UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE INGENIERÍA ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE SEGURIDAD ESTILO BOTÓN DE PÁNICO TIPO TÓTEM Y SU SISTEMA DE GESTIÓN, PARA EL CAMPUS DE LA UNIVERSIDAD POLITÉCNICA SALESIANA SEDE CUENCA

Trabajo de titulación previo a la obtención
del título de Ingeniero Electrónico

AUTOR: LUIS EDUARDO OLMEDO CUEVA
TUTOR: ING. JUAN DIEGO JARA SALTOS, MgT.

Cuenca - Ecuador
2023

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Luis Eduardo Olmedo Cueva con documento de identificación N° 1150124780, manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Cuenca, 29 de septiembre de 2023

Atentamente,

Luis Eduardo Olmedo Cueva
1150124780

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Luis Eduardo Olmedo Cueva con documento de identificación N° 1150124780, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del Proyecto técnico: "Diseño e implementación de un sistema de seguridad estilo botón de pánico tipo tótem y su sistema de gestión, para el campus de la Universidad Politécnica Salesiana sede Cuenca", el cual ha sido desarrollado para optar por el título de: Ingeniero Electrónico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, 29 de septiembre de 2023

Atentamente,

Luis Eduardo Olmedo Cueva
1150124780

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Juan Diego Jara Saltos con documento de identificación N° 0103543658, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE SEGURIDAD ESTILO BOTÓN DE PÁNICO TIPO TÓTEM Y SU SISTEMA DE GESTIÓN, PARA EL CAMPUS DE LA UNIVERSIDAD POLITÉCNICA SALESIANA SEDE CUENCA, realizado por Luis Eduardo Olmedo Cueva con documento de identificación N°1150124780, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Cuenca, 29 de septiembre de 2023

Atentamente,

Ing. Juan Diego Jara Saltos, MgT.
0103543658

AGRADECIMIENTOS

En primer lugar, quiero expresar mi profundo agradecimiento a Dios y a la Virgen del Cisne por darme la oportunidad de estar aquí y por otorgarme la vida. No puedo dejar de agradecer de todo corazón a mi madre y a mi hermano, cuyo apoyo inquebrantable ha sido fundamental para alcanzar este logro.

Asimismo, deseo manifestar mi más sincero agradecimiento al ingeniero Juan Diego Jara Salto, mi tutor, por brindarme su invaluable orientación y apoyo incondicional a lo largo de todo el proceso de elaboración de mi tesis. Sus consejos y conocimientos fueron fundamentales para culminar este proyecto.

Por último, quiero expresar mi gratitud a todos los docentes de la Universidad Politécnica Salesiana y a mis amigos, quienes han contribuido de manera significativa a mi crecimiento personal y académico a lo largo de este camino.

LUIS

DEDICATORIA

Dedico este trabajo a dos personas especiales en mi vida: mi madre, Carmen, y mi hermano, Fernando. Su apoyo incondicional ha sido un faro en mi camino durante toda mi formación académica. También quiero expresar mi gratitud a mi familia y amigos, con un agradecimiento especial para mi amigo John, cuya colaboración fue esencial para llevar a cabo las pruebas y culminar exitosamente este proyecto.

LUIS

Índice general

Índice de figuras

Índice de tablas

Resumen

La inseguridad en Ecuador en estos años ha ido incrementando exponencialmente. Los robos y asaltos cerca de instituciones educativas son cada vez mas frecuentes. En el presente proyecto se propone el diseño e implementación de un sistema de seguridad estilo botón de pánico tipo tótem con su sistema de gestión. Para la alimentación del mismo se realizara por medio de paneles solares.

En el capitulo 1 se presenta datos estadísticos de la inseguridad en el Ecuador e introduciremos los conceptos teóricos relacionados con las herramientas, tanto de hardware como de software, que se emplearon en el diseño e implementación del botón de pánico tipo tótem.

En el capitulo 2 se detalla las características fundamentales de los dispositivos electrónicos que se van a utilizar, el software para el desarrollo web y configuración de los puertos GPIO, de la centralita y usuarios SIP en la Raspberry Pi, además los cálculos de la autonomía del sistema de alimentación, finalmente se presentan los diseños de bloques y conexiones del sistema.

En el capitulo 3 se realiza la implementación del sistema de seguridad colocando los equipos utilizados dentro de la estructura tipo tótem, se realiza un análisis económico y adicional las conexiones y pruebas de funcionamiento.

Finalmente, se exponen las conclusiones, recomendaciones y posibles líneas de investigación futuras. La seguridad es un aspecto sumamente relevante en cualquier entorno, por lo que es crucial promover el uso de la tecnología para fortalecer sistemas seguros y confiables, contribuyendo así al mejoramiento del bienestar de las personas.

Palabras clave: Raspberry Pi; Sistema de seguridad; Botón de pánico;GPIO; tótem; centralita; SIP.

Abstract

Insecurity in Ecuador has been increasing exponentially in recent years. Robberies and assaults near educational institutions are becoming more and more frequent. In this project we propose the design and implementation of a totem type panic button security system with its management system. It will be powered by solar panels.

Chapter 1 presents statistical data on insecurity in Ecuador and introduces the theoretical concepts related to the tools, both hardware and software, that were used in the design and implementation of the totem panic button.

Chapter 2 details the fundamental characteristics of the electronic devices to be used, the software for web development and configuration of the GPIO ports, the switchboard and SIP users on the Raspberry Pi, in addition to the calculations of the autonomy of the power supply system, and finally the block designs and system connections are presented.

In chapter 3, the implementation of the security system is carried out by placing the equipment used inside the totem type structure, an economic analysis is performed, as well as the connections and functional tests.

Finally, conclusions, recommendations and possible lines of future research are presented. Security is an extremely relevant aspect in any environment, so it is crucial to promote the use of technology.

Keywords: Raspberry Pi;Security system; Panic button; GPIO (General Purpose Input/Output); Totem;telephone switchboard; Session Initiation Protocol-SIP ; .

Antecedentes

En Ecuador, durante el período que va desde el 1 de enero hasta el 20 de marzo de 2022, las entidades pertinentes registraron un total de 815 fallecimientos violentos. Sin embargo, en el lapso equivalente de 78 días en 2023, se reportaron 1.356 incidentes, lo que representa un aumento del 66,4 %. En el año 2022 hubo un promedio de 10,4 casos diarios de 25 a comparación de lo que va del primer semestre del 2023 es de 17.4 casos. El 85 % de los fallecimientos tienen lugar en las áreas de Prosperina, Sur de Guayaquil, Esmeraldas, Quevedo, Pascuales, Manta, Portete, Esteros, Florida, Portoviejo, Milagro, Salinas, Durán y Balzar, según lo anunciado por el titular de la cartera de Interior **antecedentes**.

De acuerdo con las encuestas llevadas a cabo en junio de este año por el Diario La Hora, se revela que 3 de cada 10 estudiantes de las universidades de Quito han sido víctimas de robo o intento de robo en alguna ocasión. Los lugares con una mayor incidencia de asaltos o robos abarcan el transporte público, áreas de entretenimiento y las vías públicas [1].

Con el propósito de disminuir los niveles de delincuencia dirigida contra los estudiantes universitarios, los rectores de las universidades de la ciudades de Cuenca y Azogues han desarrollado un plan integral. Este plan busca salvaguardar a los estudiantes y, de esta manera, contrarrestar los asaltos que ocurren después de las clases y en los bares situados en las cercanías de las instituciones educativas[2].

Justificación

Ante el incremento de la inseguridad en espacios públicos, universidades y áreas con alta concentración de personas, resulta oportuno aprovechar la tecnología contemporánea para abordar situaciones de emergencia de forma ágil y eficaz. Un sistema de seguridad elaborado de forma independiente suele ser más económico en comparación con alternativas disponibles en el mercado. Se propone diseñar e implementar un sistema de seguridad estilo botón de pánico tipo tótem gestionado vía software dentro de la Universidad Politécnica Salesiana sede Cuenca Con este sistema se pretende ubicar tótems de seguridad en lugares estratégicos para mejorar la seguridad de estudiantes y personal que labora en el campus de la Universidad.

Objetivos

Objetivo General

- Diseñar e implementar un sistema electrónico de seguridad estilo botón de pánico tipo tótem y su sistema de gestión para informar sobre emergencias dentro del campus universitario.

Objetivos específicos:

- Realizar una revisión bibliográfica sobre los dispositivo y/o aplicaciones de seguridad diseñadas e implementadas en los campus universitarios.
- Realizar el levantamiento de información de las especificaciones del escenario de implementación y las especificaciones de operación del sistema en el campus.
- Diseñar e implementar el sistema de seguridad que se adapte a las especificaciones analizadas.
- Implementación del Sistema de Gestión, mediante software que permita una interacción "Front End" de manera amigable para los operadores del centro de atención de emergencias.
- Validar el sistema de seguridad del botón de pánico mediante pruebas a través del sistema de gestión dentro de la Universidad Politécnica Salesiana sede Cuenca, utilizando una infraestructura tipo tótem movible para las pruebas.

Introducción

En los últimos años la inseguridad en Ecuador ha ido en aumento, según los informes de la Policía Nacional indican que entre enero y junio del presente año se han documentado 3.513 casos de homicidio, mostrando un incremento del 58 % en comparación con el año 2022.[3]. Según Diario El Mercurio, los lugares de más índice delictivo a estudiantes en la ciudad Cuenca son en las paradas de buses y bares que se encuentran alrededor de las universidades [2].

Es de vital importancia que tanto los estudiantes como el personal de la universidad experimenten un sentido de seguridad al interior del campus. Para ello se propone el diseño e implementación de un sistema de seguridad estilo botón de pánico tipo tótem con su sistema de gestión, este sistema estará equipado con una fuente de alimentación renovable, como son los paneles solares, Con el propósito de instalarlo en zonas externas de alta concurrencia, permitiendo así abordar emergencias en el campus de la Universidad Politécnica Salesiana.

Capítulo 1

FUNDAMENTACIÓN TEÓRICA

En este capítulo presentamos los conceptos teóricos de las herramientas tanto de hardware como de software utilizadas para el diseño e implementación del botón de pánico tipo tótem con su respectivo sistema de gestión. Como primer punto se realizó una revisión de información de los dispositivos de Emergencia que se encuentran implementados en diferentes universidades.

1.1. Factores de Inseguridad

La preocupación por la inseguridad en Ecuador es compartida por muchos, dado que los homicidios ocurren a cualquier hora del día. Desafortunadamente, hemos presenciado el surgimiento de un nuevo modus operandi: los secuestros y extorsiones, los cuales lamentablemente se han vuelto cada vez más frecuentes. Según el Diario La Prensa, con el objetivo de abordar de manera efectiva la inseguridad y la delincuencia en Ecuador, existen factores primordiales para reducir estas problemáticas. Uno de ellos radica en la lucha contra la pobreza, acompañada de iniciativas para motivar a los adolescentes a acceder a centros educativos, evitando que se involucren en actividades criminales. Asimismo, es importante tener en cuenta la continua presencia de la corrupción y el crimen organizado a lo largo de los años. Por último, no se puede pasar por alto el desafío que implica el tráfico de drogas, ya que Ecuador es reconocido como un país de tránsito para exportar sustancias controladas a otras naciones [4].

En la Figura 1.1 se puede visualizar las provincias con mayor número de muertes

violentas desde el 2012 hasta el 2022. Se observa que la provincia de Guayaquil tiene la mayor tasa.

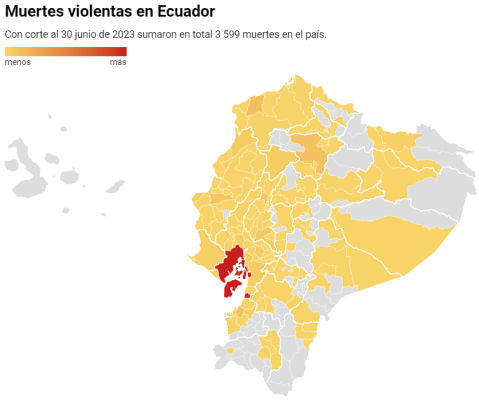


Figura 1.1: Indicadores por provincia de muertes violentas en Ecuador. [5]

1.2. Levantamiento de información

En universidades de Estados Unidos en la década de los 70's se implementó sistemas de emergencia conocidos como "blue-light emergency phones", estos tuvieron gran acogida porque ayudaron a tener un ambiente más seguro dentro del campus, lo que caracterizó a estas infraestructuras fue la interacción inmediata con el personal de seguridad, con forme a pasado los años ha aumentado la implementación y diseño de estas estructuras, a continuación se presentan algunas de ellas en diferentes universidades.

En el año 2015 The University of British Columbia(UBC) [6] implementó un sistema de seguridad tipo totem, el mismo que cuenta con un botón que al presionar dicho pulsante se realizara una llamada directa al teléfono del departamento de seguridad del campus de Vancouver. La UBC ubicó cerca de 35 teléfonos azuales de emergencia en el campus, a este sistema de seguridad en el 2016 se le agregaron cámaras de vídeo, las mismas que son monitorizadas desde el área de seguridad del campus.

Figura 1.2: Emergency Blue Phone Camera System en Universidad de la Columbia Britanica. [6]

La Universidad del Oeste de California tiene ubicados 65 sistemas de emergencia llamados Emergency Blue Light Telephones, éstos por lo general están ubicados en el estacionamiento, áreas de deporte, pasillos y entradas de los edificios. Para activar este sistema se lo puede hacer de dos maneras, la mas común es presionar el botón de emergencia, el cual va activar una luz estroboscópica de color azul, y la segunda forma es dar un golpe al costado de la estructura para activarse una alarma y active la luz azul [7].

Figura 1.3: Emergency Blue Light Telephones en Universidad del Oeste de California. [8]

A diferencia de [6] y [7] mencionadas anteriormente, La universidad de Augustana

implemento un teléfono de asistencia de emergencia el cual además de tener un botón de pánico y una luz estroboscópica, incluyeron un teclado numérico para realizar una llamada local [9].

Figura 1.4: Emergency Assistance Phones en Universidad de Augustana. [9]

1.3. Análisis comparativo de componentes de hardware

En la presente sección, llevaremos a cabo una comparación de diversos componentes electrónicos con el objetivo de identificar los dispositivos que ofrecen las características más destacadas. Nuestra intención es proporcionar una guía sólida al seleccionar componentes electrónicos que se adapten perfectamente a este proyecto. Analizaremos detenidamente aspectos como rendimiento, durabilidad, eficiencia energética y funcionalidad para aprovechar al máximo el sistema de emergencia. En la Tabla 1.1 se presenta una comparación detallada que nos ayudará a seleccionar el pulsador de emergencia más adecuado para nuestro sistema de seguridad.

Tabla 1.1: Comparación y tipos de pulsantes de emergencia.

Tipo	Resistencia a la intemperie	Durabilidad	Costo	Alimentación
Con cubierta de vidrio	IP65	10-15 años	$ 16.00	24 V CC
Con llave	IP65	5-10 años	$ 12-20	24 V CC
Selector	IP65	5-10 años	$ 25	24 V CC
Industrial	IP65	5-10 años	$ 2-5	24 V CC

Figura 1.5: Tipos de pulsantes de emergencia. [10]

En la Tabla 1.1 se puede apreciar que el pulsador más adecuado en función de su costo es el modelo industrial. Además, cuenta con semejantes características que los otros. En la Figura 1.5 se detallan los tipos de pulsadores mencionados anteriormente. Tras completar una comparación del pulsador, procedemos a llevar a cabo una evaluación similar de la plataforma de hardware fundamental que se utilizará en este proyecto. En la Tabla 1.2 se puede ver las características fundamentales de dos plataformas que son Arduino y la familia de Raspberry Pi.

Tabla 1.2: Comparación entre Arduino y Raspberry Pi.

Características	Arduino	Raspberry Pi
Procesador	Microcontrolador ATmega328P	Procesador ARM
Memoria	32 KB de RAM	1 GB de RAM
Almacenamiento	32 KB de EEPROM	Ranura para tarjeta microSD
Entradas y salias	14 pines digitales	40 pines GPIO
Sistema Operativo	—	Raspbian, Ubuntu, Linux, etc.
Programación	C/C++	Python, C, C++, Java, JavaScript, etc.
Costo	Bajo	Medio
Dificultad de uso	Fácil	Medio
Aplicaciones	Proyectos básicos	Proyectos avanzados, IoT, etc.

Finalmente, se procede a realizar una comparación de los equipos empleados en el sistema de alimentación. En este apartado hay que tener en cuenta la potencia consumida por la Raspberry Pi, en este caso una media de 10 vatios.

Tabla 1.3: Comparación de paneles solares.

Característica	Panel de 10 vatios	Panel de 20 vatios	Panel de 30 vatios
Potencia max.	10 vatios	20 vatios	30 vatios
Potencia min.	6 vatios	12 vatios	18 vatios
Voltaje	12 voltios	12 voltios	12 voltios
Corriente max.	0,83 amperios	1,66 amperios	2,5 amperios
Corriente min.	0,5 amperios	1 amperio	1,5 amperios
Eficiencia	15 % - 20 %	17 % - 22 %	19 % - 24 %
Precio	20$ - 30$	30$ - 40$	60$ - 80$
Dimensiones	185 mm x 415 mm x 18 mm	345 mm x 470 mm x 25 mm	345 mm x 605 mm x 25 mm
Peso	250 gramos	450 gramos	650 gramos

Como se puede apreciar en la *Tabla* 1.3, la opción más adecuada es emplear un panel de 20 vatios, ya que su potencia mínima satisface el promedio de requisitos necesarios. Habiendo seleccionado el tipo de panel, procedemos a evaluar qué batería es la más adecuada para el sistema en la Tabla 1.4 se puede ver la comparación entre 2 de las más utilizadas.

Tabla 1.4: Comparación entre batería de litio y batería seca.

Características	Batería de litio	Batería seca
Voltaje	12 voltios	12 voltios
Capacidad	10 amperios-hora	10 amperios-hora
Ciclos de vida	2000	1000
Costo	100$ - 200$	20$ - 50$
Autodescarga	Baja	Baja
Mantenimiento	No Requerido	No Requerido

La *Tabla* 1.4 muestra que la batería seca es la mejor elección debido a su costo más bajo.

1.4. Hardware utilizado

1.4.1. Raspberry Pi

La Raspberry Pi(RPI) fue desarrollada por Raspberry Pi Foundation en 2012, este dispositivo es un minicomputador de bajo costo y dimensiones, no cuenta con memoria interna por lo cual se necesita instalar en una micro SD un sistema operativo basado en GNU/linux ARM. La RPI realiza las mismas funciones que un computador de escritorio [11].

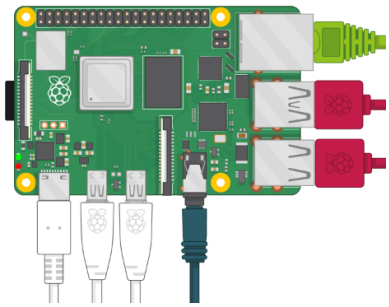


Figura 1.6: Raspberry Pi[11].

