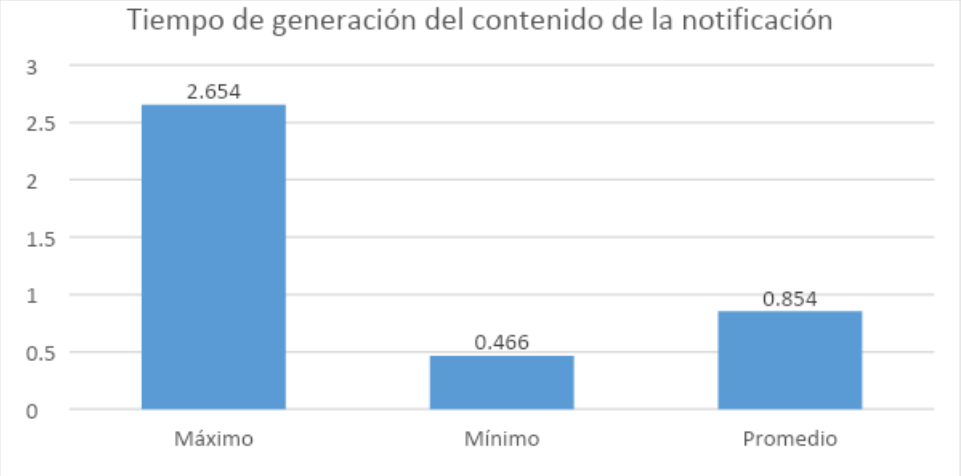

Figura 55: Resumen de tiempo de generación del contenido de notificación.

Fuente: Elaboración propia.

Para generar el mensaje personalizado se ocupa un tiempo promedio de 0,854 ms, un máximo de 2,654 ms y mínimo de 0,466 ms.

3.2.3. Latencia de notificaciones en local

Esta prueba se realizó en la misma red en donde está instalado el asistente domótico y sus integraciones utilizando como cliente un smartphone Android.

3.2.3.1. Latencia en envío desde Home Assistant hasta cliente Smartphone.

Es el tiempo ocupado en la comunicación entre Home Assistant y el cliente Smartphone mediante las notificaciones. El propósito es informar al propietario la presencia de una persona en la puerta de su casa por medio de un mensaje con la identidad del visitante y una foto.

Tabla 6: Resumen de latencia en comunicación HA – cliente, por medio de notificaciones.

Tiempo en envío de notificación	Total (ms)
Máximo	1,423
Mínimo	0,054
Promedio	0,749

Fuente: Elaboración propia.

3.2.3.2. Latencia en envío desde cliente Smartphone hasta Home Assistant.

Es el tiempo ocupado en la comunicación entre el cliente Smartphone y Home Assistant por medio de las notificaciones. Su propósito es permitir al propietario ejecutar acciones programadas mediante su confirmación en las notificaciones recibidas.

Tabla 7: Resumen de latencia en comunicación cliente - HA, por medio de notificaciones.

Tiempo en envío de notificación	Total (ms)
Máximo	0,986
Mínimo	0,194
Promedio	0,232

Fuente: Elaboración propia.

