ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN PROTOTIPO DE TIMBRE CON VISUALIZACIÓN DE LA PERSONA QUE DESEA INGRESAR Y APERTURA DE LA PUERTA, POR MEDIO DE UNA PLATAFORMA *IOT*

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES

BENEDICT ALEXANDER VILLA TIPAN

benedict.villa@epn.edu.ec

DIRECTOR: LEANDRO ANTONIO PAZMIÑO ORTIZ

leandro.pazmino@epn.edu.ec

DMQ, agosto 2023

CERTIFICACIONES

Yo, BENEDICT ALEXANDER VILLA TIPAN declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Benedict Alexander Villa Tipan
benedict.villa@epn.edu.ec
benedictvilla@hotmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por BENEDICT ALEXANDER VILLA TIPAN, bajo mi supervisión.

Mónica Vinueza Rhor, MSC
DIRECTORA (E) DE LA ESFOT
Leandro Antonio Pazmiño Ortiz
DIRECTOR
leandro.pazmino@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmó que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales corresponde al autor que ha contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Benedict Alexander Villa Tipan
benedict.villa@epn.edu.ec
CI: 1724583305

DEDICATORIA

A mis dos pilares de vida, a mis padres, por darme la oportunidad de culminar mis estudios y demostrarme que todo es posible con determinación y constancia.

A mi hermana, Doménica, por ser un gran ejemplo a su corta edad y enseñarme a soñar despierto como lo hacen los niños.

A Isa, mi guía espiritual, pronto nos volveremos a ver.

Benedict

AGRADECIMIENTO

A mi tutor de tesis, Ing. Leandro Pazmiño por alentarme y guiarme en el arduo proceso para finalizar mis estudios.

A mis profesores, por su noble labor en impartir sus conocimientos y siempre aportar con un consejo de vida.

A mis compañeros de carrera, futuros colegas, ha sido un privilegio compartir aulas y vivencias juntos, gracias por hacer de mi vida académica un recuerdo inolvidable.

A mi colega de trabajo, Ing. Vinicio Arcos por darme la oportunidad de ingresar al mundo laboral y brindarme la confianza que necesitaba para alcanzar mis metas.

Benedict

ÍNDICE DE CONTENIDO

CERTIFICACIONES 1
DECLARACIÓN DE AUTORÍA 2
DEDICATORIA 3
AGRADECIMIENTO 4
ÍNDICE DE CONTENIDO 5
RESUMEN 8
ABSTRACT 9
1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO 10
1.1 Objetivo general 11
1.2 Objetivos específicos 11
1.3 Alcance 11
1.4 Marco Teórico 11
Sistema de Seguridad 12
Domótica 12
Videoportero 12
Microcontroladores 12
ESP32-CAM 13
Raspberry PI 13
OV2640 13
Plataforma *IoT* 13
Blynk 13
Bots Conversacionales 14
Telegram 14
IDE Arduino 14
2 METODOLOGÍA 15
3 RESULTADOS 16
3.1 Identificación de los requerimientos para el diseño del prototipo 16

Selección de Tarjeta de desarrollo 16
Visualización en tiempo Real 16
Selección de plataforma *IoT* 17
Selección de Aplicación de mensajería 17
Implementación basada en *software* libre 17
3.2 Seleccionar el hardware acorde a los requerimientos establecidos 17
Selección de hardware 17
Selección de software 24
3.3 Diseño del prototipo del sistema de videoportero 25
Esquema general del proyecto 27
Funcionamiento general del sistema 28
Explicación general del código del ESP32-CAM 29
Adquisición de video mediante cámara OV2640 30
Streaming de video 31
Conexión ESP32-CAM con *Blynk* 31
Crear Bot en *Telegram* 42
Generar comandos para *bot* de *Telegram* 47
Funcionamiento de sensores y actuadores del videoportero 50
3.4 Implementar el prototipo en una maqueta 51
Diseño de la estructura de la puerta 51
Diseño 3D para prototipo de videoportero 53
3.5 Realizar pruebas de funcionamiento del prototipo 54
Funcionamiento del sistema ante un visitante 54
Pruebas de funcionamiento con *Blynk* 56
Pruebas de funcionamiento con *Telegram* 58
Funciones Adicionales 61
Video de funcionamiento del prototipo 62
Costo del prototipo 62
4 CONCLUSIONES 62

5 RECOMENDACIONES 63

6 REFERENCIAS BIBLIOGRÁFICAS 65

7 ANEXOS 67

RESUMEN

En el presente trabajo de integración curricular se desea desarrollar un prototipo de videoportero inteligente utilizando una ESP32-CAM que permite su gestión y monitoreo a través de una aplicación desarrollada en *Blynk* como plataforma *IoT*, además del *bot* @VPortero23_bot que se encuentra en *Telegram*.

En la primera sección se abordan los objetivos que enmarcan el proyecto a desarrollarse, además, se incluyen varios conceptos teóricos de forma conceptual para tener un mejor entendimiento del prototipo a elaborarse.

La segunda sección explica la metodología a emplearse para garantizar que los objetivos planteados en la primera sección se cumplan, aquí se detallan los procesos que dieron paso a la elaboración del prototipo.

En la tercera sección se exponen los resultados alcanzados acorde los objetivos previamente definidos, aquí se detallan los requerimientos que tiene el diseño del prototipo, además se presenta la selección de *hardware* y *software* empleados en el desarrollo, la estructura del código del videoportero, la elaboración e implementación de una maqueta que emula la entrada al domicilio y finalmente las correspondientes pruebas que validan un óptimo funcionamiento.

En la cuarta sección se exponen las conclusiones al finalizar el prototipo, el enfoque se lo realiza en función a los objetivos planteados y en la quinta sección se establecen las recomendaciones con el fin de prevenir cometer errores, así como realizar una mejora en el prototipo.

En las dos últimas secciones se presentan el material bibliográfico que sirve de sustento para la realización del proyecto, así como los anexos que son conformados por: certificado de originalidad, enlace al video de funcionamiento y el código del prototipo.

PALABRAS CLAVE: Videoportero, ESP32-CAM, *Blynk*, *Telegram*, *bot.*

ABSTRACT

In the present curricular integration work, the aim is to develop a smart video doorbell prototype using an ESP32-CAM, which allows its management and monitoring through an application developed on the Blynk platform as an IoT solution, in addition to the @VPortero23_bot Telegram bot.

The first section covers the objectives that frame the project to be developed. Furthermore, several theoretical concepts are included conceptually to provide a better understanding of the prototype to be created.

The second section explains the methodology to be employed to ensure the achievement of the objectives outlined in the first section. This section details the processes that led to the implementation of the prototype.

The third section presents the achieved results in accordance with the previously defined objectives. It outlines the requirements for the prototype's design, presents the selection of hardware and software used in the development, describes the structure of the video doorbell's code, the creation and implementation of a model that simulates the home entrance, and finally, the corresponding tests validating its proper functionality.

The fourth section provides conclusions upon the completion of the prototype. The focus is on the fulfillment of the set objectives. In the fifth section, recommendations are provided to prevent mistakes and suggest improvements to the prototype.

In the last two sections, the bibliographic materials that support the project's execution are presented, along with the annexes that include: a certificate of originality, a link to the operational video, and the prototype's code.

KEYWORDS: *Video doorbell, ESP32-CAM, Blynk, Telegram, bot.*

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

La seguridad en el hogar es un punto crítico en Ecuador, pues no se ha brindado un enfoque en avanzar hacia mecanismos tecnológicos que desarrollen una solución a esta problemática. En la búsqueda de subsanar este contratiempo se puede optar por la implementación de un sistema de visualización sobre la persona que se encuentra en la puerta del domicilio, así como su apertura en caso de un visitante esperado, caso contrario se puede emitir una alerta de tal forma que disuada al invitado no esperado.

Para que esto sea viable, el domicilio debe contar con unos requisitos previos a la implementación del sistema: conexión a internet mediante Wireless, servicio de red eléctrica y dispositivos inteligentes que procesen esta información.

En la realización del presente proyecto se realiza un prototipo de videoportero, que permite su gestión a través de una aplicación en *Blynk* Plataforma *IoT* y un bot en *Telegram*. El prototipo permite capturar imágenes en tiempo real de lo que está sucediendo en la puerta de su domicilio, si el usuario se encuentra en el domicilio puede realizar un *streaming* de video a través de la aplicación *Blynk*, por otra parte, si el usuario se encuentra fuera del domicilio puede optar por una fotografía en tiempo empleado la herramienta del *bot* de *Telegram.* En caso de no contar con una buena iluminación, se integró un comando que permite iluminar el rostro del visitante, mediante el *flash* que tiene integrado el ESP32-CAM

Si el usuario requiere la apertura de puerta se puede optar por realizarla mediante un botón que se encuentra integrado *Blynk* o mediante un comando escrito por el *bot* de *Telegram*. Por otra parte, el usuario puede optar por encender la alarma que se ha integrado como complemento al proyecto con el fin de disuadir al visitante no deseado, además, como funcionalidad extra se optó por colocar un sensor magnético en la entrada al domicilio y en caso de que exista un ingreso no deseado al domicilio, se envíe una notificación a *Telegram* y automáticamente la alarma entre en funcionamiento.

Todo el prototipo ha sido montado en un diseño 3D y una maqueta que emula el ingreso al domicilio del usuario final, al ser un prototipo, las credenciales de acceso se encuentran en la programación del código y son las únicas que permiten el monitoreo y control de este. El sistema es alimentado por dos adaptadores de corriente de 5 (V) para el ESP32-CAM y de 12 (V) para el cerrojo de la puerta. Con este proyecto se desea

mejorar la seguridad para el hogar, brindando un entorno amigable a través de una aplicación móvil.

1.1 OBJETIVO GENERAL

Implementación de un prototipo de timbre con visualización de la persona que desea ingresar y apertura de la puerta por medio de una plataforma *IoT*.

1.2 OBJETIVOS ESPECÍFICOS

- Identificar los requerimientos para el diseño del prototipo.
- Seleccionar el hardware acorde a los requerimientos establecidos.
- Diseñar el prototipo del sistema de timbre.
- Implementar el prototipo en una maqueta.
- Realizar pruebas de funcionamiento del prototipo.

1.3 ALCANCE

Por medio del presente proyecto se busca implementar un prototipo de videoportero que permita al dueño de la vivienda identificar de manera visual a la persona que timbró a la puerta. Por tal motivo, tendrá la capacidad de realizar las siguientes acciones:

- Alertar al dueño de la casa cuando una persona se encuentra en la puerta.
- Mostrar por medio de una aplicación en tiempo real la persona que timbró a la puerta.
- Abrir la puerta por medio de la plataforma *IoT*.

En la búsqueda por complementar el alcance antes mencionado, se agregar las siguientes características al prototipo:

- Permitir la activación de una alarma a manera disuasiva en caso de sea un visitante no deseado.
- En adición, el prototipo permite alertar al dueño si la puerta del domicilio ha sido forzada.

1.4 MARCO TEÓRICO

En la actualidad existen diversas tecnologías orientas a la seguridad del hogar, como cámaras de vigilancia, sensores de presencia, bocinas, etc. Pero en ciertas ocasiones

pueden llegar a ser demasiados costosos en su adquisición o a su vez se necesita un conocimiento técnico para su instalación. Para facilidad del usuario final y en la búsqueda de combatir la inseguridad que se está viviendo en el país se desea crear este prototipo de videoportero inteligente. A continuación, se desarrollan conceptos para una mejor comprensión sobre el funcionamiento del prototipo.

SISTEMA DE SEGURIDAD

La serie de componentes, asi como dispositivos electrónicos intercomunicados entre sí, siendo usados para la prevención de robos, intrusiones, incendios u otros eventos, es denominado sistema de seguridad. Es decir, un sistema de seguridad tiene la finalidad de detectar amenazas en nuestro alrededor, sea en hogares, oficinas, etc. Y para lograrlo hace uso de varios equipos tecnológicos como: fotodetectores, sensores magnéticos, sirenas, llaves inteligentes, entre otros [1].

DOMÓTICA

La domótica o también llamada automatización del hogar, es la aplicación de diversas tecnologías en los hogares que tienen la finalidad de controlar y administrar automáticamente varios sistemas y dispositivos. Dichos dispositivos se orientan en buscar una mejora a la seguridad, la comodidad y el bienestar de sus usuarios.

En estos sistemas domóticos se puede llegar a controlar diversos elementos como pueden ser: sistemas de seguridad, la iluminación, aire acondicionado, entre otros equipos domésticos [2].

VIDEOPORTERO

Un videoportero es un dispositivo electrónico que se utiliza para la comunicación visual y auditiva entre personas que están en diferentes ubicaciones, como una puerta de entrada y una vivienda. Generalmente consiste en una cámara de video y un sistema de audio integrados en un mismo dispositivo, permitiendo a la persona en el interior de una propiedad ver y hablar con la persona en el exterior antes de permitirles el acceso [3].

MICROCONTROLADORES

Los microcontroladores son una especie de minicomputadoras, capaces de procesar cierta información de entrada para posteriormente darle una salida que sea entendible para el usuario y así crear un sistema automático de algún proceso determinado [4], en este caso el de un videoportero. Existen diversos microcontroladores en el mercado, pero se destacarán 2 en especial.

ESP32-CAM

LA ESP32-CAM es un dispositivo que permite realizar proyectos de tipo *IoT* ya que posee conectividad *Wifi*, pines GPIO que permiten controlar sensores o actuadores y principalmente destaca por llevar integrado una cámara de vídeo pequeña denominada OV2640. Al ser un elemento versátil se puede enfocar en darle un uso como cámara de seguridad CCTV, visión remota para robots con visión artificial embebida, etc [5].

RASPBERRY PI

La *Raspberry Pi* se trata de una computadora con dimensiones pequeñas que posee la capacidad de conectarse, a través de sus puertos USB, a periféricos como lo es: un teclado, un ratón e incluso un monitor. Tiene el sistema operativo *Linux* y permite programar en lenguajes como *Scratch* o *Python* [6].

Por lo tanto, es capaz de ser usada en varios proyectos digitales como estaciones meteorológicas, reproductores de video, etc. A diferencia de la ESP32-CAM no posee una cámara integrada, esta se debe adquirir de manera separada pero debido a sus capacidades de procesamiento se puede llegar a implementar un sistema de videoportero más avanzado, incluso con visión nocturna, pero su gran desventaja es su precio elevado.

OV2640

OV2640 es un sensor de imagen utilizado en cámaras y módulos de cámaras como lo es la ESP32-CAM. Es fabricado por *OmniVision Technologies* y ofrece una resolución de 2 megapíxeles. El OV2640 es conocido por su bajo consumo de energía y su capacidad para capturar imágenes de alta calidad en una variedad de condiciones de iluminación [7].

PLATAFORMA *IOT*

Una plataforma *IoT* es una infraestructura que contiene ciertas aplicaciones necesarias para recopilar, almacenar, analizar y entregar información a un usuario final, esta información es generada por dispositivos que se encuentran conectados a Internet.

Esta plataforma se alojada en la nube, centraliza la recopilación de datos y permite compartir información fácilmente entre aplicaciones [8].

BLYNK

Blynk es plataforma *IoT* que permite el desarrollo de aplicaciones móviles diseñada en la creación de proyectos de Internet de las cosas (*IoT*). Permite a los desarrolladores

personalizar sus aplicaciones y que estas interactúen con el *hardware* a elegirse como puede ser: *Arduino*, *Raspberry Pi* y ESP8266/ESP32*. Blynk* al estar relacionado con proyectos de *IoT* despliega una interfaz gráfica fácil de utilizar para el usuario final, permitiendo controlar y monitorear los dispositivos conectados de forma remota, segura y estable [9].

BOTS CONVERSACIONALES

Son un sistema informático que actúa como un asistente virtual, se encuentra diseñado para responder preguntas y mantener conversaciones, siendo estas de manera escrita o hablada. Estos *bots* conversacionales pueden programarse para realizar diversas funciones como responder preguntas frecuentes, brindar asistencia al usuario, realizar transacciones, brindar información personalizada, etc.

Existen *bots* conversacionales simples, los cuales siguen instrucciones preconfiguradas, es decir, el cliente o usuario debe realizar una pregunta en específico o enviar un comando en concreto para que el bot pueda funcionar [10].

TELEGRAM

Telegram se trata de una aplicación enfocada a la velocidad y seguridad en torno a la mensajería instantánea [11]. Mediante esta aplicación se puede enviar o recibir imágenes, mensajes, videos, entre otros archivos multimedia. Adicional, uno de sus principales atractivos es la interfaz de programación de aplicaciones (API) que permite la creación *bots* conversacionales, es decir, ofrece una plataforma que pueden usar desarrolladores para crear herramientas personalizadas y especializadas que se pueden usar en los chats de *Telegram*, a fin de integrar servicios.

Se desea crear un Bot para que se mantenga en comunicación con la ESP32CAM, a fin de que pueda el usuario pueda escribir comandos en el chat del *bot*, la tarjeta de desarrollo lea dichos comandos y así realizar ciertas acciones configuradas, como, por ejemplo: tomar fotografías, activar iluminación, etc.

Asimismo, cuando una persona timbre la puerta se desea tomar una fotografía y posteriormente enviársela al usuario por el chat del *bot* creado.

IDE ARDUINO

El *IDE de Arduino* crea un entorno amigable para desarrollar el código y tiene compatibilidad con la placa de desarrollo ESP32-CAM. La combinación del ESP32-CAM y el *IDE de Arduino* permite a los usuarios crear proyectos *IoT* complejos y conectados. Pueden aprovechar las capacidades de conectividad del ESP32, como el *Wi-Fi* y el

Bluetooth, para establecer comunicación con otros dispositivos o servicios en la nube. Además, el *IDE de Arduino* ofrece una amplia comunidad de usuarios, bibliotecas y recursos disponibles en línea, lo que facilita el desarrollo de proyectos con el ESP32 [12].

2 METODOLOGÍA

Acorde al plan de trabajo de titulación, se realiza una segmentación por etapas, las mismas que explicarán a detalle cómo se cumplirán los objetivos.