1.4.2. Módulo de cámara Raspberry Pi

1.4.3. Adaptador de audio USB

Es una tarjeta de sonido bidireccional, consta en su entrada un módulo USB y como salida dos conectores Jack 3.5 milímetros con puertos de audio y micrófono [12]. Se utilizará para separar el audio y el micrófono con el fin de conectar un altavoz y un micrófono de forma independiente.

Figura 1.7: Adaptador de sonido USB [12]

1.4.4. Pulsador industrial

Son interruptores que se accionan manualmente para impedir o permitir que fluya el paso de la corriente [13].

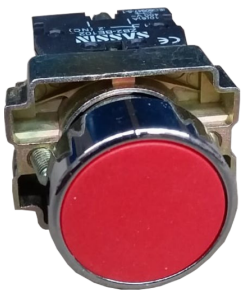


Figura 1.8: Pulsador industrial [13]

1.4.5. Panel solar

Los paneles solares son dispositivos diseñados para capturar la energía radiante del sol y convertirla en energía eléctrica utilizable. La cantidad de electricidad generada por un panel solar está directamente influenciada por la intensidad de la luz solar que incide sobre él. Aun en condiciones de sombra parcial, los paneles solares siguen generando una cantidad mínima de electricidad. La mayoría de estos dispositivos están concebidos para generar alrededor de 14 a 16 voltios cuando se encuentran sometidos a carga. Esta característica posibilita que un panel solar sea capaz de cargar una batería de 12 voltios[14].

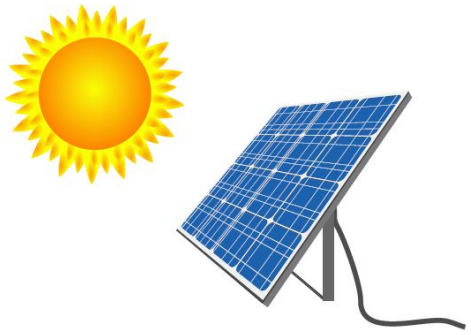


Figura 1.9: Panel Solar [15]

1.4.6. Batería

Es poco común, por no decir que es excepcional, que los paneles solares alimenten directamente equipos eléctricos. Esto se debe a que la cantidad de energía capturada por un panel solar fotovoltaico varía en función de la intensidad de la luz solar que incide sobre él. Esta fluctuación hace que la fuente de energía sea demasiado inconstante para satisfacer las necesidades de la mayoría de los dispositivos eléctricos.

En un sistema conectado a la red eléctrica, el inversor se encarga de gestionar esta variabilidad. Si la demanda supera la producción de energía solar, el sistema obtiene electricidad tanto de la red como del sistema solar. Sin embargo, en el caso de sistemas independientes de la red o sistemas de respaldo, la energía debe ser almacenada en baterías. Los paneles solares alcanzan su máxima potencia cuando están expuestos al sol en su punto más alto, no necesariamente cuando se requiere más energía.

Para garantizar un suministro constante, especialmente en momentos de poca o

ninguna exposición solar, se recurre al uso de baterías. En este contexto, las baterías de ácido de plomo de ciclo profundo resultan esenciales para almacenar la energía. Aunque comparten similitudes con las baterías automotrices, cuentan con un diseño interno diferente que les permite ser descargadas significativamente y recargadas en múltiples ocasiones, lo que es fundamental para un sistema de energía solar[14].

Figura 1.10: Batería [16]

La mayoría de las baterías de plomo-ácido tienen una configuración de 6 voltios o 12 voltios. Similar a los paneles solares, estas baterías pueden ser interconectadas para conformar conjuntos de baterías de mayor tamaño. Al igual que con los paneles solares, es posible conectar estos conjuntos en serie para incrementar la capacidad y el voltaje del sistema de almacenamiento de energía. Alternativamente, se pueden conectar en paralelo para aumentar la capacidad sin modificar el voltaje.[14].

1.4.7. Controlador de carga

Un sistema eléctrico solar demandará la presencia de un controlador destinado a gestionar la dirección de la corriente eléctrica hacia y desde la batería. Es fundamental evitar la sobrecarga de las baterías por parte de los paneles solares, ya que esto podría ocasionar daños y, a la larga, incluso la degradación completa del conjunto de baterías. Del mismo modo, es crucial no permitir que las baterías se descarguen por completo, ya que esto puede resultar en la rápida deterioración del grupo de baterías.[14].

Figura 1.11: Controlador de Carga [16]

1.5. Software

1.5.1. Raspberry Pi imager

Es un software muy facil de utilizar es compatible con sistemas operativos como: Ubuntu, macOS y Windows, permite a los usuarios instalar un sistema operativo ya sea la versión mas actual como una versión en específica. El sistema operativo mas utilizado para RPI es Raspberry Pi OS o antes también llamado Raspbian pero en este software existen otros tipos de S.O. como son: RISCO OS PI, Apertis, Ubuntu entre otros[17].

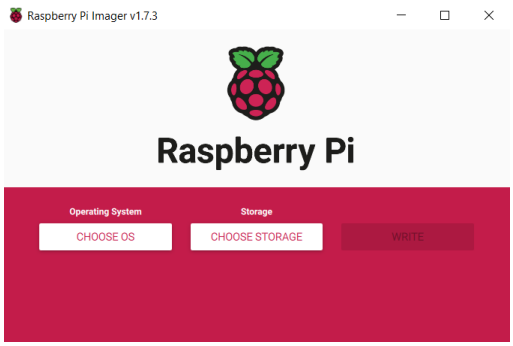


Figura 1.12: Raspberry Pi Imager[17]

1.5.2. Voz sobre el protocolo de Internet (VoIP)

También es conocido como VoIP, tuvo su inicio con la integración de internet se puede definir como un servicio de telefonía a través de internet. Existe 2 tipos principales de servicios VoIP que son: VoIP en la nube que consiste en hacer o recibir llamadas mediante aplicaciones softphone, permitiendo una reducción de costos y mantenimiento [18]. Y también VoIP Residencial la cual no necesita tener una linea telefónica analógica, como principal características son: bajo costo, calidad de llamada, funciones avanzadas, instalación y portabilidad [19].

1.5.3. Sistemas de VoIP

En la actualidad, varios sistemas de telefonía de software libre se pueden implementar en diversos proyectos según sus requerimientos. A continuación, se mencionan los más destacados.

- 3CX
- Asterisk
- Elastix
- FreePBX
- FusionPBX
- OpenPBX

1.5.4. Asterisk

Asterisk es un software de comunicación gratuita dedicado específicamente en telefonía de VoIP, IP, Private Base Exchange(PBX) entre otros [20]. Además, este sistema es compatible con Raspberry Pi.

En Asterisk, el administrador es un modelo cliente/servidor sobre TCP, el cual tiene la capacidad de recibir comandos llamados “acciones” que generan una “respuesta” de Asterisk, También envían ’Eventos’ que contienen mensajes de

información sobre modificaciones dentro de Asterisk. Con el fin de evitar que los informes extensos obstaculicen por completo la interfaz del administrador, algunas acciones en Asterisk generan una respuesta inicial y datos en forma de lista de eventos.

En Asterisk, la configuración de usuarios de administración se realiza dentro del archivo manager.conf. A estos usuarios se les otorgan permisos de lectura y escritura para recibir tipos específicos de eventos y llevar a cabo acciones particulares, respectivamente. Se lista los comandos de administración del servicio:

- **Iniciar**–>Service asterisk start
- **Estado**–>Service asterisk status
- **Parar**–>Service asterisk stop
- **Reiniciar**–>Service asterisk restart
- **Recargar**–>Service asterisk reload

1.5.5. Protocolo de inicio de sesión (SIP)

Fue creado por la Internet Engineering Task Force (IETF), esta definido por el estandar RFC 3261 (2002) y se actualiza frecuentemente, es un protocolo de la capa de aplicación del modelo OSI se encarga de señalizar y controlar sesiones de comuncación entre dispositivos de voz y vídeo, es importante mencionar que SIP no fue creado para transferir audio o vídeo [21].

El protocolo SIP es muy utilizado en telefonia IP, transmisión (streaming) de contenido multimedia, VoIP y juegos en linea. Las entidades que se definen son:

- Agentes de Usuario SIP.
- Servidores SIP: proxy, de registro, redirección y localización.
- SIP GATEWAY [22]

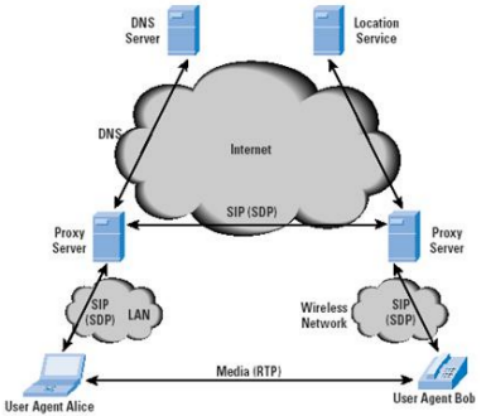


Figura 1.13: Arquitectura del protocolo SIP[22]

.

1.5.6. Protocolo de Control De Transmisión (TCP)

este protocolo tiene su función dentro de la capa de transporte del modelo OSI, organiza los datos o inforamción enviados de una manera fácil para que puedan ser transmitidos dentro de una arquitectura cliente-servidor. TCP se considera un protocolo confiable para el envio de datos, ya que cualquier perdida de paquetas que exista es detectado y resuelto. Estos paquetes son limitados hasta 1500 bytes de los cuales 20bytes es para formar el encabezado TCP y 20 bytes para el encabezado IP el restante es para envio de datos [23].

Al ser un servicio orientado a conexión TCP tiene 3 fases que son: establecimiento a la conexión, transferencia de datos y fase de terminación de la conexión [24].

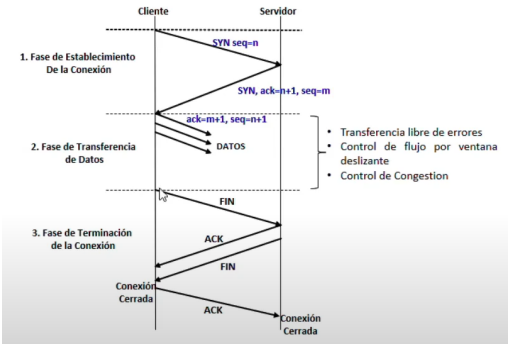


Figura 1.14: Fases de conexión protocólo TCP [24]

.

1.5.7. Protocolo de Datagramas De Usuario (UDP)

El protocolo UDP al igual que TCP son protocolos que se encuentran dentro de la capa de transporte, a diferencia del mencionado anteriormente UDP es más rápido al momento de realizar el transporte de datos, el inconveniente de este protocolo es que cuando algún paquete se pierde no puede detectar y resolverlo como sucedía con TCP. Por tal razón las aplicaciones más comunes son vídeo y multimedia [25].

1.5.8. WI-FI

WIFI es un estándar de comunicación inalámbrica que permite la conexión bidireccional de datos entre artefactos electrónicos como es PC. Smartphones. TV etc. El principio de funcionamiento tiene como base las ondas de radio AM/FM de baja frecuencia. WIFI se encuentro dentro de las bandas de frecuencia de 2.4 y 5GHz. Como se puede observar en la Figura 1.20 los componentes de una conexión inalámbrica son dos, el artefacto electrónico mencionado anteriormente el cual convierte los datos a través de las ondas electromagnéticas y un router que se encarga de recibir la señal y codificarla [26].

Figura 1.15: Dispositivos conectados mediante WIFI [26].

En la Tabla 1.5 se puede observar un resumen de los protocolos, frecuencia, Ancho de canal y velocidades de datos del protocolo WIFI [27].

1.5.9. .NET

Es una plataformas de desarrollo de Microsoft la cual contiene un conjunto de lenguajes de programación, librerías y herramientas para crear todo tipo de

Tabla 1.5: Estándares del protocolo WIFI[27].

Protocolo	Frecuencia	Ancho de canal	MIMO	Velocidad de datos max
802.11ax	2,4 o 5 GHz	20, 40, 80, 160 MHz	Usuario múltiple (MIMO-MU)	2,4 Gbps
802.11ac wave2	5 GHz	20, 40, 80, 160 MHz	Usuario múltiple (MIMO-MU)	1,73 Gbps
802.11ac wave1	5 GHz	20, 40, 80 MHz	Un solo usuario (SU-MIMO)	866,7 Mbps
802.11n	2,4 o 5 GHz	20, 40 MHz	Un solo usuario (SU-MIMO)	450 Mbps
802.11g	2,4 GHz	20 MHz	No se aplica	54 Mbps
802.11a	5 GHz	20 MHz	No se aplica	54 Mbps
802.11b	2,4 GHz	20 MHz	No se aplica	11 Mbps
Tradicional 802.11	2,4 GHz	20 MHz	No se aplica	2 Mbps

aplicaciones.

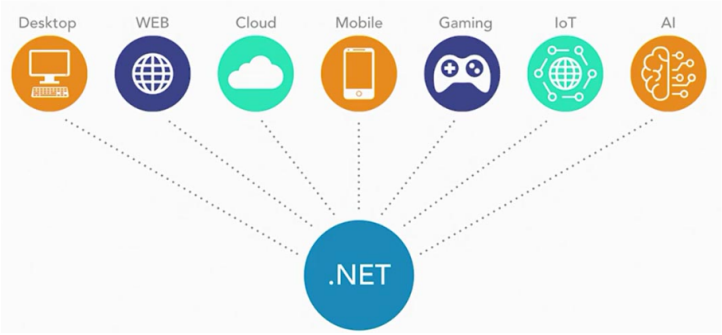


Figura 1.16: .NET [28].

Entre las principales características de .NET tenemos que es una plataforma de código abierto(open source), es multiplataforma y tiene uso con contenedores docker, es decir compacta todo el código y las dependencias de un proyecto o aplicación en un formato estándar con la finalidad de tener una rápida ejecución en entornos informáticos [28].

1.5.10. Lenguaje de programación C#

C# es un lenguaje de programación que tiene raíces en la familia de lenguajes C y es reconocible para los programadores que han trabajado con C, C++, Java y JavaScript. .

Figura 1.17: Logo C# [29].

El lenguaje de programación C# es uno contemporáneo que se enfoca en objetos y tiene seguridad de tipoS. Los desarrolladores pueden crear una amplia variedad de aplicaciones que se ejecutan en .NET, estas pueden ser sólidas y seguras mediante el lenguaje C#. Los proyectos realizados en C# se ejecutan en la plataforma .NET, un sistema de ejecución virtual llamado Common Language Runtime (CLR) y un conjunto de bibliotecas de clases. El CLR es la implementación por parte de Microsoft de la infraestructura de lenguaje común (CLI) [29].

1.5.11. Sockets

Websocket es un protocolo de comunicación bidireccional entre el navegador del usuario y un servidor mediante una conexión persistente. Permite una comunicación full dúplex ya que los datos pueden enviarse de manera bidireccional en forma de paquetes, sin suspender la conexión y sin pedidos adicionales de HTTP. Ya que es un protocolo fiable, eficiente y de rápido acceso a los datos, es muy utilizado en juegos en línea y aplicaciones en tiempo real, entre otros.

1.5.12. Blazor Server

Es un framework creado por Microsoft para construir aplicaciones web interactivas de pagina única (Single Page Aplication), es decir que todo el contenido de la aplicación web se ejecuta en una sola pagina, utilizando el lenguaje de programación C# y HTML. El entorno de ejecución esta definido dentro del servidor y controla lo siguiente:

- Compilación del código en el lenguaje de programación C#.
- Desde el interfaz de usuario(IU) se envía información que ocurren entre un navegador WEB(Chrome, Firefox) al servidor y viceversa.
- El proceso de actualizaciones del IU se envía desde el servidor al cliente y se muestra la versión actualizada de la aplicación [30].

1.5.13. Motion Eye

Es una distribución de Linux que permite convertir una computadora en un sistema de videovigilancia. El sistema operativo se basa en BuildRoot (Herramienta para generar sistemas Linux embebidos a través de la compilación cruzada), y utiliza Motion (programa configurable para monitorear señales de vídeo) como backend y MotionEye (Interfaz para el control de cámaras) como frontend

1.5.14. Visual Studio Code

Es un editor de código de uso moderno (IDE del ingles Integrated development environment) lanzado en el año 2015 por Microsoft, este software contienen herramientas necesarias para el desarrollo de una aplicación web y permite trabajar con diferentes lenguajes de programación admite gestionar atajos de teclado y reestructurar un código fuente [31].

Figura 1.18: Logo Visual Studio Code [31]

Las ventajas de utilizar este IDE son las siguientes:

Multiplataforma: disponible tanto para Windows, Lunux y MacOS

Personalización: ofrece diferentes formas de trabajo con la utilización de extensiones.

Depuración detecta errores en el código que se esta desarrollando.

Control de versiones: Control de versiones: tiene compatibilidad con git para configurar y visualizar.

1.5.15. Inventor

Inventor es un software CAD 3D, que permite el diseño mecánico 3D y simulación de productos de manera profesional que posee las siguientes características.

- Combinación potente de capacidades de diseño para métrico, directo, de formas libres y basado en reglas.
- Integra herramientas para el diseño y simulación de cualquier tipo de materiales.
- Es compatible con Trusted DWG que permite obtener una excelente definición en los modelos creados

Figura 1.19: Logo Inventor [26].

1.5.16. Altium designer

Es un conjunto de programas que permiten la creación de esquemas, simulación y diseño de circuitos impresos en todas sus fases. No se trata de un conjunto de paquetes sueltos vendidos como una suite y conectados mediante archivos externos (netlist), sino de un programa único(dxp.exe) que crea un (entorno) frontend y comunica al usuario con los distintos servidores (por ejemplo, editor de texto, editor de esquemas, editor de PCB).

Figura 1.20: Logo Altium Designer [26].

Capítulo 2

MARCO METODOLÓGICO

En este capítulo se presenta la estructura que se realizó para el diseño del sistema de seguridad estilo botón de pánico tipo totem. Además, un análisis comparativo de las características fundamentales de los dispositivos electrónicos y de cámara que se utilizaran para este proyecto, con la finalidad de poder optimizar recursos y aprovechar al máximo la eficiencia de los mismos y obtener mejores resultados.

Tras ello, utilizar software libre para la implementación del servidor web conjunto, una interfaz web de visualización para la supervisión del sistema de seguridad.

2.1. Requerimientos

En esta sección se describen los posibles lugares para colocar el tótem de seguridad, tenemos que tener en cuenta que la estructura cuenta con un panel solar por ende se debe ubicar en un lugar externo.

2.1.1. Puntos estratégicos para la ubicación del tótem de seguridad

La Figura 2.1 indica los puntos A,B y C, que representan las zonas con mayor afluencia de personas, es decir, son posibles ubicaciones seleccionadas estratégicamente para el tótem de seguridad. Las pruebas de nuestro prototipo se realizaron en el punto A.

Figura 2.1: Puntos estratégicos para ubicar el tótem en el campus El Vecino. [32]

2.2. Selección de hardware y software del sistema

Para la realización de este proyecto se emplea software de código abierto, que permite tener completa libertad de ejecutar, copiar, modificar y distribuir. Con el fin de compartir información con la comunidad de la web.

