ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

PROTOTIPO PARA GESTIÓN Y MONITOREO DE LA SEGURIDAD DEL LABORATORIO DE INTERCONECTIVIDAD DE LA EPN USANDO RASPBERRY PI

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

JÉFFERSON OMAR CÓRDOVA LEDESMA

jefferson.cordova@epn.edu.ec

RICARDO ALEXANDER FLOR JÁCOME

ricardo.flor@epn.edu.ec

DIRECTOR: Ing. FRANKLIN LEONEL SÁNCHEZ CATOTA MSc.

franklin.sanchez@epn.edu.ec

Quito, julio 2018

AVAL

Certifico que el presente trabajo fue desarrollado por Jéfferson Omar Córdova Ledesma y Ricardo Alexander Flor Jácome bajo mi supervisión.

_____

Franklin Leonel Sánchez Catota Msc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Nosotros, Jefferson Omar Córdova Ledesma, Ricardo Alexander Flor Jácome, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

JEFFERSON OMAR CÓRDOVA LEDESMA

RICARDO ALEXANDER FLOR JÁCOME

DEDICATORIA

Dedico este trabajo a mis padres Deida y Nelson, a ellos quienes siempre han confiado en mi y con sus consejos he podido afrontar problemas tanto en lo personal como en lo académico, todo lo alcanzado es gracias a ustedes.

Jéfferson

Dedico el presente trabajo a mis padres Ricardo e Isabel, a mi hermana Rommy y a mi amado hijo Franquito, es por ellos que soy lo que soy ahora. Los amo con mi vida. A ellos les dedico todo mi esfuerzo y perseverancia puestos en este trabajo.

Ricardo

AGRADECIMIENTO

Agradezco a mis padres por todo lo que incondicionalmente me han brindado y que gracias a su ánimo, apoyo y consejos he llegado hasta este punto de mi vida académica, estoy consciente de haberles sacado canas verdes al corregir mis errores mientras crecía. Mis logros y triunfos los he alcanzado gracias a ustedes. ¡Gracias Padres!

A Vanessa por permitirme compartir gratos momentos a su lado y enseñarme que con esfuerzo se puede lograr lo que uno se propone. Gracias por el inmenso cariño, paciencia y apoyo en el transcurso de nuestra vida universitaria.

A Benji porque para él la felicidad es encontrar un hueso, jugar, dormir o que le acaricie la barriga y con esto me ha enseñado que “vivir la vida con simpleza es la mejor manera de disfrutar y encontrar la felicidad.”

A mi director del Proyecto de Titulación Ing. Franklin Sánchez, gracias porque más que un profesor se ha convertido en un amigo con sus consejos.

A mi compañero Ricardo porque gracias a sus conocimientos y habilidades logramos culminar el presente y formar una excelente amistad.

A mis amigos y compañeros que compartimos aulas en la Institución, gracias por los momentos vividos. Finalmente, a la Escuela Politécnica Nacional y a mis profesores que a lo largo de la carrera universitaria supieron brindarme un poco de sus conocimientos.

Jéfferson

AGRADECIMIENTO

Le agradezco a mis padres Ricardo e Isabel por haber sido un pilar fundamental en la consecución de esta meta, por brindarme su incondicional apoyo, por los valores que me han inculcado, y porque a lo largo de mi vida siempre han velado por mi bienestar.

A mi hermana Rommy por haber sido un gran apoyo en los momentos más difíciles de mi vida, por haberme escuchado y comprendido cuando yo más lo necesitaba.

A mi primo Andrés porque más que amigo ha sido un hermano, incondicional a lo largo de toda mi vida ha estado en las buenas y en las malas, siempre escuchándome y ayudándome en cualquier situación.

A Joselin porque mientras fue mi compañera de vida siempre fue un motivo de alegría, cariño, mucha comprensión y consuelo, gracias a su paciencia y amor incondicional fue que pude seguir adelante y llegar a concluir este objetivo.

A Jefferson por haber sido un excelente compañero de tesis y amigo, por haberme tenido la paciencia necesaria y por motivarme a seguir adelante en los momentos de angustia y desaliento.

Ricardo

ÍNDICE DE CONTENID

INDICE DE FIGURAS

INTRODUCCION

METODOLOGIA

RESULTADOS Y DISCUSIÓN

INDICE DE TABLAS

INTRODUCCIÓN

METODOLOGIA

RESULTADOS

RESUMEN

La seguridad hace referencia a la ausencia de riesgo o amenazas, es un concepto del cual se derivan varios términos de acuerdo con la necesidad. En el campo de la video vigilancia se promueve la integración de mecanismos de alarma, controles de acceso, uso remoto del sistema, registro de eventos, etc., todo ello con el objetivo de detectar anomalías dentro del ambiente monitoreado.

Los laboratorios la Escuela Politécnica Nacional cuentan con equipos de alto valor, los cuales necesitan ser resguardados, de esta forma se ha considerado un caso en particular, el Laboratorio de Interconectividad de la FIEE que no dispone de seguridad alguna. Además de ofrecer servicios a la facultad existen tareas que pueden ser automatizadas o mejoradas por ejemplo la interacción del jefe de laboratorio con los usuarios y la sala.

Para solventar la problemática de seguridad del Laboratorio se plantea el desarrollo de un prototipo considerando una aplicación móvil, mediante la cual se controle el acceso en la puerta principal mediante código QR y se reciba una alerta cuando exista una apertura. Además, que sea posible visualizar los interiores de la sala a través de un stream de video y se pueda encender/apagar los computadores. Para controlar el acceso a la aplicación se tiene un registro de usuarios que son creados por el administrador del prototipo.

El prototipo implementado se somete a pruebas de funcionamiento para cada subsistema desde la aplicación móvil una vez realizado el registro de los usuarios.

PALABRAS CLAVE: Código QR, Raspberry Pi, Frameworks, Ionic

ABSTRACT

Security refers to the absence of risk or threat, it is a concept from which several terms are derived according to the need. In the field of video surveillance, several mechanisms are involved like warnings, access control, remote using of system, event registration, etc., all with the aim of detecting anomalies within the monitored environment.

The laboratories of the National Polytechnic School have high-value equipment, which need to be safeguarded, in this way the following case study has been considered in particular, the Interconnection Laboratory of the FIEE that does not have a security option. In addition to offering services to the faculty there are tasks that can be automated or improved for example, the interaction between the person in charge of the laboratory and the users or the room.

To solve the security problem of the Laboratory, the development of a prototype considering a mobile application is proposed, through which the access to the front door is controlled by QR code and an alert is received if there was an opening. In addition, it is possible to visualize the interior of the room through a video transmission and you can turn on / off the computers. To control access to the application, you have a registry of users that are created by the prototype administrator.

The prototype was implemented to perform a functional test for each subsystem from the mobile application once the user registration was made.

KEYWORDS: QR Code, Raspberry Pi, Frameworks, Ionic

1 INTRODUCCIÓN

En los tiempos actuales, la inseguridad existente en el país hace que las personas decidan tomar varias medidas y precauciones para así poder proteger sus bienes, entonces, generalmente se opta por el uso de sistemas de seguridad mediante video vigilancia ya que estos sistemas proveen de vigilancia de un área determinada en tiempo real y almacenan respaldos de las grabaciones obtenidas.

Un sistema de video vigilancia puede ser configurado e implementado basándose en diferentes parámetros y requerimientos, donde básicamente se activan alarmas ya sea por una percepción de movimiento, detección humana, identificación facial, entre otros. Con todas estas características estos sistemas presentan un amplio espectro de solución en múltiples escenarios dentro de zonas industriales o residenciales.

Por otro lado, desventajas como el espacio disponible para los equipos de un sistema de vigilancia, conectividad limitada, automatización de tareas y personalización de procesos han hecho que se empiece a utilizar dispositivos alternativos con más ventajas como es la Raspberry Pi, ya que permite agregar múltiples funcionalidades tan solo con la instalación de programas o aplicaciones a la plataforma de igual manera que permite agregar sensores y periféricos, mejorando aún más a un sistema de vigilancia tradicional.

Aprovechando estas ventajas un sistema de video vigilancia basado en Raspberry Pi, será destinado a monitorear los interiores del Laboratorio de Interconectividad de la Escuela Politécnica Nacional mediante una aplicación móvil, para brindar vigilancia a bienes materiales, estudiantes y profesores que utilizan dicho laboratorio.

1.1 Objetivos

El objetivo general de este Proyecto Técnico es: Desarrollar un sistema de vigilancia y gestión para el Laboratorio de Interconectividad de la Facultad de Ingeniería Eléctrica y Electrónica utilizando Software de código abierto.

Los objetivos específicos de este Estudio Técnico son:

- Analizar características de los dispositivos y sistemas operativos necesarios para el desarrollo del sistema mediante la revisión de sus propiedades.

- Diseñar un sistema de video vigilancia inalámbrica con Raspberry Pi y dos cámaras IP fijas, con el fin de ofrecer administración remota a través de una aplicación móvil.

- Desarrollar una aplicación móvil en Ionic para la administración y monitoreo del sistema por parte del usuario final utilizando las características y facilidades que ofrece este *framework*.

- Configurar en un servidor el control de encendido de los computadores con el fin de ejecutarlo desde la aplicación con la ayuda de la red del laboratorio.

- Realizar las pruebas de operación de cada uno de los componentes del prototipo para lograr su funcionamiento óptimo mediante ensayos de experimentación con variación de parámetros.

1.2 Alcance

Con este proyecto se pretende proveer de vigilancia al Laboratorio de Interconectividad de la Facultad, con la propuesta de un prototipo basado en dos subsistemas y una aplicación móvil. El primer subsistema es el de vigilancia, el cual ofrecerá un *stream* de video en tiempo real, y envío de notificaciones cuando se active el sensor de contacto que tendrá el prototipo. El subsistema estará conformado por: una Raspberry pi 3 Model B como elemento principal, a la cual se integran de forma inalámbrica dos cámaras IP, un dispositivo de almacenamiento para guardar videos, además de un sensor de contacto. El prototipo, adicionalmente, dispondrá de un modem 3G para enviar notificaciones SMS y una cámara USB para leer códigos QR que permitirá el ingreso de los usuarios al laboratorio. Finalmente, este prototipo contará con una fuente de alimentación alterna, para la Raspberry pi y sus periféricos (sensores, modem 3G, dispositivo de almacenamiento, lector QR.), que funcionarán durante un eventual corte de energía eléctrica

El segundo subsistema es el de monitoreo de PC's, donde se ofrecerá la funcionalidad del encendido y apagado remoto de los computadores que posee el laboratorio, por medio de la red LAN del laboratorio y un servidor Apache alojado en la Raspberry Pi. Todo este sistema será gestionado desde un smartphone, a través de una aplicación móvil.

Mediante Ionic un Framework de código abierto se desarrollará una aplicación móvil híbrida, la cual pondrá a disposición del usuario el control de funciones provistas a

través de los dos subsistemas como: la visualización del *stream* de video en tiempo real proporcionado por las cámaras IP, acceso al dispositivo de almacenamiento para la revisión de videos y el encendido/apagado remoto de computadores del laboratorio. Adicionalmente se podrá generar un código QR aleatorio a partir de los datos del usuario para permitir el acceso de estos a la sala mediante una validación de su información. Finalmente, la aplicación se acoplará con *Firebase*, plataforma de la cual se aprovechará lo servicios como autenticación y base de datos en tiempo real.

1.3 Marco Teórico

1.3.1 Antecedentes

Cuando se habla de seguridad, en los últimos años los sistemas de video vigilancia han crecido considerablemente llegando a tal punto que han logrado ser uno de los sistemas más empleados en el ámbito laboral y residencial [1]. Existen varias consideraciones al momento de la elaboración de sistemas de esta índole, por ejemplo, preguntas como ¿El sistema es de interiores o exteriores?, ¿Es necesario grabación de videos?, ¿El monitoreo se realiza desde pc, dispositivo móvil u otro artefacto?, ¿Se desea monitorear localmente o de forma remota?, son la base para encontrar requerimientos del sistema de video vigilancia. Estos sistemas se fundamentan en el uso de protocolos de transmisión de video, cuentan con la posibilidad de ser monitoreados remotamente utilizando herramientas como Internet.

Además, debido al avance tecnológico se pueden considerar dispositivos cotidianos (smartphone) para acoplar a los sistemas de video vigilancia y desarrollar un prototipo con mayores prestaciones tanto para el usuario como para el operador. Un teléfono inteligente o smartphone aprovecha el potencial que ofrecen los servicios alojados en el Internet y el poder que tiene de conectar a los usuarios sin importar distancias.

1.3.2 Laboratorio de Interconectividad de la FIEE

Ubicado en el séptimo piso del edificio Química-Eléctrica, el Laboratorio de Interconectividad de la FIEE acoge una cantidad considerable de usuarios debido a las asignaturas impartidas en el mismo. Provisto de computadores, racks de pared con sus respectivos equipos de red, routers, switches y el rack principal, este laboratorio tiene una buena infraestructura tecnológica, pero no cuenta con ningún mecanismo de seguridad.

Dentro del mismo se llevan a cabo varias tareas a diario, entre ellas están el encendido y apagado de los computadores que se realizan manualmente. Además, si una persona

ingresa a la sala fuera del horario laboral no hay manera de saberlo. De igual manera, la custodia de los bienes materiales en el interior del Laboratorio de Interconectividad se realiza personalmente, es decir si el administrador o ayudante no se encuentra en el laboratorio no se puede conocer la condición en la que se encuentra el sitio y sus interiores. Considerando la interacción del administrador con el laboratorio, el encendido/apagado manual de PC's, la baja seguridad de la sala y el alto costo de los bienes, se propone el actual proyecto, de esta forma se logra minimizar el tiempo al encender/apagar PC's y evitar que los computadores permanezcan encendidos en la noche, se agrega una forma autenticación para el ingreso a la sala y se desarrolla una interfaz para la interacción Administrador-laboratorio de forma remota.

Red de Datos

En una red de datos, las direcciones lógicas (IP) y físicas (MAC) son necesarias para lograr la comunicación entre los equipos.

Para sobrellevar problemas existentes dentro de una red se considera procedimientos establecidos en protocolos de red y que, además pueden brindar funciones y características adicionales a una red, como es el caso de *Wake On Lan* que permite controlar el encendido de un equipo a través de su tarjeta de red.

Para este proyecto se considera los computadores disponibles en el Laboratorio de Interconectividad, cada uno dispone de un punto de red conectado al switch central. Se tiene de un direccionamiento IP estático que, de ser necesario dispone de direcciones libres para conexión de otros dispositivos.

Estándar Wake On LAN (WoL)

Es un estándar de red que permite encender remotamente a los computadores, previa activación en el BIOS y configuración en el sistema operativo Windows del equipo. Una vez habilitado brinda beneficios para realizar tareas administrativas, de esta manera los administradores pueden encender los equipos de forma remota. Particularmente este mecanismo puede ser útil en Universidades o empresas que requieren el uso de una red *WAN (Wide Area Network).*

Wake on LAN utiliza un paquete denominado *magic packet* el cual es enviado a todos los dispositivos conectados a la red, de tal forma que, si este paquete es recibido, la tarjeta de red envía una señal hacia la fuente de energía para indicar el encendido del sistema.

Existen problemas y limitaciones al trabajar con este estándar, algunos de los cuales se detallan a continuación:

- ***Requerimientos del sistema*** *WoL* necesita que el hardware de red ofrezca esta funcionalidad, así como tener habilitado la activación por el *magic packet* en el computador, pero esto no representa un problema en computadores de escritorio modernos debido a que esta configuración viene por defecto en el adaptador de red.
- ***Falta de confirmación de entrega*** *WoL* fue diseñado para trabajar de manera simple (entrega rápida de paquetes), a costa de no proporcionar ningún tipo de confirmación de envío o entrega.
- ***Necesidad de dirección MAC*** Como parte de su funcionamiento es necesario la dirección MAC del ordenador destino, algo que se vuelve tedioso, confuso y extenso en redes con un alto número de ordenadores.

La Figura 1.1 muestra el funcionamiento de *WoL* desde una red externa. A través de la red se envía un *magic packet* con la dirección MAC perteneciente al computador destino, en el router de la red local de destino se utiliza ARP para encontrar esa dirección MAC y enviar el paquete. Una vez la tarjeta reconoce el *magic packet,* se enciende el computador.

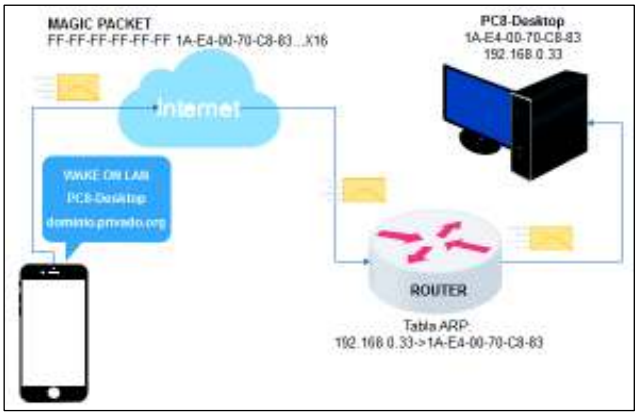

Figura 1.1 Diagrama de funcionamiento estándar *Wake On LAN*

1.3.3 Sistemas de Videovigilancia

Los sistemas de video vigilancia permiten la administración y visualización remota de las cámaras en cualquier momento, diseñados para supervisar varios entornos y actividades en las que el administrador del sistema no puede estar presente[2].

CCTV (Circuito Cerrado de Televisión), uno de los principales sistemas de vigilancia[3], permite una visualización de la actividad que se produce en el entorno ya sea en tiempo real o mediante grabaciones. Existen dos tipos de sistemas de video vigilancia, analógicos con una estructura compuesta con dispositivos del tipo analógico y digitales que son sistemas basados en IP.

Sistemas analógicos

Los CCTV analógicos están compuestos por monitores, cámaras, grabadores del tipo analógico, disponen de una funcionalidad y rendimiento básico además de costoso [3]. En los sistemas CCTV se tiene la conexión entre cámaras y multiplexor mediante cable coaxial de 75 ohmios. El monitor utilizado tiene capacidad de conexión a un VCR (*Video Cassette Recorder)* para grabar audio y video, ubicado en el cuarto de control la CCU (Unidad de Control de Cámara) permite controlar las señales de video captadas por cada una de las cámaras conectadas al multiplexor. En la Figura 1.2 se indica el diagrama de un sistema tradicional de CCTV analógico.

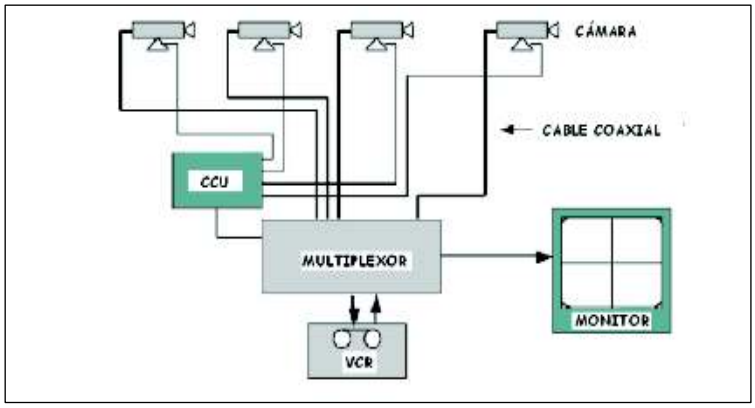

Figura 1.2 CCTV tradicional analógico[4]

Sistemas digitales basados en IP

Estos sistemas digitalizan el video y transmiten estos datos a través de la red mediante el protocolo de comunicación TCP/IP, siendo aplicables en una LAN o WAN. La principal ventaja de los sistemas basados en IP es el monitoreo remoto de un área a través de Internet, utilizando un navegador y una dirección IP, de esta manera se aprovecha las funciones provistas por los protocolos de transmisión en tiempo real.

La vigilancia IP además de permitir la transmisión de video de forma digital, ofrece mecanismos de acceso al sistema de forma segura, por ejemplo, el empleo de contraseñas para limitar el acceso.

En la Figura 1.3 se muestra el esquema de un sistema de video vigilancia basado en IP

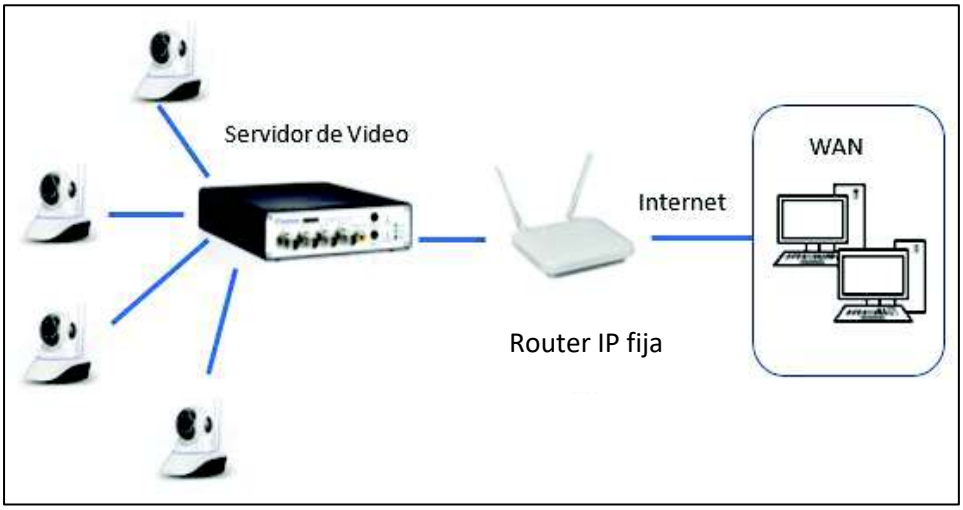

Figura 1.3 Sistema de video vigilancia basado en IP

1.3.4 Protocolos de transmisión de datos en tiempo real [5]

Un protocolo es un conjunto de reglas que permite la interacción entre dos entidades para transmitir información. Existen protocolos que se encargan de la transmisión y sincronización de audio y video además de trabajar, existen protocolos que funcionan en la capa aplicación como RTCP y RTSP, protocolos como UDP en capa de transporte y RTP a nivel de sesión se utilizan para transmitir los datos multimedia. Una transmisión eficiente de datos en tiempo real conlleva cumplir con ciertos requerimientos como ancho de banda, retraso de transmisión, fiabilidad, calidad de transmisión. A continuación, se describen los protocolos más conocidos:

- ***Transmission Control Protocol (TCP)*** Protocolo que permite una transmisión segura de datos y se asegura que los paquetes lleguen a su destino sin errores y en el orden que se transmitieron, sin embargo, no es adecuado para el envío de datos en tiempo real debido al retardo que produce, de esta forma se pueden ver afectados considerablemente este tipo de servicios. Es un protocolo orientado a conexión y trabaja sobre capa de transporte del modelo TCP/IP. Realiza control de flujo, tiene mensajes de confirmación de recepción de paquetes.