En la etapa 1 se realiza una identificación de los requerimientos necesarios para cumplir el alcance del proyecto, el mismo que lleva como características principales la notificación al usuario, visualización en tiempo real y apertura de puerta. Al tener estos parámetros y por medio de una investigación bibliográfica, se enlista una serie de elementos que servirán de base para la etapa 2.

En la etapa 2 denominada selección del *hardware*, se detallan las características que posee el módulo ESP32-CAM. Este módulo ha sido sometido a una comparación con otros módulos destacados, como la *Raspberry Pi* o el módulo *Arduino*, y ha resultado ser la opción elegida para la creación del prototipo. El módulo ESP32-CAM brinda las mejores cualidades, beneficios, funciones a bajo costo y es compatible con varias plataformas *IoT*, las mismas que serán abordadas en la etapa 3.

En la etapa 3 correspondiente a la fase del diseño, se realiza una investigación bibliográfica de plataformas *IoT* que permitan el desarrollo de código abierto mediante el entorno *IDE Arduino*, asi como compatibilidad con el módulo ESP32-CAM. En esta búsqueda, se opta por la plataforma *Blynk* debido a su facilidad de conexión, siendo una de las plataformas *IoT* más versátiles en el mercado y que se adecuara al desarrollo de la aplicación para el videoportero. Además, se elige *Telegram* para en la creación de un *bot* denominado @VPortero23_bot.

En la etapa 4 se hace la implementación de todo el sistema, es decir, acorde a las 2 etapas anteriores se realiza el montaje del *hardware* en una maqueta que emula el ingreso al domicilio y se acopla junto con el *software* desarrollado.

En la etapa final se ejecutan pruebas de operatividad en base al alcance del proyecto, los resultados obtenidos sirven para validar el correcto funcionamiento del prototipo y en caso de existir errores, se realizan las respectivas correcciones o afinaciones del

sistema en general. Este enfoque tiene como objetivo asegurar el cumplimiento total de los objetivos del proyecto y garantizar el logro exitoso de los mismos.

3 RESULTADOS

En esta sección se presentan los procedimientos que dieron paso al correcto funcionamiento del videoportero, se determinan los parámetros necesarios para identificar los requerimientos del prototipo y lograr seleccionar tanto el *software* como el *hardware* a emplearse. Una vez completa la selección de elementos, se expone como se realizó el diseño del videoportero en el módulo ESP32-CAM, módulo principal que cumple con éxito el alcance y objetivos del proyecto planteados.

Se realiza el diseño e implementación del prototipo mediante una impresión 3D donde se incorporan los elementos principales, asi como una maqueta que emula el acceso al domicilio. Se realiza la integración de *Blynk*, asi como el *bot* en *Telegram,* que permite la gestión y monitoreo del prototipo. Al contar con la implementación del videoportero se presentan las pruebas del correcto funcionamiento y se logra determinar si el sistema es funcional.

3.1 IDENTIFICACIÓN DE LOS REQUERIMIENTOS PARA EL DISEÑO DEL PROTOTIPO

Acorde al análisis realizado, se identifican los requerimientos necesarios para cumplir con el diseño del proyecto, principalmente se desea que el prototipo sea capaz de notificar al usuario cuando el visitante se encuentre en la puerta, visualizar en tiempo real al visitante y permitir la apertura de la puerta mediante una plataforma *IoT*.

SELECCIÓN DE TARJETA DE DESARROLLO

La tarjeta de desarrollo seleccionada debe ser capaz de procesar la información de captura de video o fotografías de manera óptima y enviar el contenido multimedia a través de internet. Además, debe permitir el control de las entradas o salidas GPIO de la tarjeta de desarrollo a fin de controlar los actuadores, en adición, debe contar con conexión *Wifi* facilite su acceso a internet.

VISUALIZACIÓN EN TIEMPO REAL

El prototipo debe contar con la visualización en tiempo real y garantizar el funcionamiento óptimo de las demás características, se debe desarrollar una alternativa que evite un consumo excesivo de los datos y una sobrecarga en la placa. Se plantea

realizar un *streaming* de video cuando el usuario se encuentre en el domicilio y permitir enviar una fotografía en tiempo real sin restricciones geográficas. Se debe contar con una plataforma *IoT* para realizar el *streaming* de video y una aplicación de mensajería eficiente para el envio de la fotografía.

SELECCIÓN DE PLATAFORMA *IOT*

La plataforma *IoT* debe tener la capacidad de crear un objeto de video para enlazarlo con el *streaming* de video que está realizando la tarjeta de desarrollo, este componente es esencial y muchas plataformas *IoT* no soportan esta característica o la soportan en su versión de pago, por lo cual se debe buscar una plataforma adecuada. Además, debe contar con restricciones de acceso al proyecto por medio de tokens que serán empleados en la elaboración del código.

SELECCIÓN DE APLICACIÓN DE MENSAJERÍA

La aplicación de mensajería que se busca integrar en el proyecto debe ser altamente eficiente y contar con una disponibilidad óptima para el usuario final, además, debe permitir la transmisión de contenido multimedia y el uso de una API que gestione el módulo a emplearse. La API debe contar con un ID de registro que será empleada únicamente para el usuario final, misma que se integrará en el código.

IMPLEMENTACIÓN BASADA EN *SOFTWARE* LIBRE

El prototipo al estar conformado por varios elementos debe contar con un *software* que permita una gestión y modificación de forma libre, siendo este gratuito e ilimitado. Debe garantizar una compatibilidad con elementos electrónicos que se encuentran integrados y sea de fácil configuración. Es esencial que permita habilitar el registro del token emitido por la plataforma *IoT*, asi como el ID de la API en la aplicación de mensajería. Esto contribuye en la confidencialidad de los datos del usuario final de manera efectiva.

3.2 SELECCIONAR EL HARDWARE ACORDE A LOS REQUERIMIENTOS ESTABLECIDOS

SELECCIÓN DE HARDWARE

En el mercado nacional e internacional, se tiene una variedad de elementos y componentes que permiten la creación de diversos proyectos, estos elementos a su vez tienen una serie de características que varían conforme los fabricantes, para ello se evalúa el componente que reúna las características adecuadas para el correcto funcionamiento del prototipo.

Selección de microcontrolador

El dispositivo principal se trata del microcontrolador, es donde se realiza la programación y servirá como eje central para la gestión de los demás componentes. El microcontrolador debe permitir la integración de una cámara que permite la visualización de la persona que desea ingresar a la vivienda, bajo este requisito, debe también contar con un óptimo rendimiento en el *streaming* de video. En la búsqueda de un microcontrolador que permita el correcto rendimiento se encuentran varias opciones en el mercado como lo son: ESP32-CAM, *Raspberry Pi*, ciertas placas de *Arduino*, etc.

A continuación, se presentan las características que poseen los microcontroladores usados en el desarrollo de proyectos relacionados al área de domótica.

Características de la ESP32CAM

La ESP32-CAM como se muestra en la **Figura 3.1**, es un dispositivo inteligente que posee las características de conectividad *Wifi* y *Bluetooth*, pines GPIO para la conexión de actuadores o sensores. Además, posee la cámara de video OV2640, una ranura para insertar una tarjeta MicroSD, en la cual se puede almacenar fotos o videos.

Figura 3.1 Tarjeta ESP32-CAM [5]

Algunas de sus aplicaciones típicas del módulo ESP-32CAM son:

- Tomar fotografías.
- *Streaming* de video.
- Detector de movimiento
- Reconocimiento facial con MTMN.

La principal característica para destacar es su bajo precio, por lo que es muy utilizado en el área de IoT. Además, soporta resoluciones de hasta 1622×1200 (pixeles) hasta 60 (cuadros por segundo) máximo. En el marco teórico se destacó que la ESP32CAM usa el sensor de imagen OV2640 ya que esta viene por defecto cuando se adquiere la

tarjeta. Este sensor se lo visualiza en la **Figura 3.2** y destaca por la resolución y rendimiento que brinda, además de su bajo costo.

Figura 3.2. Sensor de imagen OV2640

Características de la *Raspberry Pi 3B/4B*

Una de las tarjetas más populares en el mercado se denomina tarjeta *Raspberry Pi* 3 o 4 modelo B, la cual se observa en la **Figura 3.3**, es una tarjeta que posee grandes capacidades de procesamiento. Este dispositivo es considerado como una pequeña computadora personal, ya que tiene la capacidad de conectarse a un monitor externo mediante su puerto HDMI, integra puertos USB para conectar un ratón y teclado y finalmente posee su propio sistema operativo llamado Raspbian.

Figura 3.3. *Raspberry Pi* modelo B [6]

Como se mencionó anteriormente, existen dos modelos populares la *Raspberry Pi* 3 y *Raspberry Pi* 4, se puede decir que este último modelo posee características más prominentes, es decir, tiene un mejor procesador, más cantidad de memoria RAM, puertos USB 3.0, etc. Además, soporta la conexión de módulos externos, por ejemplo, una cámara, sensores y actuadores, pero todas estas capacidades se ven reflejadas en su precio, es decir, a cambio de tener todas estas características que se mencionaron se debe pagar un precio elevado.

Las principales aplicaciones en las que se usa una *Raspberry Pi* son:

- Como computadora personal.
- Centro multimedia.
- Automatización industrial, asi como residencial.
- Aplicaciones en Internet de las cosas (*IoT*).
- Servidor *web*.
- Consola de juegos.
- Seguridad electrónica.

Así también, existen módulos externos que se pueden usar como cámaras y son compatibles con la *Raspberry Pi*, por ejemplo, en la **Figura 3.4** se presenta el módulo de cámara V2 compatible con los modelos A y B de la *Raspberry*.

Figura 3.4 Módulo de cámara V2 compatible con *Raspberry Pi* [13]

Esta cámara posee una resolución de 3280x2464 (px) a 30 cuadros por segundo máximo, por lo que es considerada como una cámara de alta definición y es compatible con cualquier *Raspberry Pi*. Tiene integrado el sensor de imagen IMX219PQ de Sony, que ofrece imágenes de video de alta velocidad y sensibilidad.

Características de *Arduino*

Una de las placas que tiene un mayor renombre en el mercado y la misma que cuenta con varios modelos se denomina *Arduino*, su alta demanda en proyectos de Domótica se genera por la versatilidad que ofrece, asi como una extensa variedad de posibilidades en la creación de proyectos. Su compatibilidad con módulos externos y su variedad de

puertos disponibles como se muestra en la **Figura 3.5**, convierte a esta tarjeta en una de las herramientas de programación más completas.

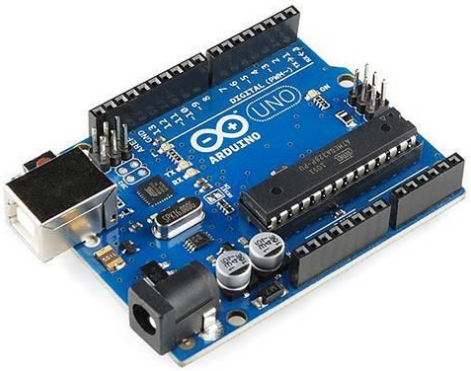

Figura 3.5. *Arduino Uno* [14]

Existen varios modelos de *Arduino* como: Mega, Nano, Uno, etc. El modelo más usado es el *Arduino Uno*, el mismo que ofrece características suficientes para implementar un proyecto de complejidad media y alta.

Algunas de las aplicaciones que se puede realizar con un *Arduino* son:

- Impresoras 3D y máquinas CNC.
- Drones y *rovers*.
- Robótica.
- *IoT*.
- Elemento de prototipado en la industria.

En el caso de este proyecto, se puede recurrir al uso de un módulo de cámara externo OV7670 la misma que se observa en la **Figura 3.6.** Pero la principal desventaja de este módulo es la poca resolución que ofrece, 640x480 (pixeles) con 30 (cuadros por segundo) máximo.

Figura 3.6. Módulo de cámara OV7670 [7]

A partir de estas características mencionadas, se elige la tarjeta ESP32-CAM ya que posee una cámara integrada y tiene la capacidad de conectar a internet mediante *Wifi*. Adicional posee los GPIO necesarios para conectar los actuadores y sensores deseados. Este módulo se lo mostro en la **Figura 3.1**.

Adicional, en la **Tabla 3.1** se realiza una comparativa con todas las especificaciones técnicas de las tarjetas de desarrollo.

Tabla 3.1 Comparacion de microcontroladores [5], [6], [14].

Parámetro	**ESP32CAM**	***Raspberry Pi* + *cámara V2***	***Arduino Uno* + OV76770**
CPU	LX6 32-bit *Dual Core*	4 núcleos Cortex-A72	*ATMega328P*
Velocidad de Reloj	hasta 160 (MHz)	1.5 (GHz)	16 (MHz)
Memoria Flash	4 (MB) SD externa 4 (GB)	SD 64 (GB)	256 (KB)
Pines GPIO	14 pines I/O	40 pines I/O	14 pines digitales y 6 analógicos
Wifi	802.11 b/g/n/e/i	802.11 ac	N/A
Bluetooth	*Bluetooth* 4.2 con BLE.	*Bluetooth* 5.0	N/A
Voltaje de operación	5 (V)	5 (V)	3.3 (V) a 5 (V)
Resolución de imagen	1622x1200 (pixeles)	3280x2464 (pixeles)	640x480 (pixeles)
FPS Max.	60	30	30
Precio	$ 13	$ 280	$ 17

En base a las especificaciones técnicas detalladas de cada microcontrolador, especialmente comparando la ESP32-CAM con el *Arduino* Uno más el módulo de cámara y la *Raspberry Pi* + cámara V2, se puede observar que la cámara de la ESP32-CAM posee mayor rendimiento en cuestiones de resoluciones, cuadros por segundo máximo y principalmente se puede destacar la conectividad a Wifi, todo esto a un bajo precio. Como se mencionó en el marco teórico, a diferencia de la *Raspberry Pi*, la tarjeta ESP32-*CAM* es mucho más barata siendo esta la principal ventaja y ofrece las capacidades para la implementación de un videoportero inteligente.

Selección de sensor

El sensor por integrar determina el estatus de la puerta, esto permite evaluar si la puerta se encuentra cerrada o abierta, se considera este parámetro importante pues garantiza al usuario final tener la seguridad de que la abierta sin que el usuario lo haya autorizado. En caso de que esto suceda se concluye que la puerta fue forzada por un visitante no deseado, de ahí su utilidad en la implementación de este prototipo, el sensor magnético a usar se muestra en la **Figura 3.7**.

Este sensor magnético para puertas es compatible en microcontroladores como lo son: *Arduino*, ESP8266, ESP32, etc. Es conveniente realizar la integración junto con módulo seleccionado ESP32-CAM. Su conexión se debe realizar en configuración *Pull Up* para su correcto funcionamiento, el voltaje de alimentación puede ser extraído del módulo y se encuentra entre 3.3 (V) y 5 (V).

Figura 3.7 - Sensor magnético para puertas [15]

Su funcionamiento es sencillo, cuando ambas partes del sensor se encuentran juntas, se cierra un circuito mediante un imán y permite el paso de voltaje, en caso de que se separen las piezas el circuito se abre. Esto quiere decir que en la tarjeta se puede realizar una lectura de voltaje para saber si la puerta está abierta o cerrada.

Selección de actuadores

Para la apertura de la puerta se hace uso de una cerradura electromagnética de 12 (V), al polarizar con el voltaje antes mencionado permite la apertura de la cerradura y junto con ella la puerta, este mecanismo se lo puede visualizar en la **Figura 3.8**. Se ha optado por este elemento pues cuenta con una similitud real al cerrojo que poseen las puertas convencionales, es sencillo de implementar y no sobrecarga de procesos al CPU lo que permite gestionar de mejor manera las otras características del sistema.

Figura 3.8 Cerradura Electromagnética [16]

Debido a que la cerradura electromagnética se debe polarizar a 12 (V) es necesario usar un relé a fin de separar la parte de control 5 (V) de la de potencia. Este elemento permite proteger la tarjeta de desarrollo de posibles daños por voltajes o corrientes elevados, se lo puede visualizar en la **Figura 3.9**, donde se muestra sus leds y pines de accionamiento.

Figura 3.9 Módulo Relé [17]

El *buzzer* es un actuador sonoro, como se muestra en la **Figura 3.10**, se encuentra integrado en el prototipo para emitir un sonido a manera de notificación, esto ocurrirá en el instante que el visitante se encuentre accionando el dispositivo, es decir, cuando presione el pulsador correspondiente al timbre, así también si existe alguna necesidad de emitir un sonido por seguridad, se lo ha integrado a manera de alarma, la cual sonará en caso de que en la puerta se encuentre un visitante no deseado.

Figura 3.10 *Buzzer* [18]

SELECCIÓN DE SOFTWARE

El Prototipo requiere ser controlado por medio de software, por ello, es necesario emplear un mecanismo que permia la programación de este, asi como la plataforma que permitirá tener un control para el usuario final. Estos dos puntos deben permitir la compatibilidad entre elementos.

Selección de IDE de programación

El módulo ESP32-CAM permite su programación en un lenguaje de tipo C. Para ello se usa el software de *Arduino IDE*, el cual es el más usado para programar este tipo de microcontroladores. Además, existen foros y soportes en internet que pueden ayudar a solucionar posibles problemas que puedan surgir durante la programación.

3.3 DISEÑO DEL PROTOTIPO DEL SISTEMA DE VIDEOPORTERO

Selección de plataforma IoT

Para el módulo ESP32-CAM se analiza la plataforma que permita la conexión de todos los pines tanto análogos como digitales, como plataformas a seleccionar se encuentran *Blynk* y *Arduino IoT Cloud.* A continuación, se realizará un análisis para seleccionar la plataforma que permita la administración adecuada del proyecto:

Blynk es una plataforma *IoT* altamente especializada y orientada a la experiencia del usuario en el control y monitoreo de dispositivos a través de una aplicación móvil. Lo que destaca a *Blynk* es su capacidad de realizar *streaming* de video en proyectos *IoT*, lo que la convierte en una elección ideal para aplicaciones que requieren una transmisión en tiempo real de imágenes y videos. Adicionalmente, permite la creación de pines virtuales que permiten un mejor control del módulo ESP32-CAM, Esto permite tener una interfaz intuitiva que permite a los usuarios acceder y controlar el streaming de video desde su dispositivo móvil de manera sencilla. Posee una aplicación en *Play Store* de manera gratuita [9].