2.2.1. Visual Studio Code

Se utiliza esta herramienta de edición de código debido a las ventajas que ofrece al programar, utilizar extensiones en este software facilita al programador tener mas funcionalidades y a medida que se vaya programando uno u otro lenguaje, se instalara las que necesitemos. En este caso se instalaron las extensiones de C# y Remote-SSH, la última nos ayuda a conectarnos via remota a la Raspberry Pi 2.
En comparación con otros editores de código como Sublime Text, Atom e IntelliJ, estos necesitan comprar una licencia para poder obtener todos los beneficios que ofrece en el caso de Sublime Text y IntellJ, también ofrecen una versión gratuita o de prueba. Mientras que Atom fue desarrollada por GitHub, actualmente pertenece a Microsoft [33].

2.2.2. .Net(DotNet)

.Net Core es lanzada en el 2016, es una plataforma exclusiva de Microsoft sobre la que se puede desarrollar cualquier tipo de proyectos como son: web, móvil, escritorio, vídeo juegos entre otros como se muestra en la Figura .Es una tecnología moderna multiplataforma, es decir, compatible con diferentes sistemas operativos como son: windows, linux y MacOs. Tiene soporte para muchos lenguajes de programación entre ellos tenemos C#, F# y VB.NET [34].

Figura 2.2: Infraestructura de .NET core. [35]

2.2.3. x | Raspberry Pi imager

Este software es proporcionado por Raspberry Pi Foundation [36], es una herramienta gratuita multiplataforma, permite instalar algunos sistemas operativos(OS) compatibles con la Raspberry Pi en diferentes versiones, también

se puede ejecutar dentro de la interfaz de la Raspberry Pi para actualizar el mismo. Después de haber cargado la imagen se puede configurar algunos aspectos importantes como son: el nombre del host, habilitar SSH para tener un control remoto, establecer un nombre de usuario y una contraseña personalizada y configurar la red SSD que se va a conectar inalambricamente como se muestra en la Figura 2.3. ver Apéndice A:1.

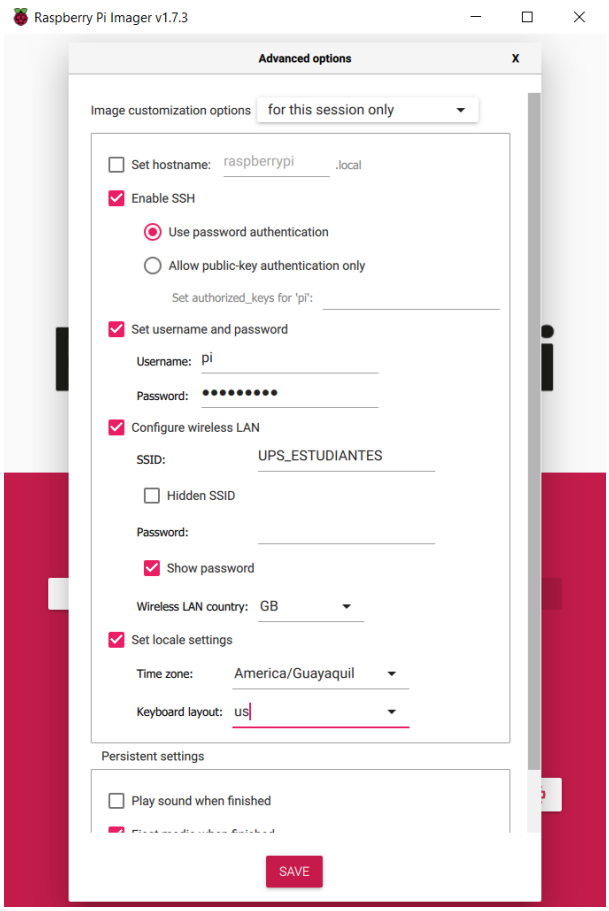


Figura 2.3: Configuración en la interfaz de Raspberry Pi Imager. [36]

2.2.4. Configuración de Asterisk en Raspberry Pi 2

La Raspberry Pi 2 se configura como una centralita, es decir, se puede gestionar, enrrutar y responder llamadas. En la Figura 2.4 se muestra un esquema de cómo se realiza una llamada VoIP utilizando una Raspberry Pi como centralita telefónica. En donde la extensión 201 se configura en el tótem y la extensión 200 en el sistema de gestión. Ver Apéndice A:2.

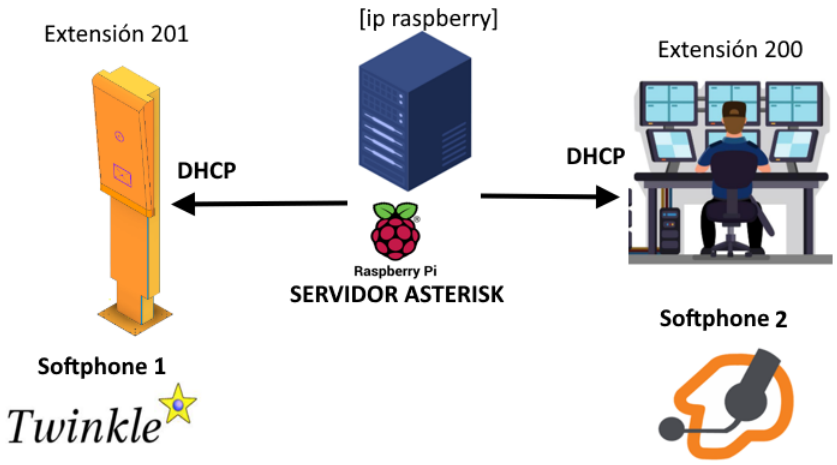


Figura 2.4: Esquema del proceso de llamada VoIP. (Fuente:Autor)

En la Figura 2.5 se visualiza el comando de instalación del servicio. Antes de hacer la instalación es recomendable utilizar el comando de actualización de la Raspberry Pi **apt-get update**.

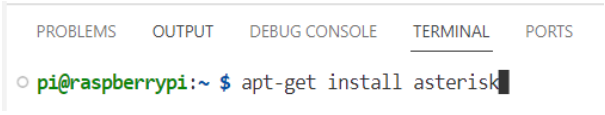


Figura 2.5: Comando para instalar Asterisk en Raspberry Pi.(Fuente: Autor)

Después de instalar, digitamos el comando para verificar la versión y otro comando de administración para ver el estado del servicio.

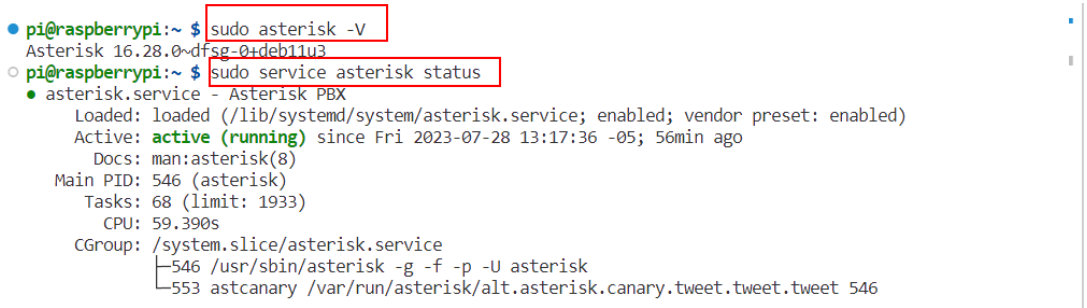


Figura 2.6: Versión y estado de asterisk.(Fuente: Autor)

En la Figura 2.6 se puede verificar que se ha instalado correctamente la versión 16.28.0 y que esta activo. Luego, se procede a configurar los dos archivos más importantes **sip.conf** y **extensions.conf**, los mismos que permiten definir canales SIP o clientes (entrada y salida de llamadas) y también el comportamiento dentro de la centralita.

2.2.5. Configuración del archivo sip.conf

Este archivo esta conformado por un sección **[general]**, donde se agregan los parámetros de configuración generales que van a tener todos los usuarios. En la siguiente Tabla 2.1 se muestra la estructura.

Tabla 2.1: Parámetros de configuración del archivo sip.conf [20].

Función	Descripción
username=200	Número de extensión de usuario
type=friend	El usuario puede enviar y recibir llamadas
host=dynamic	Cualquier equipo se puede registrar como cliente
secret=1234	Contraseña de autenticación para acceder a la extensión
context=internal-extensions	Punto de salida del telefono
disallow=all	Permite descativar codecs
allow=all	Activa todos los codecs

En la Figura 2.7 se configura dos usuarios SIP, la número 200 que está en el totem o cliente y la 201 que se encuentra en el servidor o personal de seguridad.

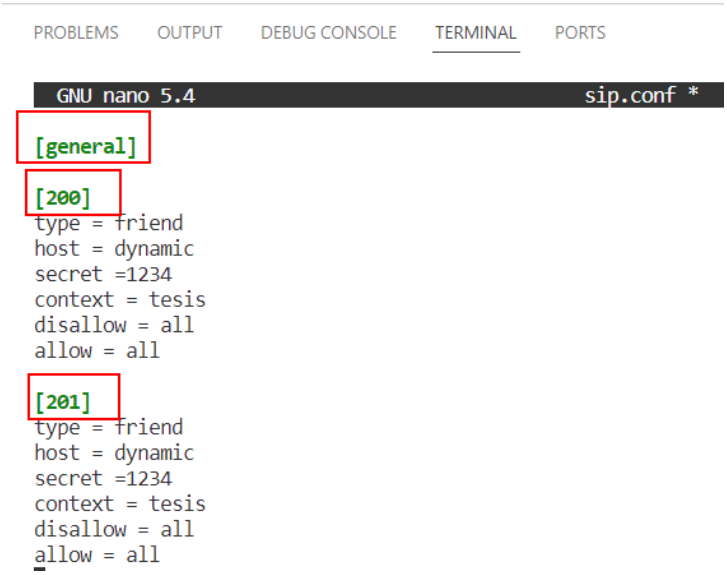


Figura 2.7: Configuración de los usuarios SIP 200 y 201.(Fuente: Autor)

Después de configurar los usuarios SIP, procedemos a digitar el comando **Service asterisk reload** de asterisk, para que se guarden las configuraciones, luego accedemos a la consola de comandos asterisk con el siguiente comando **[sudo asterisk -rvvv]**. Dentro de la consola digitamos el comando **sip show users** como se muestra en la Figura 2.8 ,aquí se muestra información de usuarios SIPs, contraseña y el contexto.

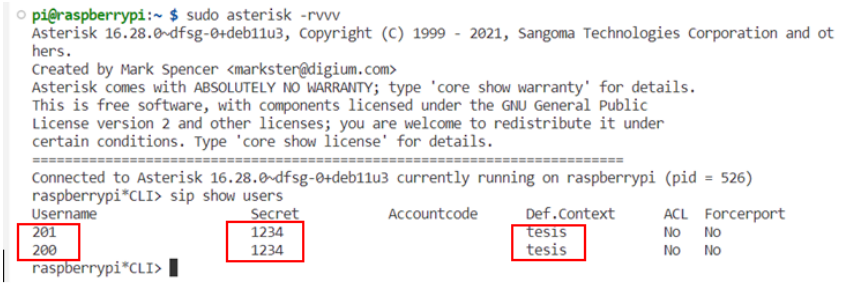


Figura 2.8: Extensiones 200 y 201 creadas.(Fuente: Autor)

2.2.6. Configuración del archivo extensions.conf

Este archivo controla el plan de llamadas de la centralita, es decir, define como se comporta las llamadas entrantes y salientes. La estructura de definir extensiones es la siguiente:

Exten =>Número; Prioridad; Aplicación a ejecutar(parámetro)

En la Tabla 2.2 se detalla cada parámetro.

Tabla 2.2: parámetros de configuración del archivo extensions.conf [20].

Función	Descripción
Exten =>	Define una extensión especifica dentro del contexto
Número	Define el número al que se va a llamar dentro del contexto
Prioridad	Indica el orden en la que se va ajecutar
Aplicación Dial	Se ejecuta la aplicación DIAL para realizar llamadas

En la Figura 2.9 se muestra la configuración del archivo **extensions.conf** realizada en las Raspberry Pi.

Figura 2.9: Configuración de Extensions.conf (Fuente: Autor)

2.2.7. Inicio de sesión de usuarios SIP en shoftphone

Una vez configurado los usuarios SIP, se registran en el softphone los parámetros de usuario, contraseña y dominio. En este caso se utiliza el software Zoiper para la extension 200 y para el usuario 201 se utilizo el software twinkle como se muestra en la Figura 2.10 y Figura 2.11respectivamente.

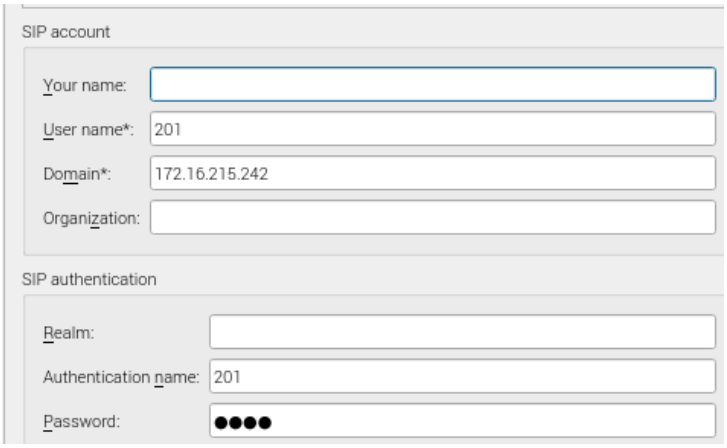


Figura 2.10: Configuración usuario SIP en Twinkle (Fuente: Autor)

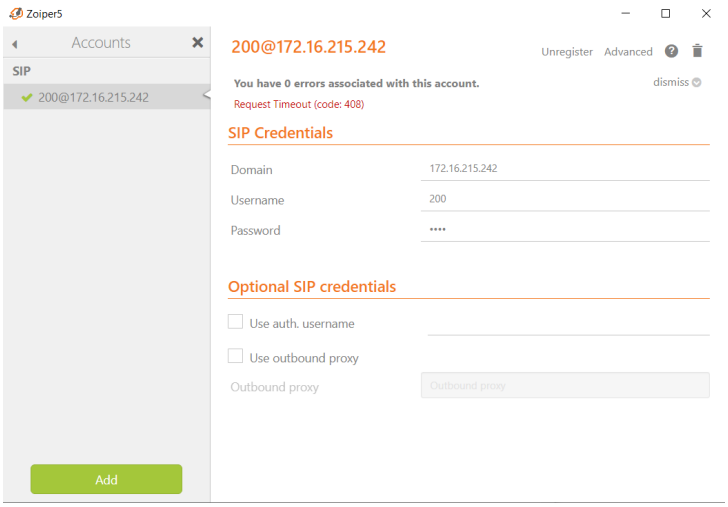


Figura 2.11: Configuración de usuario SIP en Zoiper (Fuente: Autor)

Finalmente queda configurado los usuarios SIP en los softphones para realizar la llamada de VoIP, para verificar que las extensiones 200 y 201 están ONLINE, desde la consola de asterisk digitamos el comando **sip show peers**, en la Figura 2.12 se puede observar que los dos usuarios se encuentran ONLINE.

```
raspberrypi*CLI> sip show peers
Name/username             Host                                  Dyn Forcerport Comedia    ACL Port     Status      Description
200/200                   172.16.211.73                          D  Auto (No)  No             33380    OK (12 ms)
201/201                   172.16.215.242                         D  Auto (No)  No             5064     OK (7 ms)
2 sip peers [Monitored: 2 online, 0 offline Unmonitored: 0 online, 0 offline]
raspberrypi*CLI>
```

Figura 2.12: Información de los peers SIP configuradas (Fuente: Autor)

2.2.8. Raspberry Pi 2 Model B

Para la implementación de este proyecto se utiliza una Raspberry Pi de segunda generación, ya que las características que contiene son suficientes para el mismo, también, por motivo de su bajo costo como se observa en la Tabla 2.3 en comparación con otros modelos de Raspberry Pi. Desde la terminal podemos ver el módelo de la Raspberry PI utilizando el comando **pinout**, Además se muestra características principales y los periféricos que contiene como se observa en la Figura 2.13.

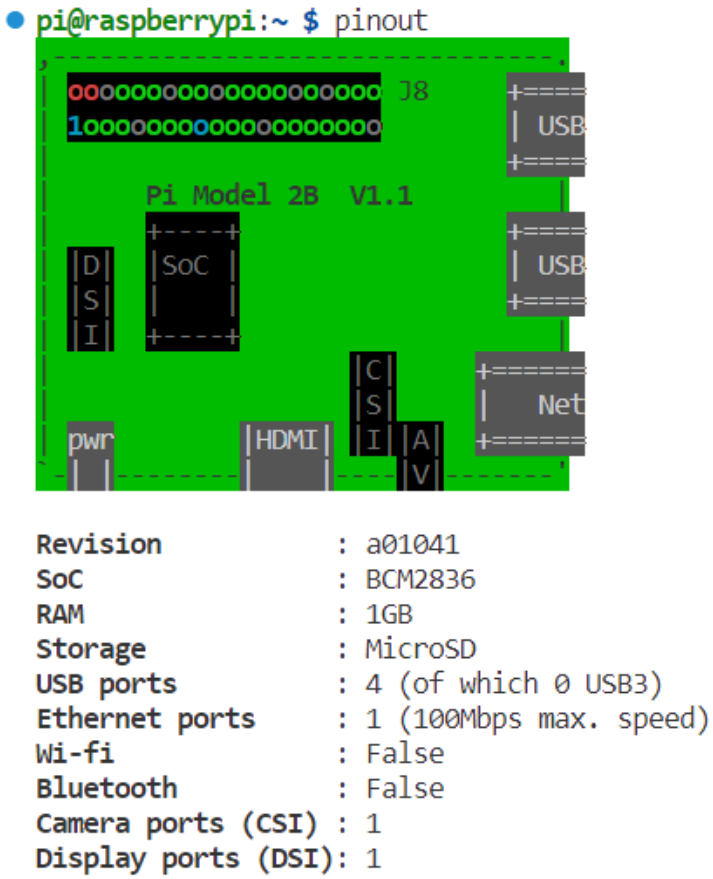


Figura 2.13: Raspberry Pi 2 Model B (Fuente: Autor)

En la Tabla 2.3 podemos ver la comparación que existe con respecto a la 3ra y 4ta

generación, respectivamente.