- ***User Datagram Protocol (UDP)*** Sobre UDP trabajan los protocolos como RTSP, RTCP los cuales se crearon con el fin de transmitir datos en tiempo real. Protocolo no orientado a conexión, realiza el intercambio de datagramas sobre la capa de transporte. No provee confirmación de mensaje ni control de flujo, de esta forma los paquetes de información se adelantan o retrasan. UDP no produce retardos en transmisión de video a costa de perder calidad en la imagen resultante, esto puede ser visto como ventaja en este tipo de servicios.

- ***Real-Time Transport Protocol (RTP)*** Permite la transmisión de datos en tiempo real sobre UDP, de manera poco fiable debido a que no maneja mecanismos que garanticen una recepción de los paquetes. Trabaja junto con RTSP en el área de control de flujo, además, cuando existe un envío de información por flujos (audio y video separados), RTP puede sincronizarlos en el destino. Protocolo no orientado a conexión trabaja en capa de sesión.

- ***Real-Time Transport Control Protocol (RTCP)*** Protocolo que trabaja en capa de sesión, es no orientado a la conexión. Basa su funcionamiento en el envío periódico de paquetes de control, mediante estos paquetes informa a todos los participantes de una sesión multimedia sobre calidad de servicio (QoS). RTCP no transporta datos multimedia, trabaja en conjunto con RTP para transportar y empaquetar esos datos.

- ***Real Time Streaming Protocol (RTSP)*** Protocolo no orientado a conexión, trabaja a nivel de aplicación, controla varios flujos sincronizados de audio y video, en otras palabras, controla el envío de datos multimedia. Para esto define diferentes requisitos y tipos de conexión de tal modo que asegura un envío de información eficiente sobre IP. RTSP es similar en sintaxis y operación a HTTP.

1.3.5 Raspberry pi 3 model b [6]

Desarrollado por Raspberry Pi Foundation en Reino Unido, es un pequeño computador de dimensiones similares a una tarjeta de crédito, el ingreso de la placa al mercado comenzó en 2006 con sus modelos A y B, en 2014 se anunció el modelo B+, en el 2016 se ingresa la Raspberry Pi 3 Model B siendo esta la tercera generación de tarjetas Raspberry Pi con hardware y software adicional respecto a modelos anteriores. Finalmente, en 2018 se lanzó el modelo RPi 3 B+ con varias mejoras incrementando la frecuencia del CPU a 1.4 GHz y mejorando las características de conectividad inalámbrica.

Software

En la página oficial de la fundación Raspberry Pi se muestran las opciones de sistema operativo que pueden correr sobre la plataforma. Raspberry Pi basa su funcionamiento en distribuciones o derivaciones de Linux el mismo que es de código abierto, distribuciones como Debian, Arch Linux y Fedora. Se muestran varias opciones de terceros las cuales son orientadas a distintas aplicaciones y se pueden apreciar en la Figura 1.4

Figura 1.4 Sistemas operativos de terceros [7]

Para este proyecto la principal aplicación será la de servidor, y para cumplir con dicha tarea es muy recomendable utilizar la versión oficial recomendada por la fundación Raspberry Pi, denominada Raspbian.

- **Raspbian**

Basada en Debian, se ofrecen las versiones: *lite* súper ligera, *desktop* con entorno gráfico y *noobs* con todas las facilidades para usuarios primerizos. El sistema dispone de algunas aplicaciones preinstaladas, además de navegadores como Chromium o Midori.

Es considerada una plataforma multilenguaje, debido a que el sistema operativo con el que trabaja además de soportar lenguajes de programación como Python, Java o Scratch[8] dispone de herramientas complementarias para desarrollar proyectos con ellos, entre estas herramientas se pueden mencionar: Mathematica, Sonic Pi, BlueJ Java IDE, Greenfoot Java IDE, Wolfram, entre otros [9]. La mayoría de estas herramientas están orientadas a la educación en programación.

Especificaciones técnicas

Las características de Raspberry pi 3 Model B se presentan en la Tabla 1.1. En general, se puede indicar que: dispone de 4 puertos USB, un socket 10/100 Base T para conexión de un conector RJ-45, interfaces inalámbricas una Wifi con 802.11 b/g/n y la otra bluetooth 4.1. Además, cuenta con salidas de audio y video, así como un conector de cámara CSI-2 y una ranura para tarjeta de memoria microSD.

Tabla 1.1 Especificaciones técnicas Raspberry Pi 3

Características	**Descripción**
CPU	1.2GHz Quad-Core ARM Cortex-A53
GPU	Dual Core VideoCore IV Multimedia, capacidad 1Gpixel/s, dispone de un codificador H.264 y una resolución 1080p
SoC (System on Chip)	Broadcom BCM2387
Salida de Video	HDMI, conector RCA para (PAL y NTSC) y una Interfaz DSI (Display Serial Interface) para la conexión de pantallas externas como un LCD
Salida de Audio	3.5mm Jack, HDMI
Ethernet	Socket RJ-45 brinda conexión 10 /100 Base-T
Puertos USB	4 puertos USB (2.0)
Conector de cámara	Interfaz serial para cámara 15 Pines (MIPI)
Conectores GPIO	40 pines 2.54 mm, Provisto de 27 GPIO entre ellos pines de +3,3V, +5V y GND
Conectividad Inalámbrica	802.11 b/g/n Wireless LAN y Bluetooth 4.1 (Clásico y Low Energy)

La Figura 1.5 indica la tarjeta Raspberry Pi 3 con cada uno de los componentes detallados en la Tabla 1.1 de esta forma se puede apreciar de mejor manera, tanto el tamaño como su distribución en la placa.

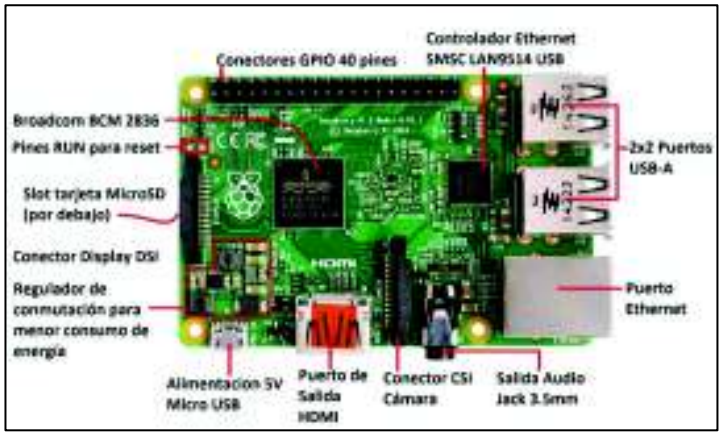

Figura 1.5 Distribución de componentes placa Raspberry Pi 3 Model B

GPIO (General Purpose Input/Output) Son considerados como una interfaz de conexión entre dispositivos externos y la placa, de este modo, se puede implementar prototipos que abarquen sensores, luminarias, entre otros, para poder analizar diversos fenómenos. La distribución de pines GPIO se ilustra en la Figura 1.6.

La tarjeta está provista de 40 conectores GPIO, los cuales se encuentran divididos en dos columnas de 20 pines cada uno y forman 4 grupos donde se tienen: 2 pines de +5V, 2 pines de +3.3V, 8 pines de GND y 28 pines que se pueden configurar como entrada o como salida dependiendo del propósito.

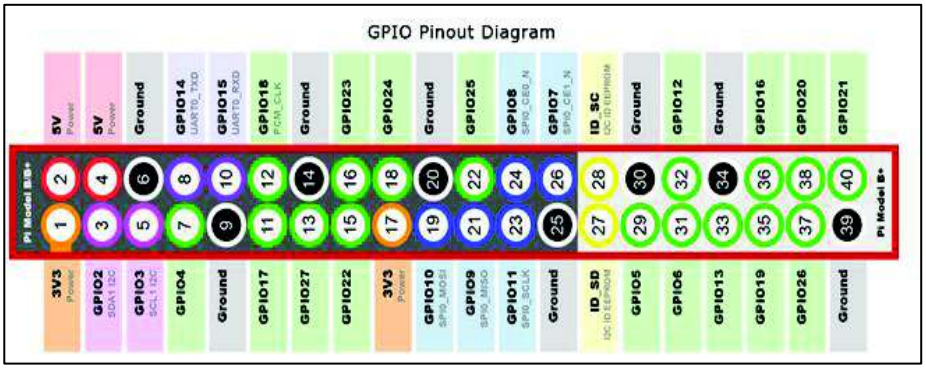

Figura 1.6 Diagrama esquemático de los 40 pines de propósitos generales de Raspberry Pi 3 Model B (GPIO)

Una utilización bastante extendida es la de instalar un servidor web en la Raspberry Pi 3 el cual permite brindar varios servicios al usuario, una opción muy completa y soportada por el sistema operativo es el servidor Apache del cual se tratará a continuación.

Servidor Web Apache (Raspberry Pi)

Apache es el servidor Web *Open Source* de mayor utilización en sistemas operativos Linux y sus derivaciones. Administra el acceso a sitios web en la nube, por ejemplo, un cliente realiza la petición de una página web, en el servidor se resuelve la petición de dicho cliente entregando la página solicitada mediante aplicaciones de navegación. Existen otro tipo de servidores con funcionalidades similares a Apache como es Light Httpd o *Internet Information Service* (IIS).

Características de Apache [10]

- Es de tipo modular, por lo tanto, puede ser adaptado a diferentes entornos y necesidades.
- Se caracteriza por ser multiplataforma debido a que puede operar en diversos sistemas operativos como Linux, Windows, MAC.
- Configuración simple a través de ficheros, además soporta HTTP 1.1.
- Desarrollado por Apache Software Foundation (ASF), siendo un servidor de código fuente robusto y gratuito.
- Es de tipo extensible, se puede incrementar la cantidad de módulos ampliando aún más su funcionalidad.
- Soporta diferentes lenguajes de programación, como Python, Perl, PHP.

Apache2 como se lo conoce al servidor web para distribuciones de Linux, permite almacenar páginas web o aplicaciones que puedan proveer de servicios tanto desde la intranet como desde Internet. La configuración de Apache2 se la realiza desde archivos alojados en el directorio "/etc/apache2" siendo el principal "apache2.conf" [11].

1.3.6 Sensores

Un sensor es considerado como un dispositivo mecánico y/o eléctrico el cual detecta magnitudes físicas como temperatura, calor, presión, movimiento, entre otros. Generalmente la salida es una señal eléctrica que varía según como se haya modificado alguna propiedad física por ejemplo la resistencia eléctrica. A continuación, se revisan características básicas de algunos sensores muy comunes dentro de los sistemas de vigilancia tradicionales.

Sensor de Infrarrojos

Dispositivo destinado a la medición de la radiación electromagnética infrarroja, dentro de su campo de visión. Es un elemento opto electrónico conformado por elementos que permiten receptar la radiación que es invisible para nuestros ojos emitida por los cuerpos. De esta forma se tiene un elemento emisor (LED emisor luz infrarroja) y un elemento receptor (Fotodiodo) como lo ilustra la Figura 1. 7. La alimentación del sensor es de 3,3V a 5V. Utilizado ampliamente en las alarmas antirrobo en donde la radiación IR emitida por el LED IR en condiciones normales cae directamente sobre el fotodiodo, de forma que si una persona intercepta dicha señal se produce en suceso de alarma.

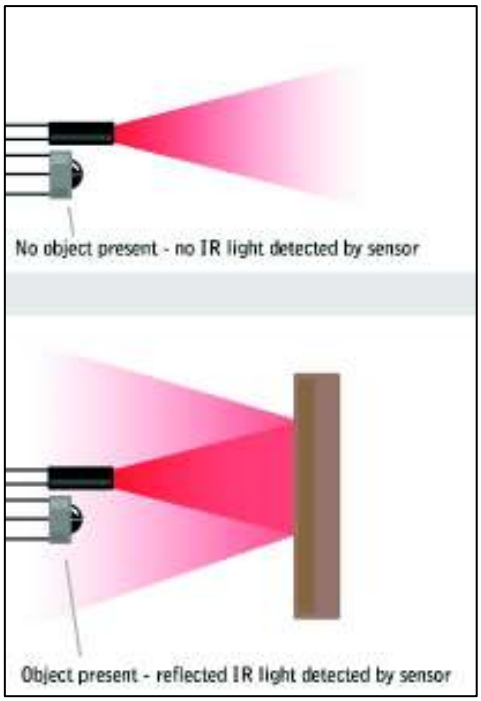

Figura 1. 7 Funcionamiento sensor de infrarrojos [12]

Sensor de detección de Movimiento

Un sensor de movimiento es un dispositivo electrónico que genera una señal eléctrica cuando existen objetos moviéndose dentro de su rango de detección, usualmente es incorporado en sistemas donde sea necesario el uso de alertas cuando existe movimiento en un entorno. Existen diversos tipos de sensores entre los que se destacan los PIR y los ultrasónicos.

PIR sensor: Un ejemplo de sensor PIR es el modelo HC-SR501, mide cambios en los niveles de radiación infrarroja producida por los objetos dentro del área detección La alimentación del sensor es de 5 a 12 V, aproximadamente consume menos de 1 mA de corriente trabajando en un rango de temperatura de -15° a 70° C. Funciones adicionales del sensor como el ajuste de sensibilidad y calibración de distancia de detección (3 a 7 metros) se las realiza mediante 2 potenciómetros, el ángulo de detección tiene una apertura de 90° a 110°, como se muestra en Figura 1.8.

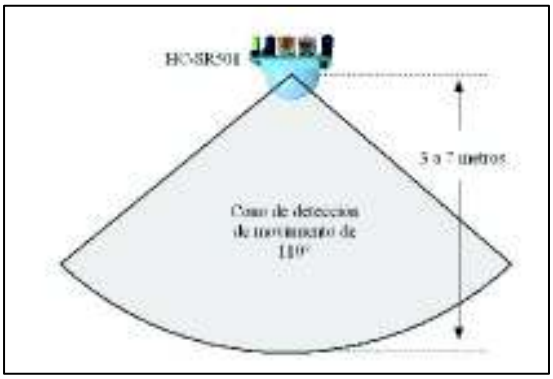

Figura 1.8 Ángulo de detección de movimiento del sensor PIR HC-SR501

Sensor ultrasónico: Sensor que trabaja en base a ondas ultrasónicas, la reflexión de las ondas determina un objeto moviéndose dentro del área de detección, de esta forma se puede determinar la distancia que existe entre el sensor y el objeto. Trabaja con una alimentación de 5V. Los sensores PIR necesitan línea de visión directa y por el contrario los sensores ultrasónicos no, por ello estos últimos sensores cubren toda un área y pueden detectar personas detrás de obstáculos, pero se debe considerar la desventaja de que son más sensibles a movimientos pequeños y en ocasiones, disparando falsas alarmas.

Sensor de contacto (magnético) [13]

Conocidos también como detectores magnéticos, se basan en una variación de campo magnético en respuesta a una variación física. Eléctricamente se lo compara como un interruptor, que tiene las posiciones (NA) normalmente abierto y (NC) normalmente cerrado. Este sensor se utiliza para monitorear el estado o posición de puertas, ventanas o cualquier interfaz física que requiera apertura y/o cierre.

Los sensores magnéticos constan de dos partes imantadas, donde no es necesario que las partes se toquen para obtener una señal eléctrica como salida. Mediante el campo magnético al cambiar de posición forman un circuito abierto o cerrado respectivamente.

Existen algunos tipos de sensores dependiendo de su uso, por ejemplo: de bornera, precableado, blindados para portones, de tipo industrial, entre otros. Al igual que la mayoría de los sensores se alimentan de 5V DC La Figura 1. 9 ilustra un sensor magnético precableado.

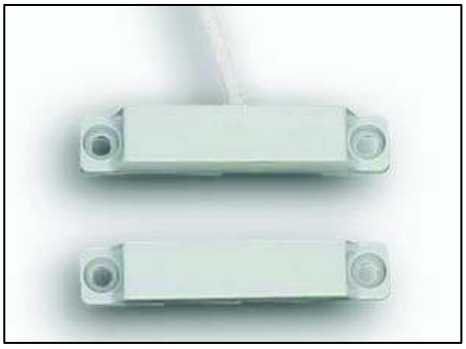

Figura 1. 9 Sensor magnético pre cableado, dos componentes imantados[14]

1.3.7 Cámaras de Vigilancia IP

En la actualidad existen sistemas de vigilancia de baja y alta complejidad, por ejemplo, siendo los sistemas contra robo y atraco los más sencillos mientras que los sistemas con funciones complementarias como mecanismos contra incendios, anti-hurtos, especiales, etc, se consideran más completos, la implementación de los mismos considera el uso de cámaras web o cámaras IP. Estos sistemas ofrecen al usuario una visualización en tiempo real del área donde se instala la cámara. Una vez conectada la cámara IP a una red de datos es fácilmente configurable mediante un acceso Web por parte del usuario. Dentro de un sistema de video vigilancia se considera un servidor de *streaming* el cual se encarga de recibir un flujo de datos (video) por parte de las cámaras y distribuirlo entre los clientes conectados a este. Existen diversos tipos de cámaras IP que se detallan a continuación: [15]

Cámaras PTZ: Cámaras mediante las cuales se puede cubrir un sitio despejado en exteriores debido a su flexibilidad de movimiento horizontal y vertical. La mayoría de las cámaras de esta índole incluye funcionalidades como estabilización de imagen, configuración de posiciones, el auto seguimiento para detección automática de movimiento, entre otras. Existen dos tipos de cámaras PTZ, las mecánicas se utilizan principalmente en interiores y en aplicaciones donde se emplea un operador, el zoom óptico de estas cámaras varía entre 10x y 26x. Asimismo las PTZ no mecánicas ofrece un campo de visión más extenso por lo que se usan en exteriores, utiliza un sensor de imagen permitiendo al operador alejar o acercar de manera instantánea el objetivo o

parte de la escena. Igualmente, las cámaras domo PTZ en consecuencia del diseño y el montaje que requieren, este tipo de cámaras son ideales para trabajar en instalaciones discretas [7]. En la Figura 1.10 se observa una cámara PTZ tipo domo.

Figura 1.10 Cámara IP tipo domo PTZ, marca HIKVISION

- ***Cámaras fijas*** Es una cámara que una vez montada provee de un campo de vista fijo, la dirección a la que apunta es claramente visible, utilizada en aplicaciones donde sea necesario que la cámara se encuentre visible hacia los usuarios, por ejemplo, en entidades financieras, centros comerciales, supermercados, etc. donde de alguna manera se pueda intimidar al sospechoso. En la Figura 1.11 se muestra una cámara IP fija.

Figura 1.11 Cámara IP fija

Además de cámaras, sensores y alarmas, existen otros dispositivos mediante los cuales se puede proveer de seguridad a diversos lugares, mediante el control de acceso en puertas. Por ejemplo, el uso de una tarjeta y un lector, el uso de un lector de huella dactilar, un lector de retina permite realizar una autenticación y poder ingresar a una sala. Sin embargo, otro método abarca la utilización de un lector de códigos QR que permite realizar una misma autenticación, pero con la ventaja de que se puede codificar información en un espacio bastante reducido. Para dicho método se necesitan dos partes básicas: la primera parte abarca la generación de códigos QR y por otro lado el módulo lector del símbolo. A continuación, se detallan los fundamentos de los códigos QR.

1.3.8 Códigos QR [16]

Un código de Respuesta Rápida (*Quick Response Code*) es una forma de representar y almacenar información dentro de una matriz cuadrada de puntos, que al ser bidimensional permite codificar y almacenar más información que su predecesor el código de barras tradicional. Se caracteriza debido a que sus elementos son cuadrados de color blanco y negro representando una unidad de información, además, dispone de 3 cuadrados de mayor tamaño en las esquinas para facilitar el proceso de lectura, en la Figura 1.12 se muestra los parámetros descritos.

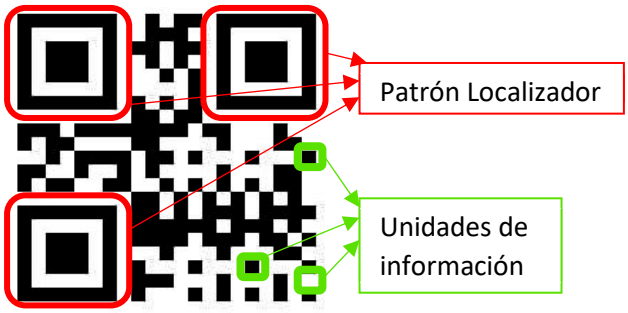

Figura 1.12 Representación Unidades de Información

Características y ventajas de los códigos QR

La ventaja principal de un código QR es que puede almacenar y codificar muchos tipos de datos que van desde números o caracteres alfanuméricos hasta los caracteres japoneses (Kanji), siendo así el mayor número de almacenamiento de 7.098 caracteres. Por otro lado, el espacio de impresión se reduce comparándolo con un código de barras en una relación 1 /10. A continuación, se detallan algunas características generales de los códigos QR:

- Alta capacidad de codificación de datos: hasta 2.953 bytes.
- Decodificación sencilla y de altas velocidades: aplicaciones de software o lectores.
- Adaptabilidad del código a los datos.
- Tiene una capacidad de corrección de errores: hasta un 30% de restauración de datos. Basado en el algoritmo Reed-Solomon
- Diferenciación de niveles claros y oscuros
- Independencia de orientación facilitando la lectura del código.

- Está constituido por módulos, dependiendo de la versión van aumentando, la primera versión consta de 21 módulos y hasta el momento la versión 40 con 177 módulos.

Estructura de un código QR [17]

Un código QR está conformado por diversas zonas, por ejemplo, considerando la zona donde se ubica la versión del modelo, si este es mayor a 7 se añade un patrón de alineamiento, algo que no sucede para códigos de menor versión. Además, mientras mayor sea la información que se codifique el tamaño del código aumenta. A continuación, se detalla los elementos que componen un código QR.

- **Información del formato.** - Contiene información acerca de la tolerancia al error, es una sección conformada de 15 bits (módulos) divididos en dos partes, 5 de datos y 10 para emplearlos en corrección de errores. Los 2 primeros bits de datos contienen información del nivel de corrección de errores que se está usando, mientras que los otros 3 módulos indican la máscara de datos usada. La información de formato se encuentra alrededor de los patrones de localización, esencialmente utilizada para la correcta decodificación del símbolo.

- **Información de versión.** - Esta sección específica la versión del código QR, en la actualidad se tiene 40 versiones diferentes. Está conformada de 18 módulos, 6 de los cuales son datos y los 12 son para corrección de errores. Los primeros 6 bits indican que versión es el código (por ejemplo 000111 es la versión 7), se encuentra ubicado por encima del patrón localizador inferior izquierdo y a lado izquierdo del patrón superior derecho. Son matrices de 6x3 módulos. La información de versión normalmente se aplica a versiones superiores a la 7.