3.2.3.3. Comparación de latencia de notificaciones en local

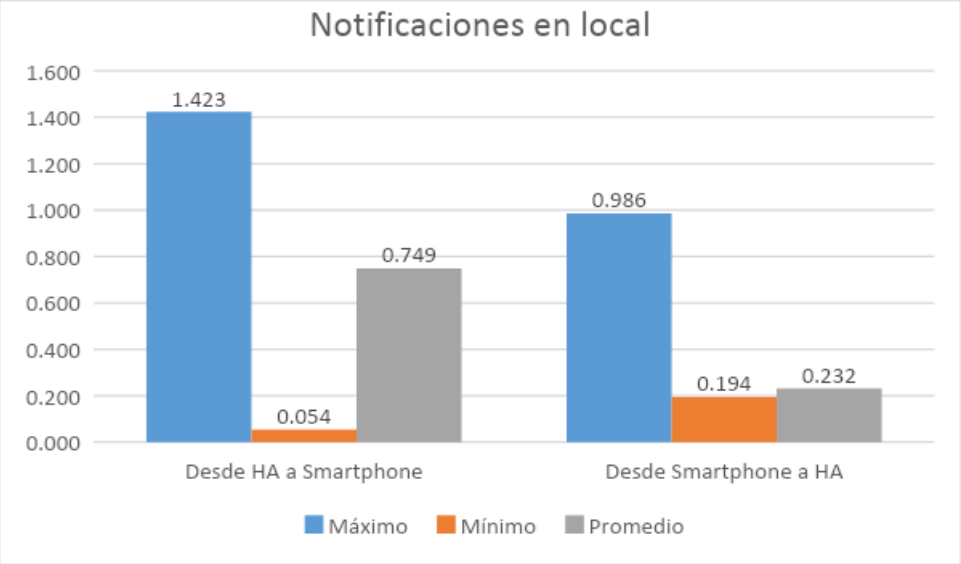

Figura 56: Resumen latencia de notificaciones en local

Fuente: Elaboración propia.

El resultado de las pruebas indica que en el envío de la notificación desde Home Assistant hasta el Smartphone hay un tiempo promedio de 0,749 ms, un máximo de 1,423 ms y un tiempo mínimo de 0,054 ms. Mientras que para la comunicación desde el Smartphone hasta Home Assistant hay un tiempo promedio 0,232 ms, máximo de 0,986 ms y mínimo de 0,194 ms.

3.2.4. Tiempo de generación y reproducción de notificación de voz mediante Mozilla TTS.

Se toma como referencia el tiempo de envío del mensaje en formato de texto (T9), procesamiento del mensaje (T10) y la devolución del mensaje en formato de audio (T11), así como su reproducción (T12). El texto tiene la siguiente estructura: *“ ‘Invitado’. Es un gusto recibirlo en mi hogar. Sea usted bienvenido.”*, en donde *Invitado* es el nombre de la persona reconocida por el sistema.

Tabla 8: Resumen de tiempo de generación y reproducción de notificación de voz mediante Mozilla TTS.

Tiempo	T9 – T10 – T11	T12	Total (ms)
Máximo	4,234	0,935	5,169
Mínimo	2,431	0,039	2,470
Promedio	3,344	0,101	3,445

Fuente: Elaboración propia.

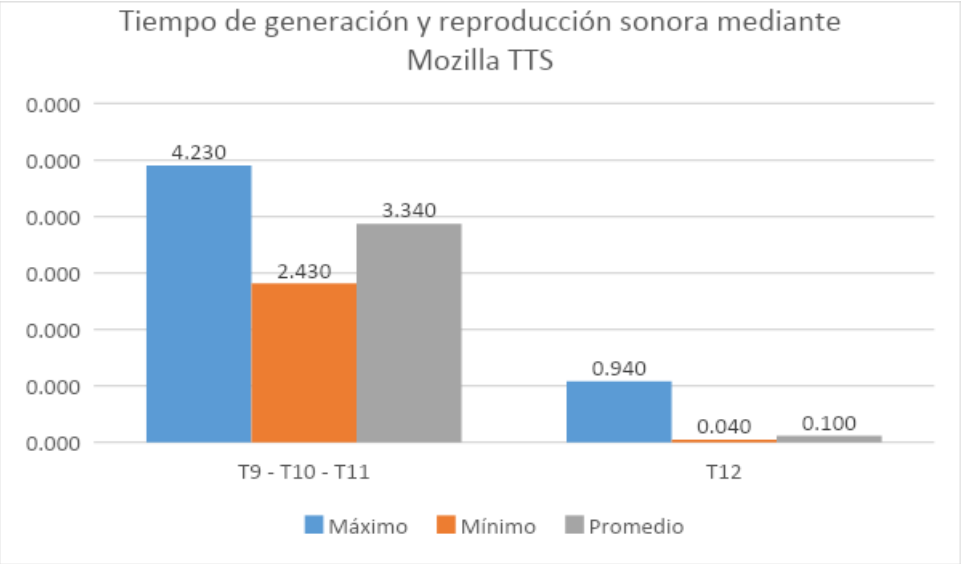

Figura 57: Resumen de tiempo de generación y reproducción de notificación de voz mediante Mozilla TTS.