Tabla 2.3: Comparación de tres generaciones de Raspberry Pi

Caracteristicas	Raspeberry Pi 2	Raspeberry Pi 3	Raspeberry Pi 4
Procesador	Broadcom BCM2836	Broadcom BCM2837	Broadcom BCM2711
Frecuencia de CPU	900 MHz	1.2 GHz	1.5 GHz
Memoria RAM	1 GB	1GB, 2GB o 4GB	2GB, 4GB o 8GB
Puertos USB	4 puertos USB 2.0	4 puertos USB 2.0	2 USB 2.0 y 2 USB 3.0
Costos	50 $	90 $	250 $

2.2.9. Amplificador de audio LM386

Se hace uso de un amplificador de audio con la finalidad de que durante la llamada, la persona que haga uso del tótem pueda escuchar de manera clara al personal de seguridad. Para nuestro caso, se utiliza el amplificador de potencia de audio de bajo voltaje **LM386**. Es un circuito integrado de **Texas Instruments** que se ha seleccionado por su buen rendimiento y fácil configuración. La Tabla 2.4 indica sus características y en la Figura 2.14 se muestra el diagrama de circuito electrónico.

Tabla 2.4: Características técnicas del amplificador de Audio LM386.

Elementos	Cantidad
Rango de voltaje de suministro	4 - 12 V
Ganancia	20 - 200
Impedancia de carga	5 - 32 Ω

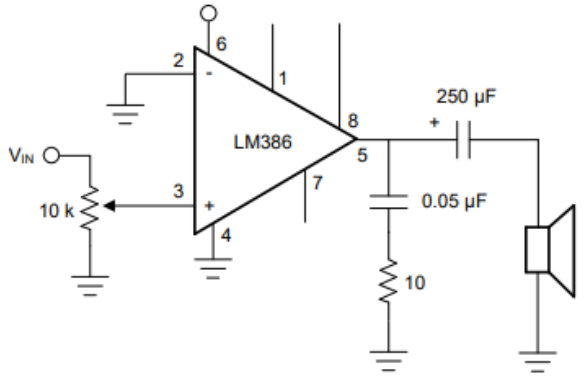


Figura 2.14: Típica aplicación del amplificador de audio LM386. [37]

2.3. Despliegue del servidor Web

Para este proyecto se utiza la tecnología .Net y el lenguaje de programación C#. Empezamos instalando el SDK(Software Development Kit) de .Net Core en la Raspberry Pi 2, este SDK incluye: compiladores, bibliotecas, herramientas de desarrollo, documentación, ejemplos y plantillas de proyectos para el desarrollo de aplicaciones. También, permite aprovechar el lenguaje de programación C#, para ello se tiene que verificar la arquitectura de la Raspberry Pi 2, se puede consular esta información escriniendo el siguiente comando **uname -m** como se pude ver en la siguiente Figura 2.15, mas detalle en el Apéndice B.

```
pi@raspberrypi:~ $ uname -m
armv7l
```

Figura 2.15: Arquitectura del procesador de la Raspberry Pi 2 (Fuente: Autor)

.

Teniendo esta información se procede a descargar e instalar .Net Core en la Raspberry Pi 2 [38]. Después se crea un directorio en nuestro caso se llama **BlazorCamera**. Dentro de este se crea el proyecto, para el cual se digita el comando **dotnet new blazorserver-empty -n BlazorCamera -o** . En la Figura 2.16 se puede observar todas las carpetas creadas a partir de este comando.

```
pi@raspberrypi:~/webserver/BlazorCamera $ ls -l
total 48
-rw-r--r-- 1 pi   pi    118 Jul  5 23:29 ActualizarServidor.sh
-rwxrwxrwx 1 pi   pi    477 Jul  4 17:58 App.razor
-rwxrwxrwx 1 pi   pi    149 Jul  5 23:52 appsettings.Development.json
drwxr-xr-x 3 root root 4096 Jul 12 15:22 bin
-rwxrwxrwx 1 pi   pi    385 Jul 12 14:05 BlazorCamera.csproj
-rwxrwxrwx 1 pi   pi    144 Jul  4 17:58 _Imports.razor
-rwxrwxrwx 1 pi   pi     58 Jul  4 17:58 MainLayout.razor
drwxr-xr-x 3 pi   pi   4096 Jul 12 15:22 obj
drwxrwxrwx 2 pi   pi   4096 Jul 28 02:28 Pages
-rwxrwxrwx 1 pi   pi    372 Jul 12 15:27 Program.cs
-rw-r--r-- 1 pi   pi    169 Jul 12 15:11 RegistroPulso.cs
drwxrwxrwx 4 pi   pi   4096 Jul  7 08:52 wwwroot
```

Figura 2.16: Despliegue del servidor web (Fuente: Autor)

.

En el archivo **BlazorCamera.csproj** se agregan las dependencias o librerias que se van utilizar con sus respectivas versiones, en nuestro caso fueron **System.Device.Gpio**

que permite tener interacción con los pines GPIO de la Raspberry Pi 2 y la librería **LiteDB** que es una base de datos para almacenar el registro de pulsos (las veces que se ha presionado el botón).

En la carpeta **Pages** que se muestra en la Figura 2.16 se encuentra el archivo **Index.razor** que abarca el frontend y backend, esto debido a que en Blazor se realiza la interfaz de usuario como la lógica de programación dentro del mismo archivo, se especifica en la documentación [39]. Para explicar las funciones backend y frontend realizadas en este proyecto se detalla de manera separada.

2.3.1. Back end

La estructura de la lógica en esta aplicación se realiza por secciones: Importación de librerías, declaración de variables o constantes y funciones principales.

Como se mencionó anteriarmente se agregaron las librerias **System.Device.Gpio** y **LiteDB**, adicional a estas utilizaremos otras que son propias de .NET en la Figura 2.17 podemos ver todas las librerías utilizadas.

Index.razor ×

Pages > Index.razor

```
1  @page "/"
2  @implements IDisposable
3  @using System.Device.Gpio
4  @using System.Diagnostics
5  @using LiteDB
6  @inject LiteDatabase db
```

Figura 2.17: Librerías utilizadas (Fuente: Autor)

.

La librería **System.Diagnostics** proporciona atributos y clases en el espacio de nombres. También implementamos el intervalo **IDisposable** que permite ejecutar la función llamada **Dispose** cuando se cierra la página, es decir, libera recursos de memoria. Además, se utiliza la directiva inject **LiteDatabase db** para proveer una instancia de la clase LiteDatebase accediendo a los métodos de la misma.

En la Figura 2.18 se muestra las variables de tipo int, bool, string y array declaradas. Estas permiten almacenar datos al momento de cambiar el estado del GPIO, verificar usuario y contraseña en la sección de log in, mitigar el efecto anti-rebote, entre otras.

```
@code
{
    int pulsante = 17;
    bool llamando = false; // llamando
    bool rebote = false;
    bool mostrarCamara = false;
    bool historial = false;

    bool autorizado = false;
    bool pinesRegistrados = false;
    string error = string.Empty;
    string usuario = string.Empty;
    string clave = string.Empty;
    RegistroPulso[] historialPulsos = Array.Empty<RegistroPulso>();
```

Figura 2.18: variables utilizadas (Fuente: Autor)

.

A continuación, se presentan los bloques de código de las principales funciones realizadas. Para el log in se utiliza una función llamada **async Task Autorizar()**, permite escribir un usuario y contraseña para el inicio de sesión, en este caso tanto el usuario y contraseña es «admin» se muestra el diagrama de flujo en la Figura 2.19. Ver Apéndice B.7

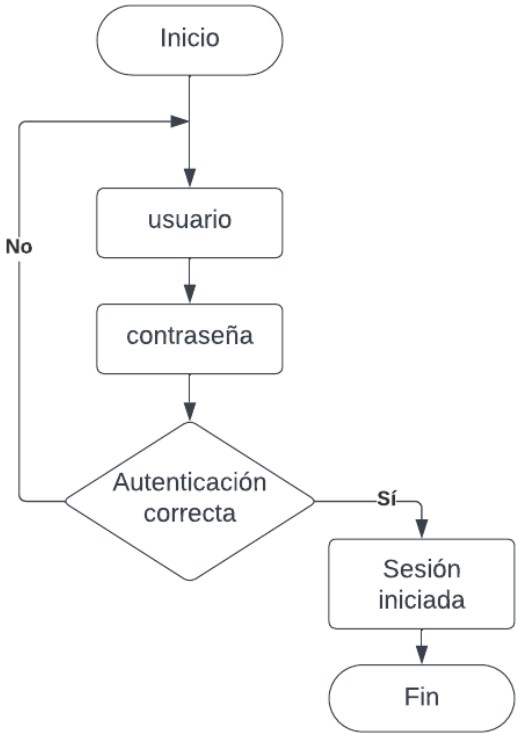


Figura 2.19: Diagrama de flujo de la función autorizar (Fuente: Autor)

.

Se crea la función llamada **async void OnPinEvent(object sender,**

PinValueChangedEventArgs args), para cuando se presione el pulsante realiza la instrucción de llamar y guardar registro. Antes de esto se configura un temporizador con el objetivo de eliminar el efecto rebote del pulsante, y se establece otro para esperar un segundo antes de realizar la llamada, se puede ver en la Figura 2.20 el diagrama de flujo. ver Apéndice B:6

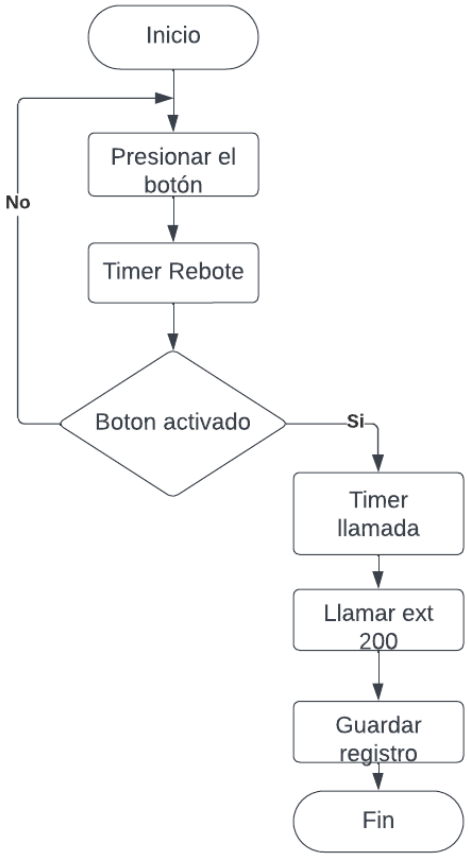


Figura 2.20: Diagrama de flujo de la activación del pulsante. (Fuente: Autor)

.

Para realizar la llamada de VoIP al momento de presionar el pulsante, primero tenemos que configurar una consola virtual u oculta, después dentro de esta consola procedemos a ejecutar el comando **«twinkle -c –call 200 –immediate"»** la cual realiza una llamada a la extensión 200 que es la del personal de seguridad, todo este proceso se encuentra en la función **async Task AbrirLlamada()** como se muestra en el diagrama de bloques de la Figura 2.21.

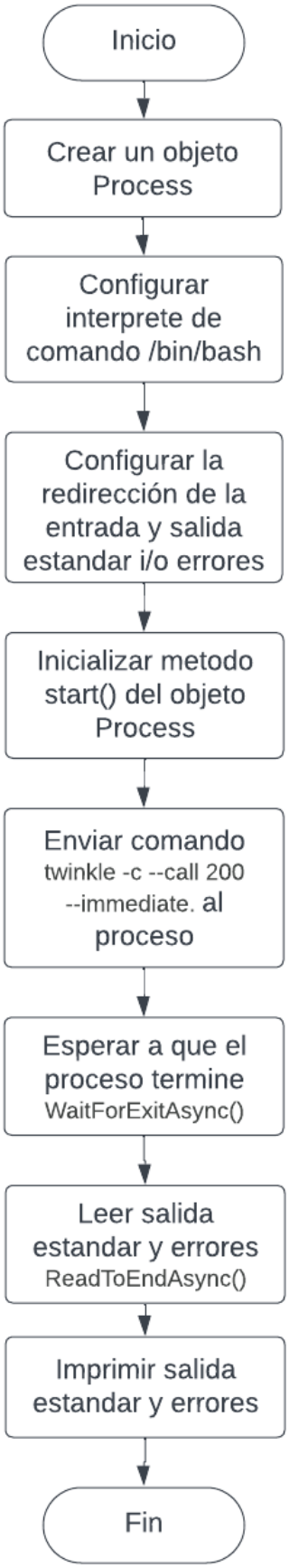


Figura 2.21: Diagrama de flujo para la función abrir llamada y consola virtual (Fuente: Autor)

.

Se crea la lógica para llevar un registro de las veces que se ha presionado el pulsante. El registro se guarda en una base de datos y se puede consultar en cualquier momento. Además, se añade la posibilidad de marcar un registro como falso o erróneo. Esta información se muestra en el historial para que el usuario pueda identificar fácilmente los registros incorrectos, en la Figura 2.22 se muestra el diagrama de bloques.

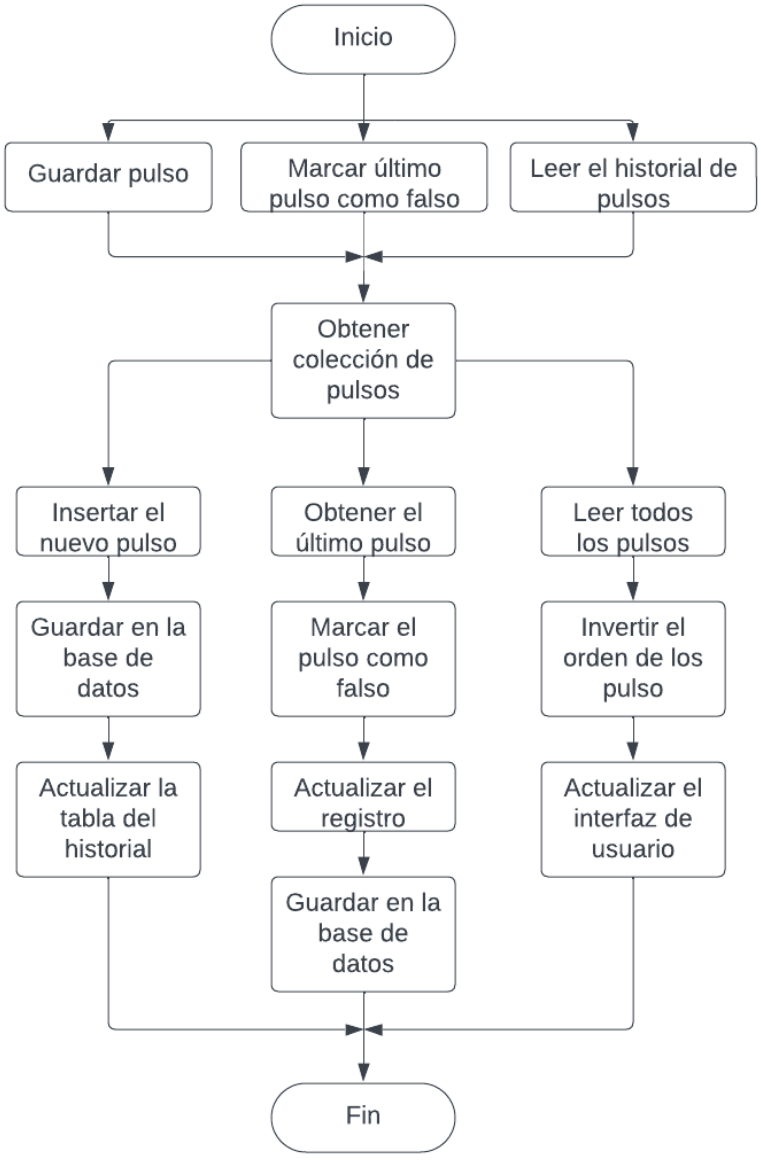


Figura 2.22: Diagrama de flujo para guardar registro al momento de pulsar el botón (Fuente: Autor)
.

Finalmente se crea la función **void CerrarSesion()** para salir de la página principal,

al ejecutarse esta función deberá permitir ingresar otra vez con el usuario y contraseña.

2.3.2. Front end

Para la interfaz de usuario se debe mostrar la estructura en HTML, estilos CSS que se encuentra dentro de una carpeta publica llamada **wwwroot** y también se utiliza los componentes de blazor(variables, funciones, etc). Como se había mencionado anteriormente el frontend y backend se programa dentro del archivo **Index.razor** por ende, se pueden utilizar las variables declaradas en la Figura 2.18

En primer lugar, empleamos el condicional, **if (autorizado == true)**, es decir, si el usuario a iniciado sesión correctamente pueda ver la página principal. Dentro de este condicional empezamos a visualizar el encabezado de la interfaz, es este caso una barra de navegación en donde se encuentran los botones de Inicio, Mostrar Historial y Cerrar sesión. como se muestra en la Figura 2.23

```
9   @if (autorizado == true)
10  {
11      <h1 class = "centrado">UNIVERSIDAD POLITECNICA SALESIANA </h1>
12      <h2 class = "centrado">SISTEMA DE SEGURIDAD </h2>
13
14
15      <ul class="menu">
16        <li><a href="#">Inicio</a></li>
17        <li><a href="#" @onclick=@(() => {historial = true;})>Mostrar Historial</a></li>
18        <li><a href="#" @onclick=CerrarSesion >Cerrar sesión</a></li>
19      </ul>
```

Figura 2.23: Barra de navegación (Fuente: Autor)

.

Después se utiliza la variable **llamando** para mostrar un mensaje en la interfaz, al momento de precionar el pulsante del totem se visualiza «SE HA PRESIONADO EL BOTON DEL TOTEM 1», se agrega color y tipo de letra como se muestra en la Figura 2.21

```
22      @if(llamando == true)
23      {
24          <h1 style="color: red; font-family: Arial, sans-serif;">SE HA PRESIONADO EL BOTON DEL TOTEM 1</h1>
25      }
```

Figura 2.24: Mensaje de aviso que se ha presionado el botón (Fuente: Autor)

.

Luego se agrega un botón en la interfaz para «abrir/ocultar cámara». Para tener imagen del módulo de cámara de la Raspberry Pi se utiliza el software «motion eye»

mencionado en el capitulo 1. Aqui me da un puerto especifico para poder monitorear la cámara, en este caso es **:8081**

```
27    @if (mostrarCamara == true)
28        {
29            <button class="btn1" @onclick=@(() => mostrarCamara = false)>Ocultar camara totem 1</button>
30            <img width="400" height="400" src="http://172.16.215.242:8081/" >
31
32        }
33        else
34        {
35            <button class="btn1" @onclick=@(() => mostrarCamara = true)>Mostrar camara totem 1</button>
36        }
```

Figura 2.25: Comando para visualizar el modulo de cámara(Fuente: Autor)

.

Adicional se agrega la opción de mostrar historial, aqui se presentan tres parámetros: ID, fecha y nota. En esta ultima se puede agregar mediante un botón llamado «Marcar Falso» cuando se presione el botón del totem de manera erronea.

```
42    @if (historial == true)
43    {
44    <div>
45    <h3>Historial pulsaciones</h3>
46    <button @onclick=MarcarUltimoRegistroComoFalso>Marcar Falso</button>
47    <button @onclick=LeerHistorialPulsos>Actualizar</button>
48    <table>
49        <tr>
50            <th>Id</th>
51            <th>Fecha</th>
52            <th>Nota</th>
53        </tr>
54        @foreach (var registroPulso in historialPulsos)
55        {
56            <tr>
57                <td>@registroPulso.Id</td>
58                <td>@registroPulso.Fecha</td>
59                <td>@(registroPulso.Falso ? "Registro falso" : "")</td>
60            </tr>
61        }
62    </table>
63    </div>
64    }
```

Figura 2.26: Historial de pulsasiones(Fuente: Autor)

.