En la Figura 1.13 se ilustra tanto la información de formato como la información de versión de un código QR.

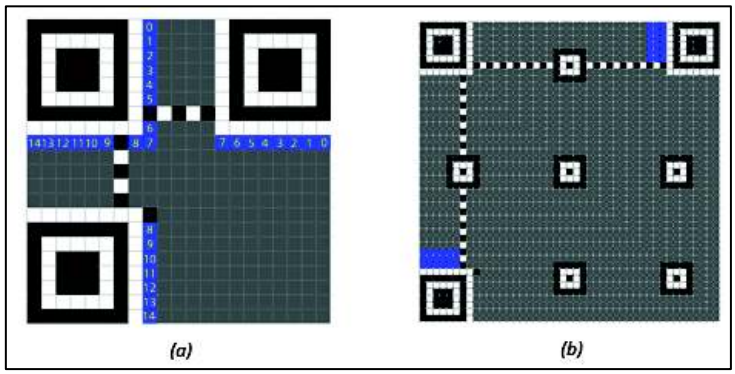

Figura 1.13 (a)Información de formato (b) Información de versión [18]

- **Patrón Localizador. -** Consiste en 3 estructuras localizadas en las esquinas del código QR excepto en la esquina inferior derecha, son fundamentales para el reconocimiento y orientación, de esta forma se indica la dirección en la cual se imprime el código. Cada patrón está basado en una matriz cuadrada negra rellena de 3x3, rodeado de una matriz cuadrada blanca de 5x5 y nuevamente de una matriz de 7x7. Además, cuenta con un patrón vertical y otro horizontal, ambos temporizadores que permiten definir la versión del símbolo y las coordenadas de los módulos.

- **Patrón de sincronización:** Utilizando este patrón que es un segmento de colores alternos, un lector de códigos puede determinar qué tan grande es la matriz de datos almacenada. Además, se lo considera como una referencia para poder determinar el ancho y las coordenadas de los módulos.

- **Patrón de alineamiento:** Patrón que ayuda con la orientación del código QR cuando este es grande, es decir estos patrones dependen de la versión del QR generado. Conformado por 3 cuadrados concéntricos de tamaño 1x1, 3x3 y 5x5, alternado los colores entre negro, blanco, negro respectivamente. El código QR de versión 1 no dispone de patrón de alineamiento.

- **Zona de amortiguamiento:** Zona que rodea al código para aislarlo de interferencias a la hora del reconocimiento, esta zona permite al lector QR distinguir entre el código y sus alrededores. Tiene un tamaño de 4 módulos.

- **Zona codificada:** Incluye la información a ser codificada, así como, información que complementa el código destinada a la versión, información de formato e información de la sección de corrección de errores. Los datos que codificar son convertidos en un flujo de bits y se almacena en un *codeword* conformado de 8 bits. La información o datos provistos en diferentes formatos se codifican en conjuntos para la creación del código, cada uno de estos subconjuntos debe estar delimitado por una cabecera de 4 bits y un contador de caracteres (longitud variable dependiendo de la versión). A continuación, se detalla algunos formatos de información:

    - **Numérico. -** Utiliza solamente los dígitos comprendidos entre 0-9, en este modo la información entrante se divide en grupos de 3 dígitos para su posterior conversión en su equivalente binario de 10 bits.

- **Alfanumérico**. - Consta de 45 caracteres entre números y letras, además de 9 caracteres simbólicos. Esta información se agrupa en pares para ser codificados asignándolos un valor entre 0 y 44**.** El primer carácter se multiplica por 45 mientras que al segundo se le suma este resultado, posteriormente se transforma a su equivalente binario de 11 bits.
- **Kanji**. - Caracteres del alfabeto japonés, tiene una densidad de 13 bits por cada 2 caracteres.
- **Binario.** - El valor de cada carácter es de 8 bits.

En la Figura 1.14 se ilustra un código QR identificando cada una de sus partes.

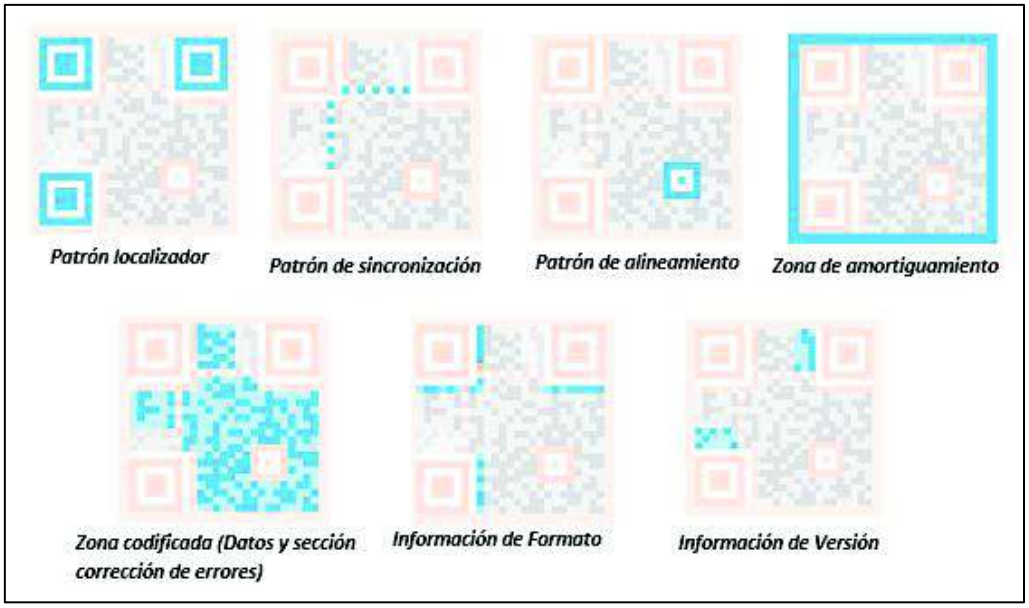

Figura 1.14 Estructura de código QR [19]

Zona Codificada

En esta sección se detalla el proceso de ubicación de la información codificada en base a la estructura del código QR, seguido de la corrección de errores y la máscara de datos.

- **Ubicación de *Codewords***

La cantidad final de *codewords* se define por el tamaño de la región de codificación, determinado por el tamaño del símbolo menos el tamaño de los patrones de función. Dependiendo de la versión y el nivel de corrección de errores elegido, los *codewords* de datos se dividen en grupos para aplicar el algoritmo de Reed-Solomon y de esta manera construir la secuencia en el símbolo. La secuencia final se conforma en orden, primero

todos los *codewords* de datos, seguido de todos los *codewords* de errores obtenidos, y al final si es necesario se completa el símbolo con módulos de relleno.

Se tienen dos formas de ubicar los *codewords* en un símbolo, regular e irregular. Su utilización depende de la posición que encuentren en el símbolo y de los obstáculos encontrados. A continuación, se describen algunas particularidades:

- La forma regular consta de un bloque de 2x4 módulos verticales u horizontales.
- La forma irregular es consecuencia de haber encontrado obstáculos en la colocación regular.
- La secuencia de bits se coloca siempre de derecha a izquierda empezando por el bit más significativo.
- La colocación de los *codewords* empieza desde la esquina inferior derecha hacia arriba hasta encontrar un límite u obstáculo, para después bajar por la columna adyacente, tal como se indica en la Figura 1.15

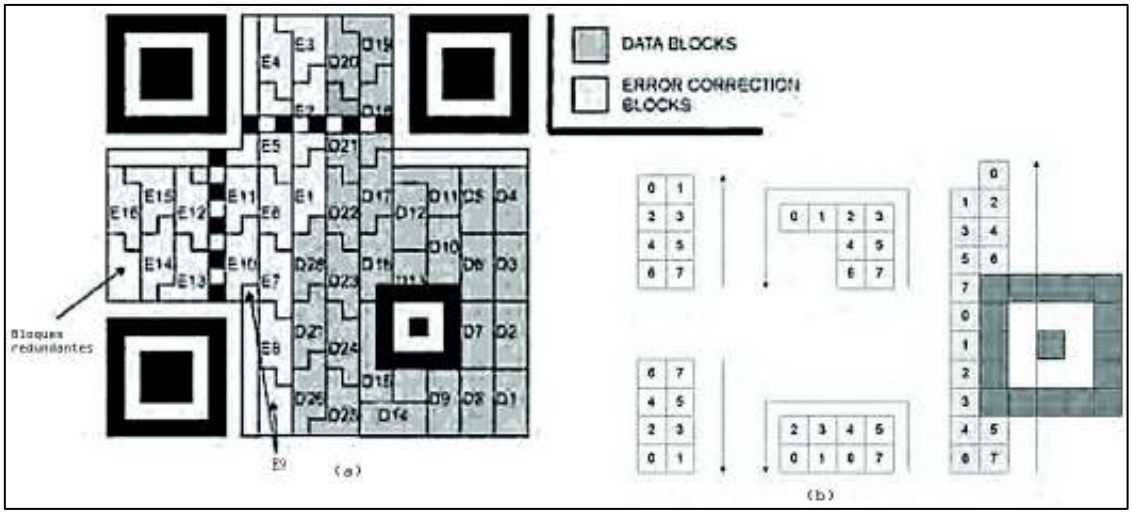

Figura 1.15 (a) Ubicación de *codewords* de corrección de errores (b) Ubicación de *codewords* de datos [20]

Una vez ubicados los datos para una mejor decodificación se hace uso de una máscara de datos que se aplica a la zona codificada.

- **Corrección de Errores**

La corrección de errores se utiliza en la mayoría de los enlaces de datos para que en el destino se pueda recuperar la información pese a errores producidos por el medio de

transmisión. Un código QR tiene capacidad de corrección de errores, lo que implica que puede recuperar datos cuando el código ha sufrido daños.[17]

Empleando codificación de errores basados en algoritmos de Reed-Salomon se crea un conjunto de *codewords* ECC *(Error Correction Codewords),* los mismos que se añaden a los datos dando redundancia al código QR. Dichos códigos permiten una corrección a nivel de byte, existen 4 niveles de corrección en un símbolo QR llegando a corregir hasta un máximo del 30% de la información o *codewords de* datos.

La Tabla 1.2 muestra los niveles de corrección de errores de un código QR.

Tabla 1.2 Niveles de corrección de errores

Nivel	**Porcentaje**
L (Low)	Corrige hasta 7 % de los *codewords* de datos
M (Medium)	Corrige hasta el 15 % de los *codewords* de datos
Q (Quality)	Corrige hasta el 25% de los *codewords* de datos
H (High)	Corrige hasta el 30% de los *codewords* de datos

- **Máscara de Datos**

Se utiliza el enmascarado para optimizar la decodificación de los símbolos, la idea es conseguir que el número de módulos blancos y negros esté equilibrado y se deben excluir secuencias encontradas en los patrones de función, como los de localización.

Para ello se aplica una máscara de datos a los *codewords* de datos y de corrección de errores solo en la zona de codificación, a través de la operación lógica XOR. Se tienen 8 patrones de mascara aplicables, para elegir cual es el más apropiado se aplican todos y se comparan los resultados, seleccionando el más adecuado de acuerdo a un cálculo de defectos según unos pesos establecidos [16], en la Figura 1. 16 se muestra las máscaras posibles.

En base a cada uno de los parámetros expuestos anteriormente se puede generar un código QR según una serie de procedimientos, los cuales se describen a continuación.

Código máscara	Condición
000	(i + j) mod 2 = 0
001	i mod 2 = 0
010	j mod 3 = 0
011	(i + j) mod 3 = 0
100	(⌊i div 2⌋ + ⌊j div 3⌋) mod 2 = 0
101	(i j) mod 2 + (i j) mod 3 = 0
110	((i j) mod 2 + (i j) mod 3) mod 2 = 0
111	((i+j) mod 2 + (i j) mod 3) mod 2 = 0

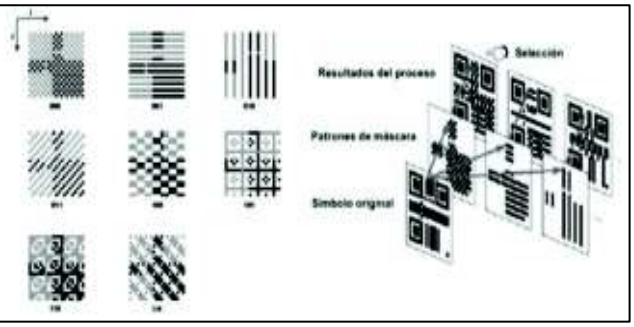

Figura 1. 16 Patrones y proceso de enmascaramiento. [16]

Generación de códigos QR

A la hora de generar un código QR su tamaño es el más importante, base a la versión se establece el tamaño del módulo. Además, es importante el área física ocupada por el código para facilitar su decodificación. Por otra parte, para la lectura de un código QR es necesario un lector de códigos o una cámara para obtener la imagen, además de un software que permita la decodificación de mismo.

- **Proceso de codificación**

La codificación de la información para generar un código QR se basa en 7 pasos:

1. **Analizar los datos.** Establecer el tipo de codificación adecuado dependiendo del tipo de caracteres a ser almacenados.

2. **Codificar los datos**. Convertir los datos entrantes en un flujo de bits según el modo establecido, formar los *codewords* de 8 bits mediante la división del flujo datos. Finalmente se añade un delimitador de inicio o indicador y uno al final de cada subconjunto de datos.

3. **Codificar la corrección de errores.** Establecer el nivel de corrección de errores, dependiendo del modo seleccionado dividir los *codewords* de información en bloques y aplicar el algoritmo de corrección. Una vez generados los *codewords* de corrección de errores se los añade al final de la secuencia de corrección.

4. **Estructurar el mensaje.** Tanto los *codewords* de datos como los de corrección de errores se entrelazan, y se añade relleno para completar la estructura.

5. **Colocar los módulos en el símbolo.** Ubicar los patrones de función de acuerdo con la versión seleccionada para complementar el código QR, además, ubicar los *codewords* de datos en la zona codificada.

6. **Enmascarar los datos.** Equilibrar tanto módulos blancos como negros de tal forma que se minimice apariciones de módulos no deseados, se lo realiza mediante la aplicación de máscaras de datos en la zona codificada.

7. **Generar información de versión y de formato**. Ubicarlas en el código para complementarlo.

El siguiente paso a la generación de un código QR es la lectura y obtención de la información. Para readquirir la información contenida en un código QR es necesario un escáner de códigos y llevar a cabo un proceso de decodificación como el descrito a continuación.

Lectura de códigos QR

Para poder descifrar los códigos QR es necesario dos elementos, el primero una cámara para captar la imagen del símbolo y el segundo un componente de software que procese y decodifique el símbolo para obtener la información contenida.

Actualmente, en un entorno doméstico no es difícil encontrar dispositivos que cumplan con el primer requisito, hay cámaras incluidas en la mayoría de los dispositivos móviles, ordenadores y demás aparatos tecnológicos. También existen módulos diseñados específicamente para realizar la tarea de decodificar un código QR. Estos generalmente están destinados a entornos industriales, y son periféricos que pueden conectarse directamente a equipos finales como una PC, para el procesamiento de los datos.

- ### Proceso de decodificación

Proceso inverso a la codificación, es decir, en base al símbolo se obtiene datos en forma de caracteres, este proceso se basa en 8 pasos:

1. Obtener una imagen del código QR, crear una matriz de "1" y "0" en la cual se reconozca los módulos blancos y negros.

2. Leer la información de formato, además obtener el nivel de corrección y la máscara de datos empleada

3. Leer la información de versión si el código dispone y determinar el tamaño del símbolo en módulos.

4. Se aplica una máscara de datos mediante la operación XOR a la matriz de datos codificados

5. Obtener *codewords* tanto de datos como de corrección de errores generados en la codificación
6. En base a *codewords* de corrección de errores y el nivel, detectar los posibles errores producidos en la información codificada
7. De acuerdo con los indicadores de modo se dividen los *codewords* de datos corregidos
8. Decodificar caracteres de acuerdo con la codificación empleada.

Los códigos QR pueden ser utilizados en diversos campos, como por ejemplo en el área industrial para el control de inventarios, en el área de seguridad para implementar un control de acceso con identificación o para compartir información y distribuir de forma fácil enlaces con contenido exclusivo, aplicación muy útil para el campo del marketing.

Dentro de un sistema de video vigilancia es necesario tener un canal de comunicación entre el administrador y la información recopilada por el sistema, para esto se podría utilizar una pequeña solución con software, que permita acceder al contenido a través de una interfaz agradable al usuario como páginas web o aplicaciones móviles y de esta manera controlar las funcionalidades provistas por dichos sistemas. Además, el estándar descrito anteriormente WoL también trabaja con aplicaciones o paquetes en diferentes distribuciones de Linux. Dicho esto, para la creación de páginas web o aplicaciones móviles se utilizan lenguajes de programación que permitan el desarrollo para que el producto final cumpla con los requerimientos.

1.3.10 Lenguajes de programación

Un lenguaje de programación es una forma básica y apropiada para describir la organización de información y la sucesión de operaciones necesarias para efectuar una labor determinada, de forma que sea un proceso compresible tanto para el usuario como para el ordenador. Cada lenguaje posee su propia sintaxis. Existen un gran número de lenguajes de programación que se utilizan según el área de aplicación, en la cual se quiera dar una solución a un problema.

Entre las formas de desarrollo de software, existen dos enfoques necesarios para la creación de aplicaciones móviles híbridas que son similares al desarrollo de sitios web, el primero es el diseño web mejor conocido como *front-end*, que se enfoca en la parte visual de un sitio web. Se utilizan básicamente los lenguajes de programación como JavaScript,

para darle funcionalidad a la página, y metalenguajes como HTML y CSS, el primero utilizado como la estructura base de la información a presentarse y el segundo usado para darle una mejor apariencia y estilos con el fin de hacer más agradable la página al usuario.

El segundo campo es el desarrollo web y también se conoce como *back-end,* que se enfoca en procesar la entrada de datos que vienen desde el *front-end.* En otras palabras, son el conjunto de pasos que el administrador del proyecto web utiliza para resolver las peticiones de los usuarios finales. Dado que un *back-end* [21] consiste básicamente de un servidor, una aplicación y una base de datos, se usan como herramientas los lenguajes Python, PHP, NodeJs, MySQL para implementar todos estos elementos. A continuación, se describen las características de mayor importancia de los lenguajes mencionados anteriormente.

- ***HTML (Hyper Text Markup Language)*** **[22]** Lenguaje sencillo desarrollado por el *World Wide Web Consortium,* de programación estructurada y uso de *hyperlinks* (enlaces). Es utilizado para desarrollar y dar forma a las páginas Web. Al desarrollar un documento HTML son necesarias etiquetas con diversos atributos mediante las cuales se inicia y finaliza una instrucción. Los diferentes tipos de etiquetas disponibles en el lenguaje permiten la integración de varios elementos a la página web, elementos como cabeceras, tablas, párrafos, entre otros. HTML no necesita ningún software especial para la visualización de la página web.

- **JavaScript** ***[23], [24]*** Es un lenguaje de programación sencillo empleado para el diseño de páginas web dinámicas junto con HTML. JavaScript proporciona funcionalidades como: efectos sobre botones, abrir enlaces, crear mensajes interactivos, animaciones con el objetivo de que la página web sea más agradable para el usuario. Inicialmente este lenguaje fue conocido como *LiveScript* pero Netscape cambió su nombre tomando relación a Java. La inclusión entre el código JavaScript y HTML se hace mediante etiquetas `<script>` de inicio y fin, este código se lo adjunta en cualquier parte del documento HTML. Una característica importante de este lenguaje es que no requiere compilación

- ***Cascading Stylesheets (CSS)[25]*** CSS es una abreviación de *Cascading StyleSheets* (Hoja de estilo en cascada), trabaja simultáneamente con HTML para poder simplificar el proceso de hacer presentables o atractivas las páginas web. Se

utiliza CSS para poder manejar ciertos parámetros de diseño como por ejemplo el color del texto, tamaño de filas y columnas, espacio entre imágenes, color e imágenes de *background*, etc. CSS trabaja en base a identificadores dentro de etiquetas, cada identificador consta de parámetros a ser cambiados acorde a las necesidades. CSS permite que el contenido sea optimizado para cada dispositivo donde se utiliza, por ejemplo, una página web diseñada para un dispositivo móvil se la puede usar con el mismo código en un navegador.

- ***SASS (Syntactically Awesome StyleSheets)*** es un metalenguaje que puede ser traducido en otro lenguaje, de esta manera SASS es traducido en CSS para su posterior representación. Provee de dos formatos diferentes para su sintaxis, siendo posible dos extensiones del fichero. Entonces "fichero.sass" se caracteriza al igual que JavaScript en no usar llaves ni punto y coma final, mientras que "fichero.scss" permite el uso clásico de llaves e incorporar código CSS como se indica en la Figura 1. 17

fichero.sass	fichero.scss

body
background: #000
font-size: 62.5%
	body {
	background: #000;
	font-size: 62.5%;
	}

Figura 1. 17 Diferencia de sintaxis entre las extensiones de CSS

- ***Python[26]*** Es un lenguaje de programación basado en estructuras de datos y con un enfoque a la programación orientada a objetos, su sintaxis hace de este un lenguaje ideal para poder realizar scripts para el desarrollo de aplicaciones, posee una extensa biblioteca que está a disposición en el sitio web de Python, adicionalmente posee enlaces a diversos programas de ejemplo y herramientas con su respectiva documentación.

  Una ventaja de este lenguaje es que permite probar porciones de código para posteriormente agregarlos al programa final, también a diferencia de otros lenguajes usa palabras en lugar de símbolos, esto resulta útil en el caso de familiarizarse con el lenguaje, además existen pequeños programas que proporcionan mejoras al intérprete de Python como el autocompletado de comandos del código o coloreado en la sintaxis de este.

1.3.11 Software disponible para el desarrollo de aplicaciones móviles [27]

Desarrollar aplicaciones para dispositivos móviles se considera como un conjunto de procesos que involucran la escritura de software destinado a dispositivos inalámbricos como teléfonos inteligentes o tablets. La creación de aplicaciones móviles se asemeja en cierta forma al desarrollo de aplicaciones web, sin embargo, las apps se diseñan para aprovechar características específicas de los dispositivos móviles.

Una aplicación móvil es un programa informático elaborado como una herramienta para el usuario, destinada al desempeño de tareas del tipo profesional, de ocio, educativas, entre otras. Existen tres tipos de aplicaciones móviles [28] las cuales se describen a continuación:

- **Aplicaciones Nativas** Son desarrolladas para un determinado sistema operativo, por ello necesitan el uso de un lenguaje específico para cada sistema y se instalan a través de las diferentes tiendas de aplicaciones.
- **Aplicaciones Web móviles dedicadas** Desarrolladas en lenguajes de programación web conocidos como HTML, JavaScript y CSS, tienen la ventaja de que el código es elaborado una vez y se puede utilizar en todas las plataformas, en otras palabras, son independientes del sistema operativo.