Fuente: Elaboración propia.

En la generación del archivo de audio con el mismo mensaje de texto da como resultado un tiempo promedio de 3,344 ms, máximo de 4,234 ms y mínimo de 2,431 ms. Mientras que para la reproducción del mismo ocupa un tiempo promedio de 0,101 ms, máximo de 0,935 ms y mínimo de 0,039 ms.

3.2.5. Latencia de video streaming en local

Consiste en identificar la demora existente entre la transmisión de video desde las cámaras hasta su recepción en Home Assistant.

Para la toma de muestras en esta prueba se realizaron 100 capturas de pantalla en donde se presenta en una ventana el tiempo en formato hh:mm:ss,000 junto con un navegador web con la interfaz gráfica de Home Assistant (cliente) en donde se muestra el video captado. Las cámaras se colocan delante de la pantalla.

Tabla 9: Resumen de latencia de video streaming en local.

Tiempo (ms)	TAPO C200	DVR	Android IP	ESP32-CAM
Máximo	1,427	1,284	11,826	1,462
Mínimo	0,144	0,287	0,548	0,638
Promedio	0,277	0,430	1,546	0,837

Fuente: Elaboración propia.

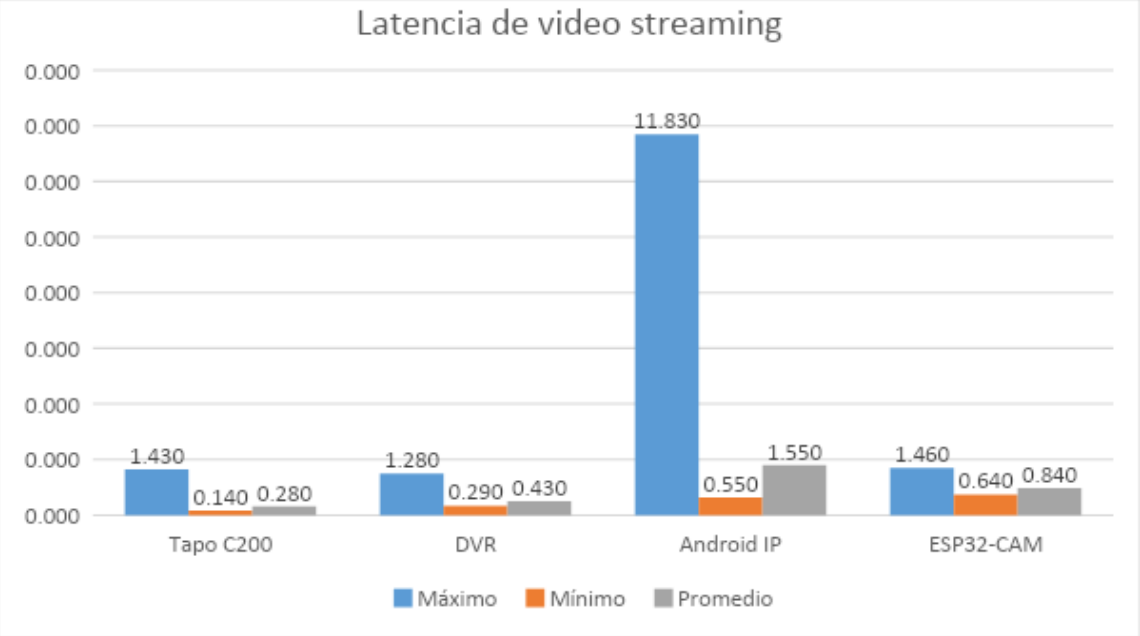

Figura 58: Resumen de latencia de video streaming en local.

Fuente: Elaboración propia.

La prueba de latencia da los siguientes promedios. La cámara Tapo con 0,277 ms, el DVR con 0,430 ms, el dispositivo Android IP Webcam con 1,546 ms y el ESP32-CAM con 0,837 ms.