Finalmente cuando el condicional **if (autorizado == false)** se presenta un contenedor «div» en HTML que contiene el formulario de inicio de sesión, como se puede ver en la Figura 2.27.

```
69  else
70  {
71      <div class="login-box">
72          <img src="../img/logoUPS.jpg" class="avatar" alt="Avatar Image">
73          <h1>SISTEMA DE SEGURIDAD </h1>
74
75          <br>
76          <label for="username">Usuario</label>
77          <input @bind-value=usuario type="text" name="username" placeholder="Enter Username" required>
78          <br>
79
80          <label for="username">Contraseña</label>
81          <input @bind-value=clave type="password" name="password" placeholder="Enter Password" required >
82          <br>
83
84          <span class="login-error">@error</span>
85          <br>
86          <input @onclick=Autorizar type="submit" value="Log In">
87
88      </div>
```

Figura 2.27: Formulario de inicio de sesión(Fuente: Autor)

.

En la Figura 2.28 y 2.29se pude ver la interfaz web conjunto todo lo mencionado anteriormente.

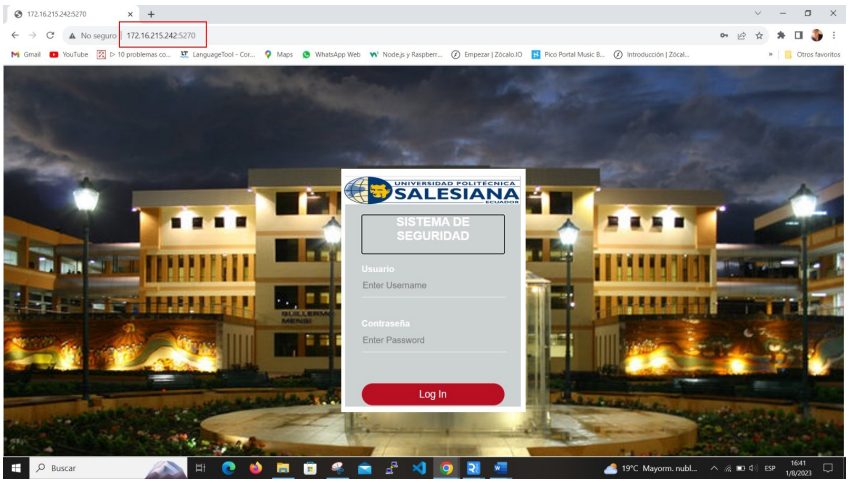


Figura 2.28: Interfaz de login(Fuente: Autor)

.

Figura 2.29: Interfaz de la página principal(Fuente: Autor)

.

2.4. Diseño general del sistema

En la Figura 2.30 se presenta un diagrama de flujo del proceso técnico y de asistencia, en caso de que el botón sea activado.

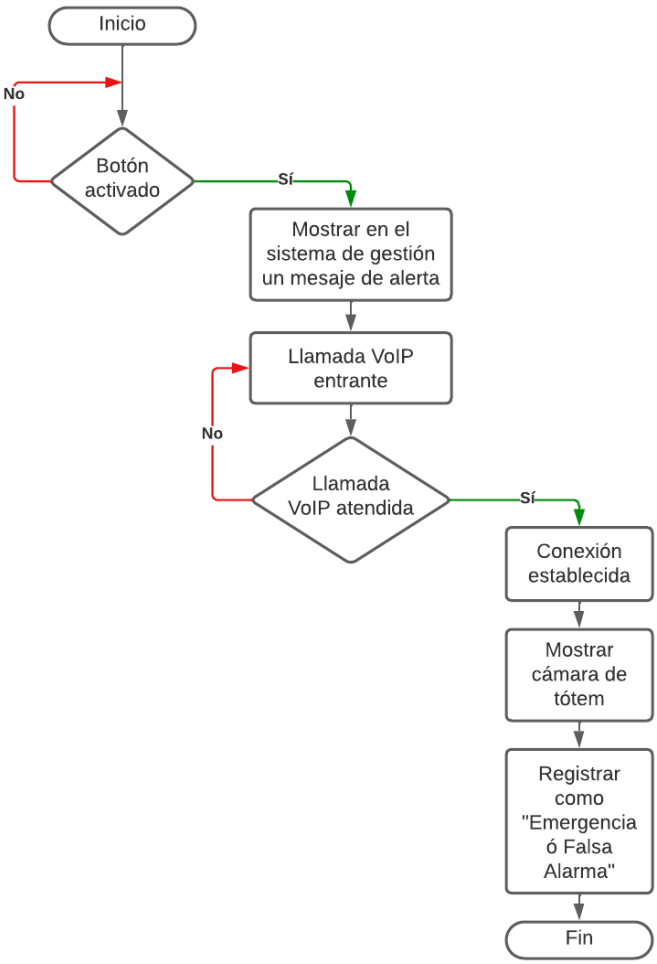


Figura 2.30: Diagrama de flujo general del sistema de seguridad(Fuente: Autor)

.

2.4.1. Estructura de tótem

Para la realización del presente trabajo se ha diseñado la estructura del tótem en el software de Autodesk INVENTOR. Para mayor detalle de las dimensiones ver Apéndice C.

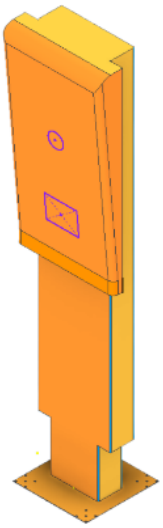


Figura 2.31: Diseño de la estructura de Tótem INVENTOR

2.4.2. Diagrama de bloques

La Figura 2.32 indica el diagrama de bloques para el funcionamiento del sistema de botón de pánico. El proceso inicia con el accionamiento de un pulsante, el cual, inicia una llamada al personal de seguridad, y al mismo tiempo inicia la transmisión de vídeo. Cuando el personal de seguridad responde, se establece una comunicación bidireccional.

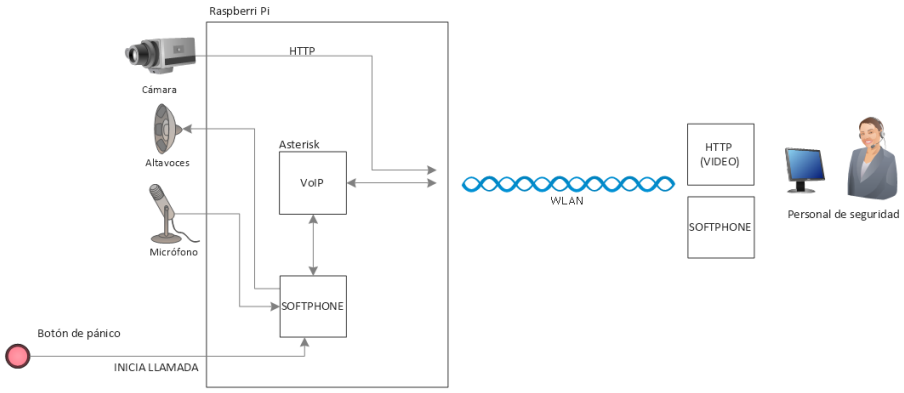


Figura 2.32: Diagrama de bloques del funcionamiento del botón de pánico. Fuente: Autores.

2.4.3. Determinación del sistema autónomo de alimentación

Para este trabajo, se plantea hacer uso de un sistema de alimentación mediante uso de un sistema fotovoltaico. Por tanto, se debe determinar la capacidad del sistema para un funcionamiento sin interrupción durante el período activo de personal y estudiantes que sería desde las 7 de la mañana hasta las 10 de la noche. En primer lugar, se realiza una Tabla de consumo de todos los equipos que forman parte del sistema. La Tabla 2.5 indica estos valores.

Tabla 2.5: Consumo eléctrico de los equipos.

Elemento	Potencia en DC (W/h)	Voltaje (V)	Horas por día	Consumo por día (W/día)
Raspberry Pi	3	12	15	45
LM 386	3	12	0.75	2.25
Total	6 W/h			47.25 W/día

Para el caso de amplificador de audio LM386, se considera sólo un 5 % del total de horas. El consumo total es de 47.25 W/día y 6 W/hora en corriente continua. Se coloca un panel de 50 W/hora en conjunto con una batería de 12V a 7 Ah y un controlador de carga de 15 A.

$$P_{bat} = 12V \times 7Ah = 84W/h \gg 6W/h$$

Por lo tanto, la batería es capaz de entregar una potencia suficiente para mantener al sistema encendido.

Tiempo de carga de la batería

Para determinar el tiempo aproximado para que la batería se encuentre cargada completamente se parte de la ecuación 2.1

$$T_{carga} = \frac{Capacidad\ de\ la\ bateria}{Corriente\ de\ carga} \tag{2.1}$$

Para nuestro caso, la capacidad de batería es de 9 Ah, y la corriente de carga está en función de la potencia que el panel solar entrega al regulador de carga, o a la corriente equivalente de carga. Se realizan mediciones de corriente de carga de la batería en diferentes escenarios; cuando la intensidad de radiación es alta y baja. Los resultados

se indican en las Figuras 2.33 y 2.34.

Por lo tanto, el tiempo de carga de la batería con una intensisdad de radiación alta es:

$$T_{carga} = \frac{9\ Ah}{1\ A} = 9\ horas$$

Y el tiempo de carga de la batería con una intensisdad de radiación baja es:

$$T_{carga} = \frac{9\ Ah}{0{,}6\ A} = 15\ horas$$

Figura 2.33: Esquema de conexiones para el proyecto botón de pánico. (Fuente: Autores)

Autonomía del Sistema

Otro parámetro importante del sistema de botón de pánico es su autonomía, que hace referencia al tiempo que puede estar activo sin recibir energía de la radiación solar.

Para determinar la autonomía se parte de la ecuación 2.2.

$$T_{auton} = \frac{Capacidad\ de\ la\ bateria}{Corriente\ de\ carga\ sistema} \tag{2.2}$$

Figura 2.34: Corriente suministrada por el panel solar en condiciones de baja radiación solar. (Fuente: Autores)

La corriente de carga del sistema es la corriente que el sistema en conjunto consume de la batería. Se puede determinar a través de la ecuación 2.3.

$$I_{sistema} = \frac{Potencia\ consumida}{Voltajedelsistema} \tag{2.3}$$

Por lo tanto;

$$I_{sistema} = \frac{6\ W}{12\ V} = 0{,}5\ A$$

En consecuencia,

$$T_{auton} = \frac{9\ Ah}{0{,}5\ A} = 18\ h$$

El tiempo que el sistema puede estar activo sin recibir energía por radiación solar es de 18 horas.

2.4.4. Diagrama de conexiones

En la Figura 2.35 se muestra un esquema gráfico de los perifericos utitilizados en este proyecto, como son: módulo USB wifi, adaptador de autio USB, este para derivar y conectar un micrófono y altavoz para la entrada y salida de audio y el módulo de cámara. Además, el pin GPIO 17 que se configura como entrada para el botón de pánico.

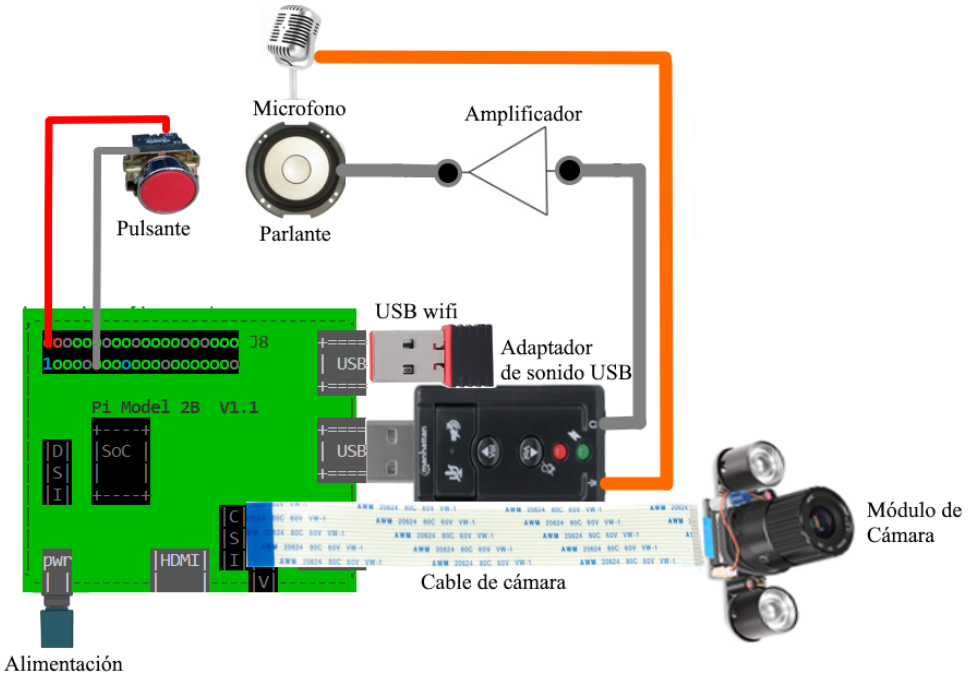


Figura 2.35: Esquema de conexiones para el proyecto botón de pánico. (Fuente: Autores)

Capítulo 3

IMPLEMENTACIÓN Y ANÁLASIS DE RESULTADOS

Una vez terminado el despliegue del servidor web, la interfaz web,la lógica del sistema botón de pánico. Se realiza las pruebas de audio al momento de hacer llamada VoIP, también, se verifica la conexión del módulo de la cámara cuando se presione el botón **mostrar/ocultar cámara** en la interfaz.

Al realizar las pruebas se presentaron errores de audio, para ello se agregó un amplificador operacional LM 386 para mejorar la señal de salida. También se presentaron pruebas de conectividad, esto debido a la atenuación de señal que provoca estar dentro de la estructura de metal, esto es conocido como la **Jaula de Faraday** [40].

3.1. Implementación del proyecto

En este apartado se muestra la fase final del proyecto, aquí se colocaran los dispositivos electrónicos dentro de la tapa en la estructura tipo tótem como se muestra en la siguiente Figura 3.1 a). Podemos observar que en la parte superior se encuentra la Raspberry Pi 2, se conecta mediante una cinta flex al módulo de cámara y en la parte inferior el pulsante.

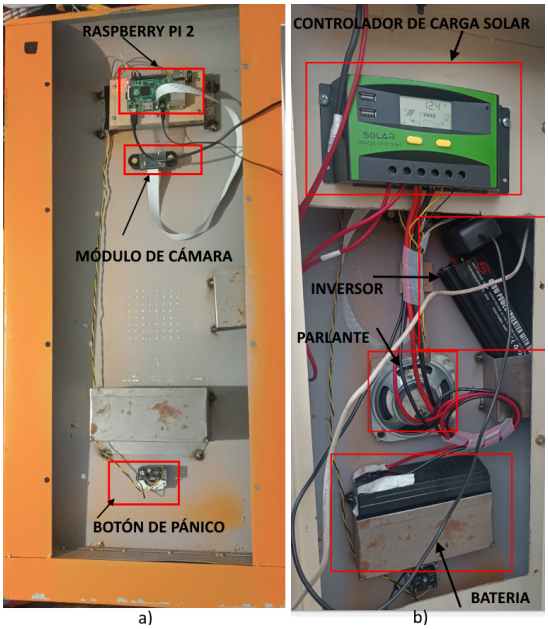


Figura 3.1: Equipos colocados dentro de la estructura tipo tótem. (Fuente:Autores)

Luego, la tapa fue colocada sobre la parte superior del tótem y se coloca en ella la batería, controlador de carga, inversor, parlante, micrófono además se realizo las conexiones respectivas como se muestra en la Figura 3.1 b). Finalmente se coloca la estructura del panel solar en la parte superior del tótem quedando así armado todo el sistema como se muestra en la Figura 3.2.

Figura 3.2: Armado del sistema de seguridad tipo tótem. (Fuente:Autores)

3.1.1. Pruebas de conectividad

Con el propósito de verificar conectividad entre el servidor Asterisk y una extensión telefónica, se realizan pruebas PING del enlace con carga y sin carga.
Cuando decimos sin carga hacemos referencia a que no existe ningún tipo de tráfico, excepto el que provoca el mismo protocolo ICMP, y una prueba de conectividad con carga hace referencia al tráfico generado en el enlace, por ejemplo, una llamada de audio.
En primer lugar se lista las direcciones IP de servidor VoIP y una extensión como se indica en la Figura 3.3. La prueba de conectividad implica realizar un ping desde la dirección IP 172.16.215.242 hasta la dirección 172.16.211.73. La Figura 3.4 muestra

```
raspberrypi*CLI> sip show peers
Name/username              Host                                  Dyn Forcerport Comedia    ACL Port     Status      Description
200/200                    172.16.211.73                          D  Auto (No)  No             33380    OK (12 ms)
201/201                    172.16.215.242                         D  Auto (No)  No             5064     OK (7 ms)
2 sip peers [Monitored: 2 online, 0 offline Unmonitored: 0 online, 0 offline]
raspberrypi*CLI>
```

Figura 3.3: Listado de direcciones IP correspondientes al servidor VoIP y a una extensión telefónica. (Fuente: Autores)

la respuesta de ping desde un teléfono con la extensión hasta el servidor VoIP. El resultado indica una latencia promedio de 188 ms, esta prueba corresponde al escenario "sin carga".

```
pi@raspberrypi:~ $ ping 172.16.211.73
PING 172.16.211.73 (172.16.211.73) 56(84) bytes of data.
64 bytes from 172.16.211.73: icmp_seq=1 ttl=64 time=357 ms
64 bytes from 172.16.211.73: icmp_seq=2 ttl=64 time=80.3 ms
64 bytes from 172.16.211.73: icmp_seq=3 ttl=64 time=304 ms
64 bytes from 172.16.211.73: icmp_seq=4 ttl=64 time=226 ms
64 bytes from 172.16.211.73: icmp_seq=5 ttl=64 time=147 ms
64 bytes from 172.16.211.73: icmp_seq=6 ttl=64 time=60.6 ms
64 bytes from 172.16.211.73: icmp_seq=7 ttl=64 time=293 ms
64 bytes from 172.16.211.73: icmp_seq=8 ttl=64 time=225 ms
64 bytes from 172.16.211.73: icmp_seq=9 ttl=64 time=136 ms
64 bytes from 172.16.211.73: icmp_seq=10 ttl=64 time=52.7 ms
^C
--- 172.16.211.73 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9013ms
rtt min/avg/max/mdev = 52.711/188.172/357.414/103.179 ms
```

Figura 3.4: Respuesta de ping desde un teléfono al servidor VoIP cuando no existe tráfico. (Fuente: Autores)

Por otra parte, la Figura 3.5 indica los resultados de una prueba de ping cuando se está ejecutando una llamada. La latencia promedio según la prueba es de 13 ms.