Arduino IoT Cloud es una plataforma que combina hardware y software para crear soluciones de Internet de las cosas, ofreciendo un enfoque más amplio en la interconexión y automatización de dispositivos. Esta más enfocado en actuadores y sensores que en realizar *streaming* de video, un punto a considerar en el proyecto puesto que se requiere alcanzar este objetivo [19].

Al realizar la comparativa de estas dos plataformas se opta por la plataforma *Blynk* pues se trata de una plataforma que permite la integración de elementos *IoT*. Esta plataforma a diferencia de *Arduino IoT Cloud,* permite realizar *streaming* de video de manera eficiente, dando como resultado el desarrollo optimo del prototipo. Adicional a esto permite al usuario relacionarse con un entorno amigable en web como se muestra en la **Figura 3.11** y con una aplicación en la Play Store para teléfonos *Andriod.* Al tener las características principales del prototipo cubiertas se considera que es la más adecuada y permite lograr todos los objetivos planteados en el proyecto.

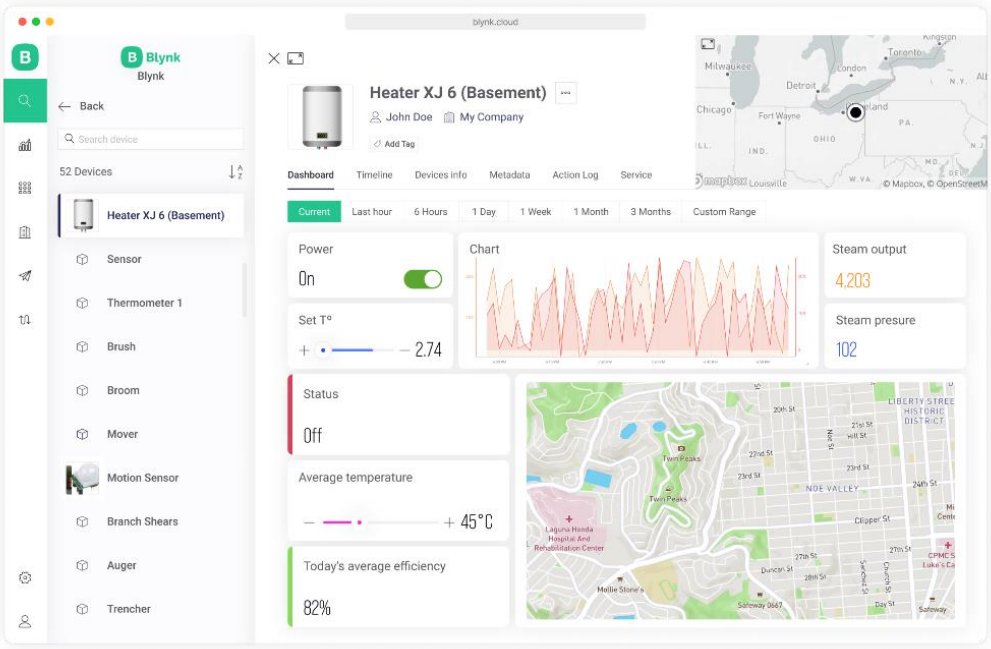

Figura 3.11 Plataforma *IoT Blynk* [9]

Selección de plataforma de mensajería

Para la plataforma de mensajería se elige *Telegram*, como se destacó anteriormente esta plataforma de mensajería posee una API para crear *bots* conversacionales, los cuales permiten integrar varios servicios. Por ejemplo, un *bot* que permita enviar o recibir comandos a la tarjeta de desarrollo y que esta ejecute las acciones programadas. Esta es la única plataforma popular que permite la integración de API y de manera gratuita, se puede visualizar un ejemplo en la **Figura 3.12**.

Figura 3.12 Plataforma de mensajería *Telegram* [11]

ESQUEMA GENERAL DEL PROYECTO

En este apartado se presenta la organización general de los componentes de *hardware* y *software* que posee el prototipo de videoportero, en la **Figura 3.11** se presenta el esquema general del proyecto. Se puede observar que la ESP32-CAM se encarga de controlador todo el sistema. Esta posee el sensor OV2640 que permitirá realizar la visualización en tiempo real, asi como el sensor magnético para puertas que permitirá la apertura, por otra parte, cuenta con actuadores como lo es: un *buzzer*, la cerradura electromagnética, un relé y el *flash* integrado en el módulo ESP32-CAM. Finalmente se puede observar que cuando se presione un pulsador (que simula el timbre de la casa), se toma una fotografía y esta se mediante internet hacia *Telegram*, asimismo se tiene la aplicación móvil en la plataforma de *Blynk* para controlar el sistema.

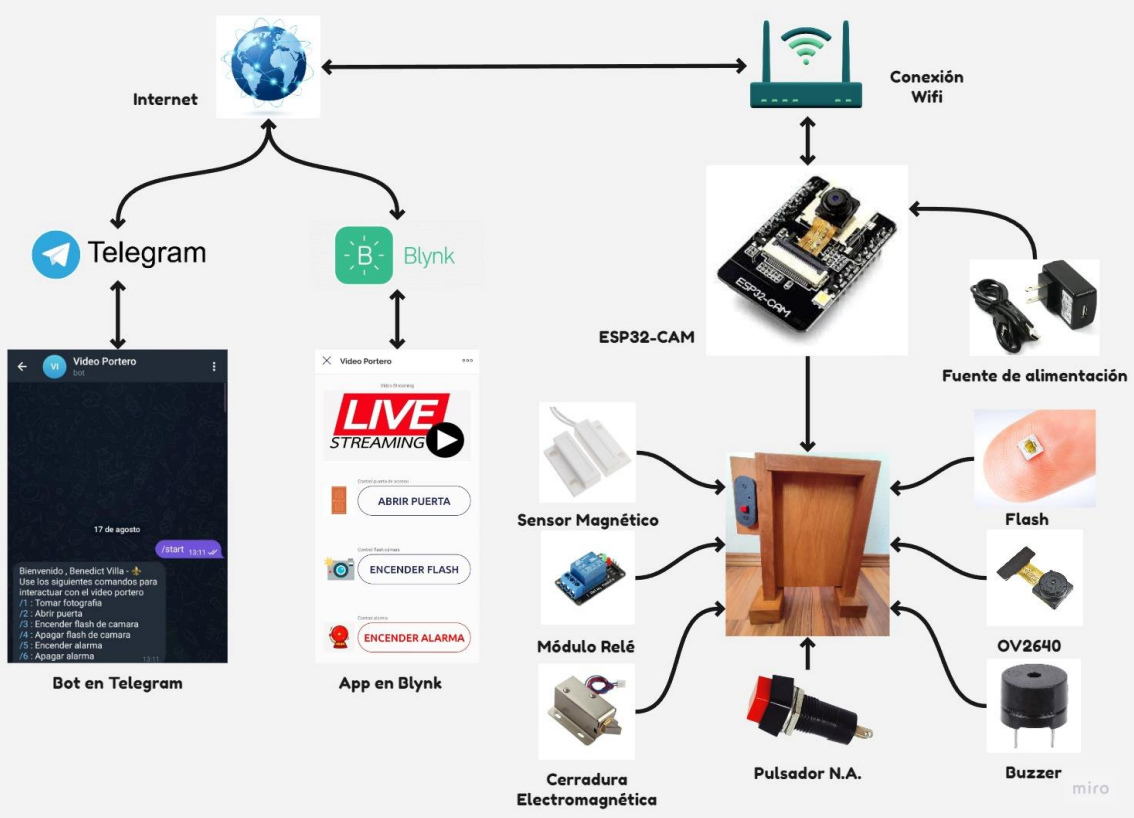

Figura 3.13. Esquema general del prototipo

El Prototipo funciona de la siguiente forma: el visitante al llegar al domicilio presionará el pulsador, el mismo que realizará la activación del sistema, acto seguido el buzzer emitirá 3 sonidos cortos y uno largo, confirmando que se está contactando con el usuario en el domicilio. Posterior a esto, el módulo ESP32-CAM mediante la cámara OV2640 se encargará de tomar una fotografía del visitante y enviarla por mensaje al usuario del domicilio.

El usuario tiene varias opciones, si se trata de un visitante esperado puede optar por abrir la puerta, en caso de que se trate de un visitante no deseado tiene la opción de alarma que servirá como método disuasivo, en caso de querer confirmar la identidad del visitante puede optar por la visualización en tiempo real por medio de Blynk que permite realizar el *streaming* de video o enviar un comando al *bot* de *telegram* para el envio de una nueva fotografía.

Además, el sensor magnético de la puerta permite monitorear el estatus de la puerta, donde se valida que la puerta se encuentra cerrada y no ha sido forzada por un visitante no deseado, en caso de que se llegue a suceder se enciende la alarma como método disuasivo y se envía un mensaje vía *Telegram* para al usuario notificando lo sucedido.

FUNCIONAMIENTO GENERAL DEL SISTEMA

A continuación, se presenta un esquema donde se representan todas las conexiones realizadas en el módulo ESP32-CAM.

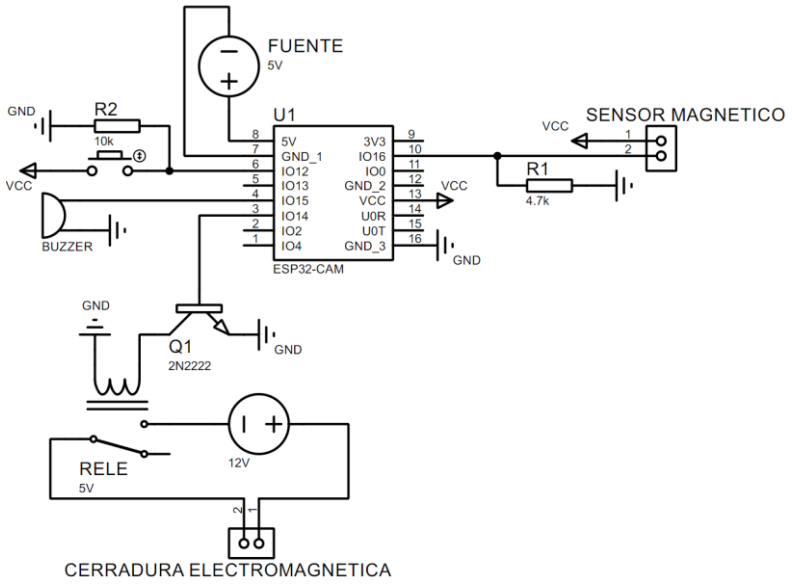

Figura 3.14 Circuito prototipo de videoportero

El circuito implementado consta de la tarjeta ESP32CAM como eje principal, a esta conecta un sensor magnético y un pulsador en configuraciones de *pull up*, adicional se tiene un *buzzer* el cual se encenderá de acorde al instante que sea ejecutado, siendo un sonido del timbre o de alarma. La cerradura electromagnética se alimenta con 12 (V) sin embargo el módulo alcanza los 5 (V), ahí nace el punto importante que es el emplear un relé, el cual permite separar la parte de control y potencia. Al emplear el relé, las salidas digitales del módulo ESP32-CAM son de 3.3 (V) es necesario emplear un

transistor 2n2222A para amplificar dicho voltaje y se pueda energizar la bobina del relé para que cambie de estado.

EXPLICACIÓN GENERAL DEL CÓDIGO DEL ESP32-CAM

En este apartado se explica de manera general el código que se ha empleado en el desarrollo del prototipo de videoportero, el código se lo puede revisar en los Anexos, por seguridad se ha oculado el token de autenticación para *Blynk*, asi como el *ID* de *Telegram*, el *SSID* de la red del hogar y su contraseña.

En el diagrama de flujo que se muestra en la **Figura 3.15**, se observa el funcionamiento general del código, en donde la tarjeta ESP32-CAM siempre se encuentra monitoreando el servidor de *Blynk* y *Telegram*, si el usuario presiona un comando lo llevará a la programación de cada plataforma, la programación en cada plataforma será explicada capítulos después. Si el pulsador entra en funcionamiento, reproduce un sonido con el *buzzer* que se encuentra conectado al sistema, se toma una fotografía y esta se envía hacia el chat de *Telegram* del usuario indicándole que alguien se encuentra en la puerta del domicilio. Asi mismo, si el sensor magnético detecta la apertura no solicitada por el usuario, activa la alarma y envía una notificación a *Telegram*.

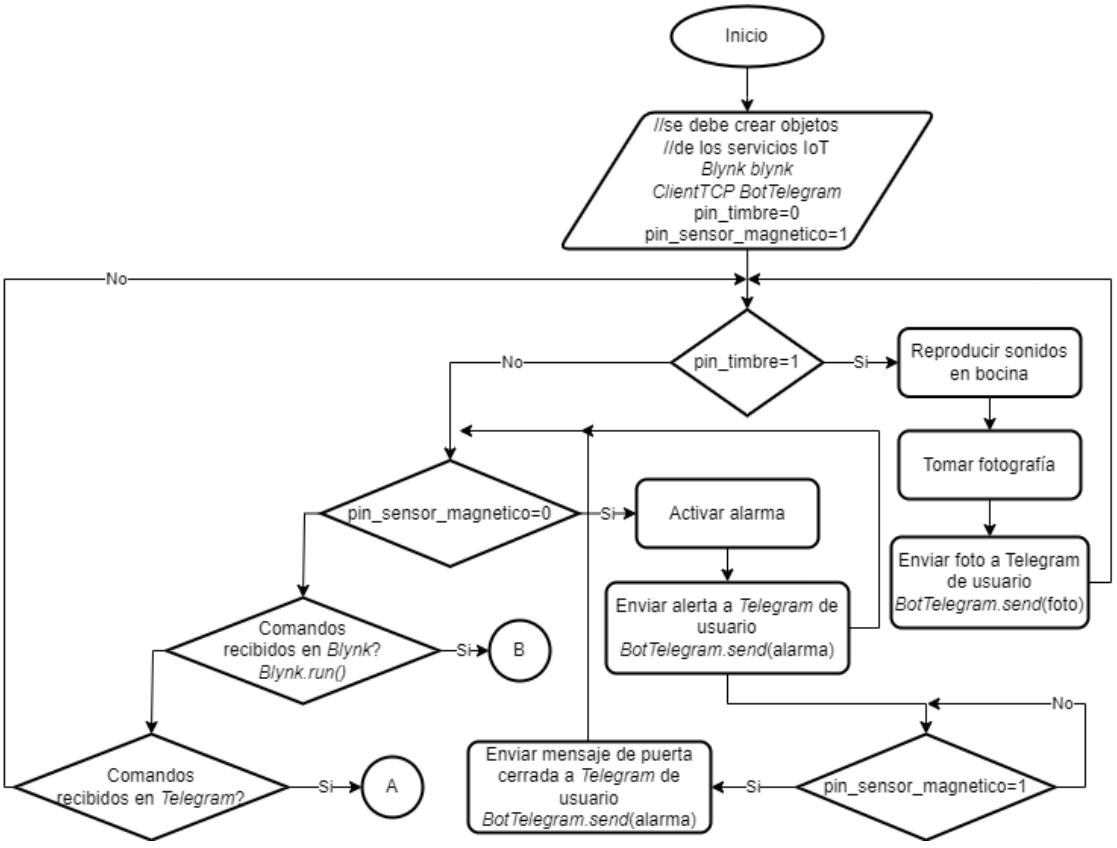

Figura 3.15. Funcionamiento general del sistema

ADQUISICIÓN DE VIDEO MEDIANTE CÁMARA OV2640

El primer paso antes de utilizar la cámara de la ESP32-CAM es inicializar el módulo OV2640. A este módulo se le debe asignar ciertas configuraciones para su correcto funcionamiento, entre dichas configuraciones se encuentra la resolución, el tipo de formato de la imagen, calidad de imagen, frecuencia de actualización y definir pines de comunicación con el microprocesador, estas configuraciones se las puede observar en la **Figura 3.16**.

Hay que considerar que existen varios modelos de placas y cámaras, por lo que la configuración puede ser diferente en cada una de estas, por ello es necesario asegurarse del modelo que se posee para realizar una correcta configuración. En este caso es un modelo de tipo "*AI THINKER*".

```
//configurar camara
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;
config.frame_size = FRAMESIZE_SVGA;
config.jpeg_quality = 10;
config.fb_count = 1;
cam.init(config);
```

Figura 3.16. Configuración de la cámara OV2640

Los parámetros para destacar son:

- Resolución de la imagen *FRAMESIZE_SVGA* (800x600): Se considera esta resolución debido a la cantidad de cuadros por segundo que ofrece. Se puede elegir una resolución de mayor tamaño, pero esto significaría un sacrificio en la velocidad de transmisión de los datos.
- Calidad de imagen (10): En este parámetro se pueden ingresar valores de 10 a 63, siendo que un menor valor significa mayor calidad. Aquí se desea tener la

máxima calidad de imagen posible para que el dueño de la casa pueda reconocer a las personas que se acerquen al videoportero con facilidad.

STREAMING DE VIDEO

Para el *streaming* de video se sigue la lógica de una conexión tipo Cliente–Servidor, donde el servidor será la tarjeta ESP32-CAM y el cliente será la aplicación móvil.

La tarjeta ESP32-CAM, incorpora conectividad *Wifi*, por lo que usa protocolos TCP/IP, esto le permite comunicarse a través internet y funcionar como servidor *web* (aunque en este caso será de forma local).

Se usa la librería "*WebServer.h*" como se muestra en la **Figura 3.17** que la tarjeta funcione en modo de servidor, esta librería se encarga de gestionar las peticiones HTTP. Es decir, cuando se detecte que un cliente va a acceder al servidor, se ejecutara una petición que mostrará un *buffer* de *frames* capturados por cámara.