3.2.6. Reconocimiento de rostros con Facebox

Al servicio de Facebox se subieron diez fotos de cinco personas para realizar esta prueba en la cual se tomaron 100 muestras de una única persona para verificar el grado de acierto en el reconocimiento. Se tomó una distancia de 1m de las cámaras y con buena intensidad de luz. La persona se desplaza realizando movimientos suaves delante de la cámara.

Tabla 10:Grado de acierto de reconocimiento de rostros con Facebox.

	Tapo C200	DVR	ESP32-CAM	Android IP Webcam
Máximo	80,33	90,96	77,26	87,74
Mínimo	55,32	56,33	54,51	51,48
Promedio	67,91	71,9531	66,9378	70,793

Fuente: Elaboración propia.

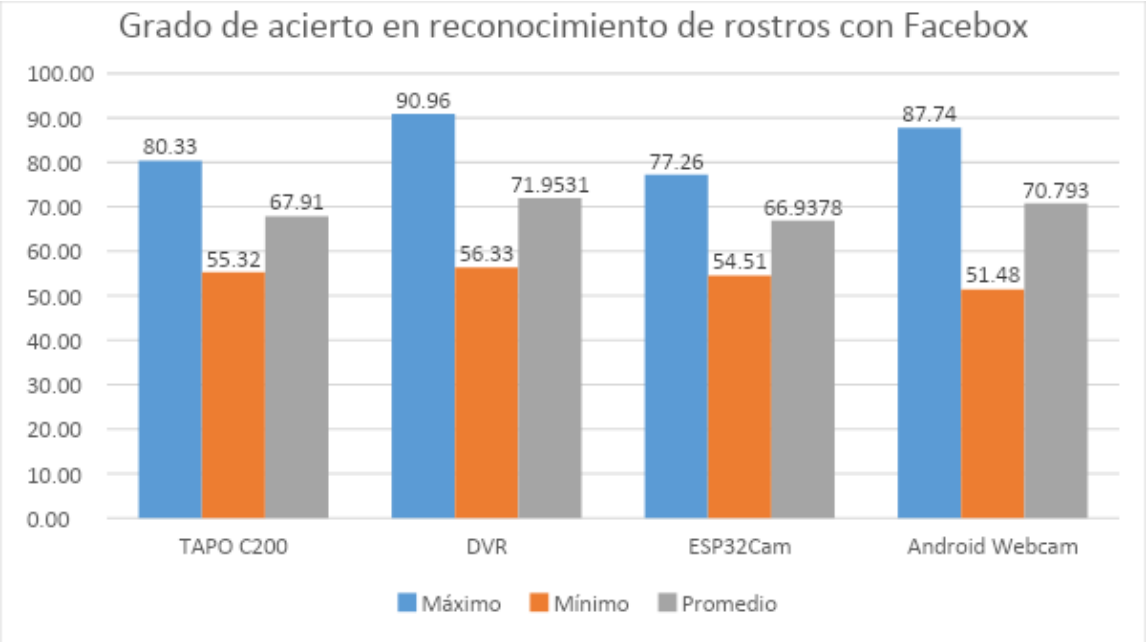

Figura 59: Grado de acierto en reconocimiento de rostros con Facebox.

Fuente: Elaboración propia.

Como resultado de las pruebas se obtiene una mayor tasa de acierto de las muestras tomadas desde el DVR con un porcentaje promedio de 71,95%,

seguido por el dispositivo Android IP Webcam con 70,793%, luego Tapo C200 con 67,91% y finalmente el ESP32-CAM con 66,93%.

3.2.7. Análisis de resultados

A continuación, se muestra la sumatoria de los tiempos promedios que se ocupan en la ejecución del escenario clasificados por cámaras.

Tabla 11: Comparación de tiempos promedios de ejecución del escenario clasificados por cámaras.

	T2	**T3,T4,T5**	**T6**	**T7**	**T8**	**T9,T10,T11**	**T12**	**Total**
Tapo C200	3,450	1,442	0,854	0,749	0,232	3,344	0,101	10,172
DVR	0,379	1,457	0,854	0,749	0,232	3,344	0,101	7,116
Android IP Webcam	2,874	1,653	0,854	0,749	0,232	3,344	0,101	9,807
ESP32-CAM	0,182	0,313	0,854	0,749	0,232	3,344	0,101	5,775

Fuente: Elaboración propia.