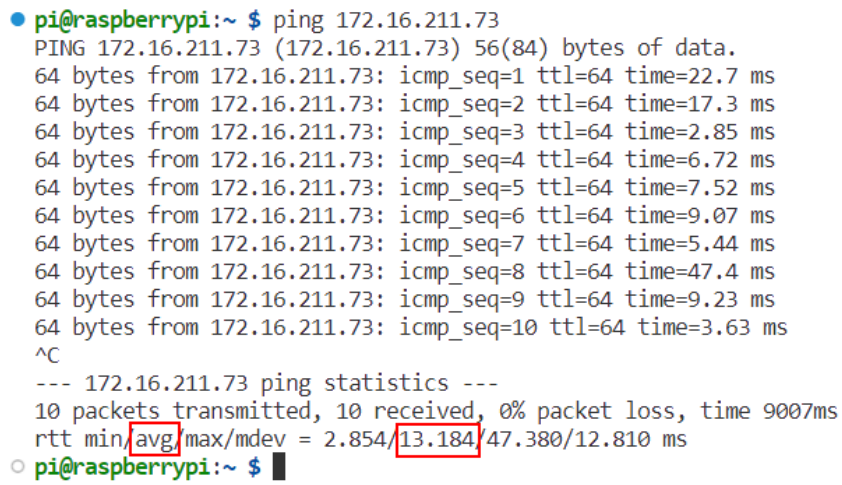


Figura 3.5: Respuesta de ping desde un teléfono al servidor VoIP cuando existe tráfico. (Fuente: Autores)

Las pruebas de conectividad indican una latencia más alta cuando no existe carga en el enlace y una latencia menos cuando se está ejecutando una llamada de voz. Estos resultados inusuales se presentan debido que en la red interna de la Universidad Politécnica Salesiana existe preferencia de tráfico para el protocolo de VoIP.

3.1.2. Análisis de resultados

El sistema de seguridad tipo tótem opera desde dos perspectivas distintas: una es la del individuo que activa el botón y la otra es la de la interfaz web o el personal de seguridad encargado.

Para ingresar a la interfaz web se escribe la dirección IP del servidor web que en este caso seria la IP de la Raspberry Pi adicional el puerto 5270, como se muestra a continuación en la siguiente Figura 3.6. El usuario y contraseña para ingresar es **admin**.

En la página principal se mostrara una barra de navegación en donde se encuentra los botones de **INICIO, MOSTRAR HISTORIAL, CERRAR SESIÓN** como se muestra en la Figura 3.7.

Cuando la persona encuentra en una situación de emergencia presiona el botón que esta en el tótem, una vez activado el botón se realiza una llamada de VoIP desde el tótem que es la extensión 200 hasta la interfaz web que contiene la extensión 201, en donde se visualiza un mensaje con el texto: **SE HA PRESIONADO EL BOTÓN DEL**

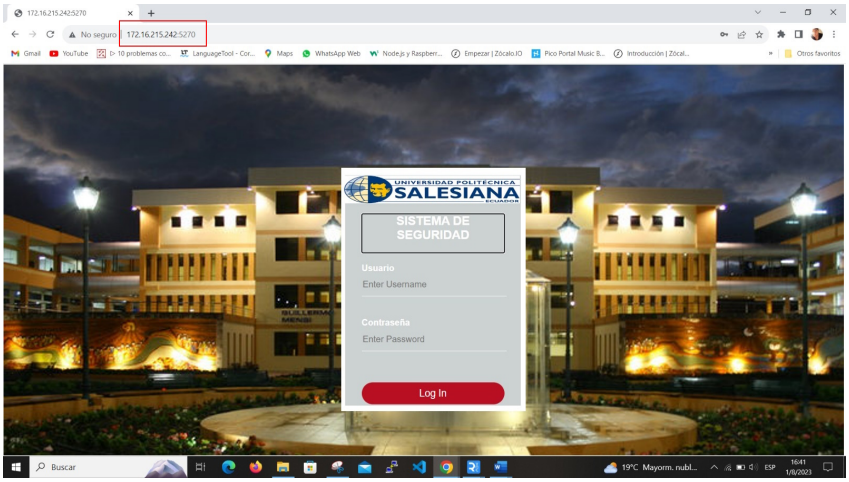


Figura 3.6: Interfaz de login. (Fuente: Autores)

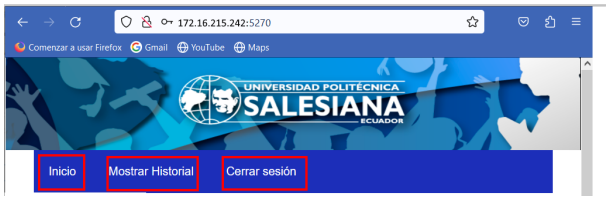


Figura 3.7: Barra de navegación. (Fuente: Autores)

tótem 1 e inmediatamente entra la llamada como se muestra en la Figura 3.8.

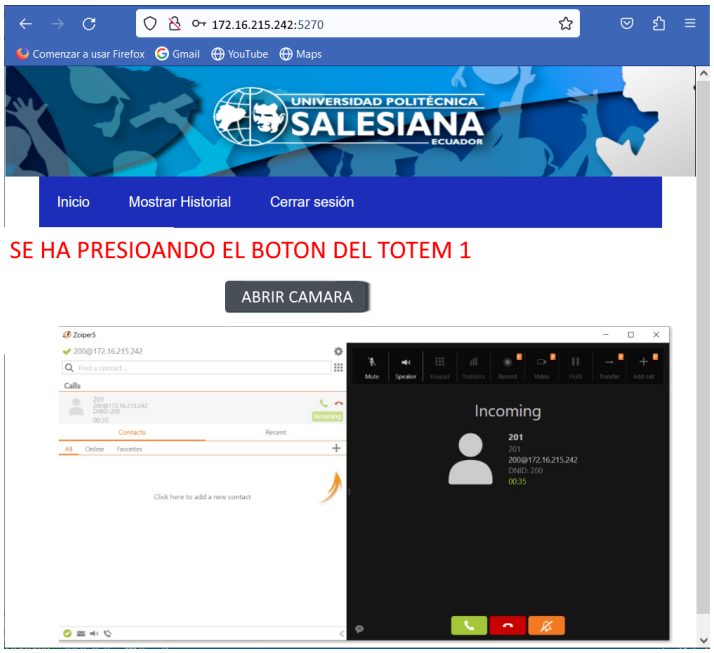


Figura 3.8: Mensaje producido al momento de presionar el botón de pánico. (Fuente: Autores)

La persona que monitorea el sistema desde la interfaz web puede presionar el botón **ABRIR CAMARA**, para visualizar la persona que ha presionado el botón como

se muestra en la siguiente Figura 3.9 .

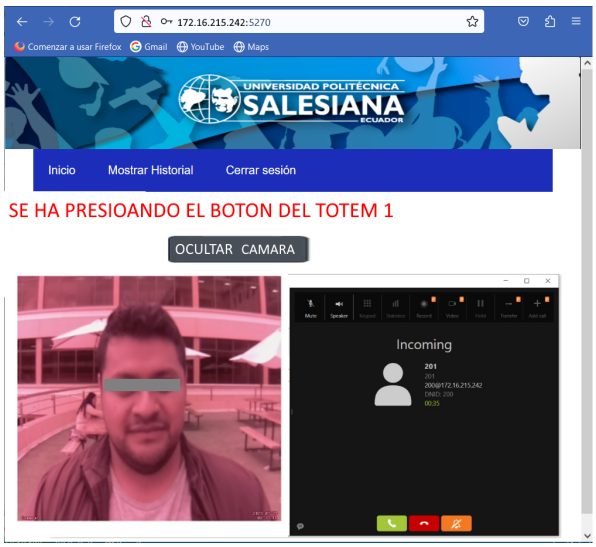


Figura 3.9: Visualización de la persona que ha presionado el botón. (Fuente: Autores)

La asistencia se atiende de manera inmediata, en el caso que de que la persona que presionó el botón lo hizo por error, la persona que monitorea tiene la opción en el interfaz web de abrir en el menú de navegación el botón **MOSTRAR HISTORIAL**, aquí va a presionar el botón **Registro Falso**, como se muestra en la siguiente Figura 3.10.

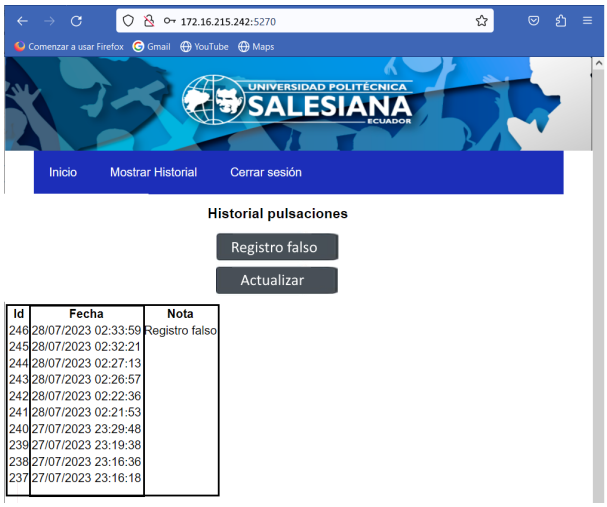


Figura 3.10: Seleccionar un registro de asistencia como falto cuando se presiona por error el botón (Fuente: Autores)

3.2. Análisis Económico

A continuación, se llevará a cabo un análisis financiero integral para la fabricación de sistemas de seguridad tipo tótem. Este análisis abarcará un período de corto plazo de 4 años e incluirá evaluaciones de CAPEX (inversiones de capital), OPEX (gastos operativos), así como los cálculos de Valor Actual Neto (VAN) y Tasa Interna de Retorno (TIR).

3.2.1. Inversión Inicial

En la siguiente Tabla 3.1 se detalla el costo de implementación del sistema de seguridad tipo tótem. Se va a dividir en 3 partes que son: sistema de alimentación, infraestructura del tótem e interfaz web.

Tabla 3.1: Costos de implementación del sistema de seguridad tipo tótem

Cantidad	Descripción	Costo
	Sistema de alimentación	
1	batería seca	$28
1	Panel solar	$35
1	Controlador de carga solar	$40
1	Inversor de corriente de 12Vcc a 110Vca	$58
1	Estructura para panel solar	$ 30
	TOTAL	**$191**
	Infraestructura y dispositivos electrónicos	
1	Raspberry Pi 2	$50
1	Adaptador wifi USB	$6
1	Adaptador de sonido USB	$3.50
1	Módulo de cámara	$40
1	Amplificador lm 386	$1.20
1	Parlante y microfono	$ 6.80
1	Pulsador industrial	$ 1.50
1	Estructura del tótem	$ 130
	TOTAL	**$239**
	Interfaz web	
	Diseño de interfaz Web	$300
	Mano de obra	$850
	TOTAL	**$1150**
	TOTAL DEL SISTEMA	**$1580**

En la Tabla 3.1 se muestra que cada estructura tendrá un costo de **$1580**.

3.2.2. Proyección

El objetivo de ventas en el primer año es de 36 unidades, los compradores potenciales serian instituciones educativas y gobiernos autónomos municipales. Para el siguiente año incrementar la producción un 50 % vendiendo 54 unidades para el segundo año. Además, se generarán ingresos por concepto de mantenimiento, cada una de estas infraestructuras requerirá mantenimiento trimestralmente. Durante el primer año, se llevarán a cabo un total de 144 mantenimientos, correspondientes a la venta de 36 totems, con un costo de mantenimiento de $20 por cada tótem. Este patrón se mantendrá en los años siguientes. En la Tabla 3.2 se muestra la proyección detallada.

Tabla 3.2: Proyección anual de ingresos del sistema de seguridad tipo tótem

PRIMER AÑO			
Descripción	**Cantidad**	**Precio unitario**	**total**
Sistema seguridad tótem	36	$ 1580	$ 56880
Mantenimiento	144	$20	$ 2880
		TOTAL	**$ 59760**
SEGUNDO AÑO			
Descripción	**Cantidad**	**Precio unitario**	**total**
Sistema seguridad tótem	54	$ 1580	$ 85320
Mantenimiento	216	$20	$ 4320
		TOTAL	**$89640**
TERCER AÑO			
Descripción	**Cantidad**	**Precio unitario**	**total**
Sistema seguridad tótem	81	$ 1580	$ 127980
Mantenimiento	324	$20	$6480
		TOTAL	**$134460**
CUARTO AÑO			
Descripción	**Cantidad**	**Precio unitario**	**total**
Sistema seguridad tótem	120	$ 1580	$ 189600
Mantenimiento	480	$20	$9600
		TOTAL	**$199200**
TOTAL DE INGRESOS 4 AÑOS			**$483060**

3.2.3. Análisis de CAPEX Y OPEX

En la Tabla 3.3 se detalla los gastos de capital y operativos del proyecto durante el periodo de proyección, se contrata dos técnicos para programación y otro para ofrecer mantenimiento inmediato en caso de requerir dando mejor servicio al cliente. También, los apartados de *estructura del tótem*, *sistema de alimentación* y *materia prima*

se calculó en base a la cantidad de sistemas de seguridad proyectadas para cada año como se puede ver en la Tabla 3.2

Tabla 3.3: Gastos de capital y gastos operativos

		CAPEX			
Descripción	**1er Año**	**2do Año**	**3er Año**	**4to Año**	**Total**
Compra de un vehículo	$ 30000	-	-	-	$ 30000
Equipos de oficina	$ 1200	-	-	-	$ 1200
Herramientas básicas	$ 2000	-	-	-	$ 2000
Estructura del tótem	$ 1300	$1950	$ 2860	$ 4290	$ 10400
Sistema de alimentación	$ 1910	$2865	$ 4202	$ 6303	$ 15280
Materia prima	$ 1090	$1635	$ 2398	$ 3597	$ 8720
	$ 37500	**$ 6450**	**$9460**	**$ 14190**	
				TOTAL DE CAPEX $67600	
		OPEX			
Descripción	**1er Año**	**2do Año**	**3er Año**	**4to Año**	**Total**
Marketing	$ 1320	$1320	$ 1320	$1320	$ 5280
Renta de local	$ 1200	$1200	$ 1200	$1200	$4800
Servicios básicos	$ 200	$250	$ 300	$350	$ 1100
Mantenimiento de vehículo	$ 2000	$ 2000	$ 2500	$ 2500	$ 9000
Técnico de programación	$ 7200	$7200	$ 7200	$7200	$ 28800
Técnico de mantenimiento	$ 7200	$7200	$ 7200	$7200	$ 28800
Secretaria	$ 5400	$5400	$ 5400	$5400	$ 21600
Director financiero	$ 15000	$15000	$ 15000	$15000	$ 60000
Gerente	$ 34800	$34800	$ 34800	$34800	$ 139200
	$ 74320	**$74370**	**$74920**	**$ 74.970**	
				TOTAL DE OPEX $298580	
				TOTAL DE GASTOS $366180	

3.2.4. Tiempo de recuperación de la inversión

Para evaluar el tiempo para recuperar la inversión y determinar la viabilidad económica del proyecto, se emplean estimaciones de flujo de caja y el porcentaje de rentabilidad durante el periodo de proyección que es 4 años, en la tabla 3.4 se puede observar que es positiva a partir del segundo año. Ver Apéndice C

Para el calculo del flujo de caja utilizamos la siguiente ecuación:

$$\text{Flujo de Caja} = \text{Ingreso} - \text{Costo} \tag{3.1}$$

Aquí determinaremos el flujo de caja de cada año según el ingreso y costo proyectado en las tablas 3.2 y 3.3, respectivamente.

para la rentabilidad utilizamos la siguiente ecuación:

$$\text{Rentabilidad} = \left(\frac{\text{Ingreso} - \text{Costo}}{\text{Costo}}\right) \times 100 \tag{3.2}$$

Tabla 3.4: Periodo de recuperación de la inversión.

Inversión Inicial	Costos	Ingresos	Flujo Caja	Rentabilidad %
Año 0	$ -67600		$-67600	
Año 1	$ 74320	$ 59760	$ -14560	-19,5 %
Año 2	$ 74370	$ 89640	$ 15270	20,5 %
Año 3	$ 74920	$ 134460	$ 59540	79,4 %
Año 4	$ 74970	$ 199200	$ 124230	165,7 %

Para el cálculo del VAN se utiliza la ecuación 3.3. De acuerdo con la Resolución No. 133-2023-M emitida por el Banco Central del Ecuador, se establece que la tasa de interés activa efectiva máxima es del 10.33

$$\text{VAN} = \frac{CF_t}{(1+r)^t} - Inversion_{inicial} \tag{3.3}$$

Donde:

$$\begin{aligned} \text{VAN} &: \text{Valor Actual Neto} \\ CF_t &: \text{Flujo de caja en el período } t \\ r &: \text{Tasa de descuento} \\ n &: 4 \text{ años} \\ Inversion_{inicial} &: \$\ 67600 \end{aligned}$$

Sustituyendo valores:

$$\text{VAN} = \frac{-14560}{(1+0,103)^1} + \frac{15270}{(1+0,103)^2} + \frac{59540}{(1+0,103)^3}\frac{124230}{(1+0,103)^4} - 67600 \tag{3.4}$$

$$\text{VAN} = -13200,36 + 12551,28 + 44369,27 + 83931,39 - 67600 \tag{3.5}$$

$$\text{VAN} = \$60{,}051,58 \tag{3.6}$$

Para calcular la Tasa Interna de Retorno (TIR) se iguala la ecuación 3.3 a cero de la siguiente manera:

$$0 = \frac{CF_t}{(1+r)^t} - Inversion_{inicial} \tag{3.7}$$

Sustituyendo valores:

$$0 = \frac{-14560}{(1+r)^1} + \frac{15270}{(1+r)^2} + \frac{59540}{(1+r)^3}\frac{124230}{(1+r)^4} - 67600 \tag{3.8}$$

Despejando r tenemos

$$\mathrm{r} = 0,304159 = 30,4159\,\%. \tag{3.9}$$

La Tasa Interna de Retorno (TIR) es de 30,4 %.

Basándonos en los cálculos del VAN y TIR, podemos concluir que el proyecto es rentable. Esto se debe a que el porcentaje de la TIR es superior al porcentaje de la tasa de interés activa establecida por el Banco Central del Ecuador.

Capítulo 4

CONCLUSIONES Y TRABAJOS FUTUROS

4.1. Conclusiones

Se verifico que la implementación del sistema de seguridad disminuye tiempo de respuesta al momento de presentarse una emergencia, teniendo una comunicación directa con el personal de seguridad, haciendo que estudiantes y personal que labora dentro del campus se sientan seguros.

Se utilizó la tecnología .Net porque en comparación con las demás, esta me ofrece una mayor documentación,gran cantidad de bibliotecas y una amplia compatibilidad con otros lenguajes de programación.

Durante las pruebas, se observó que la señal de audio emitida por el parlante presentaba una potencia reducida. Para abordar este problema, se decidió incorporar un amplificador operacional de bajo coste y fácil acceso.