Aplicaciones Híbridas Son aplicaciones resultantes de la combinación de una aplicación web y una aplicación nativa. Se desarrollan en lenguajes de programación propios de una aplicación web, pero al mismo tiempo posibilitan el acceso a las características propias del smartphone.

Para que una aplicación móvil pueda ser agregada a un dispositivo móvil es necesario disponer de un Android Application Package (APK)[29] el cual es similar a los archivos ejecutables disponibles para Windows (.exe o .msi) mediante los cuales se instala software en el computador. En un archivo con extensión apk básicamente se tiene una compresión parecida al Zip, en su interior contiene archivos para instalar una aplicación en Android.

En la siguiente parte se detalla criterios de diseño para la creación de interfaces de una aplicación.

Criterios de diseño [30]

Las interfaces de la aplicación móvil son diseñadas en base a ciertos criterios para que

trabaje eficientemente, a continuación, se mencionan algunos de estos parámetros:

- **Usabilidad:** Comprende la facilidad de uso de la aplicación, juntamente con la navegación entre páginas, el diseño y la interacción interfaz-usuario.
- **Contenido:** Convergencia entre el contenido de la aplicación con el contexto propuesto.
- **Diseño Visual:** Comprende el estilo de la aplicación considerando la presentación de las páginas, ícono principal y de interiores, la pantalla inicial, navegación, color, texto, entre otros, de esta forma el usuario la verá agradable.
- **Tipografía:** En el diseño se toma en cuenta que el texto de la aplicación sea legible y se lea con total claridad, evitando que el lenguaje utilizado de espacio a una mala interpretación.

Dependiendo de la aplicación se considera parámetros adicionales en el diseño, como los que se presentan a continuación:

- Proporcionar mensajes de aviso en la interfaz de encendido y apagado remoto
- Utilizar un lenguaje acorde al contexto propuesto y de forma familiar para el usuario
- Facilidad de uso de cada una de las interfaces que componen la aplicación
- Proveer un entorno gráfico agradable para el usuario.

Para el desarrollo de aplicaciones móviles sean web, nativas o híbridas existen herramientas que facilitan enormemente este trabajo, estas herramientas son conocidas como *frameworks.*

Frameworks

Un *framework* es una estructura de software que brinda varias herramientas facilitando el desarrollo de un proyecto, evitan escribir código repetitivo, y además faculta al desarrollador de software para realizar diseños más complejos, dando así la opción para diseñar aplicaciones mucho más rápido [31]. Existen varios *frameworks* útiles y disponibles para la creación de aplicaciones móviles híbridas, como por ejemplo Xamarin, PhoneGap, Framework 7, Appcelerator Titanium, entre otros. Ionic, un potente *framework* de código abierto que utiliza herramientas como *Angular* y *TypeScript* permite desarrollar aplicaciones móviles interactivas en poco tiempo, y gracias a ello será utilizado en este proyecto por lo que a continuación se detalla sus principales características debido a que

posee mejores prestaciones a los mencionados en cuanto a utilización de lenguajes de programación y construcción de apk.

Ionic [32]

Ionic es un Framework de código abierto, basado en Apache Cordova y AngularJs para el desarrollo de aplicaciones móviles híbridas. Estas aplicaciones son multiplataformas, por lo tanto, con el mismo código se puede generar apks que funcionan en diferentes plataformas móviles como Android, iOS y en la web a través de un navegador. Utilizan lenguajes de programación como HTML, JavaScript, CSS para su desarrollo. Entre sus ventajas se tienen las siguientes:

- Su curva de aprendizaje es mucho menor respecto a otros *frameworks* para aplicaciones hibridas.
- Provee de una gran cantidad de componentes de la interfaz de usuario usando directivas y servicios.
- Permite desarrollar aplicaciones de gran escala.

  Permite emular la aplicación en un navegador web con una recarga en tiempo real de la aplicación durante el desarrollo, o un emulador de Android cualquiera, así como en un dispositivo móvil real.

Ionic ha ido evolucionando y mejorando, basándose principalmente en las actualizaciones que ofrece AngularJS en cada una de sus versiones. La Tabla 1.3 presenta características de cada versión acorde a sus componentes como los es Angular

Tabla 1. 3 Características de las versiones de Ionic

Ionic V1	**Ionic V2**	**Ionic V3**
Usa AngularJS como base. Ofrece componentes de la interfaz de usuario con un diseño mucho más agradable. Aprovecha Apache Cordova para comunicarse con la	Usa Angular2 como base. Mayor rendimiento, mayor optimización. Definición más simple de componentes y servicios mediante clases. Inyección de	Usa Angular 4 como base. No se notan cambios de sintaxis, pero la aplicación resultante ocupará menos espacio y será más rápida. Soporta TypeScript 2.2.

plataforma nativa solo cuando lo necesita. Permite construir aplicaciones móviles híbridas con un aspecto y experiencia similar a que si fueran nativas.	dependencias simplificada con TypeScript Detección de errores en fase de “compilación” con TypeScript.	Lazy Loading: permite retrasar la petición de los recursos necesarios de una vista (html, js, imágenes, etc.) hasta el momento en que se navega a dicha vista.

AngularJS [33]

AngularJS es un *framework* de código abierto y gratuito creado por Google basado en JavaScript que permite crear aplicaciones web dinámicas, eficazmente. Una de sus principales características es el *databinding,* mediante el cual une en tiempo real los datos de dos componentes, cuando uno de ellos cambia, se actualiza también el otro elemento. Esto evita la recarga de la página web innecesariamente, en el caso de hacer operaciones o cambios por parte del usuario. También utiliza la inyección de dependencias que permite una vez creada o importada una librería, utilizarla en cualquier parte del código sin tener conflictos con las instancias y reduciendo la complejidad. Por último, cuenta con directivas que básicamente permiten otorgarle funciones adicionales al código HTML normal.

Apache Cordova [34]

Apache Cordova es un *framework* para desarrollo móvil de código abierto que permite acceder a las funciones nativas del dispositivo como los sensores: acelerómetro, cámara, GPS. Además, brinda acceso a los datos del teléfono y de la red, todo esto mediante CSS3, HTML5 y JavaScript.

Dado que en algunos dispositivos Android antiguos existe incompatibilidad en el soporte de los navegadores con HTML5, Apache Cordova incrusta el código HTML5 dentro de un *WebView* de la aplicación nativa que es la que se distribuye en las tiendas de aplicaciones.

Una aclaración importante por mencionarse es que Cordova no provee ningún *widget* de interfaz de usuario o *framework* de Modelo-Vista. Cordova solo provee el ejecutor en tiempo real en el cual operan dichos complementos.

Firebase

Firebase es un sistema que se aprovecha como un complemento para el desarrollo de aplicaciones web y móviles. *Firebase* se puede considerar como *Backend as a Service* BaaS, un servicio que ofrece almacenamiento en la nube, gestión de usuarios, la posibilidad de envío de notificaciones, integración con redes sociales, entre otros, es decir servicios en la nube, necesarios para desarrollar el *backend* para una aplicación. Además, su gran ventaja es la comunicación en tiempo real, es decir no existen consultas de actualización al servidor, simplemente los datos se van actualizando en la página.

Firebase cuenta con la capacidad de acoplarse a diferentes lenguajes de programación entre los cuales están Python, C++ o PHP. Los productos ofrecidos por *Firebase* aumentan las posibilidades de desarrollar apps complejas y de mayores funcionalidades, siendo los productos más utilizados la autenticación y las bases de datos en tiempo real.

De la misma manera se tienen muchos más productos [35] provistos por *Firebase* de los cuales se detallan los más importantes a continuación:

- **Realtime Database (Base de datos en Tiempo Real)** Mediante una base de datos no SQL que se encuentra alojada en la nube, *Firebase* almacena y sincroniza aquellos datos existentes entre los usuarios y los dispositivos en tiempo real, permaneciendo estos disponibles aun cuando los usuarios pierdan conexión de Internet

  La base de datos que *Firebase* ofrece se almacena como un árbol de objetos JSON. Esta estructura tipo árbol es muy útil y flexible para todos los tipos de datos, de esta forma se obtienen o añaden datos de una manera más organizada para su posterior uso en la aplicación.

- **Cloud Messaging (Mensajería en la nube)** Producto destinado para el envío notificaciones a los usuarios sea cual sea la plataforma de trabajo (Android, iOS, Web). Las notificaciones pueden ser enviadas a un grupo de dispositivos, a un individuo o a un grupo específico de usuarios.

- **Authentication (Autenticación)** Provee diversos métodos de autenticar usuarios, por ejemplo, mediante un correo electrónico y contraseña, mediante cuentas existentes o utilizando redes sociales, es decir una autenticación a través de Facebook, Twitter, Google, etc.

- **Cloud Storage (Almacenamiento en la nube)** Destinado al almacenamiento de imágenes, video, audio en la nube o cualquier otro tipo de contenido.
- **Test Lab for Android** Producto mediante el cual se puede realizar pruebas de funcionamiento de la app en dispositivos virtuales alojados en Google, para posteriormente crear un apk e instalar en un dispositivo físico.
- **Crash Reporting (Informes de fallos)** Esta característica crea informes detallados acerca de los errores que se producen en la emulación de la aplicación y los prioriza de acuerdo con la gravedad, ayuda al diagnóstico de problemas.

Figura 1.18 Compatibilidad del servicio *Firebase* con iOS, Android y Web

Una característica importante de *Firebase* es que permite trabajar con diversos dispositivos con sistemas operativos como iOS y Android, además de ser utilizado para desarrollo web. En cada caso su uso es diferente, por ejemplo, para aplicaciones móviles en iOS o Android se necesita el ID de la app, mientras que, en una aplicación web se copia el código de inicialización, que contiene datos del proyecto de *Firebase*, al código HTML. En la Figura 1.18 se muestra la opción de selección del SO con el cual se desea trabajar una vez ingresado al proyecto de consola de *Firebase*

2. METODOLOGÍA

En esta sección se describirá el proceso de diseño del prototipo detallando cada uno de los subsistemas que lo conforman, se considera requerimientos de hardware y software para la selección de dispositivos. Finalmente se describirá la implementación del subsistema de vigilancia y el de monitoreo de PC's con los módulos y procesos respectivos, concluyendo con el desarrollo de la aplicación móvil.

2.1 REQUERIMIENTOS DEL PROTOTIPO

El sistema planteado se enfoca en brindar vigilancia al interior del Laboratorio de Interconectividad de la FIEE, considerando el control de algunos dispositivos. Para la obtención de información acerca de requerimientos de la sala se realizaron encuestas a 12 estudiantes incluidos el ayudante y jefe de laboratorio, de esta forma a partir de la información obtenida se establecen diversas tareas para solucionar algunos de los problemas encontrados.

2.1.1 Encuestas

Las encuestas realizadas abarcan preguntas sobre el Laboratorio de Interconectividad, además se incluyen interrogantes acerca de posibles formas de automatizar tareas como acceso a la sala, avisos de alarma o procesos diarios. Finalmente se pregunta sobre un medio de interacción del usuario con el prototipo. La encuesta se la puede visualizar en el ANEXO I

Población objetivo

Las encuestas mencionadas fueron dirigidas al jefe de Laboratorio de Interconectividad, ayudante y a 10 usuarios seleccionados aleatoriamente de los estudiantes que a diario usan la entidad, dando un total de 12 personas encuestadas. En el ANEXO II se puede apreciar los resultados de las encuestas realizadas.

Información obtenida

En base a las encuestas realizadas y tomando en cuenta una entrevista con el jefe del Laboratorio se obtuvo la siguiente información sobre la situación de la sala.

El Laboratorio de Interconectividad siendo una entidad de aprendizaje posee equipos que deben ser resguardados o vigilados de alguna manera.

El encendido y apagado de los computadores de la sala se realiza manualmente, siendo

una tarea tediosa para el asistente. Sin embargo, puede darse el caso que el asistente no preste atención al momento de cerrar el laboratorio y todos los PC's queden encendidos.

El ingreso a la sala se realiza solamente con la ayuda del asistente o del jefe del laboratorio, en casos que ninguno de los dos se encuentre disponible para la apertura y sea horario de clase no existe un mecanismo para que el profesor de la asignatura pueda ingresar.

Tomando en cuenta el punto anterior, de ser el caso que el profesor pueda ingresar no existe un registro de acceso a la sala, ni un sistema de alarma que pueda notificar el ingreso.

El uso de una aplicación móvil para el manejo remoto de un sistema es una opción viable para los usuarios de la sala (Administrador, asistente, profesores) debido a que el smartphone es un dispositivo de la vida diaria.

Requerimientos obtenidos

En base a la información recolectada se puede determinar los siguientes requerimientos para el desarrollo del prototipo propuesto para el laboratorio de Interconectividad:

- **Vigilancia interior de la sala:** Proveer un mecanismo de vigilancia a través de una visualización remota de video provista por dos cámaras IP instaladas en el interior de la sala.
- **Ingreso al laboratorio:** Crear un mecanismo que permita la apertura de puerta mediante una herramienta que valide información registrada del usuario en el sistema mediante la lectura de códigos QR.
- **Notificaciones:** Proveer un sistema de alarma que permita conocer, si la puerta se encuentra abierta o cerrada. En esta etapa se realiza el envío de notificaciones al teléfono de los administradores del Laboratorio
- **Registro de acceso:** Proveer una función que permita almacenar la información del usuario al acceder a la sala en caso de haber usado el sistema de código QR.
- **Encendido y apagado remoto de computadores:** Proveer un mecanismo de gestión para el encendido y apagado remoto de los computadores.
- **Control del sistema:** Implementar una aplicación móvil que permita manejar las funciones del prototipo de forma remota, además para visualizar los eventos ocurridos en el caso que exista un suceso de alarma.

2.2 SELECCIÓN DE COMPONENTES

En esta sección se describe los componentes necesarios para el prototipo, considerando que se divide en 3 subsistemas basados en hardware y herramientas de software.

2.2.1 Hardware

El subsistema de vigilancia como el de monitoreo de *PC's* se basan en dispositivos físicos (*hardware*) configurados de manera correcta para que cumplan los requerimientos del prototipo. A continuación, se realiza una comparación mediante tablas y se selecciona el de mejores prestaciones para el proyecto. La Tabla 2.1 muestra las características más importantes de los dispositivos para ser el elemento central del prototipo. Debido a que se necesita un mayor procesamiento y además 3 puertos USB disponibles para conexión de dispositivos se opta por Raspberry Pi 3 Model B.

Tabla 2.1 Características Modelos Raspberry Pi

	RPi Model A	**RPi Model B**	**RPi 2 Model B**	**RPi3 Model B**
CPU	700Mhz	700MHz	900MHz	1.2Ghz
RAM	256M	512M	1 Gb	1G
USB	1	2	4	4
Red		Ethernet 10/100	Ethernet 10/100	Eth. 10/100 Wifi, Bluetooth

Las cámaras IP utilizadas son del tipo PTZ para interiores las cuales disponen de conexión mediante *Wifi* para acceder a su configuración, además pueden ser controladas mediante una aplicación móvil. Las características de estas se muestran en el ANEXO III

Los sensores de contacto se presentan de diversos tipos, ***normal con bornera*** puede ser atornillado en la puerta, ***precableado*** igual que el anterior, pero dispone de cables para conexión, ***sensor de embutir*** son de forma cilíndrica se empotran en las puertas, ***del tipo industrial*** los cuales tienen doble tamaño que el normal y los ***sensores blindados para portón.*** Debido a que el sensor precableado está destinado a interiores, permite su instalación sin perforar ya que es auto adherible y es compatible con el 99,9% de los sistemas de alarmas, se prefiere esta opción.

El modem 3G se seleccionó en base a la tabla que se encuentra en el ANEXO III, el modem3G HUAWEI E173s-2 provee de una mayor velocidad de trasmisión de datos. Para el dispositivo de almacenamiento se considera un disco duro de aproximadamente

1Tbyte, una *memory flash* de 8GB marca Kingston, tomando en cuenta el tamaño de los dispositivos, la capacidad y el espacio que utilizarán los recursos a ser alojados se determinó lo siguiente. El disco duro no es una buena opción porque tiene un tamaño desproporcionado en relación con el tamaño del prototipo, incrementaría innecesariamente el espacio de instalación. Igualmente, su capacidad estaría subutilizada dado que los recursos no ocupan tanto espacio en memoria. Por el contrario, la *memory flash* tiene un tamaño conveniente para agregarla al prototipo sin perjudicar el espacio total. Además, la capacidad que ofrece es suficiente para almacenar los recursos multimedia, motivos por los cuales se seleccionó la memory flash como dispositivo de almacenamiento.

El sensor de presencia infrarrojo es un tipo de sensor de proximidad, en la Tabla 2.2 se detalla características de dos tipos de sensores.

Tabla 2.2 Características sensores IR

Modelo	**E18-D80NK**	**IR FC-51**
Alimentación	5v corriente de 25 a 100mA	3 a 5v DC
Dimensiones	17x 45 mm	14x31 mm
Área de detección	Hasta 80 cm	1 a 30 cm
Observación		Dispone de LEDS para indicar una detección

Por lo tanto, debido a que el proyecto no necesita una distancia grande para la detección y al armar el módulo lector código QR las dimensiones del dispositivo deben ser pequeñas se opta por el sensor de proximidad FC-51.

La Tabla 2.3 muestra características de cámaras USB, se realiza una comparación y se selecciona la cámara X5tech XW-360 debido a que tiene una mayor resolución y dispone de un conector USB 2.0 compatible con el puerto USB de la Raspberry Pi.

Tabla 2.3 Características cámaras USB

	Genius Eye 110	**Logitech Webcam C920**	**Genius Eye 312**	**X5tech XW-360**
Angulo de Giro	360°	360°	360°	360°
Resolución	352x288	352x288	640x480	640x480
Micrófono	no	si	si	no

Puerto USB	1.1	2.0	1.1	2.0
Compatibilidad Windows	si	si	Ethernet 10/100	si

2.2.2 Software

Para el diseño e implementación del prototipo se selecciona herramientas de software que ayudan en solventar los requerimientos anteriormente detallados además de acoplarse con la aplicación móvil para su desarrollo.

Para trabajar en Raspberry Pi existen SO que son derivaciones de Linux como los que se mencionan a continuación, ***pidora,***[36] optimizada para trabajar en ARM y no muy usada como SO de Raspberry, ***ArchLinux*** [37] otro SO que se caracteriza por simplicidad y elegancia, pero no de muy fácil uso, ***Kano OS*** [38] es un sistema especialmente pensado en niños, ***Raspbian OS*** [9] es la distribución más completa y optimizada basada en Debian, provista de herramientas de desarrollo como Python y Scratch. Para el presente proyecto se optó por utilizar Raspbian debido a que las funcionalidades del prototipo se realizarán desde scripts desarrollados en Python y la instalación de paquetes en el sistema es mucho más simple en comparación a los otros sistemas mencionados.

La Tabla 2.4 muestra características de *frameworks* disponibles para desarrollo de aplicaciones móviles

Tabla 2.4 Características de *frameworks* para desarrollo aplicaciones móviles

Framework	Características
Xamarin	Trabaja en lenguaje C#
PhoneGap	Trabaja con Apache Cordova Utiliza HTML, Javascript, CSS
Ionic	Es de código abierto Con el mismo código fuente se crea apk para Android, iOS Utiliza HTML, JavaScript, CSS del tipo SASS
Appcelerator Titanium	Tiene un ambiente de desarrollo mixto dado por *Xamarin y PhoneGap*

Para el desarrollo de la aplicación móvil se utiliza Ionic el cual fue seleccionado para este proyecto por la facilidad de utilización del mismo código para obtener una aplicación móvil en Android o iOS y poder comprobar el funcionamiento de esta mediante navegadores web, además del acoplamiento con *Firebase.*

2.3 DISEÑO DE LA BASE DE DATOS

En esta sección se detallará el proceso de obtención de datos de los usuarios registrados en la *Firebase Database*, esta información será almacenada en archivos de texto dentro de la Raspberry Pi para posteriormente ser utilizada en los procesos del prototipo.

Obtención de Información de la Firebase Database.

La base de datos estará conformada por un nodo raíz que se subdividirá en 3 nodos debido a que la estructura de la base es un árbol de objetos JSON, estos nodos se encontrarán distribuidos de la siguiente manera, el nodo *tokens* contendrá información acerca de los dispositivos donde se utilizará la aplicación, el nodo *UsuarioActual* dispondrá de información del usuario que generará el código QR y el nodo *Usuarios* se subdividirá en dos nodos adicionales *Administrador*/*Ayudante* y *Profesores* los cuales dispondrán información básica de los usuarios del sistema, todo esto con el objetivo de facilitar la descarga de datos desde el controlador. Cabe mencionar que los nodos que contendrá la base de datos se definieron en base a los procesos que se realizarán posteriormente, por ejemplo, la generación del código QR necesitará información de usuario para codificarlo, el envío del SMS se realizará en base al número telefónico del usuario, además, la disposición de estos campos facilitará la descarga de datos de forma ordenada desde Raspberry Pi. La información descargada se almacenará en archivos de texto para su posterior lectura y que de esa manera esté disponible para las tareas siguientes que se describen más adelante. Este proceso se ilustra en el diagrama de la Figura 2.1.

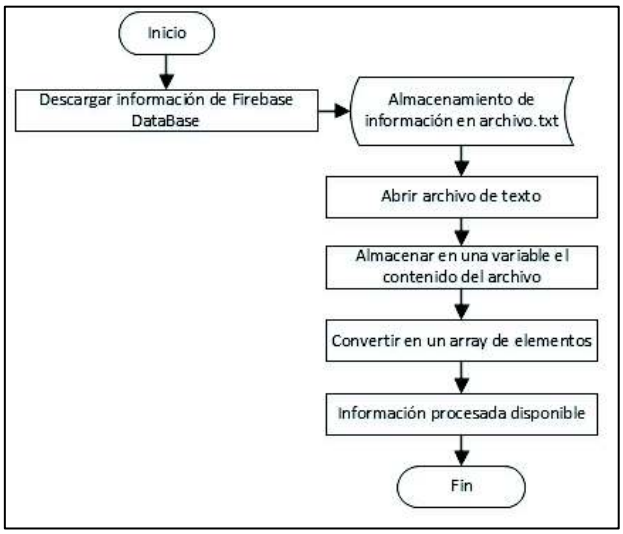

Figura 2.1 Lectura de archivo de texto

Obtención de información Administrador / Ayudante

El diagrama de la Figura 2.2 indica los pasos para desarrollar el script que se encargará en descargar y almacenar la información en un archivo de texto. Posteriormente se ejecutarán dos scripts para obtener la información del usuario (Nombre, Número) en otros archivos de texto respectivamente, este proceso se realiza para el usuario administrador y usuario ayudante. En la Figura 2.3 se ilustra el diagrama de flujo para generar el script de obtención de nombres de usuario.