En este tipo de peticiones HTTP es necesario declarar un "*header*", el cual contiene la información del contenido a mostrar, en la **Figura 3.17** se muestra el *header* usado para la transmisión de video:

```
//header necesario para streaming
const char HEADER[] = "HTTP/1.1 200 OK\r\n"
                      "Access-Control-Allow-Origin: *\r\n"
                      "Content-Type: multipart/x-mixed-replace; boundary=123456789000000000000987654321\r\n";
const char BOUNDARY[] = "\r\n--123456789000000000000987654321\r\n";
const char CTNTTYPE[] = "Content-Type: image/jpeg\r\nContent-Length: ";
```

Figura 3.17. *Header* para petición HTTP

Los *headers* HTTP siempre tienen un código de respuesta, en este caso el 200 (conexión OK), el tipo de contenido para que el cliente sepa que va a recibir y su codificación, lo cual seria los *frames* de la cámara.

Finalmente, se ejecutar una sentencia que permita mantenerse en escucha de algún cliente, es decir, si alguien accede al enlace del *streaming* se debe llamar al *header* y posteriormente enviar los *frames* capturados por la cámara.

CONEXIÓN ESP32-CAM CON *BLYNK*

El primer paso es instalar la aplicación de *Blynk* en el teléfono, para ello se debe ir a la tienda de aplicaciones del celular, buscar la aplicación *Blynk,* registrarse en la plataforma y continuar con las configuraciones que serán empleadas en la interfaz para controlar el sistema de videoportero, esto se lo puede observar en la **Figura 3.18**.

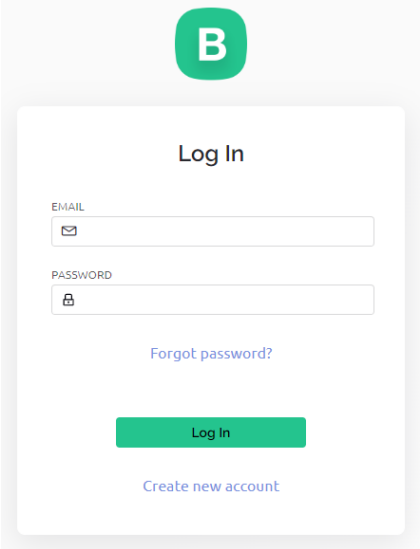

Figura 3.18. *Login* de *Blynk*

Al realizar el registro vía celular, con las mismas credenciales de acceso se ingresa vía *web*, aquí se puede observar el panel de administración de la plataforma *Blynk* como se muestra en la **Figura 3.19**.

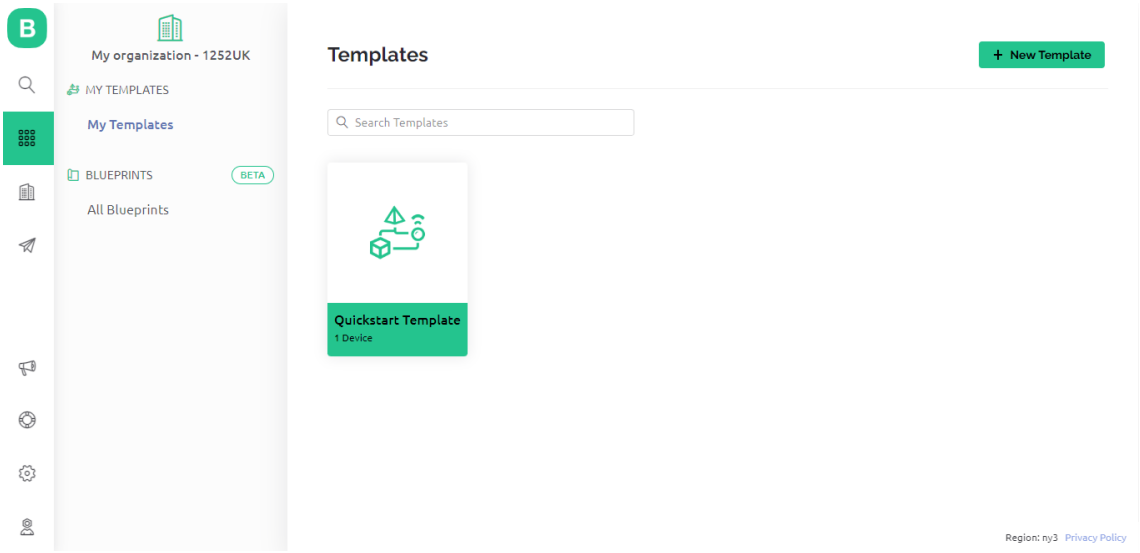

Figura 3.19. Panel de administración de *Blynk*

En el panel de administración se puede crear una plantilla, dicha plantilla contiene la configuración de los pines de la tarjeta ESP32CAM. Se debe seleccionar la opción de "*New Template*". Se abre una ventana como se muestra en la **Figura 3.20** en donde se registra el nombre de la plantilla, la placa a conectar (en este caso se selecciona una ESP32), el tipo de conexión que será mediante *WiFi* y finalmente una descripción de la plantilla a crear.

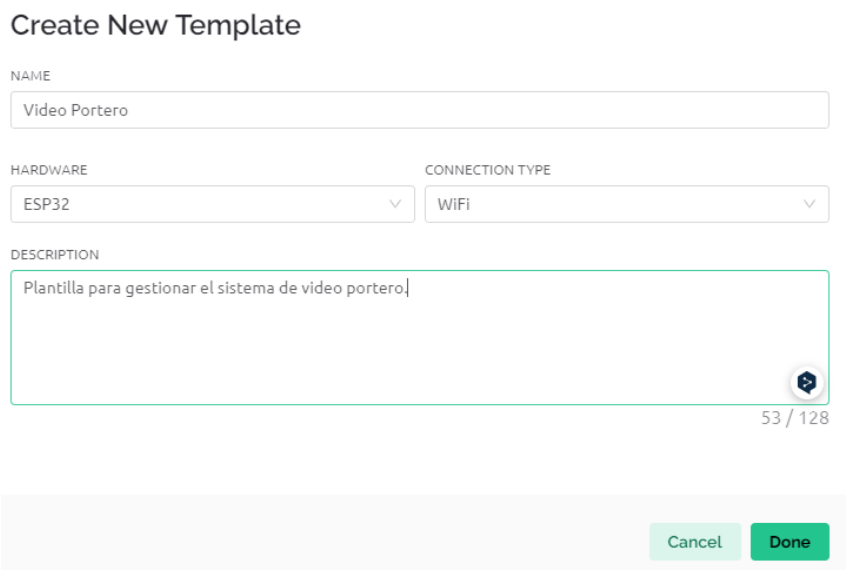

Figura 3.20. Crear nueva plantilla en *Blynk*

Una vez se crea la plantilla, *Blynk* proporciona un *token* para realizar la conexión con el módulo ESP32-CAM como se muestra en la **Figura 3.21**. Adicional, aquí se muestra como declarar este *token* en el *software* de programación usado.

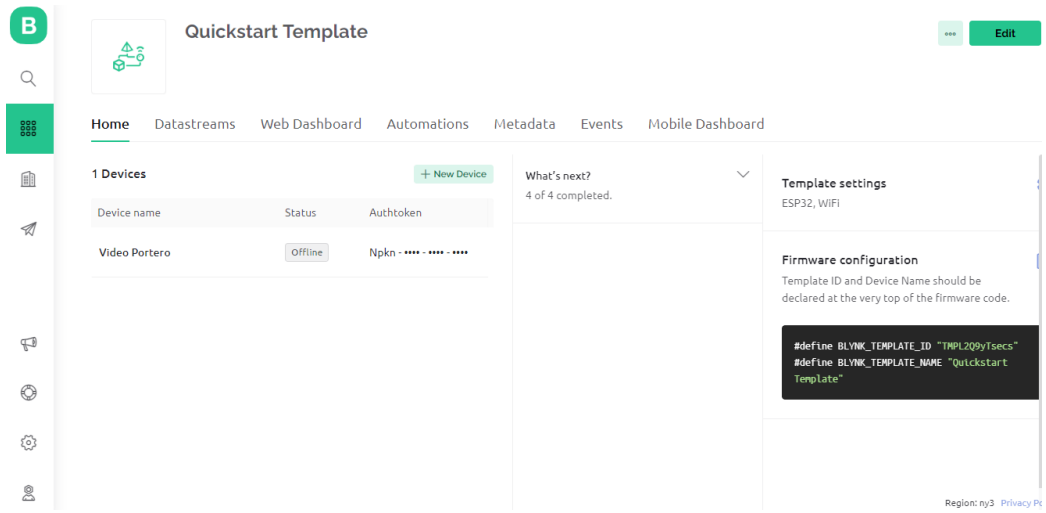

Figura 3.21. *Token* generado en *Blynk*

Ahora se debe crear los flujos de datos, es decir las variables que se enviaran al módulo ESP32-CAM, estos pueden ser físicos o virtuales, estos pines virtuales se los puede observar en la **Figura 3.22**. Por ejemplo, se puede crear una variable virtual que cuando se cambie de estado a través de la aplicación de *Blynk*, se ejecute ciertas acciones en la ESP32-CAM como abrir la puerta. Para crear estos pines se debe seleccionar la opción de "*DataStreams*".

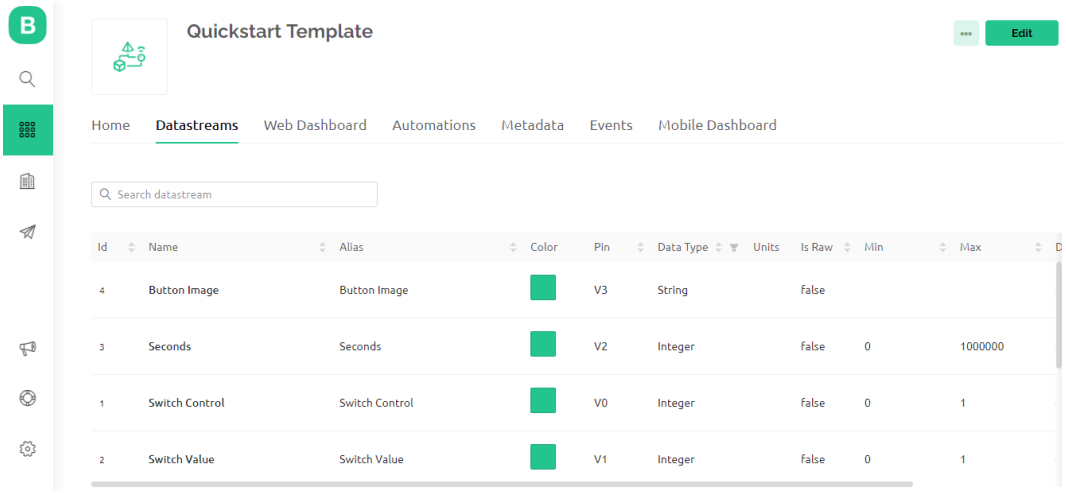

Figura 3.22. Pantalla para crear flujos de datos en *Blynk*

Una vez se ingresa a esta opción, se procede a crear los flujos de datos deseados. En la esquina superior derecha se encuentra la opción de "*Edit*", esto permite agregar o eliminar variables previamente creadas o las que vienen por defecto, tal como se muestra en la **Figura 3.23**. Una vez se selecciona la opción para editar las variables, se debe presionar el botón de "*New DataStream*".

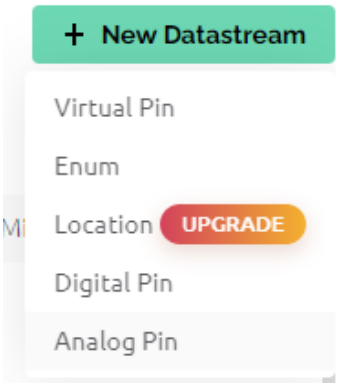

Figura 3.23. Opción para crear un nuevo flujo de datos

Se puede observar que se puede crear variables que tendrán comunicación con pines virtuales, digitales, analógicos, etc. En este caso se desea crear las siguientes variables tal cual se muestra en la **Tabla 3.2**.

Tabla 3.2. Flujo de datos creados en Blynk

Nombre de la Variable	Tipo del Pin	Dirección del Pin
Control Puerta	Virtual	V6
Control Iluminación	Virtual	V7
Control Alarma	Virtual	V8

Al seleccionar la opción para crear un pin virtual, se abre una ventana como se muestra en la **Figura 3.24**. Aquí se debe configurar un nombre para la variable, seleccionar el número de pin con el que se tendrá comunicación, el tipo de dato de la variable, unidades (en caso de que se requiera) y los valores máximos y mínimos. Para las variables planteadas se desea que sea de tipo entero y que adquieran solo valores de 0 y 1 (ON - OFF).

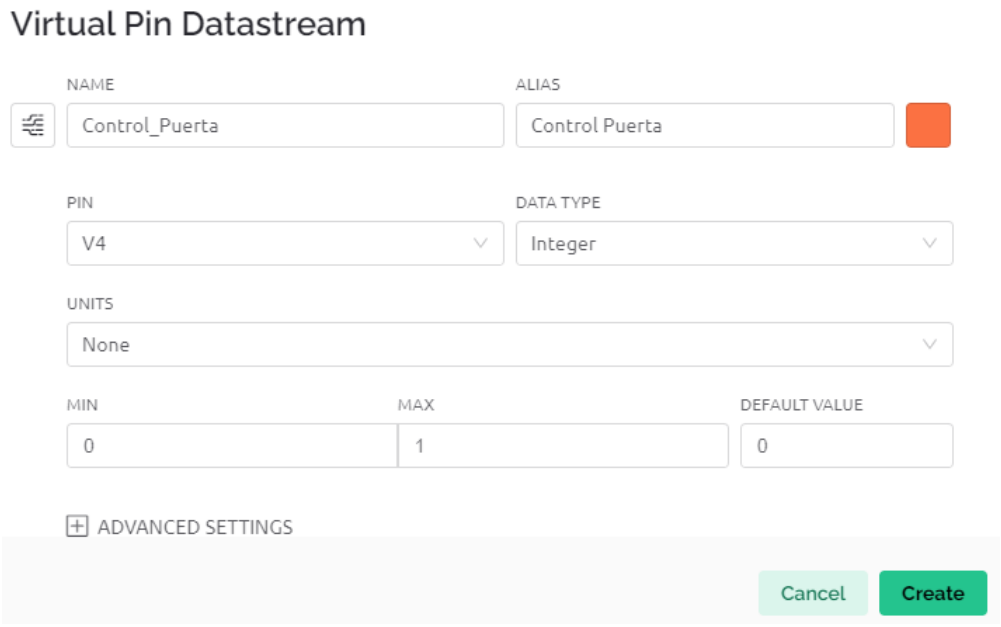

Figura 3.24. Configuración de un pin virtual en *Blynk*

Posteriormente, se realiza la configuración de los pines necesarios tal como se muestra en la **Figura 3.25**:

Figura 3.25. Variables creadas en *Blynk*

Ahora se procede a crear la interfaz en la aplicación móvil de *Blynk*, esta puede ser descargada de la *Play Store*.

Una vez descargada e instalada se procede a abrirla y se debe iniciar sesión, automáticamente aparece el *template* creado anteriormente en la versión *web* de *Blynk*.

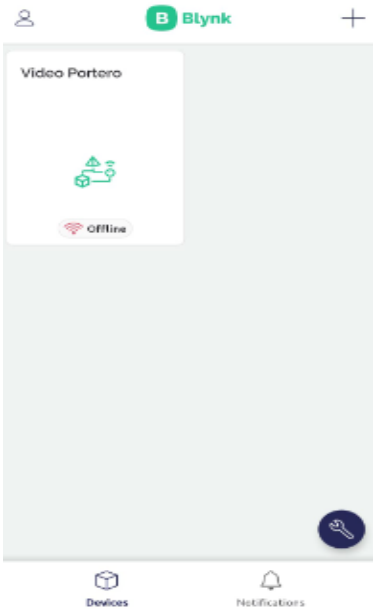

Figura 3.26. Interfaz de la aplicación móvil de *Blynk*

Al seleccionar la plantilla como se muestra en la **Figura 3.27**, se puede observar que está vacío, es decir, no existe ningún componente para mostrar al usuario. Adicional, en pantalla se muestra una recomendación que dice que se debe ingresar al modo desarrollador para empezar a añadir los *widgets* deseados, esto se realiza presionando en el botón azul que se encuentra en la esquina inferior derecha.

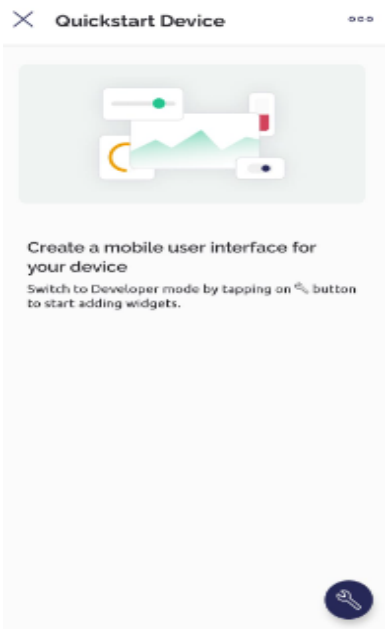

Figura 3.27. Interfaz para usuario sin añadir *widgets* en *Blynk*

Al presionar el botón indicado, se accede al modo desarrollador. En esta pantalla que se muestra en la **Figura 3.28** se puede añadir los *widgets* deseados, en este caso se desea añadir botones para manejar los actuadores del sistema y para acceder al *streaming*.

Figura 3.28. Modo desarrollador en *Blynk*

Se puede observar un icono de “+” en la esquina superior derecha, cuando se presiona se despliega una barra lateral con todos los *widgets* que se puede agregar a la interfaz como se muestra en la **Figura 3.29**, algunos *widgets* son versión premium, es decir, son de paga.