Para determinar la ubicación óptima del sistema de seguridad tipo tótem, se llevó a cabo un análisis detallado de los puntos de mayor concurrencia tanto para estudiantes como para el personal que trabaja en el campus, tal como se ilustra en la Figura 2.1. Además, se consideraron aquellos lugares donde la presencia de los guardias de seguridad es menos constante.

Las pruebas realizadas han validado la con-fiabilidad del sistema de seguridad al evidenciar la ausencia de cambios de estado en la entrada del pin GPIO. Además, se

ha verificado que el sistema de alimentación mediante paneles solares proporciona una autonomía adecuada para garantizar el suministro suficiente a la Raspberry Pi 2.

El proyecto presenta una sólida viabilidad financiera. El VAN positivo indica que se espera que el proyecto genere un flujo de efectivo neto favorable a lo largo de su vida útil proyectado.

4.2. Recomendaciones y Trabajos Futuros

Cómo trabajo futuro se puede agregar en la parte superior del tótem una cámara tipo domo para tener una vista panorámica al rededor de la estructura, también, adicional a este se puede mejorar la interfaz web agregando el softphone dentro de la misma y un visualizador que permita saber el porcentaje de carga y la duración máxima de la batería.

Se recomienda implementar estos sistemas de seguridad en más universidades y centros educativos, colocando las estructuras en lugares estratégicos de mayor afluencia de personas.

Se aconseja la utilización de un módulo de antena para establecer la conexión de la Raspberry Pi con la red local, ya que la estructura del tótem es metálica. Esto se debe a que la presencia de dicha estructura puede ocasionar una atenuación de la señal.

Realizar talleres de capacitación para tener un buen uso del sistema de seguridad, de manera que no sucedan falsas alarmas. Además, diseñar unidades de montaje en pared e implementar en lugares internos del campus, como puede ser, secretaria, tesorería, corredores de laboratorios y pasillos de las aulas.

Glosario

HTTP Protocolo de Transferencia de Hipertexto – Hypertext Transfer Protocol.

IDE Entorno de Desarrollo Integrado – Integrated Development Environment.

.NET Tecnologías Habilitadas para la Red – Network Enabled Technologies.

RaPi Raspberry Pi.

SIP Protocolo de Inicio de Sesión – Session Initiation Protocol.

TCP Protocolo de Control de Transmisión – Transmission Control Protocol.

UDP Protocolo de Datagrama de Usuario – User Datagram Protocol.

VoIP Voz sobre protocolo de Internet – Voice over Internet Protocol.

WIFI conexión inalámbrica – Wireless Fidelity.

Referencias

[1] *3 de cada 10 estudiantes universitarios en Quito han sido víctimas de robo o de intento de robo*. dirección: https://www.lahora.com.ec/pais/3-cada-10-estudiantes-universitarios-quito-victimas-robointento-robo/.

[2] *Universidades del Austro preparan un plan para combatir la inseguridad en los alrededores de las instituciones de educación superior*. dirección: http://www.elmercurio.com.ec/2022/11/23/universidades-austro-inseguridad-plan/.

[3] *La inseguridad en Ecuador escala a niveles históricos y se impone como prioridad del próximo Gobierno*. dirección: https://elpais.com/internacional/2023-07-10/la-inseguridad-en-ecuador-escala-a-niveles-historicos-y-se-impone-como-prioridad-del-proximo-gobierno.html.

[4] *La inseguridad y la delincuencia en Ecuador: Retos y soluciones urgentes*. dirección: https://prensa.ec/2023/06/02/la-inseguridad-y-la-delincuencia-en-ecuador-retos-y-soluciones-urgentes/.

[5] *Muertes violetas crecieron en 110 cantones en el primer semestredel 2023*. dirección: https://www.primicias.ec/noticias/en-exclusiva/muertes-violentas-crecieron-cantones/.

[6] *EMERGENCY BLUE PHONES*. dirección: https://security.ubc.ca/home/safety-prevention-resources/emergency-blue-phones/.

[7] *UWF has 65 blue lights emergency phones strategically placed around campus.* dirección: https://uwf.edu/finance-and-administration/departments/police/services/blue-lights/.

[8] *Emergency Blue Light Telephones*. dirección: https://www.shaw.af.mil/News/Features/Display/Article/214499/20th-communications-squadron-keeping-airmen-connected/.

[9] *EMERGENCY ASSISTANCE PHONES.* dirección: https://www.augie.edu/student-life/campus-safety/services/emergency-assistance-phones.

[10] *PULSADOR*. dirección: https://www.areatecnologia.com/electricidad/pulsador.html.

[11] *Raspberry pi*. dirección: https://https://www.raspberrypi.org/.

[12] *TARJETA DE SONIDO EXTERNA USB*. dirección: https://www.informaticamoderna.com/Adaptador_USB_Son.htm.

[13] *PULSADOR*. dirección: https://www.areatecnologia.com/electricidad/pulsador.html.

[14] M. Boxwell. 2010, ISBN: 978-1-907670-00-8.

[15] *PanelSolar*. dirección: https://sabermas.umich.mx/archivo/tecnologia/133-numero-1755/268-paneles-solares-generadores-de-energia-electrica.html.

[16] *BATERIA*. dirección: https://www.velasco.com.ec/velasco/producto.php?id=3282.

[17] *Raspberry Pi Imager*. dirección: https://www.raspberrypi.com/news/raspberry-pi-imager-imaging-utility/.

[18] C. L. Velasquez Pluas et al., «Voz sobre IP en la Nube,» B.S. thesis, 2017.

[19] *Voz sobre IP residencial*. dirección: https://www.vonageforhome.com/what-is-voip/.

[20] D. J. Sanchez Espinoza, «Diseño e implementación de una central telefónica Voip de bajo costo mediante Asterisk y Raspberry Pi para pequeñas o medianas empresas,» 2021.

[21] *What is Session Initiation Protocol (SIP)?* Dirección: https://www.metaswitch.com/knowledge-center/reference/what-is-session-initiation-protocol-sip.

[22] J. F. Correa Aguirre et al., «Diseño e implementación piloto de un centro de contactos con protocolo SIP,» B.S. thesis, Espol, 2019.

[23] *TCP (Transmission Control Protocol): retrato del protocolo de transporte*. dirección: https://www.ionos.es/digitalguide/servidores/know-how/que-es-tcp-transport-control-protocol/.

[24] D. R. Rodríguez Herlein, «Análisis del rendimiento del protocolo TCP en redes de acceso wireless,» Tesis doct., Universidad Nacional de La Plata, 2020.

[25] *Protocolo de datagrama de usuario (UDP)*. dirección: https://es.khanacademy.org/computing/ap-computer-science-principles/the-internet/.

[26] *tecnología WIFI*. dirección: https://www.adslzone.net/reportajes/tecnologia/que-es-wifi-como-funciona/.

[27] *Caractersticas de los estandares WIFI*. dirección: https://www.intel.la/content/www/xl/es/support/articles/000005725/wireless/legacy-intel-wireless-products.html.

[28] *Qué es .NET Core*. dirección: https://openwebinars.net/blog/que-es-net-core/.

[29] *A tour of the C# language*. dirección: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/.

[30] *Blazor de ASP.NET Core*. dirección: https://learn.microsoft.com/es-es/aspnet/core/blazor/?view=aspnetcore-7.0.

[31] *Documentación de Visual Studio*. dirección: https://learn.microsoft.com/es-es/visualstudio/windows/?view=vs-2022.

[32] G. R. Solano Sánchez y D. S. Zhagñay Castro, «Diseño, implementación y prueba de una red de fibra óptica para el laboratorio de telecomunicaciones de la Universidad Politécnica Salesiana,» B.S. thesis, 2021.

[33] *Qué es Visual Studio Code y qué ventajas ofrece*. dirección: https://openwebinars.net/blog/que-es-visual-studio-code-.

[34] L. A. Llerena Ocaña, F. A. Viscaino Naranjo y W. V. Culque Toapanta, «Desarrollo de software con Net Core,» *Revista Universidad y Sociedad*, vol. 14, n.º 2, págs. 85-89, 2022.

[35] *<APIs en .NET Core >Tecnología*. dirección: https://cloudappi.net/net-core/.

[36] *Raspberry Pi OS*. dirección: https://www.raspberrypi.com/software/.

[37] T. I. Incorporated, *LM386 Low Voltage Audio Power Amplifier datasheet (Rev. C)*.

[38] *How to Install .Net Framework on Raspberry Pi*. dirección: https://linuxhint.com/install-dotnet-framework-raspberry-pi/.

[39] *ASP.NET Core Blazor project structure*. dirección: https://learn.microsoft.com/en-us/aspnet/core/blazor/project-structure?view=aspnetcore-7.0#blazor-server.

[40] A. R. R. Juárez, «Carga Eléctrica,»

Apéndice A

Instalación del sistema operativo y dependencias en la Raspberry Pi

A.1: Instalación del sistema operativo en la Raspberry Pi

Abrimos el software Raspberry Pi Imager, y seleccionamos el sistema operativo más actual para instalarlo en la microSD.

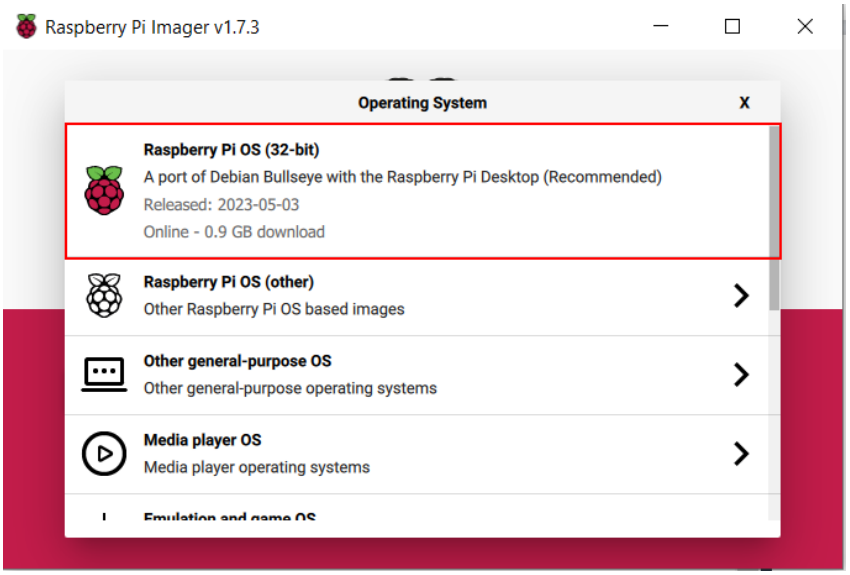


Figura A.1: Selección del sistema operativo a instalar.

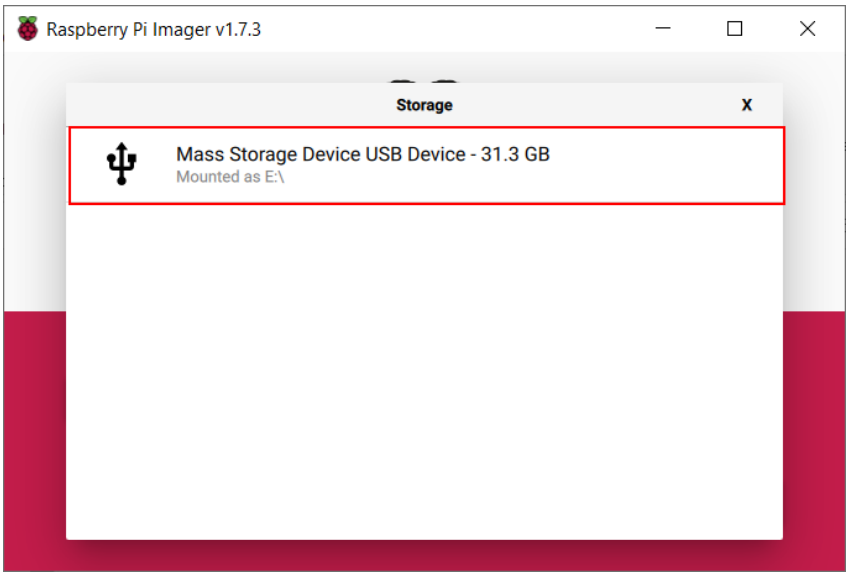


Figura A.2: Selección del medio de almacenamiento

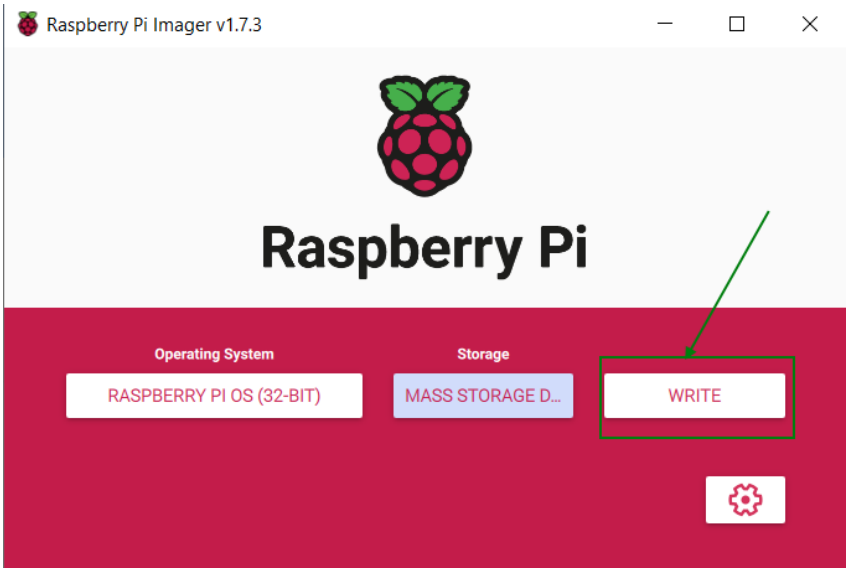


Figura A.3: Selección para escribir la imagen del S.O.

Una vez instalada la imagen del sistema operativo, se coloca la tarjeta microSD en la Raspberry Pi y se enciende.

A.2: Instalación del servidor Asterisk

Para iniciar sesión como superusuario en la Raspberry Pi se escribe el comando **sudo -i**, esto es para tener acceso completo a todo el sistema.

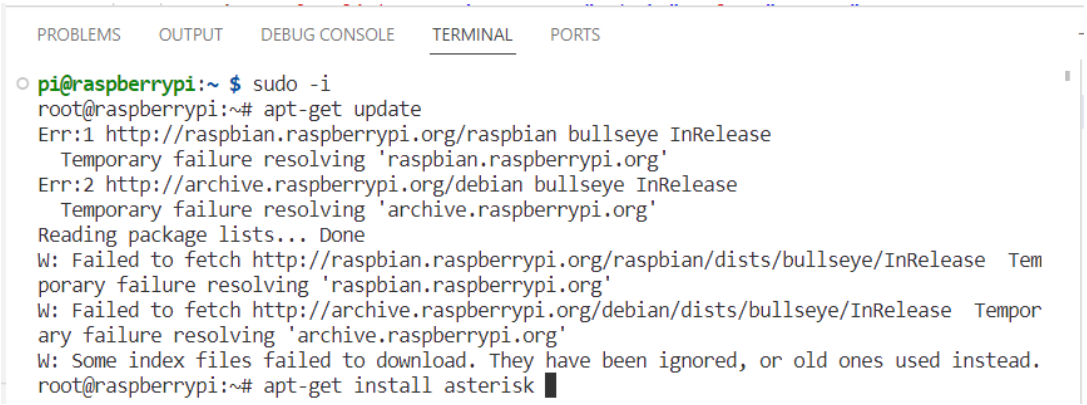


Figura A.4: Instalación del servidor asterisk.

Una vez instalado Asterisk, se descarga los siguientes paquetes.

Figura A.5: Instalación de paquetes de asterisk.

Para ver los paquetes instalados escribimos el siguiente comando.

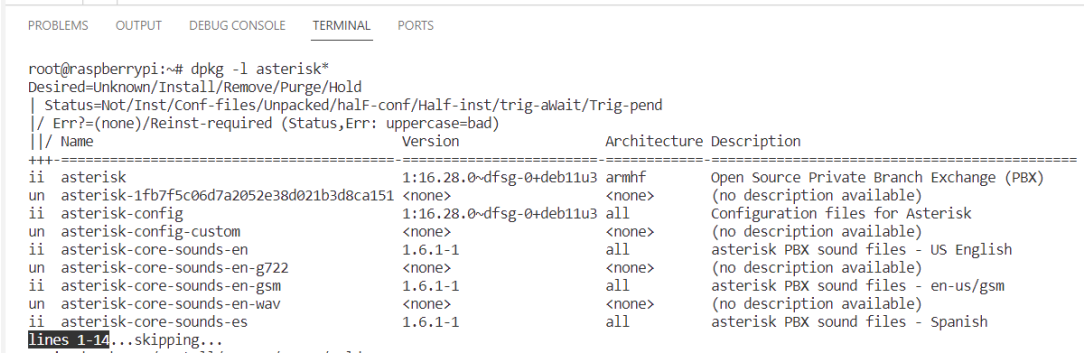


Figura A.6: Visualización de paquetes instalados.

A continuación se verifica el estado del servicio.

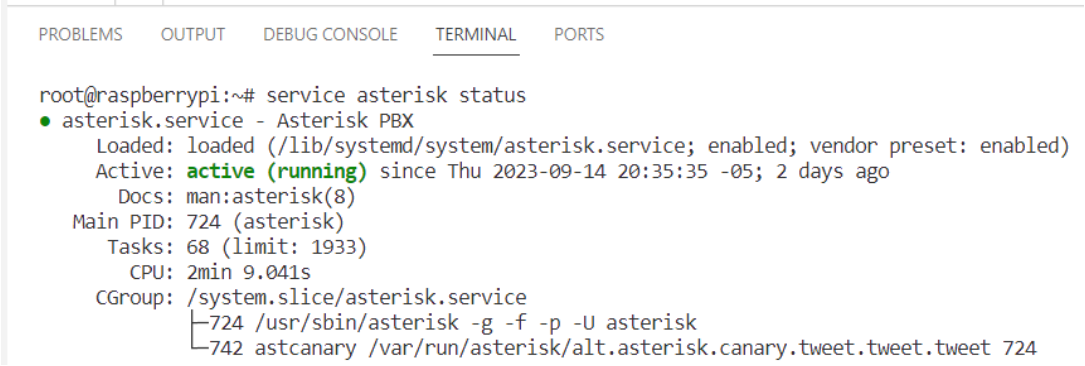


Figura A.7: Estado del servicio de asterisk activado.

Se configura los usuarios SIP y las extensiones de los mismos.

Figura A.8: Configuración de usuarios SIP.

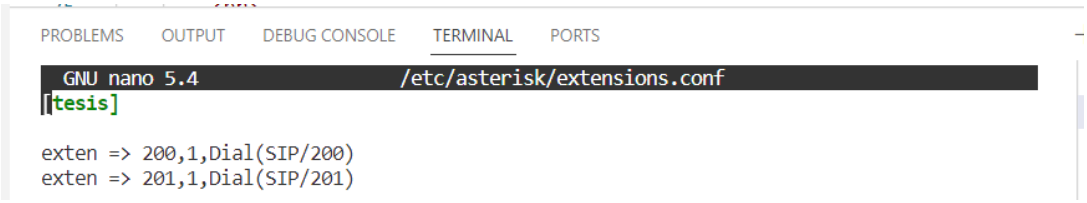


Figura A.9: Configuración de las extensiones.