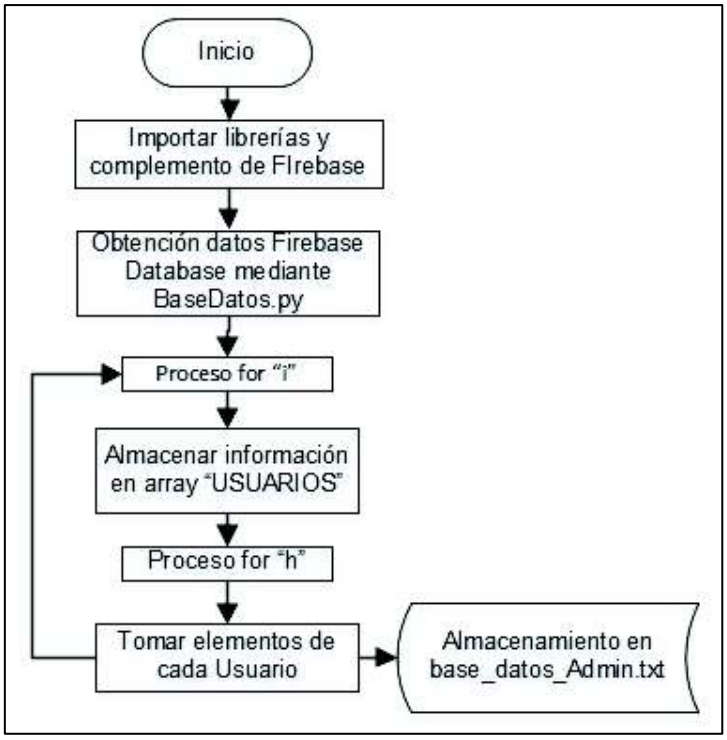

Figura 2.2 Diagrama de flujo para la obtención de información Administrador/Ayudante

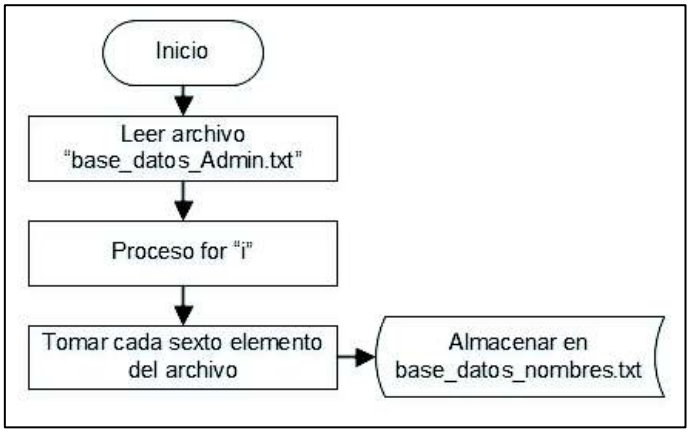

Figura 2.3 Diagrama de flujo para obtener nombres de usuario

Los usuarios registrados en la base de datos dentro del nodo *Profesores* dispondrán de otros campos como *asignatura* y *horarios* por lo que tendrá dos procesos *for* adicionales en el script de descarga, en la Figura 2.4 para obtener datos de profesores.

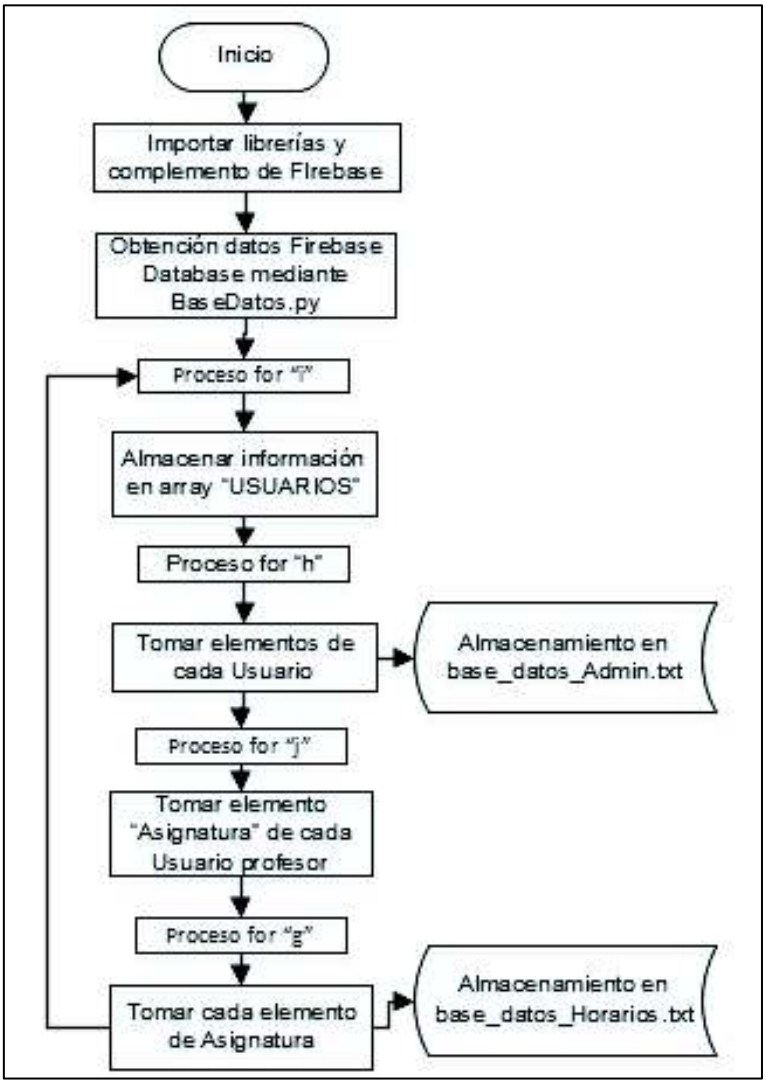

Figura 2.4 Diagrama de flujo para obtención de información Profesores

2.4 ESTRUCTURA DEL PROTOTIPO

El prototipo de gestión y monitoreo para el Laboratorio de Interconectividad de la FIEE estará conformado por tres subsistemas o módulos definidos acorde a los requerimientos mencionados en la sección anterior. La estructura del subsistema de vigilancia considera la conexión de dos cámaras IP, una cámara USB, dos sensores uno de contacto ubicado en la puerta y un infrarrojo (sensor IR) ubicado en el exterior de la sala, además dos

dispositivos USB (*memory flash* y modem 3G).

Un servidor apache instalado en Raspberry Pi y los computadores de la sala conformarán el subsistema de monitoreo de *PC's*.

Finalmente, la inclusión de una aplicación móvil como interfaz de interacción usuario-prototipo. De esta forma el prototipo está enfocado a brindar soluciones que ofrezcan vigilancia y seguridad al Laboratorio de Interconectividad, cumpliendo con los requerimientos mencionados en la sección anterior. La Figura 2.5Figura 2.5ilustra la integración de los subsistemas para conformar el prototipo completo. los subsistemas para conformar el prototipo completo.

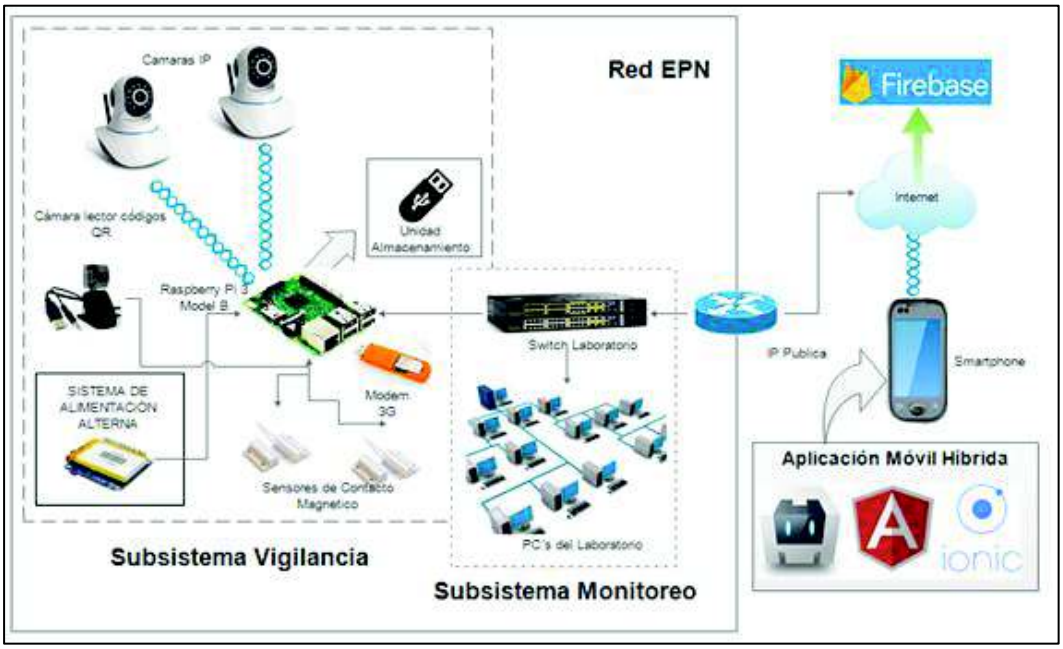

Figura 2.5 Diagrama de los subsistemas que conforman el prototipo

2.4.1 SUBSISTEMA DE VIGILANCIA

Considerando la estructura mencionada, el subsistema de vigilancia proporciona una transmisión de video en tiempo real mediante cada cámara que lo constituye, además si se detecta movimiento en el interior de la sala se inicia la grabación de video y que posteriormente podrá ser visualizado. Otra función es el envío de una notificación SMS y una "push" cuando exista suceso de alarma en la puerta. Finalmente, el acceso a la sala se realiza mediante la lectura de un código QR a través del lector QR propuesto, la cámara USB capta la imagen y realiza una comparación de modo que si la información leída concuerda la almacenada la chapa magnética se abre, de otra forma se debe generar nuevamente un símbolo QR.

Dispositivo Controlador (Raspberry Pi 3)

Como se definió en la propuesta de este proyecto, la Raspberry Pi será el elemento principal del sistema, configurada como AP será el punto central donde se conectan las cámaras de forma inalámbrica, además será la encargada de recibir las señales de salida tanto del sensor de contacto como del sensor IR Además de la conexión de los sensores a los pines GPIO se tiene un relé conectado para controlar la apertura de la chapa magnética. A continuación, se detalla el diseño de cada uno de los módulos que conforman el subsistema de vigilancia con sus procesos respectivos.

Diseño Módulo Vigilancia-Alarma

El módulo de Vigilancia-Alarma estará conformado por la Raspberry Pi como punto de acceso para la conexión de las cámaras, tanto el sensor de contacto como la chapa magnética colocados en la puerta serán controlados mediante los pines GPIO 18 y 17 respectivamente. Finalmente, el modem 3G y la *memory flash* se conectan en dos de los cuatro puertos USB del controlador. La Figura 2.6Figura 2.6ilustra el diagrama de conexión de los dispositivos. Este módulo abarca dos procesos “Proceso de Obtención de Stream de Video y Visualización de imágenes” y “Proceso de Apertura de Puerta Principal envío de notificaciones *push* y SMS” los cuales se describen a continuación. los cuales se describen a continuación.

Figura 2.6 Diagrama Módulo Vigilancia-Alarma

Obtención Stream de Video y Visualización de recursos multimedia

Proceso mediante el cual se observará los interiores de la sala mediante una trasmisión de video provista por las cámaras, esto se logrará con el paquete MOTION dentro de la

Raspberry Pi y el servidor apache donde se ubicará la galería dinámica para la visualización de recursos. La Figura 2.7 ilustra el diagrama de flujo del proceso para poder visualizar la sala.

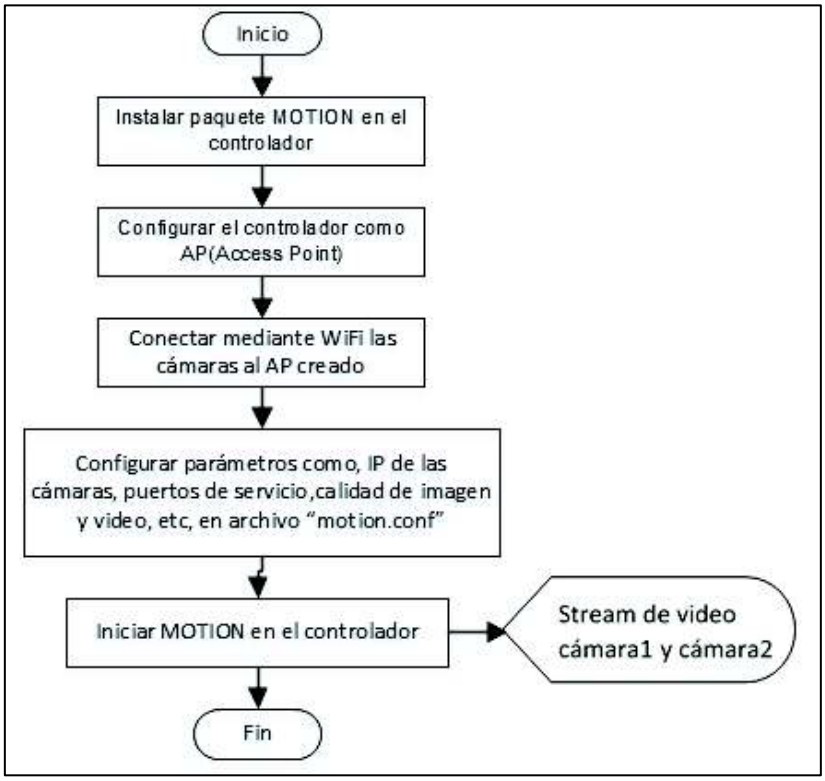

Figura 2.7 Proceso para obtener una visualización del interior de la sala

Cabe mencionar que en este diseño se ha utilizado las mismas cámaras para realizar una detección de movimiento en el interior de la sala mas no un sensor de movimiento, debido a que el paquete MOTION será configurado para que realice una detección de cambios en la imagen, además, provee parámetros de configuración para ejecución de procesos en la Raspberry Pi, de esta forma si la cámara detecta un cambio en la imagen se guarda un video y se observa en la galería dinámica.

Para la visualización de las imágenes y videos grabados se realizará un proceso de selección entre archivos de extensión *.jpg y .mp4* mediante scripts, estos se distribuirán en los directorios correspondientes a la galería. En la Figura 2.8 se ilustra el procedimiento a seguir mediante un diagrama de flujo.

En el proceso de visualización de recursos intervendrán scripts que realicen procesos complementarios, por ejemplo, para la selección y clasificación de archivos se tendrá dos scripts que cumplan con condiciones de búsqueda y de movimiento de archivos.

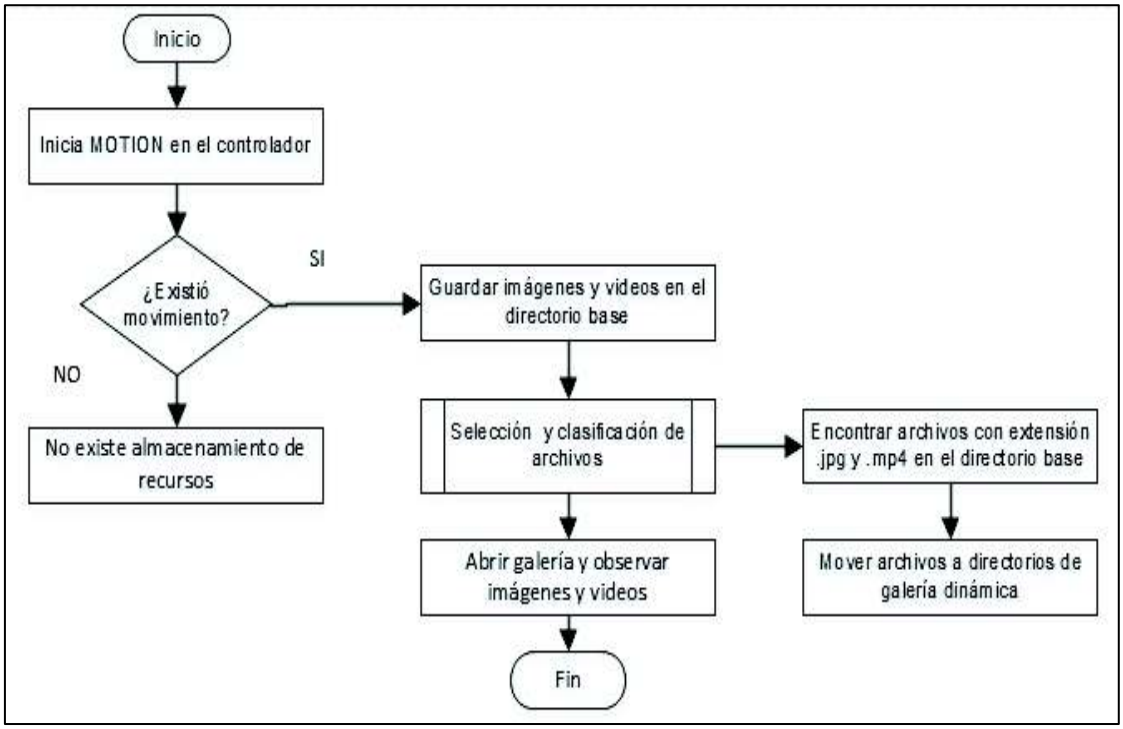

Figura 2.8 Proceso para visualizar los recursos en la galería dinámica

Una vez que los recursos puedan ser visualizados se integrarán subprocesos para respaldar archivos antiguos y liberar espacio del dispositivo de almacenamiento los cuales se describen a continuación.

Respaldo de recursos multimedia

Proceso que se realiza en base a un script el cual considera el tamaño de los directorios de la galería donde se ubican tanto imágenes como videos, una vez se sobrepase un umbral establecido (tamaño del directorio) se moverán archivos antiguos a directorios ubicados en el dispositivo de almacenamiento con el objetivo de tener más espacio disponible para nuevos eventos grabados. El script se desarrollará considerando el procedimiento que se ilustra en la Figura 2.9.

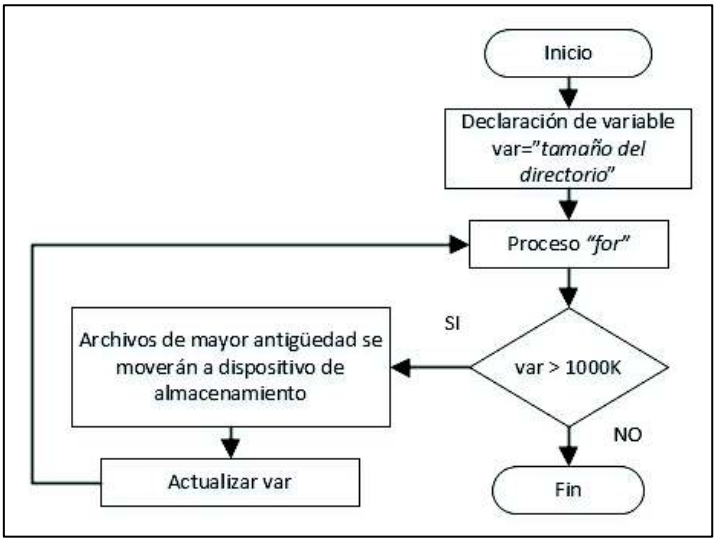

Figura 2.9 Script respaldo de recursos

Liberación espacio dispositivo de almacenamiento

En este proceso se desecha archivos antiguos almacenados en la *memory flash* mediante la ejecución periódica de un script que contendrá variables como el tamaño del directorio y una referencia de tamaño, de esta forma si se sobrepasa este umbral (referencia) se empieza a eliminar los archivos *.jpg y .mp4* más antiguos. En la Figura 2.10 se ilustra el diagrama de flujo del script a ejecutarse. Cabe mencionar que la referencia se selecciona tomando aproximadamente el tamaño de unas 100 imágenes y 100 videos, si el tamaño del directorio sobrepasa esto empieza la eliminación de archivos antiguos.

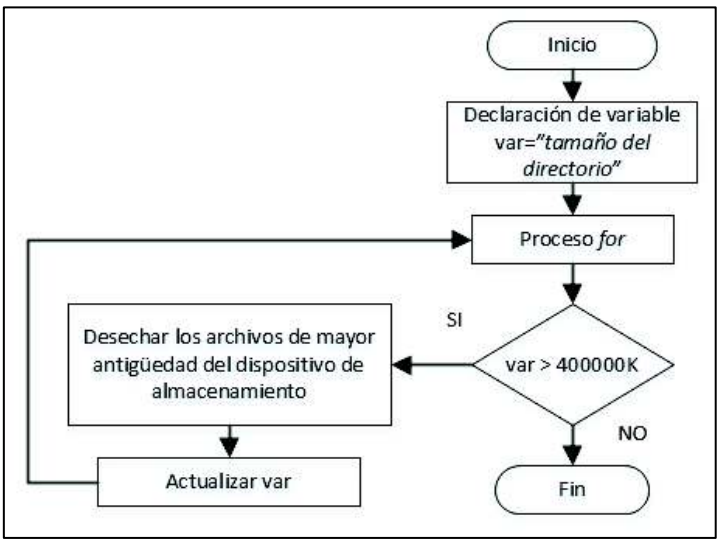

Figura 2.10 Script de liberación de espacio

Apertura de Puerta Principal y Envío de notificaciones push, SMS

Proceso mediante el cual se recibirá notificaciones en el *smartphone* como un sistema de alarma el momento en que exista una apertura de la puerta de la sala, abarcará dos subprocesos mediante la ejecución de scripts uno enviará una notificación de tipo push y otro realizará el envío de un SMS los cuales se detallan a continuación. La conexión del sensor de contacto que se ubicará en la puerta será en el GPIO 22 de la Raspberry Pi además de ser manejado por un script realizado en *python* y que se denominará "aperturaPuerta.py". Un diagrama de conexión tanto del sensor de contacto como el modem 3G se indica en la Figura 2.11 además la Figura 2.12 indica los pasos para el desarrollo del script que controla el sensor de contacto mediante un diagrama de flujo.

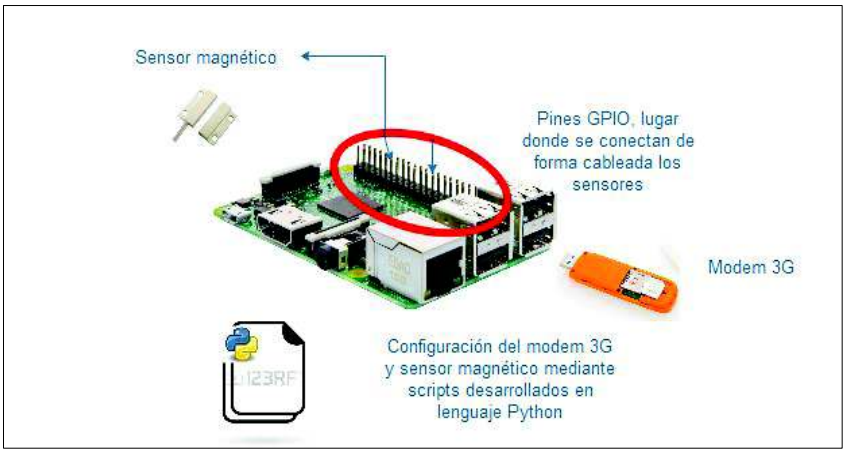

Figura 2.11 Diagrama de la conexión de sensores y modem 3G

Envío notificaciones *push*

La ejecución de este script tendrá como objetivo enviar una notificación del tipo *push* al *smartphone* del usuario el momento en que se detecte una apertura de puerta mediante el sensor de contacto, el script se denominará "pushNotification.py" y se muestra en Figura 2.13.