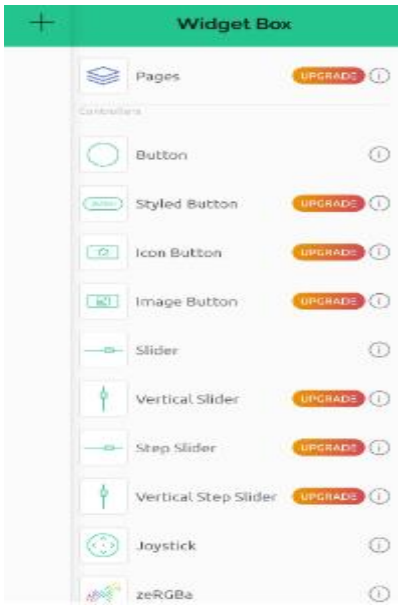

Figura 3.29. *Widgets* de *Blynk*

Primero se desea añadir un *widget* que permita abrir el *streaming* de video que realiza la ESP32-CAM vía *web*. En este caso existe un botón denominado "*WebPage Image Button"* mostrado en la **Figura 3.30**, este botón permite abrir un sitio *web* cuando se lo presiona.

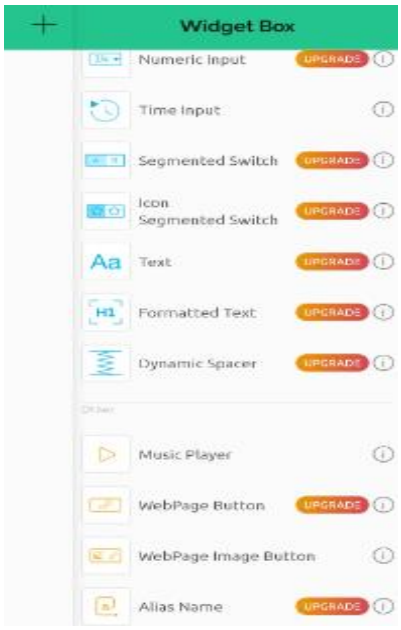

Figura 3.30. *Widget* para abrir página *web* con un botón en *Blynk*

Al seleccionarlo, aparece automáticamente en la interfaz. Este se puede redimensionar y colocar en cualquier sitio deseado como se observa en la **Figura 3.31.** Una vez se establece su ubicación y tamaño se le debe dar un clic para configurarlo.

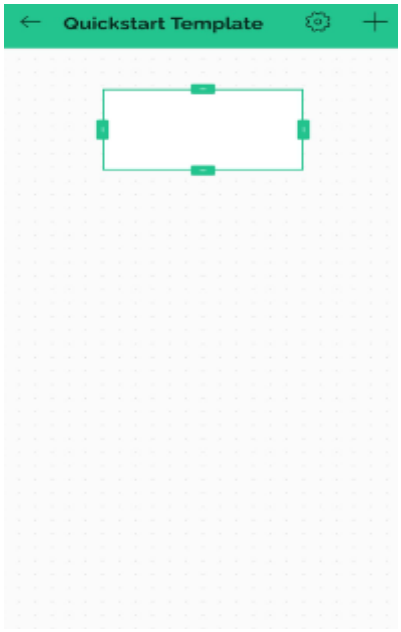

Figura 3.31. Añadir *widget* "*WebPage Image Button*" en *Blynk*

En las configuraciones de este botón se observan en la **Figura 3.32**, en el campo de URL se debe añadir la dirección IP donde se encuentra transmitiendo la ESP32-CAM. Otros parámetros para configurar son si se desea abrir el *streaming* en el navegador *web* o en la misma aplicación de *Blynk*, en este caso se desea usar la misma aplicación móvil. Además, en la parte de diseño, se puede agregar una imagen al botón.

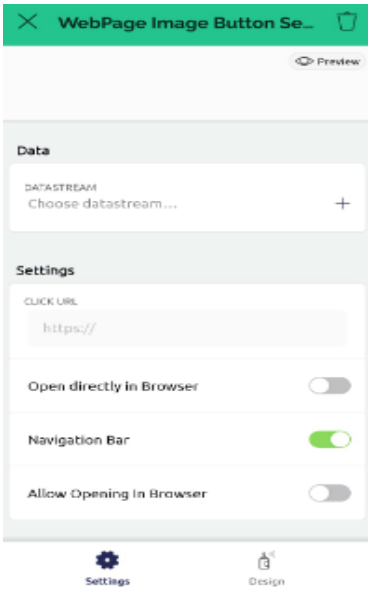

Figura 3.32. Configuración del *widget* "*WebPage Image Button*" en *Blynk*

Ahora se procede a añadir un *widget* correspondiente a un botón sencillo, para ello se repite los pasos anteriores y se selecciona el componente "*Button"*, en la **Figura 3.33** se puede observar el botón colocado en la interfaz.

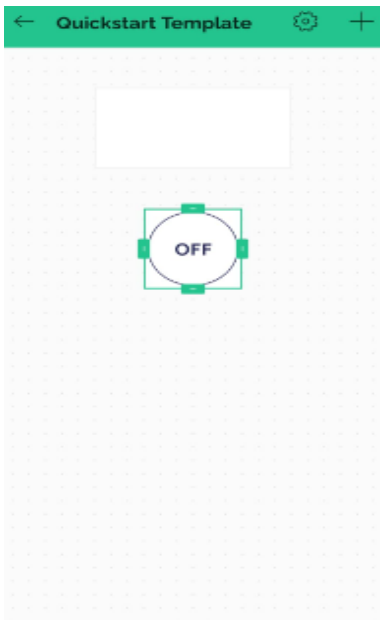

Figura 3.33. *Widget* "*button*" en *Blynk*

De igual manera al darle clic al botón se lo puede configurar. Los parámetros que permiten configurarse por medio de los botones son:

- *Data*: En este parámetro se debe vincular las variables virtuales creadas anteriormente para cada botón creado en la interfaz
- *Mode*: Se refiere al comportamiento del botón, si funciona como un pulsador o como un *switch*, en la **Figura 3.34** se puede la pantalla de configuración.

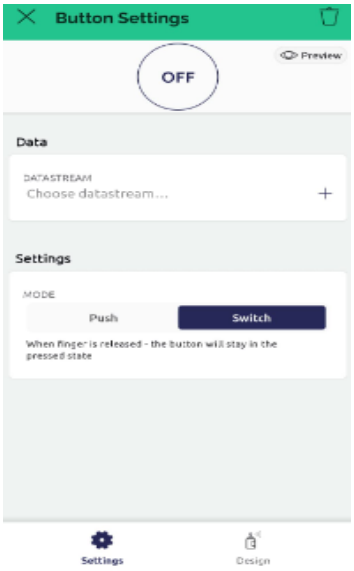

Figura 3.34. Configuración de "*button*" en *Blynk*

Por ejemplo, al presionar el parámetro de *Data*, se abre la siguiente pantalla y aquí se puede observar las variables creadas (V6, V7, V8), las cuales se pueden observar en la **Figura 3.35**.

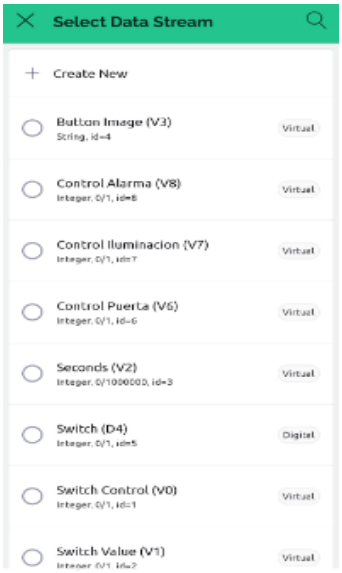

Figura 3.35. Vincular variables creadas con los botones en *Blynk*

Una vez se agregan todos los botones deseados y se haya vinculado las variables a los botones, se puede modificar los diseños y tamaños al gusto del usuario, agregar imágenes, colores, etc. Al final se obtiene un diseño como el que se despliega en la **Figura 3.36**:

Figura 3.36. Interfaz final para usuario en *Blynk*

El funcionamiento de cómo se encuentra trabajando *Blynk* se plasma en la **Figura 3.37**, diagrama de flujo que permite complementar el paso a paso que abordo en el desarrollo de la conexión del módulo ESP32-CAM con *Blynk.*

En el diagrama de se empieza por declarar los pines virtuales como se tenía previsto en la **Tabla 3.2**, además se añade la URL donde permite visualizar el *streaming* de video, se prioriza el uso del *streaming* el mismo que se mantendrá activo hasta que el usuario haya dejado de usarlo.

A manera de ejemplo se puede visualizar en la **Figura 3.37** que la lógica a emplear del cerrojo es la apertura y posterior cierre en un tiempo de 3 segundos, asi como el encendido del *flash* para una mejor iluminación y tambien la activación del *buzzer* que servirá como método disuasivo en caso de tener visitantes no deseados.

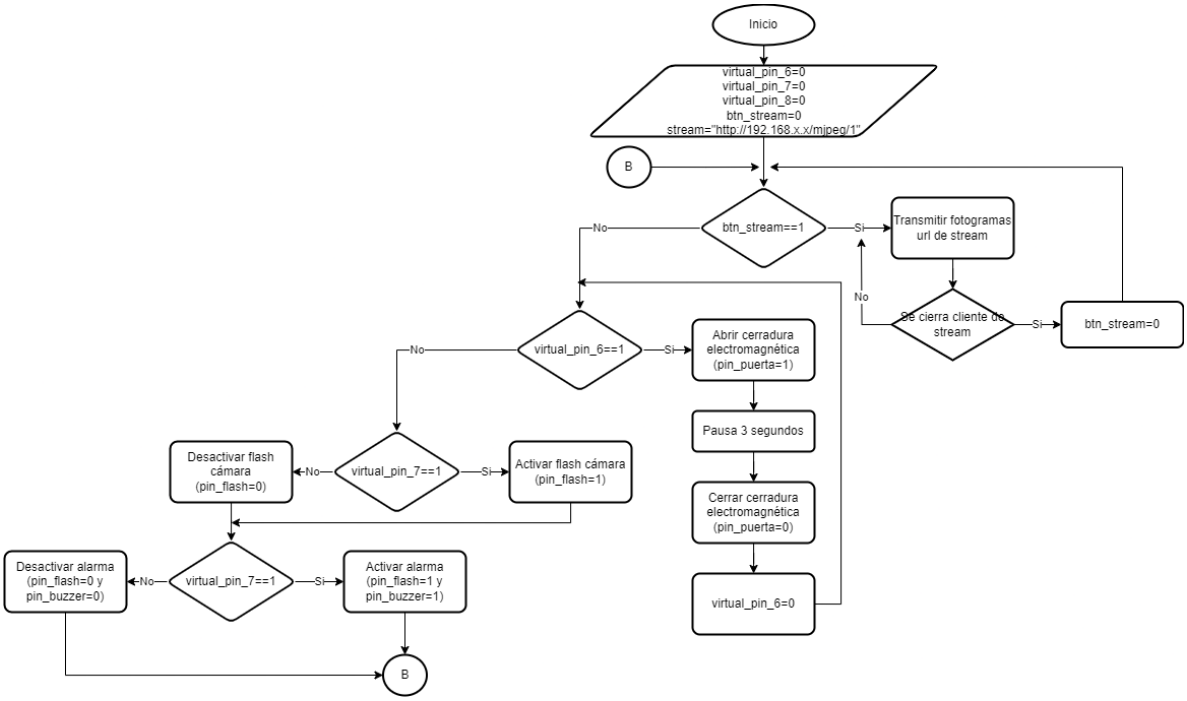

Figura 3.37 Diagrama de funcionamiento *Blynk*

CREAR BOT EN *TELEGRAM*

Para crear un bot se debe usar la API de *Telegram*, el primer paso es descargar la aplicación de *Telegram* ya sea para *Android* o *iOS*. Posteriormente, se crea la cuenta de usuario registrando el número de celular en la aplicación. Una vez se crea la cuenta de usuario, se procede a buscar un administrador de *bots* denominado *BotFather*. Este *bot* tiene el propósito de administrar a otros *bots*, es decir, crearlos, modificarlos, eliminarlos, entre otras opciones y todo por cuenta propia.

Para usar este *bot*, accedemos a la aplicación de *Telegram*, en el buscador que se encuentra en la esquina superior derecha (icono de lupa), presionamos y escribimos la palabra *BotFather*, aquí aparecen varias opciones que son el *bot* y grupos de chats similares a foros acerca de este *bot* como se muestra en la **Figura 3.38**

Figura 3.38. *BotFather* en *Telegram*

Al abrir este chat se muestra una pantalla informativa de lo que puede hacer el *bot*, y en la parte inferior un botón de iniciar, que permite empezar a interactuar con este *bot* tal como se muestra en la **Figura 3.39**.

Figura 3.39. Pantalla informativa de *BotFather*

Al presionar el botón de iniciar se muestra toda la lista de comandos disponibles con los que se pueden interactuar en el chat como se muestra en la **Figura 3.40**.

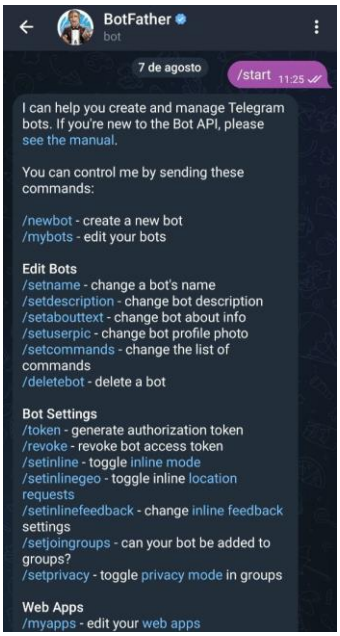

Figura 3.40. Lista de comandos de *BotFather*

Para crear un nuevo *bot* se usa el comando "/*newbot*". Al ingresar los comandos, el chat responde con instrucciones de lo que se debe realizar para crear el *bot* de manera correcta. El primer paso es ingresar un nombre para el *bot* como se muestra en la **Figura 3.41**.

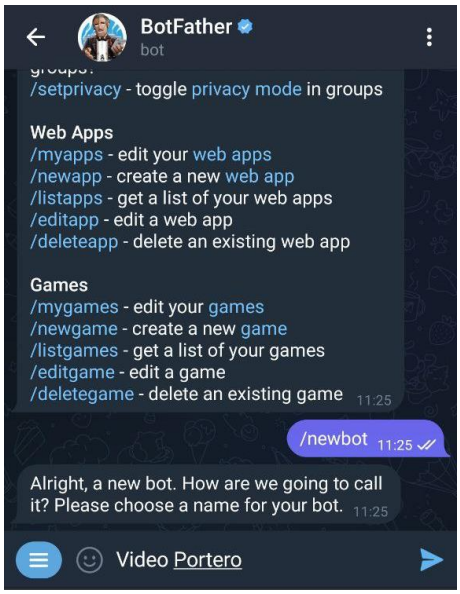

Figura 3.41. Comando para crear un nuevo *bot*

Una vez se ingresa el nombre del *bot*, se procede a ingresar un nombre de usuario para el mismo, este debe ser único y debe contener la palabra *bot* al final del nombre como se muestra en la **Figura 3.42**. En caso de que se repita, el chat notifica que se debe escribir otro nombre de usuario.

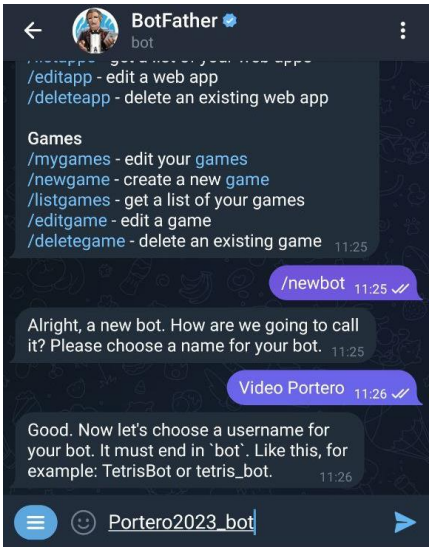

Figura 3.42. Configurar nombre de usuario para el *bot*

Finalmente, se muestra el token con el que se puede conectar la ESP32CAM con el *bot* de *Telegram* como se muestra en la **Figura 3.43**.

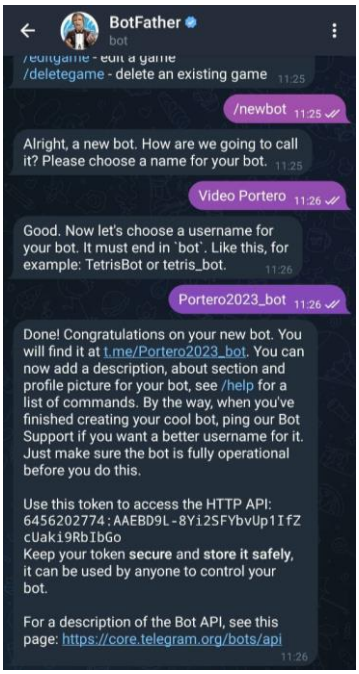

Figura 3.43. Token del *bot* en *Telegram*

Se puede escribir el comando "/*mybots*" para verificar los *bots* creados como se muestra en la **Figura 3.44**. Asimismo, se los puede seleccionar, para verificar su estado y tokens.