Finalmente por medio del comando **asterisk -rvvv** se ingresa a la consola de asterisk para ver las configuraciones realizadas.

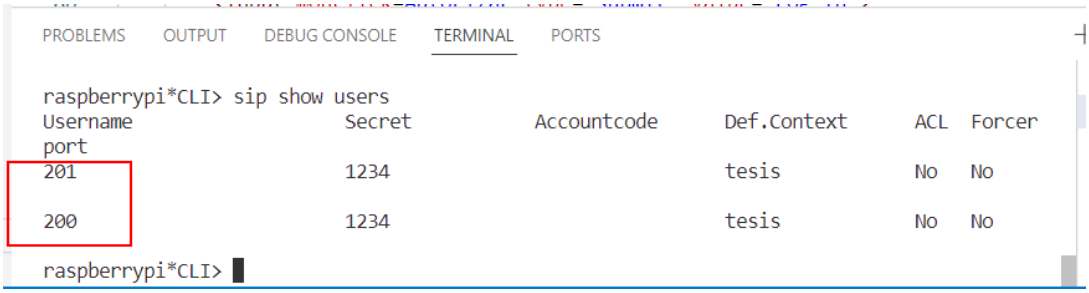


Figura A.10: Visualización de usuarios SIP creados.

A.3: Instalación de motion eye

Primero se instala las siguientes librerías.

Figura A.11: Librerias de motion eye

Segundo descargamos el software desde el siguiente enlace.

Figura A.12: Descarga del software motion eye

tercero se descarga el archivo get-pip.py, que se puede utilizar para instalar pip.

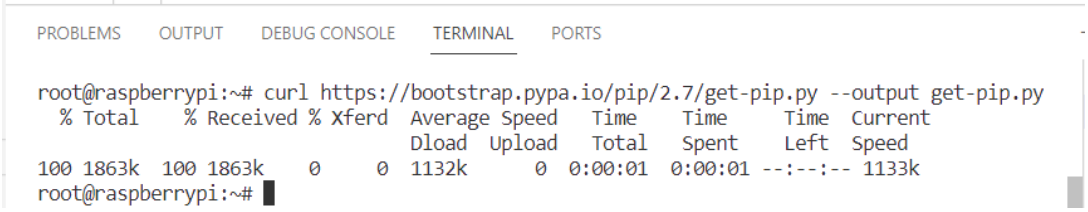


Figura A.13: Instalación de pip

Finalmente se instala el software motion eye

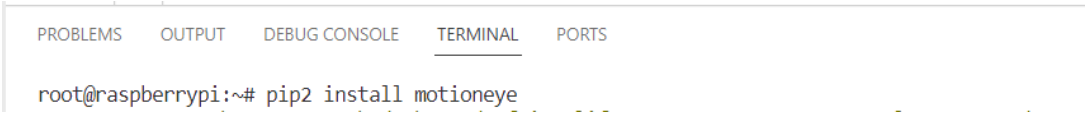


Figura A.14: Instalación del software motion eye

A.4: Instalación de .Net Core

Para instalar .Net Core se agrega el repositorio de Microsoft en la Raspberry Pi utilizando los siguientes comandos.

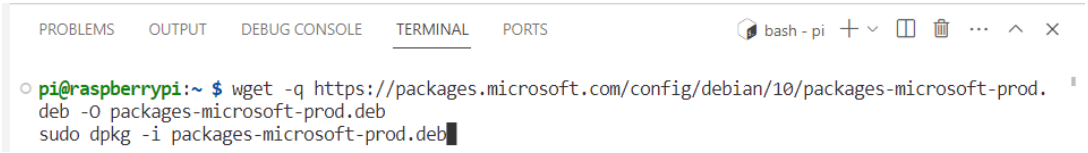


Figura A.15: Repositorio de Microsoft en Raspberry Pi.

Finalamente se procede a instalar el SDK actual de .NET core con el siguiente comando.

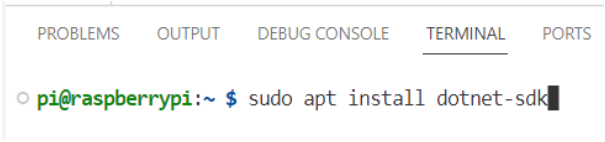


Figura A.16: Instalación de .NET core .

Apéndice B

Despliegue del servidor WEB, Back end y Front end

Para el despliegue del servidor web creamos un directorio dentro de la Raspberry Pi(BlazorCamera) y ahi se crea el proyecto de aplicación blazor server escribiendo el comando **dotnet new blazorserver**. El cual genera archivos que definen el proyecto y un archivo llamado **index.razor** que contiene el back end y front end de la aplicación.

Figura B.1: Creación del proyecto blazor server.

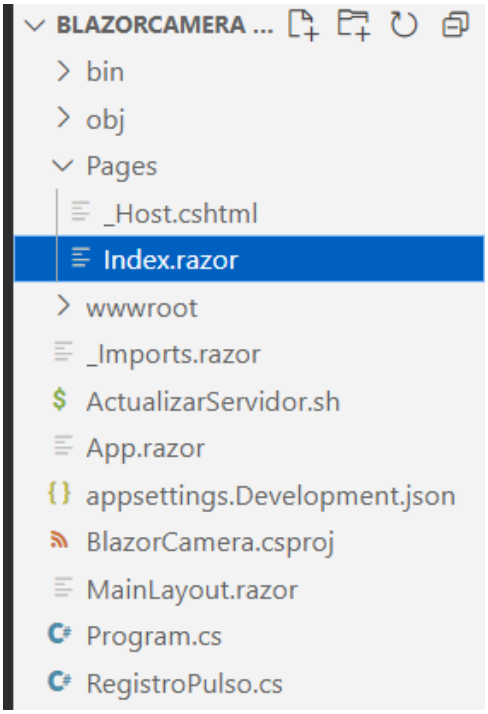


Figura B.2: Archivos generados para la aplicación blazor server.

Las siguientes lineas de código realizan el proceso de back end y front end.

FRONT END

B.1: Declaración de librerías y dependencias.

```
@page "/"
@implements IDisposable
@using System.Device.Gpio
@using System.Diagnostics
@using LiteDB
```

```
@inject LiteDatabase db
```

B.2: Autenticación de usuarios y visualización de la pagina principal.

```
@if (autorizado == true)
{
    <h1 class = "centrado">UNIVERSIDAD POLITECNICA SALESIANA </h1>
    <h2 class = "centrado">SISTEMA DE SEGURIDAD </h2>
    <ul class="menu">
      <li><a href="#">Inicio</a></li>
      <li><a href="#" >Mostrar Historial</a></li>
      <li><a href="#" >Cerrar sesion</a></li>
    </ul>

    @if(llamando == true)
    {
        <h1 style="color: red;">SE HA PRESIONADO EL BOTON DEL TOTEM</h1>
    }

    @if (mostrarCamara == true)
        {
            <button class="btn1" @onclick=@(() => mostrarCamara = false)<
            <img src="http://172.16.215.242:8081/">
        }

    @if (historial == true)
    {
    <div>
    <h3>Historial pulsaciones</h3>
    <button @onclick=MarcarUltimoRegistroComoFalso>Marcar Falso</button>
    <button @onclick=LeerHistorialPulsos>Actualizar</button>
    <table>
        <tr>
```

```
            <th>Id</th>
            <th>Fecha</th>
            <th>Nota</th>
        </tr>
        @foreach (var registroPulso in historialPulsos)
        {
            <tr>
                <td>@registroPulso.Id</td>
                <td>@registroPulso.Fecha</td>
                <td>@(registroPulso.Falso ? "Registro falso" : "")</td>
            </tr>
        }
    </table>
    </div>
    }
}
else
```

B.3: Formulario de inicio de sesión.

```
{
    <div class="login-box">
        <img src="../img/logoUPS.jpg" class="avatar" alt="Avatar Image">
        <h1>SISTEMA DE SEGURIDAD </h1>
        <br>
        <label for="username">Usuario</label>
        <input @bind-value=usuario type="text" name="username">
        <br>
        <label for="username">Contrasena</label>
        <input @bind-value=clave type="password" name="password">
        <br>
        <span class="login-error">@error</span>
        <br>
```

```
        <input @onclick=Autorizar type="submit" value="Log In">
    </div>
}
```

BACK END

B.4: Declaración de variables.

```
@code
{
    int pulsante = 17;
    bool llamando = false; // llamando
    bool rebote = false;
    bool mostrarCamara = false;
    bool historial = false;

    bool autorizado = false;
    bool pinesRegistrados = false;
    string error = string.Empty;
    string usuario = string.Empty;
    string clave = string.Empty;
    RegistroPulso[] historialPulsos = Array.Empty<RegistroPulso>();
```

B.5: Confgiuración del timer para efecto rebote del pulsante.

```
    GpioController controller = new GpioController();
    System.Timers.Timer timerRebote = new(1000);
    System.Timers.Timer timerLlamada = new(5000);

    // Se ejecuta al abrir la pagina
    protected override void OnAfterRender(bool firstRender)
    {
        if (!firstRender) return;

        // Configurar el timer para evitar el rebote del pulsante
```

```
        timerRebote.AutoReset = false;
        timerRebote.Elapsed += (s, e) => rebote = false;
        timerLlamada.AutoReset = false;
        timerLlamada.Elapsed += async (s, e) =>
        {
            llamando = false;
            await InvokeAsync(StateHasChanged);
        };

        // Registrar el pin como pulsante y asignar la funcion OnPinEvent
        controller.OpenPin(pulsante, PinMode.Input);
        controller.RegisterCallbackForPinValueChangedEvent(pulsante, PinE
        pinesRegistrados = true;
        StateHasChanged();
        Console.WriteLine("pin registrado 2");
    }
```

B.6: Función al presionar el pulsante.

```
    async void OnPinEvent(object sender, PinValueChangedEventArgs args)
    {
        if (pinesRegistrados == false) return;

        if (llamando == false && rebote == false)
        {
            rebote = true;
            llamando = true;
            timerRebote.Start();
            timerLlamada.Start();

            Console.WriteLine("llamando...");

            await GuardarRegistroPulso();
```

```
            await AbrirLlamada();

            await InvokeAsync(StateHasChanged);
        }
    }
```

B.7: Credenciales para inicio de sesión.

```
    async Task Autorizar()
    {
        error = string.Empty;

        if (usuario == "admin" && clave == "admin")
        {
            autorizado = true;
            await LeerHistorialPulsos();
        }
        else
        {
            autorizado = false;
            clave = string.Empty;
            error = "Usuario o clave incorrectos";
        }

        StateHasChanged();
    }
```

B.8: Función para cerrar sesión de la pagina principal.

```
    void CerrarSesion()
    {
        autorizado = false;
        usuario = string.Empty;
        clave = string.Empty;
```

```
        StateHasChanged();
    }
```

B.9: Configuración de un proceso en segundo plano para abrir softphone y ejecutar la opción de llamar.

```
    async Task AbrirLlamada()
    {
        // Comando a ejecutar
        // Configurar proceso para ejecutar comando
        var cmd = new Process();
        cmd.StartInfo.FileName = "/bin/bash";
        cmd.StartInfo.RedirectStandardInput = true;
        cmd.StartInfo.RedirectStandardError = true;
        cmd.StartInfo.RedirectStandardOutput = true;
        cmd.StartInfo.CreateNoWindow = false;
        cmd.StartInfo.UseShellExecute = false;
        cmd.Start();

        // Ejecutamos el comando y esperamos a que finalice
        await cmd.StandardInput.WriteLineAsync("twinkle -c --call 200 --i
        await cmd.StandardInput.FlushAsync();
        cmd.StandardInput.Close();
        await cmd.WaitForExitAsync();

        // Leemos el resultado del comando una vez finalizado
        var output = await cmd.StandardOutput.ReadToEndAsync();
        var error = await cmd.StandardError.ReadToEndAsync();

        Console.WriteLine(cmd.ExitCode);
        Console.WriteLine(output);
        Console.WriteLine(error);
    }
```

B.10: Configuración para guardar un registro.

```
async Task GuardarRegistroPulso()
{
    // Obtener coleccion de pulsos
    var pulsos = db.GetCollection<RegistroPulso>("pulsos");

    // Insertar nuevo registro
    pulsos.Insert(new RegistroPulso());

    // Guardar en el archivo
    db.Checkpoint();

    // Actualizar tabla de historial
    await LeerHistorialPulsos();
}

async Task MarcarUltimoRegistroComoFalso()
{
    /// Obtener coleccion de pulsos
    var pulsos = db.GetCollection<RegistroPulso>("pulsos");

    // Obtener ultimo pulso
    var ultimoPulso = pulsos.FindAll().LastOrDefault();

    // Si no existen datos cancelar
    if (ultimoPulso is null)
        return;

    // Marcar como pulso falso
    ultimoPulso.Falso = true;
```

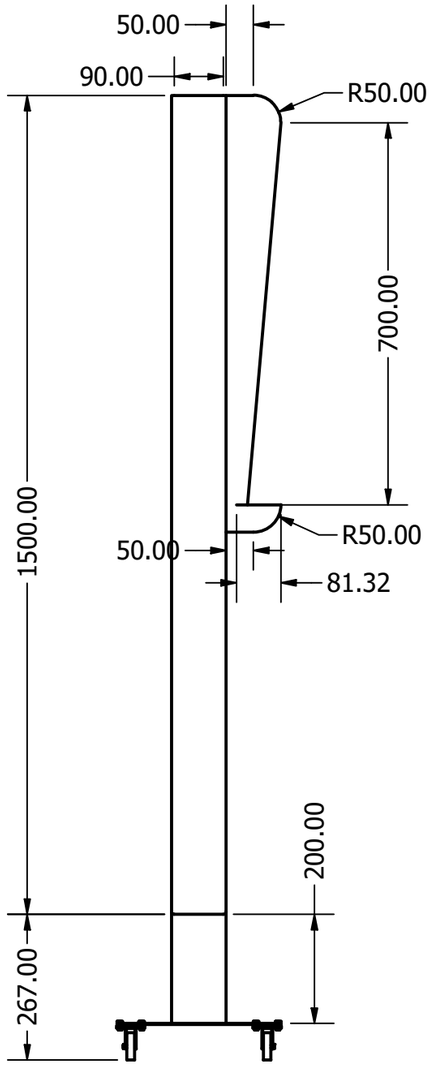
```
        // Actualizar registro
        pulsos.Update(ultimoPulso);

        // Actualizar tabla de historial
        await LeerHistorialPulsos();

        // Guardar en el archivo
        db.Checkpoint();
    }

    async Task LeerHistorialPulsos()
    {
        var pulsos = db.GetCollection<RegistroPulso>("pulsos");

        // Leer todos los pulsos e invertir el orden
        historialPulsos = pulsos.FindAll().Reverse().ToArray();

        // Actualizar interfaz de usuario
        await InvokeAsync(StateHasChanged);
    }
```

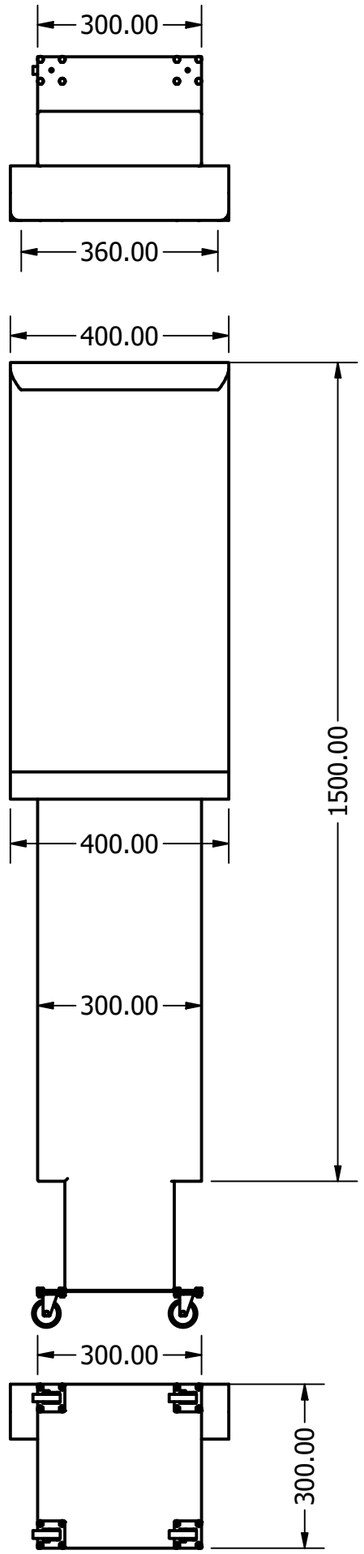
B.11: Liberar recursos en memoria al cerrar la pagina.

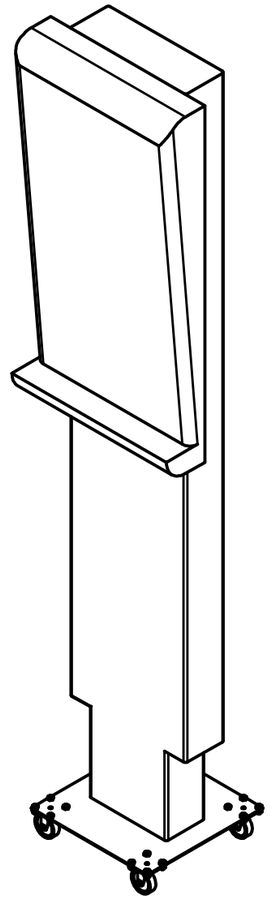
```
    public void Dispose()
    {
        controller.Dispose();
    }
}
```

Apéndice C

Diseño de la estructura y Análisis financiero







DRAWN Luis Olmedo	13/2/2023	TITLE		
CHECKED				
QA				
MFG				
APPROVED				
		SIZE A3	DWG NO dimensionesTotem	REV
		SCALE 0.089 : 1	SHEET 1 OF 1	

TASA DE DESCUENTO	CANTIDAD TOTEMS	MANTENIMIENTO ANUAL	COSTO IMPLEMENTACIÓN
0,103	36	144	$ 1.580,00
	54	216	$ 1.580,00
	81	324	$ 1.580,00
	120	480	$ 1.580,00

Años	CAPEX	OPEX	INGRESOS	FLUJOS NETO CAJA	RENTABILIDAD %
	$ 67.600,00			$ -67.600,00	
1		$ 74.320,00	$ 59.760,00	$ -14.560,00	-19,59095802
2		$ 74.370,00	$ 89.640,00	$ 15.270,00	20,53247277
3		$ 74.920,00	$ 134.460,00	$ 59.540,00	79,4714362
4		$ 74.970,00	$ 199.200,00	$ 124.230,00	165,7062825

VAN	$60.051,58	RENTABLE
TIR	30%	RENTABLE