Envío notificaciones SMS

Conjuntamente al envío de notificaciones *push* se enviará un mensaje de texto con información de hora y fecha, de forma que si la red de datos no funciona un mensaje de texto se considerará como la alerta que le llegará al usuario. El script que se denominará "envioSMS_AT.py" se muestra mediante el diagrama de flujo de la Figura 2.14.

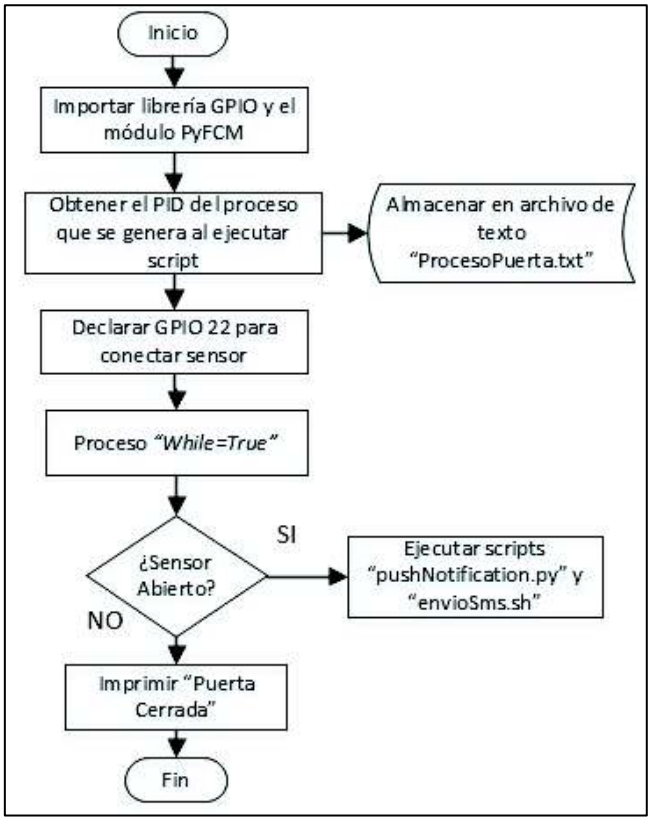

Figura 2.12 Script "aperturaPuerta.py"

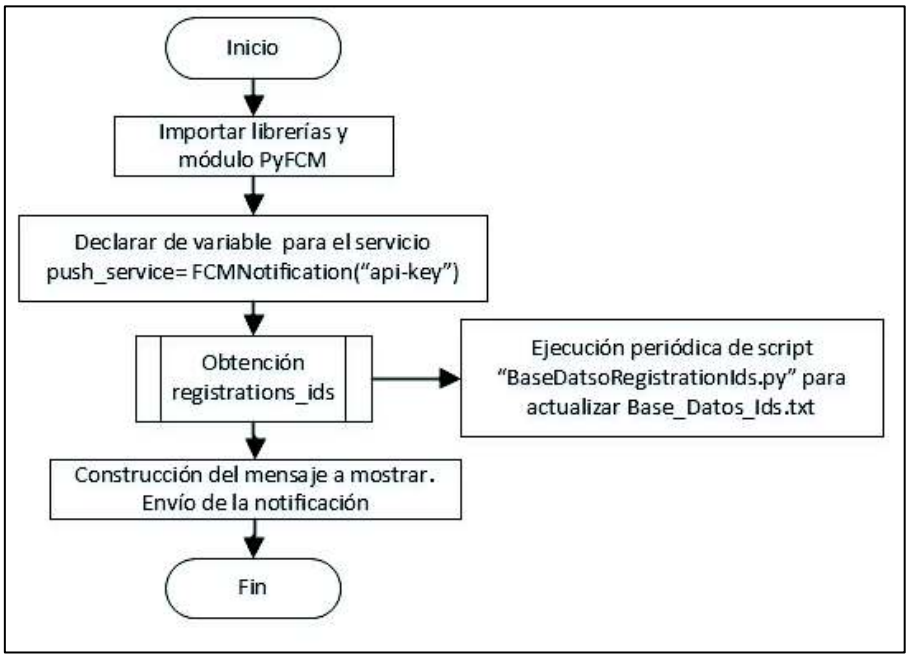

Figura 2.13 Script "pushNotification.py"

Figura 2.14 Script "envioSMS_AT.py"

Adicionalmente, dentro del subsistema de vigilancia se diseña un lector de códigos QR para una posible autenticación e ingreso al laboratorio. Se define dos componentes principales, el primero una cámara, para captar la imagen del símbolo y el segundo un script, que procese y decodifique el símbolo para obtener la información contenida en él.

Diseño Módulo Lector Código QR

El módulo Lector Código QR estará conformado por la Raspberry Pi como punto de conexión para la cámara USB, el sensor IR se conectará directamente al pin GPIO 15 y los tres diodos LEDs a los GPIO 14, 24 y 04 respectivamente que se encenderán como respuesta al proceso, complementando el módulo el relé que activará la cerradura magnética se conecta en el pin GPIO 17.

En la Figura 2.15 se observa el diagrama de conexión de los dispositivos. Este módulo abarca tres procesos "Proceso de Generación del código QR", "Proceso Activación del sensor IR" y "Proceso de Lectura del Código QR y apertura cerradura magnética" los cuales se describen a continuación.

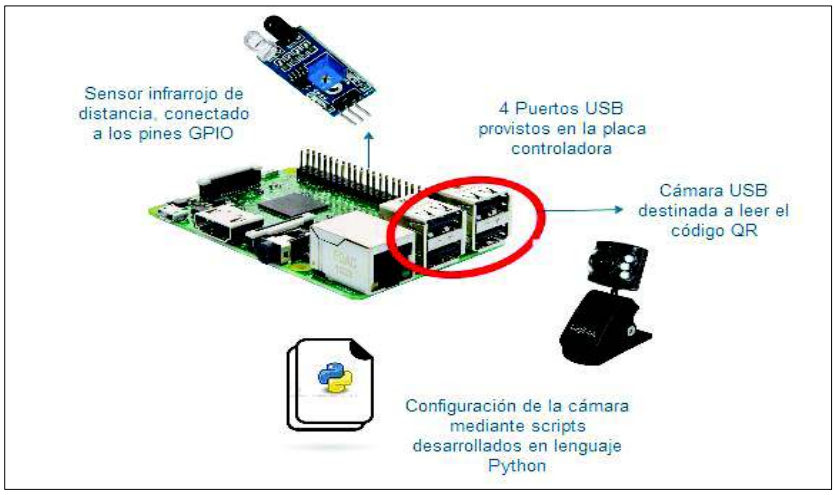

Figura 2.15 Diagrama de conexión dispositivos del Módulo Lector código QR

Generación código QR

La generación del código QR tomará en consideración información de Usuario que se almacenará en archivos de texto una vez descargada de la *Firebase Database* como se mencionó en la sección de diseño de base de datos. La Figura 2.16 indica el script que creará los respectivos archivos para la generación del código QR de acuerdo con el usuario. El objetivo es mostrar la imagen a través del servidor y visualizarlo en la aplicación móvil para su posterior lectura.

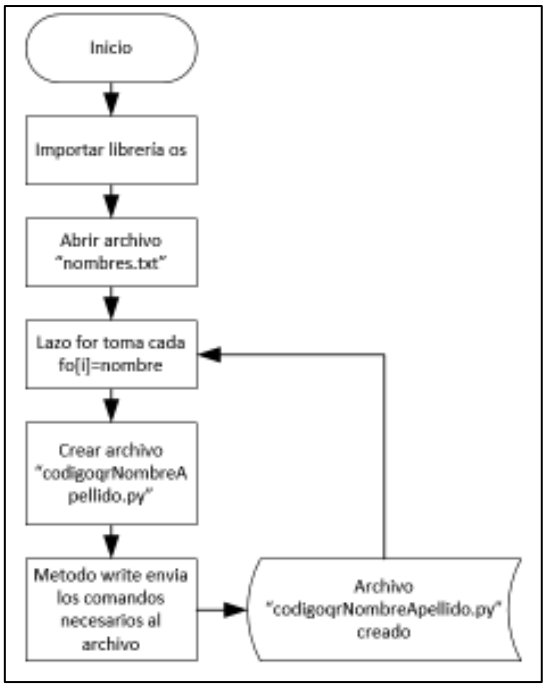

Figura 2.16 Scripts Creación códigos QR

Para que se puedan crear los códigos QR con los datos de cada usuario es necesario que se genere un script para cada uno debido a que desde la aplicación se pretende correr un script dependiendo del usuario que se encuentra ingresado en ese momento., por lo tanto, si existen 3 usuarios agregados a la base de datos, el script de creación de códigos QR creará 3 scripts diferentes de acuerdo con el nombre.

Activación sensor IR

El sensor se manejará a través de un script que se ejecutará cada vez que la Raspberry Pi encienda, de esta forma si se detectara un objeto aproximadamente a 0.5 cm de distancia emite una señal que será recibida por la Raspberry, de esta forma se ejecutará otro script que permita captar una imagen mediante la cámara USB. El proceso de configuración del sensor mediante un script se indica en la Figura 2.17.

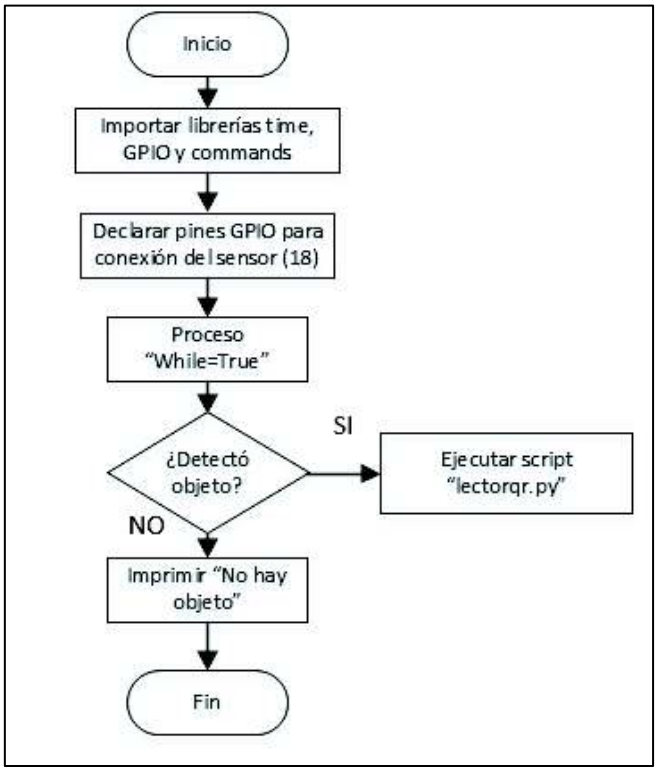

Figura 2.17 Script para activar sensor IR

Lectura código QR y Apertura cerradura magnética

La lectura del QR se realizará en la parte exterior de la sala una vez concluida la instalación de los dispositivos. Un script que se denominará "lectorqr.py" se ejecutará el momento en que se active el sensor IR y se encenderá un LED color amarillo indicando que inicio el proceso de captura de imagen. Para el desarrollo del script será necesario

paquetes como *PIL y zbarlight* en la Raspberry Pi para procesar la imagen y decodificarla. La Figura 2. 18 muestra el script de lectura de código QR.

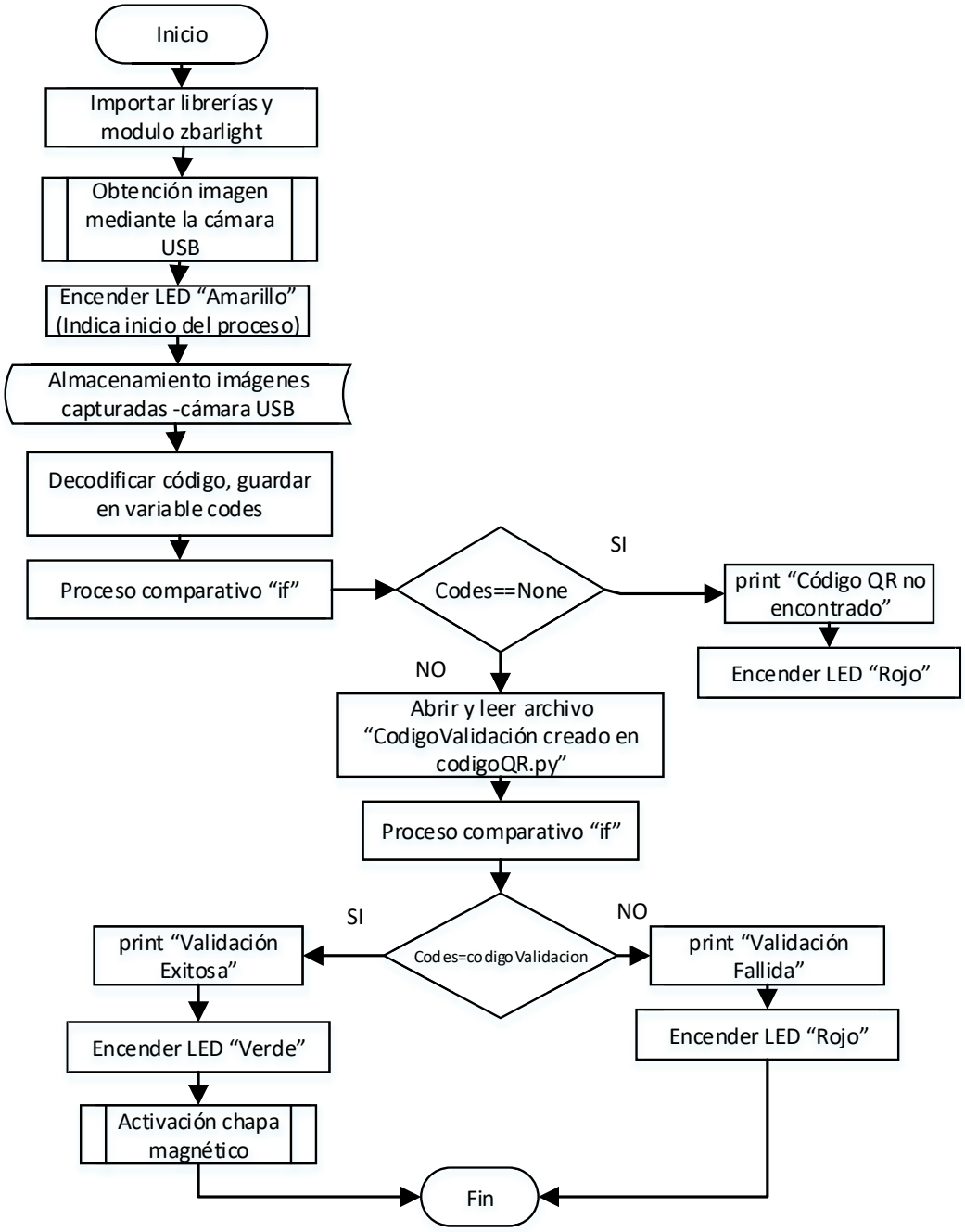

Figura 2. 18 Script "lectorqr.py"

La información decodificada de la imagen captada se almacenará en una variable para su posterior comparación con los datos que generarán el símbolo QR, de modo que si la información concuerda se tendrá una validación exitosa dando lugar a la apertura de la

chapa magnética y se encenderá el LED color verde en respuesta a esta validación. Por otro lado, si el LED color rojo se encendiera indicará un error en la lectura para que el usuario genere otro código QR

2.4.2 Subsistema de Monitoreo de Pc's

El subsistema de monitoreo de *PC's* proporcionará la función de encendido/apagado remoto de los computadores teniendo como base la ejecución de scripts desde un servidor apache instalado previamente en el controlador. Utilizará el estándar *Wake On LAN*, direcciones tanto IP como MAC de los *PC's* y comandos propios de Linux.

Diseño Módulo Monitoreo PC's

El controlador se conectará mediante cable UTP al *switch* principal del laboratorio, de esa forma se asignará una dirección IP fija que se encuentra dentro de la red de los computadores, serán necesarios paquetes de configuración como *wakeonlan* y *samba-common* para cumplir con el requerimiento establecido. En la Figura 2.19 se muestra la conexión descrita. Este módulo comprende 3 diferentes procesos los cuales se cumplen con la ejecución de scripts, entre estos se encuentra el encendido remoto de computadores, el apagado de estos y finalmente un subproceso que indicará si el computador esta encendido o apagado, estos procesos se describen a continuación.

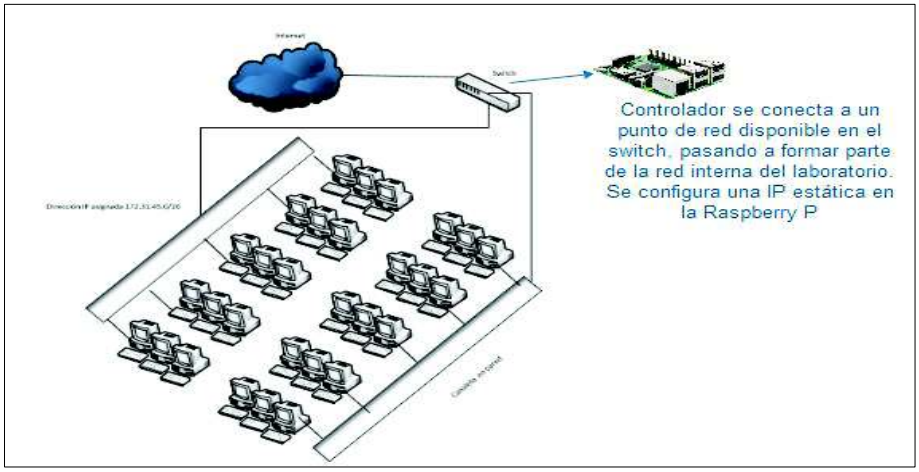

Figura 2.19 Esquema de conexión del controlador con la LAN del laboratorio

Encendido de Computadores

El encendido de un computador dentro de la sala, o a su vez de todos los computadores considera la dirección MAC y la tarjeta de red del computador. Este proceso se realiza a través de la ejecución de un script el cual se muestra en Figura 2.20

Figura 2.20 Script para encender PC's "wakeAll.py"

Apagado de computadores

Este proceso permitirá al usuario apagar de forma remota un computador o todos a la vez, se ejecutará un script desarrollado en Python el cual se observa en el diagrama de la Figura 2.21

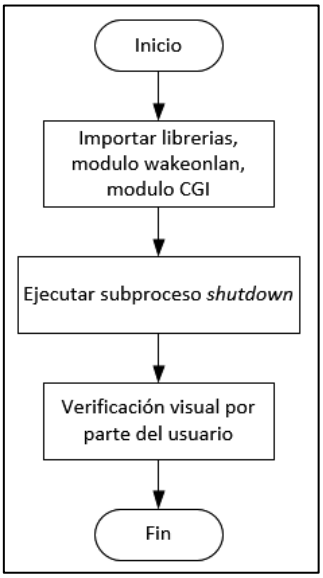

Figura 2.21 Script para apagar los PC's "shutAll".py

Obtención de información sobre el estado de un computador

La ejecución periódica de un script que se denominará "estadosPCs.py" el cual se

muestra en el diagrama de flujo de la Figura 2.22 . Se utilizará el comando *ping* y de acuerdo con el resultado se almacenará en un archivo de texto la palabra *"activado" o "desactivado"* respectivamente, de esta manera la información obtenida será utilizada en una función de la aplicación móvil propuesta.

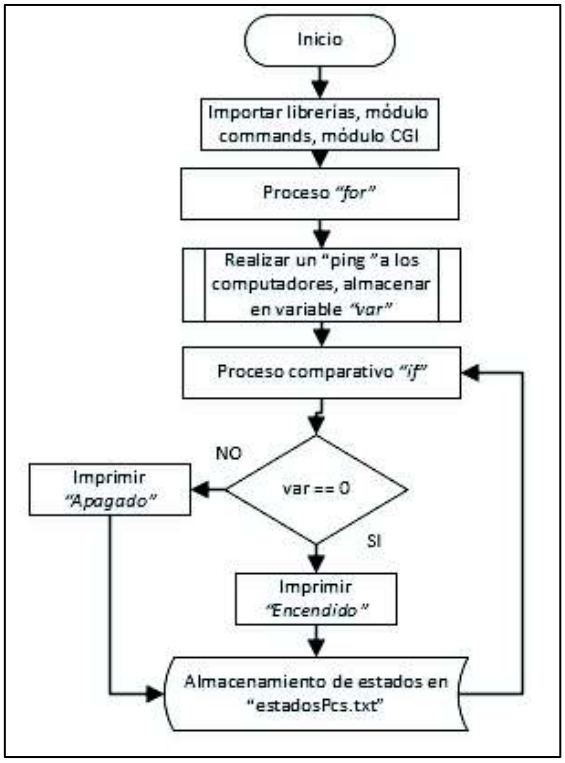

Figura 2.22 Script para conocer el estado de un PC

2.5 DISEÑO GENERAL DE APLICACION MÓVIL

En esta sección se describirá los usuarios, diagramas de caso de uso de la aplicación móvil, seguido del diseño de cada una de las interfaces o páginas que contendrá.

2.5.1 Usuarios del prototipo

El prototipo será administrado por un super-usuario que en este caso es el jefe del Laboratorio, además de usar toda la aplicación sin limitaciones tiene la opción de agregar cuentas tanto para el ayudante como para profesores. De esta manera se tendrán tres tipos de usuarios para la aplicación móvil.

El usuario ayudante tiene el control de toda la aplicación exceptuando la creación de cuentas mientras que el usuario profesor tiene acceso solamente a páginas de monitoreo y de creación del código QR para el ingreso al laboratorio.

2.5.2 Diagramas de Caso de Uso

Son diagramas representados por símbolos los cuales delimitan y representa las acciones posibles que el usuario podrá realizar en el sistema [39]. Los elementos que conforman el diagrama son Actor, Caso de Uso y Sistema. Se han considerado los siguientes casos, el usuario no autenticado, un usuario autenticado administrador, un usuario autenticado ayudante y finalmente un usuario autenticado profesor.