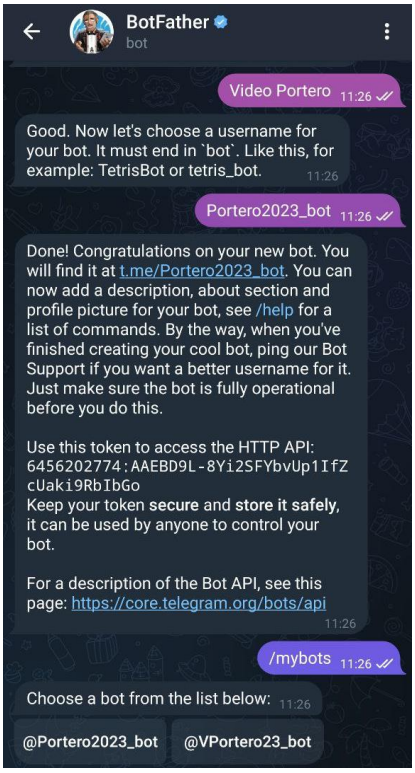

Figura 3.44. *Bot* creado en *Telegram*

Adicional, otra característica que se puede añadir al *bot*, es la de restringir usuarios no autorizados, es decir, que los comandos que se ingresen al chat para controlar la ESP32CAM solo puedan ser ingresado por cierto usuario en específico. Esto se puede hacer condicionando el *ID* del usuario, en este caso sería el dueño de la casa, para encontrar este *ID*, se debe escribir en el buscador un *bot* llamado "*IDBot*" como se muestra en la **Figura 3.45**.

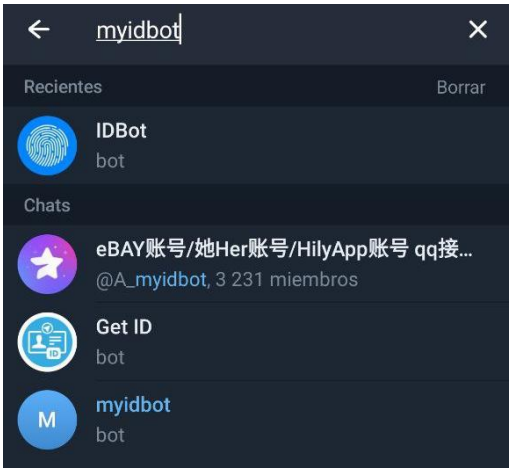

Figura 3.45. *Bot* para determinar *ID* de usuario

Al abrir el *bot* se muestra una pantalla informativa de las funcionalidades de dicho *bot* las cuales se despliegan en la **Figura 3.46**.

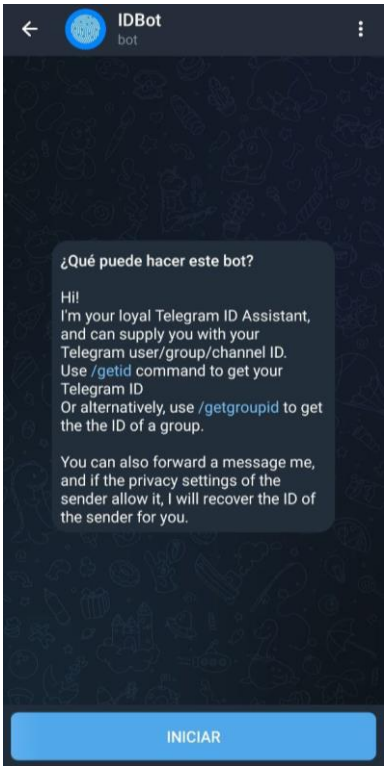

Figura 3.46. Pantalla informativa de *IDBot*

Para obtener el ID se debe escribir el comando "/*getid*". Se muestra un código como se puede observar en la **Figura 3.47**, el cual debe ser copiado y pegado en la sección de programación necesario.

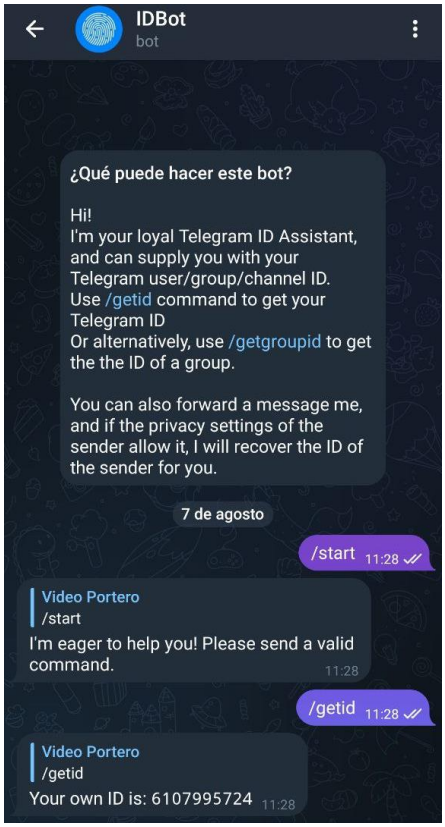

Figura 3.47. *ID* de usuario en *Telegram*

GENERAR COMANDOS PARA *BOT* DE *TELEGRAM*

Los comandos que se pueden usar en el chat del *bot* creado se deben programar en la ESP32-CAM. Por lo tanto, la ESP32-CAM debe mantenerse en modo "escucha" de los comandos que se reciben y para lograr esto se puede usar la librería Universal de *Telegram*, la cual fue creada por Brian Logh, y esta librería provee una interfaz fácil de usar para la API de *Telegram*.

Para instalar esta librería se debe abrir el *IDE* de *Arduino*, y en la barra lateral de la izquierda se debe seleccionar la opción del gestor de librerías como se muestra en la **Figura 3.48**. Se abre un pequeño menú en donde se puede escribir la librería mencionada anteriormente y se procede con la instalación.

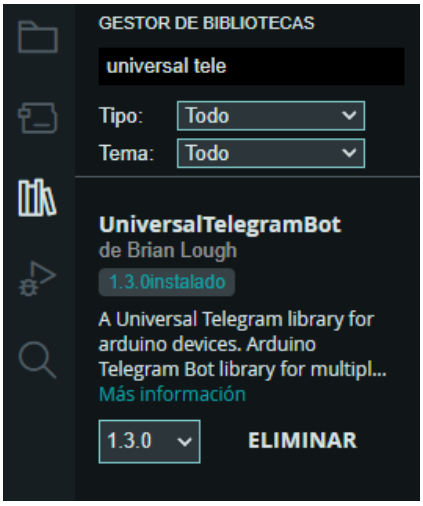

Figura 3.48. Instalación de biblioteca Universal de *Telegram*

Posteriormente se debe realizar la instalación de otra librería llamada ArduinoJson, el cual se encarga de manejar los objetos *JavaScript Object Notation* (JSON) que utiliza la API de *Telegram* para enviar o recibir mensajes, esta librería es importante ya que funciona en conjunto con la librería Universal de *Telegram*, en la **Figura 3.49** se muestra la instalación.

JSON, se puede decir que es una forma estándar de representar un objeto de tipo *JavaScript* como una cadena de datos. Los datos se presentan en un formato textual básico que se estructura de manera similar a un diccionario, es decir, contiene una clave o valor, esto facilita diversos procesos de intercambio de información por lo que se usa comúnmente en las API web para comunicar dispositivos diferentes.

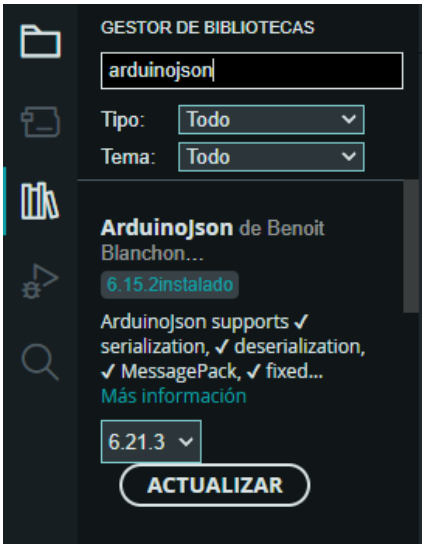

Figura 3.49. Instalación de biblioteca *ArduinoJson*

Una vez se instalan las librerías necesarias, estas se deben importar al código del proyecto. Posteriormente, se sigue la siguiente lógica para la comunicación con el *bot*.

- Se declara 3 variables auxiliares:
  - *Timer*: contiene el valor de cronometro, se usa la función millis()
  - Consulta: contiene el valor (constante) de cada cuanto se desea realizar la consulta de si existen mensajes existentes en el chat del *bot*, en este caso es cada segundo.
  - Msg: almacena el comando que se envía a través del chat
- Cuando se cumple el cronometro para verificar si existen nuevos mensajes, se usa un método de la librería para obtener los mensajes escritos en el chat. En caso de que se encuentre una cadena de caracteres esta se verifica si coincide con los 5 comandos propuestos para el sistema los cuales son:
  - "/1": este comando llama a la función para tomar una fotografía con la tarjeta ESP32-CAM y posteriormente se envía hacia el chat, utilizando la librería de *ArduinoJSON*.
  - "\2": este comando energiza el pin en el que se encuentra conectado el actuador de la puerta, es decir el relé y la cerradura electromagnética. Esto causa que el pestillo de la cerradura se contraiga y por lo tanto la puerta se abre, después de 3 segundos el pestillo vuelve a su posición normal.
  - "/3": este comando se encarga de encender el *flash* de la tarjeta ESP32-CAM, para lograr esto, se debe energizar el pin número 14.
  - "/4": este comando apaga el *flash* de la tarjeta ESP32-CAM.
  - "/5": este comando se encarga de encender la alarma de la casa, es decir, se enciende de manera intermitente el *buzzer* y el *flash* de la cámara.
  - "/6": este comando se encarga de apagar la alarma de la casa.

Cuando se escriben estos comandos, el *bot* de *Telegram* responde con una confirmación la acción solicitada por el usuario. Además, se recibe mensajes cuando la puerta fue forzada, cerrada o cuando alguien está en la puerta tocando el timbre.

En la **Figura 3.50** se muestra el diagrama de flujo de la lógica implementada.

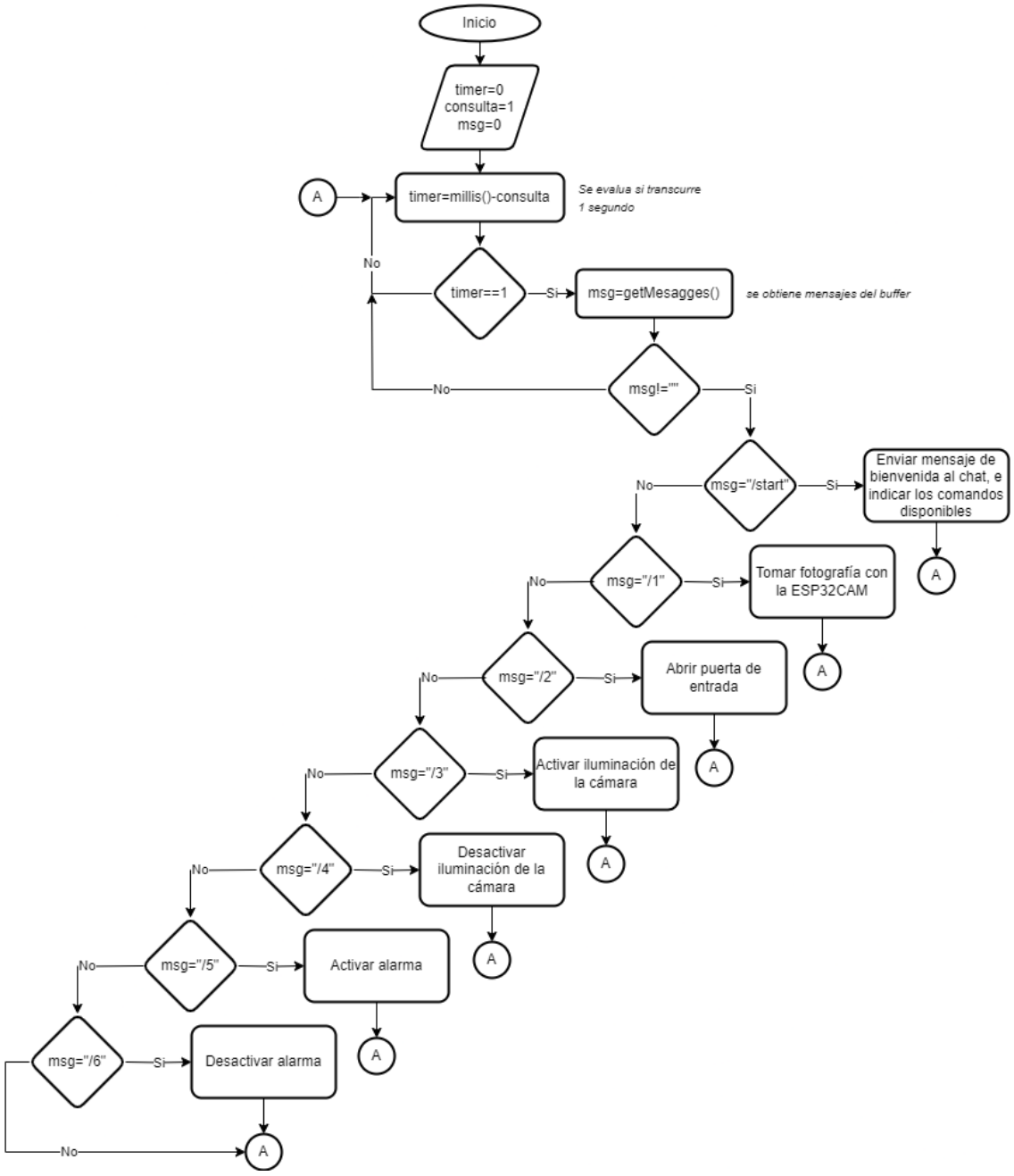

Figura 3.50. Diagrama de flujo para escucha de mensajes del *bot* de *Telegram*

FUNCIONAMIENTO DE SENSORES Y ACTUADORES DEL VIDEOPORTERO

Los actuadores pueden activarse o desactivarse a través de la plataforma de *Blynk* o mediante la de *Telegram*.

En el caso de *Telegram* se describió los comandos disponibles para manipular los actuadores, los cuales son: tomar foto, abrir cerradura, encender *flash* de la cámara y encender alarma del sistema.

En el caso de *Blynk*, los actuadores que se pueden controlar son: abrir cerradura, encender *flash* de la cámara, encender alarma (al activar la alarma se tiene una secuencia de encendido y apagado del *flash* de la cámara y del *buzzer*) y finalmente abrir el *streaming*.

Adicional, se tiene el pulsador que simula un timbre de un hogar, cuando este es pulsado se reproduce unos pitidos con el *buzzer* para alertar al dueño de la casa que alguien se encuentra en la entrada de la casa y posteriormente se llama a la función para tomar la fotografía con la ESP32-CAM para que sea enviada al chat del usuario.

Finalmente, se tiene el sensor magnético de la puerta, este sensor advierte al dueño de la casa si la puerta es forzada, ya que cuando las piezas del sensor son separadas se corta el voltaje que está ingresando al pin destinado a la conexión de dicho sensor. En caso de que la puerta se abra sin permiso, ósea sin que el actuador de la cerradura haya sido activado, se enciende la secuencia de la alarma para encender y apagar el *flash* de la cámara y el *buzzer* de manera repetitiva.

Pero en caso de se cumpla la condición de que la cerradura haya sido activa, ya sea por la aplicación de *Blynk* o de *Telegram*, esta secuencia de la alarma no debe ser activada, aunque el pin de la tarjeta se encuentre sin voltaje.

3.4 IMPLEMENTAR EL PROTOTIPO EN UNA MAQUETA

La maqueta desarrollada para demostrar el funcionamiento del sistema consta de dos partes.

DISEÑO DE LA ESTRUCTURA DE LA PUERTA

Para el diseño de la puerta, se toma la estructura básica de una puerta real que se tiene en los hogares, es decir, un marco que sostiene la puerta mediante bisagras.

Para el marco de la puerta se tiene las medidas de 24x37 (cm), en esta se añaden las bisagras necesarias para sujetar la puerta, la maqueta de la puerta se muestra en la **Figura 3.51**

Figura 3.51. Diseño del marco de la puerta

La puerta se realiza en medidas de 20x30 (cm), esta debe tener un espacio para anclar la cerradura electromagnética, tal como se muestra en la **Figura 3.52**

Figura 3.52. Diseño de la puerta

DISEÑO 3D PARA PROTOTIPO DE VIDEOPORTERO

En la búsqueda por acoplar los elementos principales al prototipo, se opta por diseñar un sólido que permita contener elementos principales como lo es el módulo ESP32-CAM, el pulsador y el *buzzer* del sistema. Este solido será acoplado a un lado de la puerta, mismo que simulará un videoportero para domicilio.

El diseño de este solido se lo realiza por medio del programa *SolidWorks*, donde se diseña en base a croquis y medidas milimétricas que garantizan la ubicación correcta de los elementos. Al finalizar el diseño se exporta el archivo en formato STL y se imprime en una impresora 3D en material PLA.

Para el diseño se consideró realizar 3 aberturas, en la primera abertura se coloca la cámara OV2640 de la tarjeta, en la segunda abertura se coloca el pulsador y finalmente en la tercera abertura se coloca el *buzzer*, en la **Figura 3.53** se muestra la parte frontal de la caja diseñada.

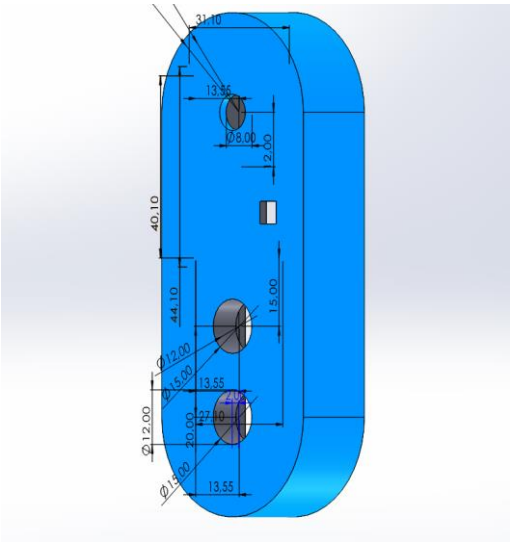

Figura 3.53. Parte frontal de la caja

En la parte posterior de la caja, se diseña soportes para los elementos mencionados, es decir, se dejan contornos donde van a embonar la tarjeta ESP32-CAM, el pulsador y el *buzzer*. Se puede visualizar la parte posterior del solido en la **Figura 3.54** y una vista en corte como se muestra en la **Figura 3.55**, esta vista muestra un destaje que servirán de soporte para que los elementos sean acoplados sin necesidad de usar un aditivo para sujeción.