Como resultado se obtienen tiempos desde los más bajos que parten de 5,775 ms, en el dispositivo ESP32-CAM hasta altos como 10.172 ms en el dispositivo Tapo C200.

Al finalizar la etapa de pruebas se obtienen tiempos razonables que confirman que la ejecución de las automatizaciones con base en las cámaras se cumplen en un tiempo adecuado y con una fiabilidad aceptable capaz de poder implementarse en cualquier hogar debido a su bajo costo y beneficios exorbitantes.

3.3. CONCLUSIONES

- Se logró instalar el sistema domótico Home Assistant en la plataforma Raspberry Pi 4 a la que se le instaló un SSD para mayor eficiencia.
- Al sistema domótico se integraron cuatro tipos de cámaras siendo el ESP32-CAM el más rápido en sus operaciones a costa de una menor resolución de imagen debido a sus características inherentes.
- Se integraron además los eventos de detección de movimiento de las cámaras al asistente domótico para que ejecute los automatismos. El ESP32-CAM no cuenta con uno por lo que se le instaló un sensor PIR el

cual tiene una mejor precisión que las técnicas de detección de movimiento que aplican las demás cámaras.

- Para la integración del flujo de video de las cámaras se identificó que la tecnología WebRTC ofrece tiempos de latencia bastante bajos y con mínimo consumo de recursos de hardware. Aunque el complemento RTSP2WebRTC a la fecha no es compatible con todas las cámaras.
- Para la captura de fotos se comprobó que la utilización de las APIs que ofrecen los fabricantes están optimizadas para funcionar a buenas velocidades a diferencia de alternativas como el uso del comando FFMPEG. Un claro ejemplo está en el DVR son su protocolo de integración ISAPI.
- Con base en la seguridad se lograron integrar servicios de reconocimiento facial como Facebox y Mozilla TTS para respuestas por voz. Ambos de manera local y con un rendimiento aceptable.

3.4. RECOMENDACIONES

- La integración de cámaras a los asistentes domóticos tiene un gran potencial si se los complementa con algoritmos de inteligencia artificial, pues son la manera de otorgarles visión para tomar decisiones en cada escenario. En este trabajo se implementó Facebox para el reconocimiento facial pero el mismo tiene la limitante de que la persona debe estar cerca para obtener excelentes resultados. Se pretende a futuro desarrollar un contenedor propio cuya implementación esté basada en redes neuronales.
- Otra idea bastante futurista es el desarrollo de algoritmos capaces de detectar comandos por medio de lenguaje de señas para personas con trastornos del desarrollo del lenguaje.

BIBLIOGRAFÍA

[1] R. C. Parocha and E. Q. B. MacAbebe, “Implementation of home automation system using OpenHAB framework for heterogeneous IoT devices,” in *Proceedings - 2019 IEEE International Conference on Internet of Things and Intelligence System, IoTaIS 2019*, Nov. 2019, pp. 67–73, doi: 10.1109/IoTaIS47347.2019.8980370.

[2] V. Kepuska and G. Bohouta, “Next-generation of virtual personal assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home),” in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference, CCWC 2018*, Feb. 2018, vol. 2018-January, pp. 99–103, doi: 10.1109/CCWC.2018.8301638.

[3] S. N. Swamy and S. R. Kota, “An Empirical Study on System Level Aspects of Internet of Things (IoT),” *IEEE Access*, vol. 8, pp. 188082–188134, Oct. 2020, doi: 10.1109/ACCESS.2020.3029847.

[4] H. Boyes and T. Watson, “Towards a secure and resilient IoT architecture for smart home energy management,” in *IET Conference Publications*, 2019, vol. 2019, no. CP756, doi: 10.1049/cp.2019.0161.