- **Diagrama de usuario no autenticado:** El usuario podrá observar la pantalla de bienvenida de la aplicación y el menú principal donde se muestra la opción Iniciar Sesión. La Figura 2.23 ilustra el diagrama de caso de uso de un usuario no autenticado.

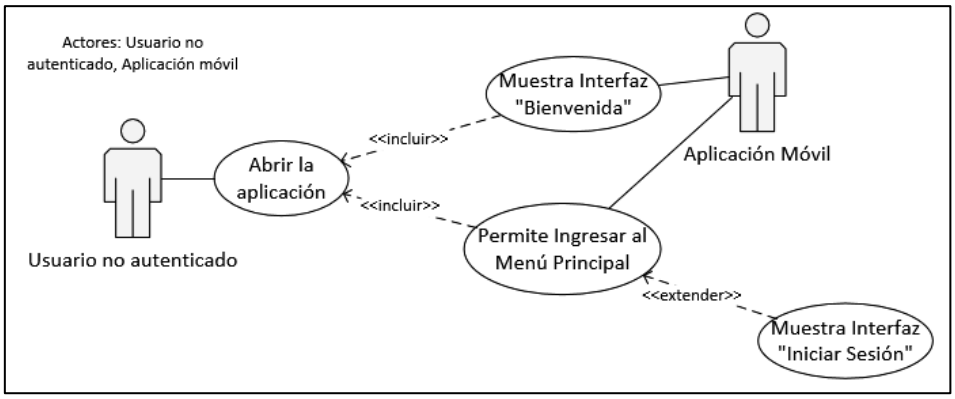

Figura 2.23 Diagrama de caso de uso “Usuario no autenticado

- **Diagrama de usuario autenticado (Administrador):** El administrador, una vez valide sus credenciales empieza la navegación a través de las páginas provistas por la app, las mismas que se indican en la Figura 2.24.

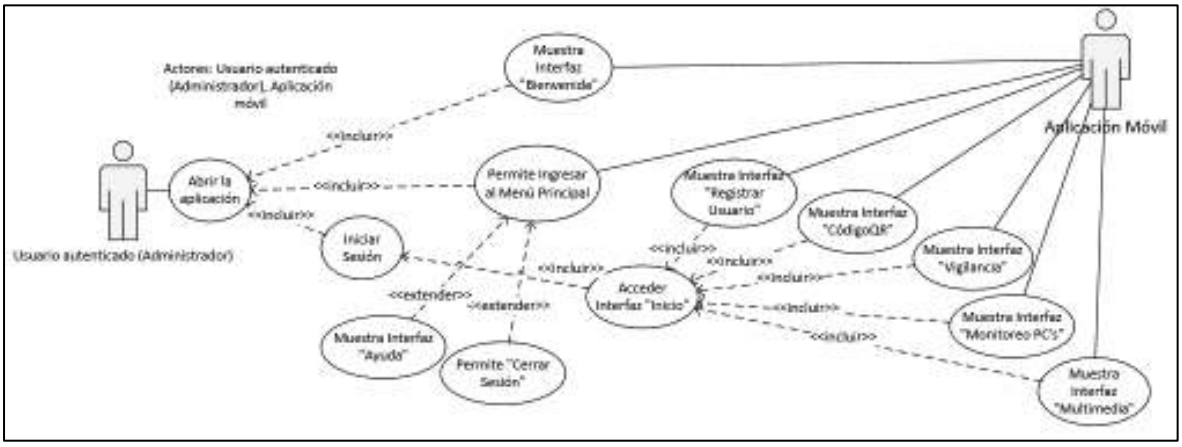

Figura 2.24 Diagrama Caso de Uso Administrador

- **Diagrama de usuario autenticado (Ayudante):** A través del administrador se crean las credenciales para autenticar el usuario *Ayudante,* una vez validadas estas el usuario puede navegar en la aplicación con limitaciones de acceso a interfaces. La Figura 2.25 ilustra las páginas accesibles para este usuario.

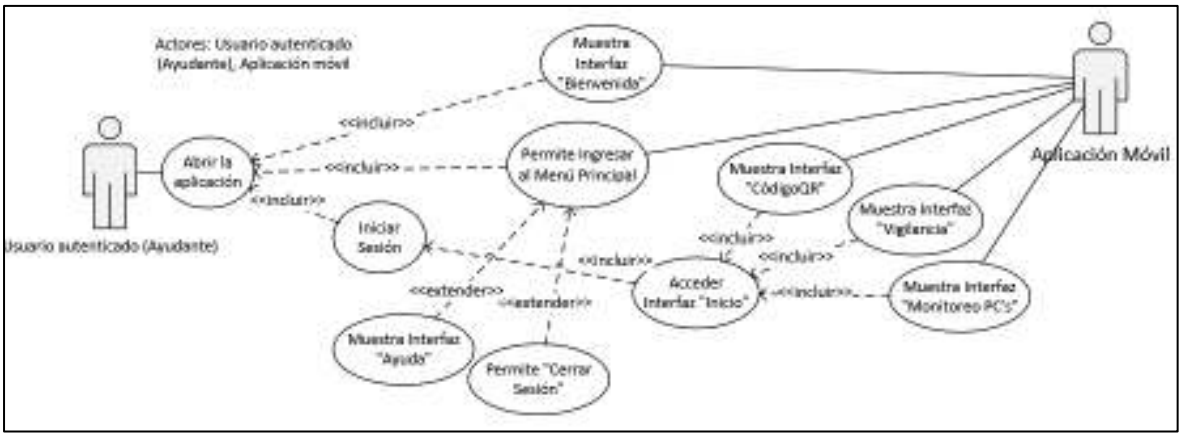

Figura 2.25 Diagrama de caso de uso “Ayudante”

- **Diagrama de usuario autenticado (Profesor):** Tiene un acceso limitado después de ingresar a la aplicación, puede acceder solo a las interfaces de monitoreo y código QR como indica el diagrama de la Figura 2.26

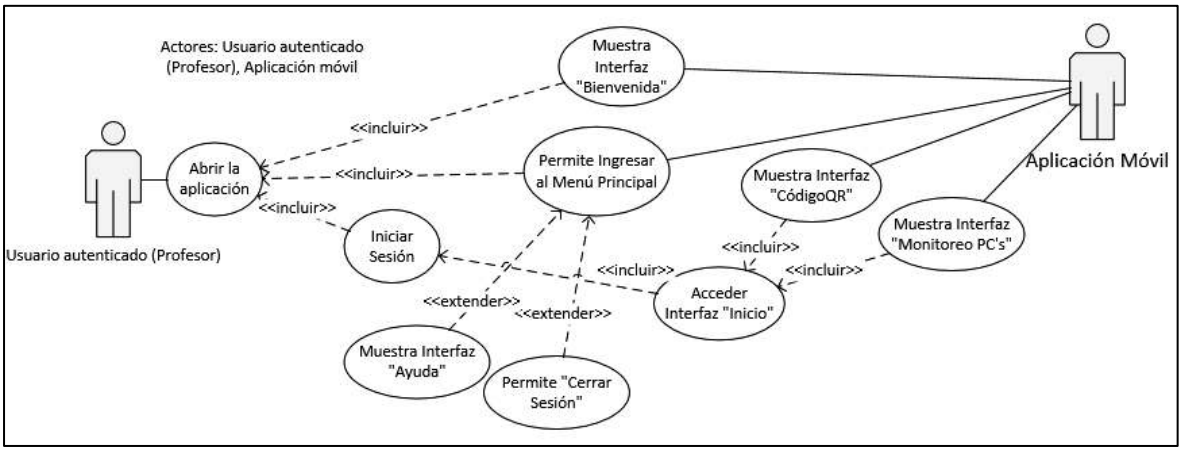

Figura 2.26 Diagrama de caso de uso “Profesor”

2.5.3 Diseño de las interfaces de la aplicación movil

La aplicación móvil está destinada a ser una interfaz que permite al usuario gestionar las funciones del prototipo de forma remota. Para ello se ha considerado elaborar varias interfaces que se detallan a continuación.

- **Interfaz de Bienvenida**

Primera interfaz visualizada por el usuario posee un botón de Inicio de Sesión. Se complementa con una imagen de la FIEE como fondo y el logo de la EPN. Además, se agrega texto donde se describe el nombre de la Universidad, la Facultad y el título del proyecto de titulación. La Figura 2.27 ilustra un bosquejo inicial de la interfaz de bienvenida.

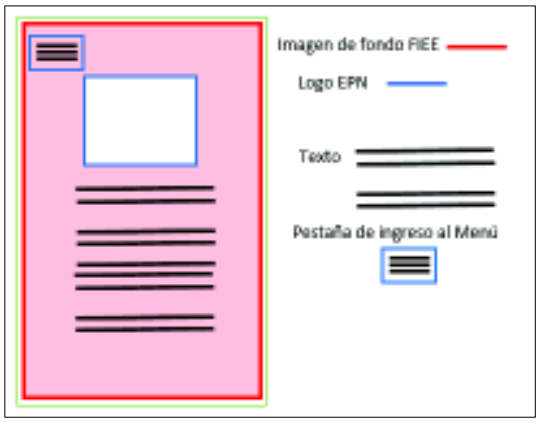

Figura 2.27 Boceto de la interfaz Bienvenida

- **Menú principal**

Presentará dos opciones, la primera cuando el usuario no se encuentra autenticado mostrará solamente la opción de Iniciar Sesión, sin embargo, al momento que el usuario valide credenciales serán dos las opciones Cerrar Sesión y Ayuda.

- **Interfaz Iniciar Sesión (Login)**

Diseñada de tal forma que permita el ingreso de un correo electrónico y contraseña para poder acceder. Contiene el logo de la Universidad y un espacio para ingresar texto tanto para el Usuario como contraseña. En la Figura 2.28 se observa un boceto inicial de la interfaz de Inicio de Sesión además del diagrama de funcionamiento de la validación de credenciales.

Figura 2.28 Boceto interfaz Inicio Sesión

- **Interfaz de Inicio (Home)**

Interfaz que presentará diferentes opciones como Registro de Usuario, Código QR, Multimedia, entre otros, estas se mostrarán de acuerdo con el rol de Usuario autenticado. A modo de etiquetas, se presentan botones para direccionar a las distintas interfaces que complementan la aplicación. Adicionalmente se tiene la opción Inicio en el pie de página La Figura 2.29 ilustra un boceto de la página diseñada.

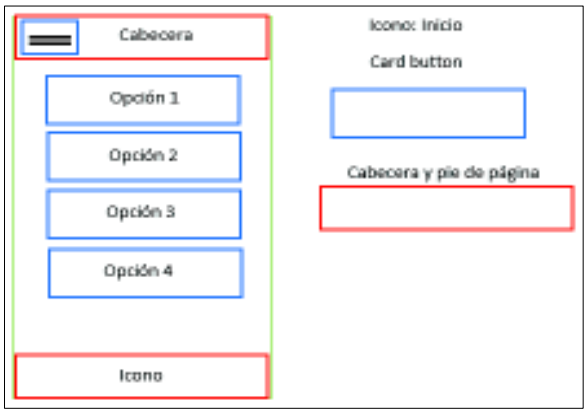

Figura 2.29 Boceto de la Interfaz Inicio (Home)

- **Interfaz Generación del código QR**

Conformada por una cabecera con el nombre de la interfaz, un botón que al presionar muestra el código generado en pantalla y un mensaje de ayuda para el usuario. En el pie de página se dispone del botón Inicio.

- **Interfaz Vigilancia**

Permitirá al usuario la visualización del interior del laboratorio mediante el stream de video proporcionado por las cámaras. Al abrir la página se tendrá dos recuadros y una pequeña descripción. Para observar con mayor detalle se presiona en la imagen, de esta forma se puede realizar un pequeño acercamiento o alejamiento

- **Interfaz Multimedia**

Muestra una galería de imágenes y videos distribuidos en filas acorde a las cámaras (1 y 2), la galería se actualiza cada vez que finalice un evento de Motion, almacenando la primera imagen con la que se crea el video. Permite la visualización de las imágenes y la reproducción de los videos.

- **Interfaz Monitoreo PC's**

Permitirá al usuario encender o apagar todos los computadores en el laboratorio mediante un botón. Sin embargo, dependiendo del estado del computador los botones cambiarán de color. Al pulsar un botón se mostrará un mensaje de alerta y las opciones de encender, apagar y cancelar de ser el caso. Cada botón estará numerado de acuerdo con el último dígito de la dirección IP asignada.

- **Interfaz agregar usuarios**

Interfaz destinada únicamente para el uso del Administrador, permitirá agregar información acerca del ayudante y profesores que utilizan el laboratorio. Esta interfaz permitirá el ingreso de Nombres y Apellido, Correo electrónico, Contraseña mediante teclado, mientras que la agregación del Cargo y el Horario de Asignaturas se realiza mediante selección de opciones. Una vez completos los campos al presionar el botón se subirá la información a la base de datos del Firebase.

Los bocetos de las interfaces secundarias de la aplicación móvil se ilustran en el ANEXO IV.

2.6 IMPLEMENTACIÓN DEL PROTOTIPO

En esta sección se describe el proceso de instalación y operación de paquetes propios de Raspberry Pi, seguido del desarrollo de scripts en lenguaje Python para la manipulación de los componentes físicos. Por último, se detalla la creación de cada una de las interfaces que conformarán la aplicación.

2.6.1 Instalación del sistema operativo

Dado que Raspberry Pi 3 no cuenta con un disco duro es necesario que el Sistema Operativo arranque desde una memoria MicroSD. La capacidad de memoria de la MicroSD que contendrá el sistema operativo debe ser como mínimo de 8GB[40]. Una vez descargado Raspbian de la página web oficial se obtiene el archivo *.iso* que es una imagen del SO que posteriormente se cargará en la MicroSD

Posteriormente, a través del programa *SDFormatter* se configura la microSD para que posea características *booteables,* esta configuración se realiza en la ventana que muestra Figura 2.30 en la cual se ingresa en *Option*, se selecciona "*Erase*" u "*Overwrite*" y finalmente "*ON*". El proceso puede tardar varios minutos.

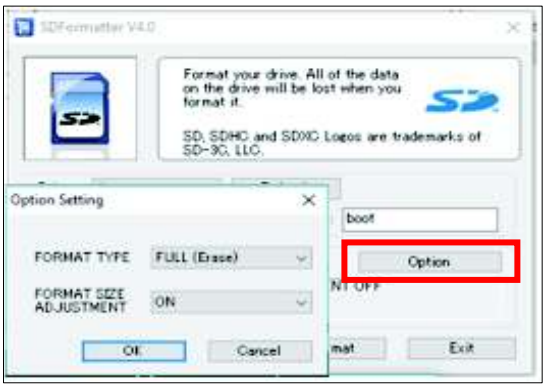

Figura 2.30 Formateo Tarjeta MicroSD

Una vez configurada la tarjeta de memoria de forma adecuada, mediante el programa Win32DiskImager se carga el S.O en esta, la Figura 2.31 indica la ventana que despliega al ejecutarse. Se completa los campos con el directorio donde se encuentra la imagen ISO además de la unidad de disco asociada a la SD. Finalmente se selecciona *Write* y el proceso de escritura empieza**.**

Figura 2.31 Escritura del SO en la tarjeta MicroSD

Finalmente, se inserta la SD en la ranura de Raspberry Pi 3, se energiza la tarjeta y se conectan los periféricos básicos: Pantalla, mouse y teclado para poder manejar la Raspberry por primera vez. Para prescindir de los periféricos mencionados en el presente proyecto se le asigna una dirección IP a la Raspberry Pi y se utiliza un acceso remoto desde clientes SSH.

Protocolo SSH (Conexión Remota)

El protocolo SSH permite acceder remotamente a la Raspberry Pi a través de una dirección IP estática que pertenezca a la red del laboratorio la cual es asignada a la interfaz Ethernet. En la Raspberry el protocolo SSH se activa en la pestaña *"preferencias"* y se selecciona la opción SSH *enabled*, o un segundo caso ingresando mediante teclado el comando `sudo raspi-config` [41]

La Figura 2.32 muestra dos clientes para conexión remota, uno de estos es *Putty* mediante el cual la interacción con el usuario es a través de línea de comandos, sin embargo, si es necesario una conexión remota considerando una interfaz gráfica se tiene el escritorio remoto de Windows. En este último caso se considera configuraciones adicionales en el controlador como se detalla a continuación

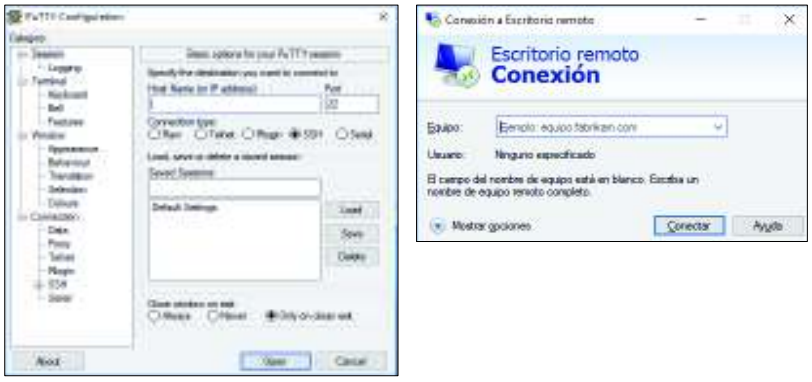

Figura 2.32 *Putty* y Conexión a Escritorio Remoto provisto por Windows

INTERFAZ GRÁFICA RASPBERRY PI 3

Un entorno gráfico facilita al usuario una mejor interacción con características propias del controlador.

- **Conexión Escritorio Remoto**

Mediante el escritorio remoto podemos acceder al entorno gráfico de Raspberry Pi teniendo un acceso total de forma visual a todo su contenido.

Para dicho proceso se instala un paquete xRDP, siendo este una derivación de código abierto de *RDP (Remote Desktop Protocol*). Se instala el servidor *VNC (Virtual Network Computing)* el mismo que se ejecuta mediante un terminal. En el *CLI (Command line Interface)* del controlador se digita los siguientes comandos `sudo apt-get install xrdp` y `sudo apt-get install tightvncserver`, para la instalación de xRDP y del servidor VNC respectivamente [42].

Finalizada la instalación de paquetes se reinicia la Raspberry Pi mediante el comando `sudo reboot`, de esta forma podemos acceder al entorno gráfico mediante el escritorio remoto de Windows ingresando la dirección IP asignada y credenciales de autenticación que por defecto son el usuario `pi` y la contraseña `raspberry` las cuales pueden ser modificadas posteriormente. En la Figura 2.33 se ilustra la interfaz gráfica generada por la conexión de escritorio remoto de Windows.

Figura 2.33 Entorno gráfico de Raspberry Pi 3

2.6.2 Implementación subsistema de vigilancia

En esta sección se describe el proceso de implementación del subsistema de vigilancia considerando la fase de diseño descrita en la sección anterior, de esta forma se empieza con el proceso de obtención de datos, seguido de la implementación del módulo vigilancia alarma y el módulo lector código QR

Obtención de información Firebase Database

La implementación del sistema de vigilancia considera la información de la base de datos, a continuación, se detalla el desarrollo de los scripts que realizan el proceso obtención de

datos descritos en la sección de diseño.

Obtención de información Administrador / Ayudante

El desarrollo del script inicia agregando las librerías y paquetes adicionales, seguido de agregar el "api key" del proyecto de *Firebase* [43] y el nombre del nodo ubicado en la base de datos, en este caso ***Usuarios/Administradores***. Se considera el procedimiento descrito en la Figura 2.2. Se realiza varios lazos *for* para obtener los elementos del *array* conformado con datos del usuario, al mismo momento se van almacena en un archivo de texto Base_Datos_Admin.txt. El segmento de código de la Figura 2.34 muestra cómo se obtiene los datos desde *Firebase*, en el ANEXO VI se muestra el código completo del script.

```
 GNU nano 2.2.6            File: respaldoNumerostratarfor.py

#!/usr/bin/env python
from firebase import firebase
import os
import time
firebase = firebase.FirebaseApplication('https://authlabointercon.firebaseio.com/')
result = firebase.get('/usuarios/',None)
values = result.values();# sacar valores de los usaurios
```

Figura 2.34 Obtención de datos nodo Usuarios Administrador/Ayudante

El script para obtener solamente los nombres de usuarios registrados se desarrolla en base al diagrama de la Figura 2.3. Se agrega librerías y se abre el archivo de lectura Base_Datos_Admin.txt, de esta forma se obtiene un *array* con los elementos del .txt**.** El valor de un elemento del *array* se obtiene ingresando el nombre del vector y la posición del dato, en la Figura 2.35 se muestra una porción de código del script "leernombres.py".

```
 GNU nano 2.2.6            File: leertxt1.py                    Modifie

#!/usr/bin/python
archivo=open("base_datos_ids.txt","r")# abrir el archivo de texto
fo=archivo.readlines() #leer linea por linea y almacer en array
archivo.close() #cerrar archivo de texto
print fo[2]
print fo[3]# Imprimir en pantalla el tercer cuarto elemento
print fo[0]
```

Figura 2.35 Lectura archivo de texto Base_datos_Ids. txt.

Tomando en consideración la forma de lectura de archivos txt descrito en el párrafo anterior se obtiene datos como los nombres, números y registration_id del usuario para procesos posteriores.

IMPLEMENTACIÓN DEL MÓDULO VIGILANCIA-ALARMA

Obtención Stream de Video y Visualización de recursos multimedia.

A continuación, se describe la implementación y desarrollo de scripts de los procesos que conforman el módulo

Configuración del Controlador como AP (Access Point) [44]

El prototipo usa una conexión inalámbrica entre las cámaras y el controlador, por lo tanto, a través de la instalación de paquetes como *hostapd, udhcpd e iptables* y su respectiva configuración de archivos la Raspberry Pi cumple el objetivo de trabajar como un Access Point. El *Acces Point* dispone de un direccionamiento IP interno que puede ser configurable de acuerdo a los dispositivos a conectarse.

Hostapd es un *daemon* que permite la creación de puntos de acceso, se instala ingresando el comando `sudo apt-get install hostapd`. El paquete *udhcpd*, una derivación del protocolo *dhcp*, que provee de una configuración más sencilla, se instala con el comando `sudo apt-get install udhcpd`. Finalmente, *iptables* un firewall de linux no solamente permite filtrar paquetes, sino que provee de la traducción de direcciones de red (NAT), se instala en la Raspberry mediante el comando `sudo apt-get install iptables`.

- **Configuración udhcpd**

Una vez descargado e instalado el paquete, se trabaja en el archivo `/etc/default/udhcpd.conf`, donde se realiza configuraciones como el rango de la red interna, máscara de red, *gateway* la interfaz para el mapeo de direcciones, como lo indica la Figura 2.36. Para el presente proyecto son necesarias dos direcciones IP para dos cámaras, de esta forma se evita que otro dispositivo se conecte al AP.