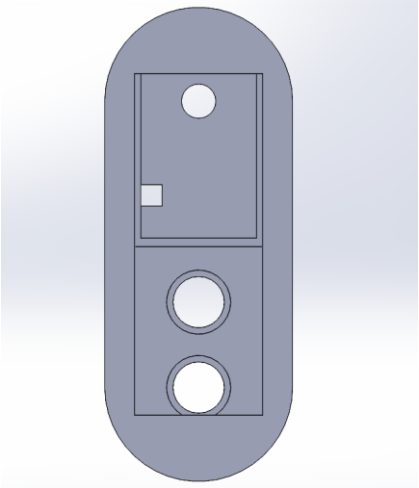

Figura 3.54. Parte posterior de la caja

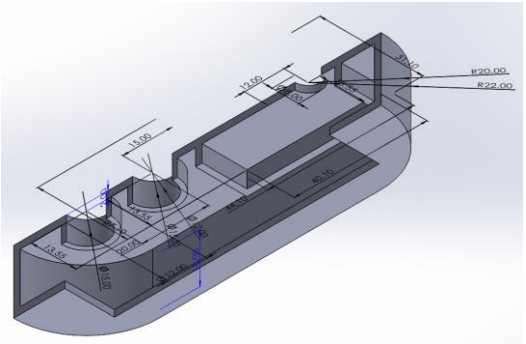

Figura 3.55 Vista con corte en la mitad

3.5 REALIZAR PRUEBAS DE FUNCIONAMIENTO DEL PROTOTIPO

FUNCIONAMIENTO DEL SISTEMA ANTE UN VISITANTE

Para las pruebas de funcionamiento se desea simular el comportamiento general de un visitante al llegar de visita al domicilio. El visitante por naturaleza se acerca a la puerta del domicilio y en la búsqueda por hacerle saber al usuario del domicilio que ha llegado, se encuentra con un diseño 3D como se muestra en la **Figura 3.56**, el mismo que se ha colocado en el extremo del marco de la puerta.

Figura 3.56 Ingreso al Domicilio

Al presentar estas características llamativas, por intuición, la persona va a presionar el pulsador del timbre, sonará el *buzzer* dando a entender que se está comunicando con el usuario del domicilio, se enciende el *flash* de la cámara como se observa en la **Figura 3.57** indicando al visitante que se encuentra capturando una fotografía. En este punto se notifica al usuario de la casa el mismo que puede optar por las acciones constituidas en los objetivos planteados anteriormente.

Figura 3.57 Encendido del prototipo

En la **Figura 3.58** se muestra que la llegada de la notificación que alguien se encuentra afuera del domicilio, en donde el usuario puede validar de que es un visitante esperado o a su vez puede realizar alguna de las acciones mediante la aplicación *Blynk* o via el *bot* de *Telegram.*

Figura 3.58 Notificación de visitante en la puerta

PRUEBAS DE FUNCIONAMIENTO CON *BLYNK*

Las pruebas de funcionamiento a realizar con *Blynk* corresponden a que, si se pulsa los botones de la interfaz desarrollada en *Blynk*, los actuadores deben encenderse o apagarse según corresponda.

En la **Figura 3.59** se muestra que la puerta se abre cuando se presiona el botón de abrir puerta.

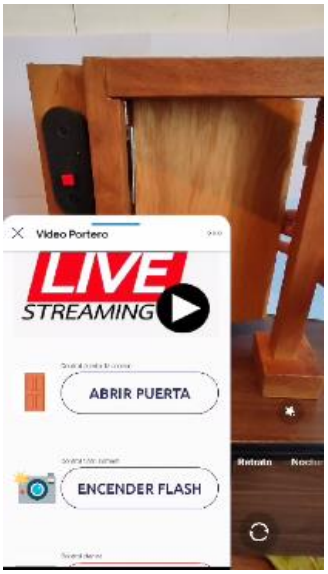

Figura 3.59. Apertura de la puerta mediante *Blynk*

En la **Figura 3.60** se muestra que se enciende el *flash* de la cámara cuando se presiona el botón de encender *flash*

Figura 3.60. Flash encendido mediante *Blynk*

En la **Figura 3.61** se muestra que la alarma se enciende cuando se presiona el botón de encender alarma.

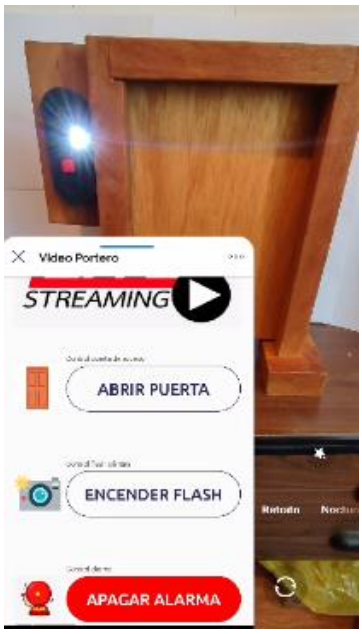

Figura 3.61. Alarma encendida mediante *Blynk*

En la **Figura 3.62** se muestra el video obtenido en la aplicación de *Blynk* cuando se presiona el botón de video *streaming*:

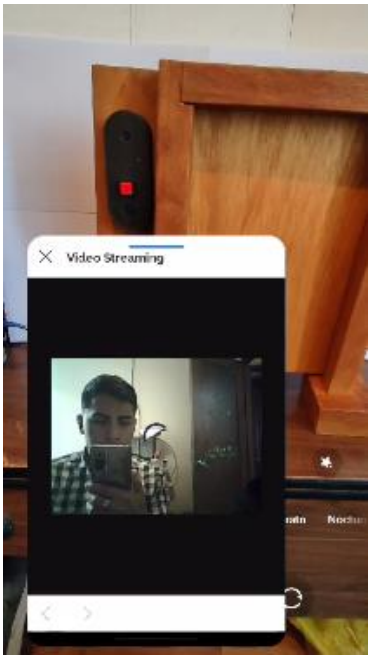

Figura 3.62. *Streaming* mediante *Blynk*

PRUEBAS DE FUNCIONAMIENTO CON *TELEGRAM*

Las pruebas de funcionamiento a realizar con *Telegram* corresponden a que, si se escriben los comandos propuestos, los actuadores deben encenderse o apagarse según corresponda.

En la **Figura 3.63** se presenta la fotografía tomada cuando se escribe el comando para realizar esta acción “/1”.

Figura 3.63. Comando para tomar foto en *Telegram*

En la **Figura 3.64** se observa la apertura de la puerta cuando se escribe el comando para realizar esta acción “/2”.

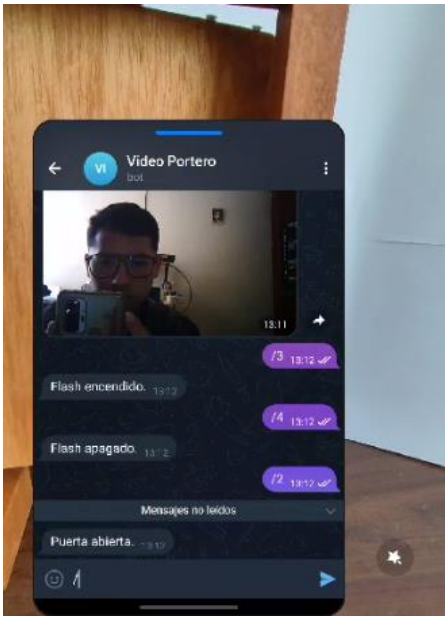

Figura 3.64. Comando para abrir puerta en *Telegram*

En la **Figura 3.65** se muestra que el *flash* de la cámara se enciende cuando se escribe el comando para realizar esta acción “/3”.

Figura 3.65. Comando para encender *flash* en *Telegram*

En la **Figura 3.66** se observa que el *flash* de la cámara se apaga cuando se escribe el comando para realizar esta acción “/4”.

Figura 3.66. Comando para apagar *flash* en *Telegram*

En la **Figura 3.67** se demuestra que la alarma del sistema se enciende cuando se escribe el comando para realizar esta acción “/5”.

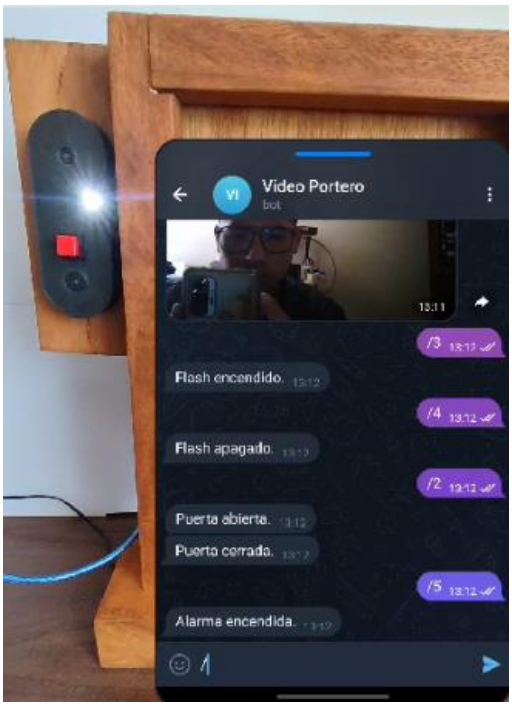

Figura 3.67. Comando para encender alarma en *Telegram*

En la **Figura 3.68** se muestra que la alarma del sistema se apaga cuando se escribe el comando para realizar esta acción “/6”.

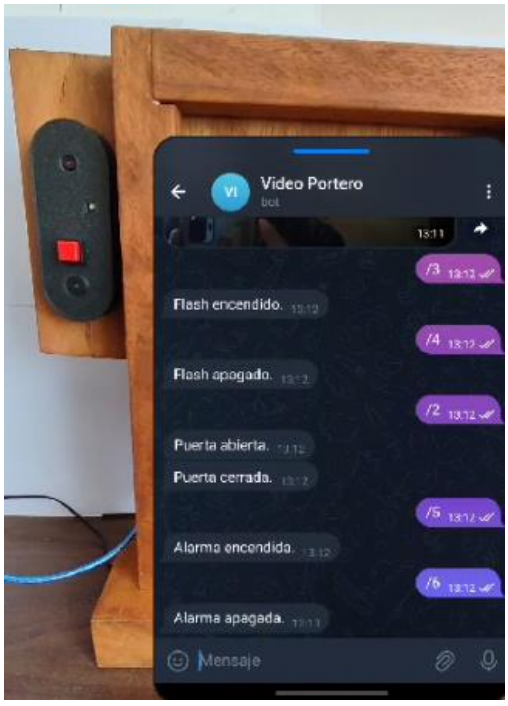

Figura 3.68. Comando para apagar alarma en *Telegram*

FUNCIONES ADICIONALES

Adicional, en la siguiente sección se muestran funciones adicionales del sistema como, por ejemplo, en la **Figura 3.69** se observa el mensaje que llega a *Telegram* cuando la puerta de entrada es forzada.

Figura 3.69. Mensaje en *Telegram* cuando se fuerza la puerta

En la **Figura 3.70** se muestra que cuando un usuario que ajeno a la casa trata de usar el *bot* de *Telegram* se le restringe el acceso.

Figura 3.70. Mensaje en *Telegram* para usuarios no autorizados

VIDEO DE FUNCIONAMIENTO DEL PROTOTIPO

En los anexos, se evidencia a profundidad el desempeño en tiempo real del prototipo el mismo que se invita al lector a visualizar y tener una retroalimentación vía multimedia. Esta retroalimentación permitirá validar el correcto y optimo funcionamiento del prototipo.

COSTO DEL PROTOTIPO

En la **Tabla 3.3** se muestra el costo de los materiales empleados en el desarrollo del prototipo, estos valores son obtenidos en referencia a Mercado Libre Ecuador.

Tabla 3.3. Costos del prototipo

Materiales	Cantidad	Precio unitario	Costo
Tarjeta ESP32-CAM	1 (u)	$ 13.00	$ 13.00
Cerradura electromagnética	1 (u)	$12.00	$ 12.00
Sensor magnético para puerta	1 (u)	$ 2.50	$ 2.50
Modulo relé 1 canal a 5 (V)	1 (u)	$ 1.50	$ 1.50
Buzzer 5 (V)	1 (u)	$ 1.00	$ 1.00
Pulsador N.A.	1 (u)	$ 1.00	$ 1.00
Resistencias 10 (kΩ)	2 (u)	$ 0.15	$ 0.30
Transistor 2n2222A	1 (u)	$ 0.15	$ 0.15
Cables varios	10 (u)	$ 0.30	$ 3.00
Impresión 3D de cubierta de timbre	1 (u)	$ 5.00	$ 5.00
Maqueta de puerta en madera	1 (u)	$ 20.00	$ 20.00
Mano de obra	20 (h)	$ 5.00	$ 100.00
		Total	**$159,45** **$139,45**

4 CONCLUSIONES

- Uno de los requerimientos principales identificados para desplegar el desarrollo del presente trabajo es la necesidad de un dispositivo que permita realizar *streaming* de video, en función de esto se determinó una característica clave para la selección del hardware del prototipo. El módulo ESP32-CAM cumple con todos los requerimientos planteados, siendo el principal el *streaming* de video a

través de la cámara OV2640 que cuenta con una resolución adecuada para visualizar con claridad el rostro de los visitantes

- En cuanto a la selección de la plataforma para el proyecto se determinó que *Blynk* es la más adecuada debido a que es la única como *plataforma IoT* que permite realizar *streaming* de video, y este es uno de los requerimientos principales del proyecto, adicionalmente, facilita la programación debido a que se la realiza por medio de bloques permitiendo optimizar el tiempo empleado para este fin.
- Con la finalidad de fortalecer la seguridad del prototipo, en la programación del dispositivo y de la plataforma se emplean tokens de seguridad, estos evitan que otras personas puedan ingresar al proyecto de *Blynk*, al *bot* de *Telegram* impidiendo que este pueda ser controlado por personas no autorizadas.
- Para mejorar la presentación del prototipo, así como proteger a los dispositivos que lo integran de una libre manipulación y de la acumulación de polvo, se diseñó un carcaza (case) 3D para montar los diferentes elementos del prototipo. Esto permite acoplar el módulo ESP32-CAM, el *buzzer* y el pulsador en un solo sitio, facilitando así su instalación en la maqueta y cumpliendo con los objetivos antes mencionados.
- La maqueta creada emula el ingreso al domicilio y fue realizada a escala, con el fin de replicar el ingreso a un domicilio real, esto permite demostrar su correcto funcionamiento y evidencia su eficacia en caso de ser empleado en un modelo real.
- Al realizar las pruebas de funcionamiento, todo el sistema se encuentra en óptimas condiciones, cumpliendo con cada uno de los objetivos que se deseaban alcanzar. Este proceso garantizó la fiabilidad y eficacia del sistema, sentando las bases para su eventual implementación a mayor escala.

5 RECOMENDACIONES

- Como trabajo a futuro se plantea la posibilidad de realizar un sistema más robusto usando elementos de mayor complejidad, por ejemplo, usar una *Raspberry Pi* para controlar todo el sistema, esto permite tener un mejor rendimiento debido a sus capacidades de hardware. Al tener este sistema más robusto se puede emplear una comunicación bidireccional con la persona que se encuentra fuera del domicilio, identificar por medio de inteligencia artificial si se trata del usuario o un visitante, entre otras.

- Debido a que este sistema es completamente electrónico, debe permanecer conectado a la toma eléctrica de manera permanente, por lo que en caso de un corte en la red eléctrica el sistema quedaría sin funcionamiento. Se sugiere implementar un respaldo de energía mediante un UPS para asegurar que el circuito siempre este energizado o a su vez, implementar un sistema de apertura y bloqueo manual para la puerta del hogar.
- El registro de usuarios para el prototipo actual se lo escribe en código mediante los tokens de cada plataforma, para ello se invita a desarrollar un prototipo que permita realizar un registro de manera dinámica sin la necesidad de que sea una variable en el sistema.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] Verisure, «Sistemas de seguridad: qué son y tipos,» Diciembre 2018. [En línea]. Available: https://www.verisure.pe/consejos-y-ayuda/preguntas-frecuentes/que-son-sistemas-de-seguridad. [Último acceso: 26 Junio 2023].

[2] E. Sarachu, «¿Qué es la domótica? ¿Cómo funciona?,» E-ficiencia, 10 Abril 2023. [En línea]. Available: https://e-ficiencia.com/domotica-que-es-y-como-funciona/. [Último acceso: 26 Junio 2023].

[3] «Revista Seguridad 360,» [En línea]. Available: https://revistaseguridad360.com/destacados/que-es-un-videoportero/. [Último acceso: 07 06 2023].

[4] Microcontroladores, «Todo Sobre Microcontroladores,» Agosto 2017. [En línea]. Available: https://microcontroladoress.com/. [Último acceso: 26 Junio 2023].

[5] C. Pascual, «ESP32 CAM,» 2019. [En línea]. Available: https://programarfacil.com/esp32/esp32-cam/. [Último acceso: 26 Junio 2023].

[6] Raspberry Pi, «¿Que es Raspberry Pi?,» [En línea]. Available: https://raspberrypi.cl/que-es-raspberry/. [Último acceso: 26 Junio 2023].

[7] «Arducam,» [En línea]. Available: https://www.arducam.com/ov2640/. [Último acceso: 07 06 2023].

[8] Cognizant, «Plataforma del internet de las cosas,» 2021. [En línea]. Available: https://www.cognizant.com/es/es/glossary/internet-of-things-platform. [Último acceso: 26 Junio 2023].

[9] «Blynk: a low-code IoT software platform for businesses and developers,» [En línea]. Available: https://blynk.io/. [Último acceso: 08 06 2023].