[5] I. Ramírez-Morales and B. Mazon-Olivo, “Análisis de Datos Agropecuarios.”, 2018

[6] D. L. Hernández-Rojas, T. M. Fernández-Caramés, P. Fraga-Lamas, and C. J. Escudero, “Design and practical evaluation of a family of lightweight protocols for heterogeneous sensing through BLE beacons in IoT telemetry applications,” *Sensors (Switzerland)*, vol. 18, no. 1, p. 57, Dec. 2018, doi: 10.3390/s18010057.

[7] T. Chaurasia and P. K. Jain, “Enhanced Smart Home Automation System based on Internet of Things,” in *Proceedings of the 3rd International Conference on I-SMAC IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2019*, Dec. 2019, pp. 709–713, doi: 10.1109/I-SMAC47947.2019.9032685.

[8] N. H. Motlagh, S. H. Khajavi, A. Jaribion, and J. Holmstrom, “An IoT-based automation system for older homes: a use case for lighting

system," in *2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA)*, Nov. 2018, vol. 2019-Janua, pp. 1–6, doi: 10.1109/SOCA.2018.8645771.

[9] K. Agarwal, A. Agarwal, and G. Misra, "Review and Performance Analysis on Wireless Smart Home and Home Automation using IoT," in *Proceedings of the 3rd International Conference on I-SMAC IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2019*, Dec. 2019, pp. 629–633, doi: 10.1109/I-SMAC47947.2019.9032629.

[10] J. Jaihar, N. Lingayat, P. S. Vijaybhai, G. Venkatesh, and K. P. Upla, "Smart Home Automation Using Machine Learning Algorithms," in *2020 International Conference for Emerging Technology (INCET)*, Jun. 2020, pp. 1–4, doi: 10.1109/INCET49848.2020.9154007.

[11] Y. Li, W. Sheng, G. Yang, B. Liang, Z. Su, and Z. Chen, "Home Assistant-Based Collaborative Framework of Multi-Sensor Fusion for Social Robot *," in *2018 13th World Congress on Intelligent Control and Automation (WCICA)*, Jul. 2018, vol. 2018-July, pp. 401–406, doi: 10.1109/WCICA.2018.8630380.

[12] HomeAssistant, "Home assistant," 2020. https://www.home-assistant.io/ (accessed Apr. 16, 2021).

[13] "openHAB." https://www.openhab.org/ (accessed Apr. 16, 2021).

[14] S. Khattar, A. Sachdeva, R. Kumar, and R. Gupta, "Smart home with virtual assistant using raspberry pi," in *Proceedings of the 9th International Conference On Cloud Computing, Data Science and Engineering, Confluence 2019*, Jan. 2019, pp. 576–579, doi: 10.1109/CONFLUENCE.2019.8776918.

[15] A. Rasyid *et al.*, "Pothole Visual Detection using Machine Learning Method integrated with Internet of Thing Video Streaming Platform," in *2019 International Electronics Symposium (IES)*, Sep. 2019, pp. 672–675, doi: 10.1109/ELECSYM.2019.8901626.

[16] "GitHub - SensorsIot/IOTstack: Docker stack for getting started on IOT on the Raspberry PI." https://github.com/SensorsIot/IOTstack (accessed Apr.

16, 2021).

[17] “ZeroTier - Redes de área global.” https://www.zerotier.com/ (accessed Apr. 16, 2021).

[18] V. V. Pirozhenko and V. D. Grigoriev, “Video Stream Processing and Compression with Codec Choice Ability,” in *Proceedings of the 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, EIConRus 2020*, Jan. 2020, pp. 455–457, doi: 10.1109/EIConRus49466.2020.9039387.

[19] F. B. Rukmana and R. F. Sari, “Performance Evaluation of Video Streaming Application via Named Data Network (NDN),” in *2019 IEEE Eurasia Conference on IOT, Communication and Engineering, ECICE 2019*, Oct. 2019, pp. 141–144, doi: 10.1109/ECICE47484.2019.8942721.

[20] S. Gaikwad, S. Tafleen, R. Gottumukkala, and K. Elgazzar, “Fault Tolerance of Real-time Video Streaming Protocols over SDN Networks,” in *2018 14th International Wireless Communications and Mobile Computing Conference, IWCMC 2018*, Jun. 2018, pp. 101–107, doi: 10.1109/IWCMC.2018.8450440.