[10] Aunoa.ai, «¿Qué es un chatbot conversacional?,» Junio 2018. [En línea]. Available: https://aunoa.ai/bot-conversacional/. [Último acceso: 02 Agosto 2023].

[11] TELEGRAM, «¿Qué es Telegram?,» Octubre 2013. [En línea]. Available: https://telegram.org/faq/es#p-que-es-telegram-que-puedo-hacer-aqui. [Último acceso: 02 Agosto 2023].

[12] «Aprendiendo Arduino,» [En línea]. Available: https://aprendiendoarduino.wordpress.com/2016/12/11/ide-arduino/. [Último acceso: 07 06 2023].

[13] A. S.A.U., «Amidata S.A.U,» [En línea]. Available: https://es.rs-online.com/web/p/camaras-para-raspberry-pi/9132664. [Último acceso: 07 06 2023].

[14] C. Cinjordiz, «INFFOTEC.net,» [En línea]. Available: https://www.infootec.net/arduino/. [Último acceso: 07 06 2023].

[15] I. Importadora, «Impormel Importadora,» [En línea]. Available: http://www.impormel.com/domotica/1805-sensor-contacto-magnetico-para-alarma-en-puertas-ventanas-nc.html. [Último acceso: 07 06 2023].

[16] Novatronic, «Novatronic,» [En línea]. Available: https://novatronicec.com/index.php/product/cerradura-electrica-solenoide-dc12v-654030/. [Último acceso: 06 07 2023].

[17] A. Electronics, «AV Electronics,» [En línea]. Available: https://avelectronics.cc/producto/modulo-rele-1-canal/. [Último acceso: 07 06 2023].

[18] G. E. Store, «Grupo Electro Store,» [En línea]. Available: https://grupoelectrostore.com/shop/componentes-electronicos-basicos/buzzers/buzzer-zumbador-5v-activo/. [Último acceso: 07 06 2023].

[19] Arduino, «Arduino,» [En línea]. Available: https://cloud.arduino.cc/. [Último acceso: 07 06 2023].

[20] «Programar fácil con Arduino,» [En línea]. Available: https://programarfacil.com/esp32/esp32-cam/. [Último acceso: 07 06 2023].

7 ANEXOS

La lista de Anexos se muestra a continuación:

ANEXO I. Certificado de originalidad

ANEXO II. Enlaces

ANEXO III. Conjunto de datos extensos

ANEXO I: CERTIFICADO DE ORIGINALIDAD

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 25 de agosto de 2023

De mi consideración:

Yo, **LEANDRO ANTONIO PAZMIÑO ORTIZ**, en calidad de Director del Trabajo de Integración Curricular titulado **IMPLEMENTACIÓN DE UN PROTOTIPO DE TIMBRE CON VISUALIZACIÓN DE LA PERSONA QUE DESEA INGRESAR Y APERTURA DE LA PUERTA, POR MEDIO DE UNA PLATAFORMA IOT** elaborado por el estudiante **BENEDICT ALEXANDER VILLA TIPAN** de la carrera en **TECNÓLOGÍA SUPERIOR EN REDES Y TELECOMUNICACIONES**, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 12 %.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: En la carpeta compartida de entrega del TIC se encuentra el reporte completo generado por Turnitin.

Atentamente,

Leandro Pazmiño Ortiz

Docente

Escuela de Formación de Tecnólogos

ANEXO II: ENLACES

Anexo II.I Código QR de la implementación y pruebas de funcionamiento

Link: Benedict_VIlla_Video Final.mp4

ANEXO III: CÓDIGOS FUENTE

```
//librerias necesarias
#include "OV2640.h"
#include <WiFi.h>
#include <WebServer.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include "fd_forward.h"
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "SPI.h"
#include "ESP32_MailClient.h"
#include <FS.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>

//definir modelo de la camara (ESP32CAM)
#define CAMERA_MODEL_AI_THINKER
#include "camera_pins.h"

//credenciales de red
#define SSID1 " ******** "  //cambiar por las credenciales del domicilio
#define PWD1 " ******** " //cambiar por las credenciales del domicilio

//pines a utilizar
#define P_FLASH 4
#define P_CERRADURA 14
#define P_BOCINA 15
#define P_TIMBRE 12
#define P_PUERTA 16

//credenciales blynk
char auth[] = " ******** "; //cambiar por el Token en Blynk

//credenciales telegram
String token_bot_telegram = " ******** "; //cambiar por Token del bot en Telegram
```

```
String id_del_chat = " ******** "; //cambiar por el ID en Telegram

//objetos
BlynkTimer timer;
SMTPData smtpData;

//nombre del archivo a guardar de la foto
#define ARCHIVO_FOTO "/foto.jpg"
//objetos auxiliares
WiFiClientSecure clientTCP;
UniversalTelegramBot bot(token_bot_telegram, clientTCP);
//lecturas de tiempo
int tiempo_lectura_bot = 1000;
unsigned long ultima_consulta_bot;

OV2640 cam;
WebServer server(80);

//header necesario para streaming
const char HDR[] = "HTTP/1.1 200 OK\r\n"
                   "Access-Control-Allow-Origin: *\r\n"
                   "Content-Type:                    multipart/x-mixed-replace;
boundary=123456789000000000000987654321\r\n";
const char BDR[] = "\r\n--123456789000000000000987654321\r\n";
const char CTN[] = "Content-Type: image/jpeg\r\nContent-Length: ";


const int hdrLong = strlen(HDR);
const int bdrLong = strlen(BDR);
const int cntLong = strlen(CTN);

int out1 = 0;
int out3 = 0;
bool enviar_foto = false;
bool open_puerta = false;
bool on_alarma = false;
bool una = false;
bool dos = false;
```

```
bool abertura = false;

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);   //quitar notificaciones de voltaje
  Serial.begin(115200);

  //configurar camara
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 10;
  config.fb_count = 1;
  cam.init(config);

  WiFi.mode(WIFI_STA);
  Blynk.begin(auth, SSID1, PWD1, "blynk.cloud", 80);
  clientTCP.setCACert(TELEGRAM_CERTIFICATE_ROOT);
```

```
//verificar conexion a wifi
int i = 0;
while (WiFi.status() != WL_CONNECTED) {
  Serial.print(".");
  i++;
  if (i > 20) ESP.restart();
  delay(500);
}

Serial.println(F("WiFi conectado"));
Serial.print("Link del stream: http://");
Serial.print(WiFi.localIP());
Serial.println("/mjpeg/1");

//timer para metodo alarma se comprueba estado cada 20ms
timer.setInterval(20, alarma);

//definir pines
pinMode(P_FLASH, OUTPUT);
pinMode(P_CERRADURA, OUTPUT);
pinMode(P_BOCINA, OUTPUT);
pinMode(P_TIMBRE, INPUT);
pinMode(P_PUERTA, INPUT);

digitalWrite(P_FLASH, LOW);
digitalWrite(P_CERRADURA, LOW);
digitalWrite(P_BOCINA, LOW);

//ejecutar servidor segun headers
server.on("/mjpeg/1", HTTP_GET, TransmisionVideo);
server.onNotFound(TransmisionError);
server.begin();
//encender flash para indicar finalizacion de configuracion
for (int i = 0; i <= 3; i++) {
  digitalWrite(P_FLASH, HIGH);
  delay(500);
  digitalWrite(P_FLASH, LOW);
  delay(500);
```

```
  }
}

void loop() {
  //cuando se tiene un cliente, iniciar stream
  server.handleClient();
  //si se timbra tomar fotografia y enviar por telegram
  if (digitalRead(P_TIMBRE)) {
    suena_timbre();
    bot.sendMessage(id_del_chat, "Alguien esta en tu puerta.", "");
  }

  if (enviar_foto) {
    tomar_foto();
    enviar_foto = false;
    delay(100);
  }

  //si se abre el sensor magnetico y el cerrojo no fue abierto, se activa la alarma
  if ((!digitalRead(P_PUERTA) && abertura == false && open_puerta == false)) {
    digitalWrite(P_BOCINA, HIGH);
    digitalWrite(P_FLASH, HIGH);
    delay(100);
    digitalWrite(P_BOCINA, LOW);
    digitalWrite(P_FLASH, LOW);
    delay(100);
    if (!una) {
      bot.sendMessage(id_del_chat, "Tu puerta fue forzada.", "");
      una = true;
    }
  }

  if (digitalRead(P_PUERTA)) {
    if (abertura || open_puerta) {
      bot.sendMessage(id_del_chat, "Puerta cerrada.", "");
    }
    delay(250);
    una = false;
```

```
    abertura = false;
    open_puerta = false;
  }

  if (on_alarma) {
    digitalWrite(P_BOCINA, HIGH);
    digitalWrite(P_FLASH, HIGH);
    delay(100);
    digitalWrite(P_BOCINA, LOW);
    digitalWrite(P_FLASH, LOW);
    delay(100);
  }

  if (millis() > ultima_consulta_bot + tiempo_lectura_bot) {
    int numero_mensajes = bot.getUpdates(bot.last_message_received + 1);
    while (numero_mensajes) {
      Serial.println("Mensaje: ");
      leer_mensajes(numero_mensajes);
      numero_mensajes = bot.getUpdates(bot.last_message_received + 1);
    }
    ultima_consulta_bot = millis();
  }

  //blynk en escucha
  Blynk.run();
  timer.run();
}

//Stream de video
void TransmisionVideo(void) {
  char buffer_video[32];
  int ax;

  WiFiClient client = server.client();

  client.write(HDR, hdrLong);
  client.write(BDR, bdrLong);
```

```
  //si esque el servidor detecta una conexion de un cliente, envia fotogramas al servidor
  while (true) {
    if (!client.connected())
      break;

    cam.run();  //metodo de libreria para enviar fotogramas
    ax = cam.getSize();
    client.write(CTN, cntLong);
    sprintf(buffer_video, "%d\r\n\r\n", ax);
    client.write(buffer_video, strlen(buffer_video));
    client.write((char*)cam.getfb(), ax);
    client.write(BDR, bdrLong);  //todo el buffer de fotogramas se envian al servidor
  }
}

const char JHDR[] = "HTTP/1.1 200 OK\r\n"
                    "Content-disposition: inline; filename=capture.jpg\r\n"
                    "Content-type: image/jpeg\r\n\r\n";  //configuracion del server (va a recibir datos de tipo imagen (200 transmision ok)) este header es para ver netamente una imagen
const int jhdLong = strlen(JHDR);


//estado del servidor
void TransmisionError() {
  String mensaje_error = "Server is running!\n\n";
  mensaje_error += "URI: ";
  mensaje_error += server.uri();
  mensaje_error += "\nMethod: ";
  mensaje_error += (server.method() == HTTP_GET) ? "GET" : "POST";
  mensaje_error += "\nArguments: ";
  mensaje_error += server.args();
  mensaje_error += "\n";
  server.send(200, "text / plain", mensaje_error);
}


//metodos auxiliares se llama este metodo cuando se timbra
```

```
void suena_timbre() {
  bool one_time = false;
  Blynk.logEvent("timbre_puerta");

  while (one_time == false) {
    //timbre
    for (int i = 0; i <= 2; i++) {  //sonar sonido tipo timbre solo se ejecuta una vez en todo
el bucle
      digitalWrite(P_BOCINA, HIGH);
      delay(500);
      digitalWrite(P_BOCINA, LOW);
      delay(500);
    }
    digitalWrite(P_BOCINA, HIGH);
    delay(1500);
    digitalWrite(P_BOCINA, LOW);
    one_time = true;
  }
  //llamar metodo tomar foto y mandar por telegram
  tomar_foto();
}

void leer_mensajes(int numero_mensajes) {
  Serial.print("# de mensajes: ");
  Serial.println(numero_mensajes);

  for (int i = 0; i < numero_mensajes; i++) {

    String chat_entrante = String(bot.messages[i].chat_id);
    if (id_del_chat != chat_entrante) {
      bot.sendMessage(chat_entrante, "Usuario no autorizado.", "");
      continue;
    }

    String mensaje_texto = bot.messages[i].text;
    Serial.println(mensaje_texto);

    String nombre_usuario = bot.messages[i].from_name;
```

```
    if (mensaje_texto == "/start") {
      String mensaje_mostrado = "Bienvenido , " + nombre_usuario + "\n";
      mensaje_mostrado += "Use los siguientes comandos para interactuar con el video portero\n";
      mensaje_mostrado += "/1 : Tomar fotografia \n";
      mensaje_mostrado += "/2 : Abrir puerta \n";
      mensaje_mostrado += "/3 : Encender flash de camara \n";
      mensaje_mostrado += "/4 : Apagar flash de camara \n";
      mensaje_mostrado += "/5 : Encender alarma \n";
      mensaje_mostrado += "/6 : Apagar alarma \n";
      bot.sendMessage(chat_entrante, mensaje_mostrado, "");
    }
    if (mensaje_texto == "/1") {
      enviar_foto = true;
    }
    if (mensaje_texto == "/2") {
      digitalWrite(P_CERRADURA, HIGH);
      bot.sendMessage(chat_entrante, "Puerta abierta.", "");
      delay(3000);
      digitalWrite(P_CERRADURA, LOW);
      open_puerta = true;
      delay(100);
    }
    if (mensaje_texto == "/3") {
      digitalWrite(P_FLASH, HIGH);
      bot.sendMessage(chat_entrante, "Flash encendido.", "");
    }
    if (mensaje_texto == "/4") {
      digitalWrite(P_FLASH, LOW);
      bot.sendMessage(chat_entrante, "Flash apagado.", "");
    }
    if (mensaje_texto == "/5") {
      on_alarma = true;
      bot.sendMessage(chat_entrante, "Alarma encendida.", "");
    }
    if (mensaje_texto == "/6") {
      on_alarma = false;
      bot.sendMessage(chat_entrante, "Alarma apagada.", "");
```

```
    }
  }
}

String tomar_foto() {
  //bool enviarPhoto = false;
  digitalWrite(P_FLASH, HIGH);
  const char* dominio_telegram = "api.telegram.org";
  String encabezado_mensaje = "";
  String cuerpo_mensaje = "";

  camera_fb_t* frame_cam = NULL;
  frame_cam = esp_camera_fb_get();
  if (!frame_cam) {
    Serial.println("Falla en la camara");
    delay(1000);
    return "Falla en la camara";
  }

  if (clientTCP.connect(dominio_telegram, 443)) {
    Serial.println("Conexion a Telegram establecida.");

    String head_telegram = "--videoportero\r\nContent-Disposition: form-data; name=\"chat_id\"; \r\n\r\n" + id_del_chat + "\r\n--videoportero\r\nContent-Disposition: form-data; name=\"photo\"; filename=\"esp32-cam.jpg\"\r\nContent-Type: image/jpeg\r\n\r\n";
    String tail_telegram = "\r\n--videoportero--\r\n";

    uint16_t imageLong = frame_cam->len;
    uint16_t extraLong = head_telegram.length() + tail_telegram.length();
    uint16_t totalLong = imageLong + extraLong;

    clientTCP.println("POST /bot" + token_bot_telegram + "/sendPhoto HTTP/1.1");
    clientTCP.println("Host: " + String(dominio_telegram));
    clientTCP.println("Content-Length: " + String(totalLong));
    clientTCP.println("Content-Type: multipart/form-data; boundary=videoportero");
    clientTCP.println();
    clientTCP.print(head_telegram);
```

```
  uint8_t* fbBuffer = frame_cam->buf;
  size_t fbLong = frame_cam->len;
  for (size_t num = 0; num < fbLong; num = num + 1024) {
    if (num + 1024 < fbLong) {
      clientTCP.write(fbBuffer, 1024);
      fbBuffer += 1024;
    } else if (fbLong % 1024 > 0) {
      size_t rem = fbLong % 1024;
      clientTCP.write(fbBuffer, rem);
    }
  }

  clientTCP.print(tail_telegram);
  esp_camera_fb_return(frame_cam);

  int tiempo_espera = 50000;  // timeout 10 seconds
  long tiempo_inicio = millis();
  boolean comprobar_estado = false;

  while ((tiempo_inicio + tiempo_espera) > millis()) {
    Serial.print(".");
    delay(100);
    while (clientTCP.available()) {
      char clt = clientTCP.read();
      if (comprobar_estado == true) cuerpo_mensaje += String(clt);
      if (clt == '\n') {
        if (encabezado_mensaje.length() == 0) comprobar_estado = true;
        encabezado_mensaje = "";
      } else if (clt != '\r')
        encabezado_mensaje += String(clt);
      tiempo_inicio = millis();
    }
    if (cuerpo_mensaje.length() > 0) break;
  }
  clientTCP.stop();
  Serial.println(cuerpo_mensaje);
} else {
```

```
    cuerpo_mensaje = "Fallo conexion a api.telegram.org";
    Serial.println("Fallo conexion a api.telegram.org");
  }
  digitalWrite(P_FLASH, LOW);
  return cuerpo_mensaje;
}


//lectura pines virtuales de Blynk
BLYNK_WRITE(V6) {
  out1 = param.asInt();
  if (out1 == 1) {
    digitalWrite(P_CERRADURA, HIGH);
    delay(3000);
    digitalWrite(P_CERRADURA, LOW);
    abertura = true;
    delay(100);
  }
}

//v7 iluminacion
BLYNK_WRITE(V7) {
  int out2 = param.asInt();
  if (out2 == 0) {
    digitalWrite(P_FLASH, LOW);
  } else if (out2 == 1) {
    digitalWrite(P_FLASH, HIGH);
  }
}

//v8 alarma
BLYNK_WRITE(V8) {
  out3 = param.asInt();
  if (out3 == 0) {
    digitalWrite(P_BOCINA, LOW);
    digitalWrite(P_FLASH, LOW);
  }
}
```

```
void alarma() {
  if (out3 == 1) {
    digitalWrite(P_BOCINA, HIGH);
    digitalWrite(P_FLASH, HIGH);
    delay(100);
    digitalWrite(P_BOCINA, LOW);
    digitalWrite(P_FLASH, LOW);
    delay(100);
  }
}
```