[21] J. Wang, S. Duan, J. Xu, S. Wang, and Q. Ling, “Synthesis of multiple video streams through multi-thread programming,” in *Proceedings of the 32nd Chinese Control and Decision Conference, CCDC 2020*, Aug. 2020, pp. 1152–1156, doi: 10.1109/CCDC49329.2020.9164266.

[22] A. Altonen, J. Rasanen, J. Laitinen, M. Viitanen, and J. Vanne, “Open-Source RTP Library for High-Speed 4K HEVC Video Streaming,” in *IEEE 22nd International Workshop on Multimedia Signal Processing, MMSP 2020*, Sep. 2020, pp. 1–6, doi: 10.1109/MMSP48831.2020.9287162.

[23] W. Hamidouche, M. Raulet, and O. Déforges, “4K real-time and parallel software video decoder for multilayer HEVC extensions,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 169–180, Jan. 2016, doi: 10.1109/TCSVT.2015.2478705.

[24] M. M. Nasralla, M. Razaak, I. Rehman, and M. G. Martini, “A Comparative

Performance Evaluation of the HEVC Standard with its Predecessor H.264/AVC for Medical videos over 4G and beyond Wireless Networks," in *2018 8th International Conference on Computer Science and Information Technology, CSIT 2018*, Jul. 2018, pp. 50–54, doi: 10.1109/CSIT.2018.8486153.

[25] E. Intriago, "Análisis de tecnologías de streaming: Evaluación de protocolos y diseño de un caso de estudio," 2016.

[26] J. Funasaka, "Adaptability Enhancement for Progressive Download Methods Based on Timer-Driven Requesting Schemes Using Multiple TCP Flows on Multiple Paths," in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Oct. 2018, no. 2, pp. 1605–1610, doi: 10.1109/SmartWorld.2018.00275.

[27] H. T. Le, T. Nguyen, N. P. Ngoc, A. T. Pham, and T. C. Thang, "HTTP/2 Push-Based Low-Delay Live Streaming over Mobile Networks with Stream Termination," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 9, pp. 2423–2427, Sep. 2018, doi: 10.1109/TCSVT.2018.2850740.

[28] F. Salih and S. A. Mysoon Omer, "Raspberry pi as a Video Server," in *2018 International Conference on Computer, Control, Electrical, and Electronics Engineering, ICCCEEE 2018*, Aug. 2018, pp. 1–4, doi: 10.1109/ICCCEEE.2018.8515817.

[29] A. Yang, L. Liu, and L. Chen, "Video transmission control for networked vision system based on RTCP," in *ANZCC 2018 - 2018 Australian and New Zealand Control Conference*, Dec. 2019, pp. 7–10, doi: 10.1109/ANZCC.2018.8606560.

[30] L. Nunez and R. M. Toasa, "Performance evaluation of RTMP, RTSP and HLS protocols for IPTV in mobile networks," in *Iberian Conference on Information Systems and Technologies, CISTI*, Jun. 2020, vol. 2020-June, doi: 10.23919/CISTI49556.2020.9140848.

[31] K. Matsuzono, H. Asaeda, I. Naladala, and T. Turletti, “Efficient Pull-based Mobile Video Streaming leveraging In-Network Functions,” in *2020 IEEE 17th Annual Consumer Communications and Networking Conference, CCNC 2020*, Jan. 2020, pp. 1–7, doi: 10.1109/CCNC46108.2020.9045507.

[32] S. Han, Y. Go, H. Noh, and H. Song, “Cooperative server-client http adaptive streaming system for live video streaming,” in *International Conference on Information Networking*, May 2019, vol. 2019-January, pp. 176–180, doi: 10.1109/ICOIN.2019.8718151.

[33] N. Eswara *et al.*, “Streaming Video QoE Modeling and Prediction: A Long Short-Term Memory Approach,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 3, pp. 661–673, Mar. 2020, doi: 10.1109/TCSVT.2019.2895223.