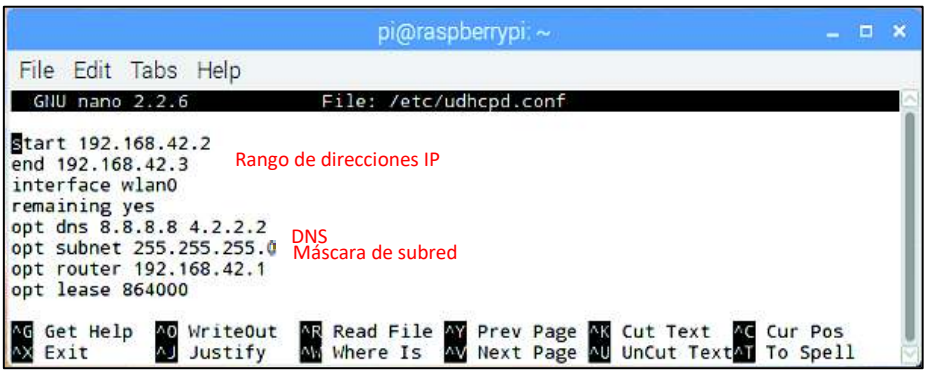

Figura 2.36 Archivo de configuración udhcpd.conf

- **Configuración hostapd**

Parámetros propios de un AP tales como SSID, contraseña, interfaz, parámetros de seguridad, etc, se configuran en el archivo `hostapd.conf` ubicado en el directorio `/etc/hostapd` una vez instalado el paquete [45]. En la Figura 2.37 se observa la configuración de parámetros en el archivo `hostapd.conf`.

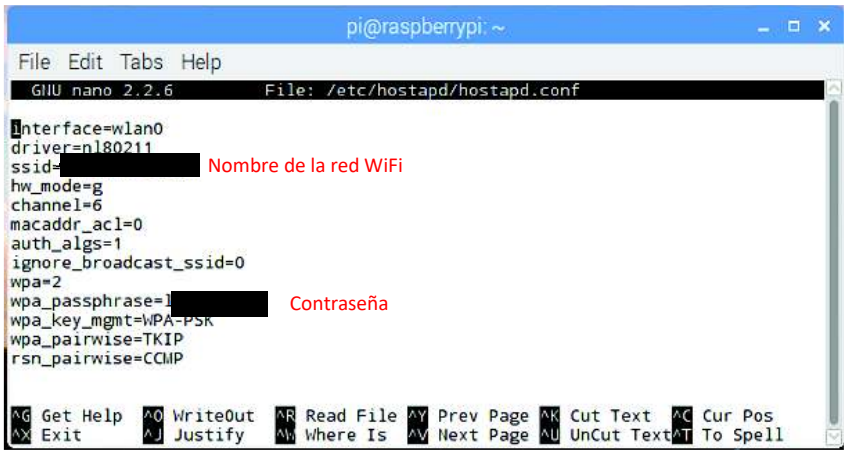

Figura 2.37 Archivo de configuración hostapd.conf

- **Configuración Iptables (Port forwarding)**

Iptables, cortafuego utilizado en distribuciones de Linux. Considerado una herramienta que permite la traducción de direcciones de red (NAT). Da lugar a la creación de reglas que analizarán los paquetes de datos entrantes o salientes de nuestra máquina y dependiendo de su estado permitir o denegar el paso hacia su destino [46]. Las reglas analizan aspectos como:

- Tipo de datos
  - INPUT: Paquetes que llegan
  - OUTPUT: Paquetes que salen
  - FORWARD: Paquetes que pasan
- Interfaz por la que paquetes entran o salen: (-i=input) (-o=output)
  - wlan0, wlan1, eth0, eth1, entre otras
- Dirección IP origen y destino: (-s= source) (-d= destino)
  - Puede escribirse una dirección IP o el rango de la red.
- NAT (modifica la IP origen y destino para conectarse con otras interfaces ya sea en la red o a Internet)

- PREROUTING: Se filtran los paquetes antes de enrutar
- POSTROUTING: Se filtran paquetes después de enrutar

- ACCION: Acciones a realizarse en la configuración de la regla de Iptables
  - ACCEPT: Aceptar la petición
  - REJECT Rechazar la petición e informar al emisor.
  - DROP: Rechazar la petición

Se configura el mapeo de direcciones en la Raspberry mediante Iptables considerando cada una de sus reglas, en este caso se realiza entre la interfaz eth0 y la wlan0. Se añade una regla con el comando ***–A*** y el tipo de datos ***FORWARD,*** el comando ***– j*** nos indica que se debe realizar una ***ACCION.*** Los establecimientos de conexiones pueden ser nuevos o establecidos usando los comandos ***NEW y ESTABLISHED*** respectivamente. La Figura 2.38 indica la configuración en la Raspberry Pi del mapeo de direcciones entre eth0 y wla0.

Finalmente en el archivo de configuración **`/etc/sysctl.conf`** se habilita la línea comentada con el siguiente texto ***"net.ipv4.ip_forward=1",*** posteriormente para que se habiliten las configuraciones se reinicia la Raspberry..

```
 GNU nano 2.2.6       File: /home/pi/scripts/creacionAP.sh

sudo echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED $
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

Figura 2.38 Configuración Iptables (eth0 y wlan0)

Finalmente ingresando el comando ***sudo ap [nombre del AP] [contraseña]*** se crea el AP con su respectivo SSID que es apreciado en dispositivos con la característica de conectarse a una red mediante Wifi como se indica en la Figura 2.39**.**

Figura 2.39 Visualización a través de un móvil al Access Point creado

La automatización de scripts permite que estos se ejecuten automáticamente al iniciar la Raspberry Pi. De esta forma se asegura el funcionamiento continuo de Raspberry un Access Point. A continuación, se detalla cómo se realiza la ejecución automática de scripts.

Automatización de scripts (cron)

Procesos que involucran al prototipo deben tener la capacidad de ejecutarse automáticamente en el momento que el controlador se inicie, tal es el caso de la configuración de AP y de la activación de alarma, para este proceso se utiliza archivos en el módulo de `crontab`[47]. Sin embargo, para la ejecución de procesos como la activación de alarma se considera el horario de clases establecido, de esta forma se automatiza este proceso de acuerdo al Usuario y hora actual.

A continuación, se muestra un ejemplo de la configuración del archivo `crontab -e` mediante el cual se automatiza los procesos descritos.

- `* * * * * sudo python /usr/lib/cgi-bin/activarAlarma.py`
- `@reboot sudo bash /home/pi/scripts/CreacionAP.sh`

Conexión de cámaras IP

Las cámaras seleccionadas se conectan a la red del laboratorio mediante su interfaz Ethernet a través de un cable de red, mediante un navegador se accede a la página de configuración. Una vez dentro se selecciona la opción *Network y se ingresa* SSID correspondiente al AP creado con Raspberry Pi (WIFIRASPI3LAB) además se ingresa la contraseña respectiva.

Para la asignación de direcciones IP a las cámaras se considera el rango provisto por el archivo `udhcpd.conf`. Se desconecta el cable Ethernet y se accede a la cámara desde un navegador ingresando la IP fijada y las respectivas credenciales.

- **Instalación Paquete MOTION**

Motion es un paquete destinado al manejo y control de cámaras ya sean del tipo USB o IP, se instala en la Raspberry Pi a través del comando `sudo apt-get install Motion` , en este proyecto se dispone de la versión 4 de MOTION, la cual presenta mayores prestaciones y parámetros de configuración en comparación a versiones anteriores**.** El archivo principal está ubicado en el directorio `/etc/motion` y se denomina `motion.conf`**.**. La Figura 2.40 ilustra la versión del paquete además de las opciones

de arrancar el programa ingresando el comando `motion -help`

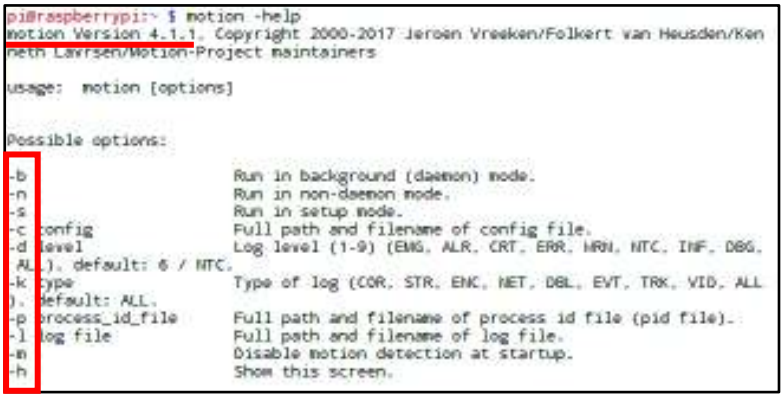

Figura 2.40 Versión de paquete MOTION y parámetros de arranque

Opciones de inicio del programa permiten apreciar errores de ser el caso, como por ejemplo la opción –n permite ver la conexión de la cámara, el puerto donde muestra el Stream, el inicio y final de un evento cuando detecta movimiento, entre otros.

En la Figura 2.41 se observa una porción del archivo de configuración de una cámara correspondiente al Stream de video. Los archivos de configuración completos se pueden apreciar en el ANEXO VI

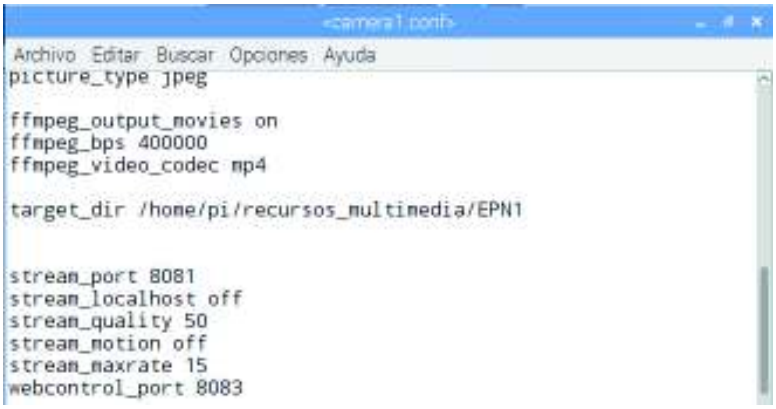

Figura 2.41 Configuración parámetros Stream para la cámara 1

Montaje del dispositivo de almacenamiento

El dispositivo de almacenamiento que en este caso es una memory flash se conecta al puerto USB de la Raspberry Pi, mediante el comando `sudo fdisk -l` se muestra el nombre asignado al dispositivo. Se edita el archivo `/etc/fstab` para realizar el montaje, ingresando el nombre asignado al dispositivo y el directorio, en este caso tenemos como nombre asignado `/dev/sda1` y el directorio a montarse `/home/pi/multimedia` [48].

En la Figura 2.42 se observa el archivo de configuración `fstab`.

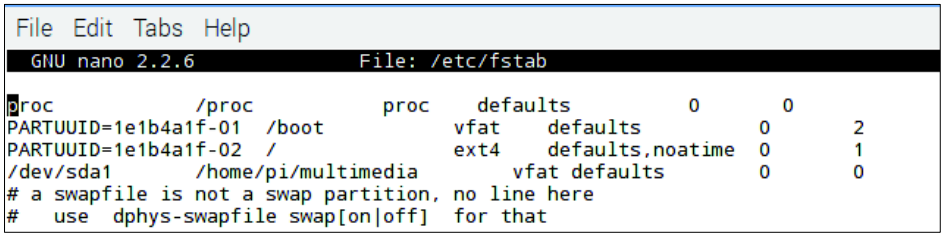

Figura 2.42 Montaje memory flash en /home/pi/multimedia

El comando `mount -a` permite el montaje del dispositivo en el directorio asignado.

Visualización de Recursos Multimedia

Finalizado un evento por las cámaras entra en acción el script de filtración dando lugar a una reubicación de archivos del tipo .jpg y .mp4. De esta forma las imágenes se ubican en una carpeta denominada "Imágenes-Camara1" e "Imágenes-Camara2" respectivamente, lo mismo sucede con los videos, dos directorios "Videos-Camara1" y "Videos-Camara2".

La función principal del script de reubicación se basa en encontrar los archivos con extensión .jpg y moverlos a otro directorio, de la misma forma pasa para archivos de formato mp4.

Con el comando `find [directorio]` se encuentra todos los archivos mientras que con `exec [comando a ejecutar]` se ejecuta la acción para dichos archivos [49]

Respaldo de recursos multimedia

Los archivos antiguos ubicados en la galería se obtienen a través del comando `find` y parámetros de ordenamiento como `sort`. La condición se basa en función al tamaño del directorio. Es decir, si el directorio de imágenes sobrepasa aproximadamente 2 MB y el de videos 100 MB los archivos serán movidos hacia el dispositivo de almacenamiento. Este script se desarrolla en base al diagrama de la Figura 2.9. En la Figura 2.43 se observa una porción de código del script descrito

Figura 2.43 Segmento de código Script "respaldo_img.sh" proceso *for*

Liberación espacio dispositivo de almacenamiento

Al igual que el script de respaldo, la liberación de espacio permite desechar recursos no necesarios en el dispositivo de almacenamiento. Se determina un valor límite de almacenamiento, si este tamaño es sobrepasado en el dispositivo empieza la eliminación de archivos antiguos hasta llegar a la referencia tanto de videos como imágenes. El desarrollo de este considera el diagrama de flujo de la Figura 2.10. En la Figura 2.44 se observa una porción de código del script descrito.

Figura 2.44 Segmento de código Script "liberación_espacio_imgs.sh"

Galería dinámica

En el presente proyecto se utiliza una plantilla del tipo *Open Source* para la galería, se descarga de la página oficial [50]**.** Una vez descargada el archivo.tar se descomprime en el directorio principal de Apache2 con el comando `sudo tar -xvf nombre_archivo.tar`. La galería *"foliogallery"* está conformada por un archivo.php y los directorios donde se almacenan las imágenes y videos. Además, está provista de archivos de configuración para poder manipular parámetros como por ejemplo colores, tamaños de los objetos, entre otros y hacerla más agradable para el usuario. La Figura 2.45 ilustra la galería desde un navegador Web

Figura 2.45 Visualización de recursos multimedia a través de un navegador

Complementando el subsistema de vigilancia se configura los dispositivos para el proceso de apertura de puerta considerado este módulo.

Apertura de Puerta Principal y Envío de notificaciones push, SMS

Configuración e Instalación sensor de contacto.

La instalación y funcionamiento del sensor de contacto considera un script desarrollado en Python, además de parámetros provistos por *Firebase* para cumplir con el requerimiento de envió de notificaciones al dispositivo móvil una vez instalada la aplicación. A continuación, se describe la implementación del proceso.

- **Conexión sensor pines GPIO**

El sensor de contacto posee dos terminales imantados precableados de tal forma que un terminal se conecta en el pin 22 y el otro terminal a GND. Dichos pines se configuran desde un script en Python. Para su desarrollo se utiliza el IDLE Python 2.7 preinstalado internamente en Raspberry.

- **Librería GPIO, Uso de pines GPIO**

La librería RPi.GPIO brinda un fácil acceso y control de los pines de propósito general de Raspberry Pi. Se instala el paquete de *Python Development Toolkit,* mediante el comando: `sudo apt-get install python-dev python-rpi.gpio`.

Los pines se pueden declarar de dos maneras: considerando su numeración exterior o con la provista por el chip BROADCOM, de este modo la configuración parte con los siguientes comandos: `gpio.setmode(gpio.BOARD)` para numeración exterior y `gpio.setmode(gpio.BCM)` en el caso de numeración del chip[51] . La Figura 2.46 ilustra la configuración de un pin en el modo BCM.

```
#Declarar GPIO en el modo BCM
io.setmode(io.BCM)

# Ingresar el numero del pin de conexion del sensor
door_pin = 22
```

Figura 2.46 Configuración pines GPIO con respectivo número de pin

INCLUSION DE FIREBASE EN PYTHON

Firebase, un servicio de Google que permite trabajar juntamente con Python proporciona facilidades para realizar procesos que abarquen el uso de sus productos, en este caso el

envío de notificaciones a la aplicación móvil.

- **FCM (*Firebase Cloud Messaging*)**

Firebase Cloud Messaging es una pequeña solución de mensajería multiplataforma que permite enviar mensajes de forma segura y gratuita [52]**.** Se utiliza FCM de tal forma que se puede enviar mensajes a un dispositivo o a un grupo de dispositivos. Estos mensajes incluyen *keys* del tipo JSON las cuales son interpretadas por el sistema operativo del dispositivo móvil. La creación de un proyecto en *Firebase* se detalla en el ANEXO VII.

PyFCM es un cliente de Python que trabaja junto con el *Firebase Cloud Messaging* para proporcionar funciones de envío de notificaciones, se instala con el comando**:** `pip install pyfcm`.

Una vez obtenido y almacenado en la base de datos el *device registration_id* al instalar la aplicación en el dispositivo móvil y considerando el *key* del proyecto de *Firebase* se procede con el desarrollo del script encargado de enviar las notificaciones *push* denominado "PushNotification.py" el cual se describe a continuación y se basa en el diagrama de la Figura 2.13.

Envío notificaciones push

El script mencionado anteriormente "pushNotification.py" realiza el envío de notificaciones al dispositivo móvil de esta forma se notifica al usuario sobre un determinado suceso. El desarrollo del script inicia agregando las librerías y el cliente del paquete PyFCM [53]. Posteriormente se ingresa el parámetro necesario para el envío de notificaciones que es *FCMNotification* el cual posee como argumento el **`"api-key"`** del proyecto de *Firebase* con la siguiente sintaxis: `variable=FCMNotification(api_key=" ")`, por otro lado, la sintaxis para los registration_ids está determinada por: `registration_ids=["registration_id1","registtration_id2 ",….]`.

El título, el mensaje, la hora, necesariamente deben ir dentro de etiquetas `message_title` y `message_body` para ser reconocidos por PyFCM.

La Figura 2.47 ilustra una porción de código donde se aprecia las opciones que caracterizan el envío de notificaciones.

Figura 2.47 Segmento de código pushNotification.py

Finalmente se activa el servicio para varios dispositivos con la línea `push_service.notify_multiple_devices["parámetros necesarios"]`, utilizando criterios como `regitration_ids, el message body, message_title,` además de la intensidad de vibración al llegar la notificación.

Envío notificaciones SMS

Para el envío de notificaciones SMS descrito en la sección de diseño previamente se conecta y configura el modem 3G, este proceso se detalla a continuación.

Configuración Modem 3G[54]

En el caso de existir una activación del sensor de contacto ubicado en la puerta y considerando algún problema con la red o que el usuario no disponga de una conexión estable a Internet se requiere de un medio que informe sobre dicho suceso. Por lo tanto, se opta utilizar un modem 3G para solventar esta necesidad a través del envío de un mensaje de texto hacia el Smartphone del usuario.

El controlador “sakis3g” actúa como un interfaz entre Raspbian y el modem 3G para su configuración, se instala mediante el comando `sudo wget http://raspberry-at-home.com/files/sakis3g.tar.gz` y el paquete se descomprime mediante `tar -xvzf sakis3g.tar.gz` , finalmente se agrega permisos con el comando `chmod +x sakis3g` y de esta forma está listo para ejecutarse.

- **Configuración de parámetros**

El funcionamiento del modem USB considera el controlador sakis3g en su modo interactivo al que se ingresa mediante el comando `sudo ./sakis3g –interactive “connect”` desplegando una ventana de configuraciones como se ilustra en la Figura 2.48.

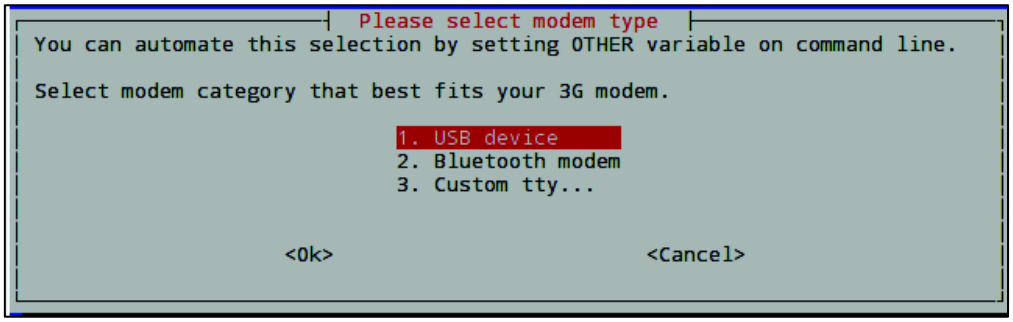

Figura 2.48 Modo interactivo sakis3g

En la ventana de configuración se selecciona la categoría del modem "`USB Device`", seguido del tipo de dispositivo que se encuentra conectado con la opción "`HUAWEI Mobile`".

El prototipo consta de un modem HUAWEI E173, se crea un APN personalizado para utilizar los parámetros de movistar como se ilustra en la Figura 2.49. Tanto el usuario como la contraseña se definen como "movistar" y el APN "internet.movistar.com.ec" los cuales se ingresa por teclado.

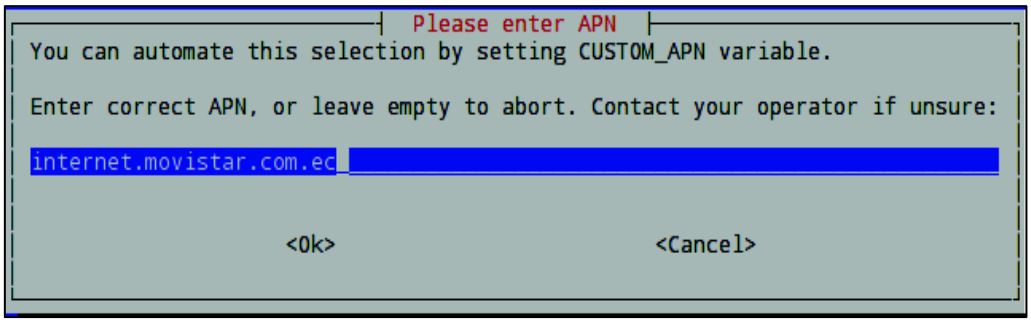

Figura 2.49 Creación de APN personalizado de Movistar

Finalmente, para el envío de SMS se utiliza comandos AT, los cuales son instrucciones simples reconocidas por el modem para enviar un SMS a través de un enlace serial. Se verifica si el modem reconoce estos comandos mediante el paquete "cu" y el comando `cu -l [nombre asignado al modem]`, de esta forma se ingresa en un terminal donde se digita los comandos AT que se detallan a continuación.

AT: Este comando es utilizado para comprobar la conexión del modem, si la respuesta a este comando es OK significa que la conexión del modem es correcta.

AT+CMGF=1: Comando utilizado para comprobar que el modem opere en el modo de envío SMS, si el resultado es OK indica que el comando fue exitoso, si el resultado es

ERROR el modem no soporta el modo de operar para enviar SMS.

AT+CMGS [número telefónico]: Comando utilizado para enviar el SMS, se ingresa el número telefónico y posteriormente el mensaje a enviar, una vez escrito el mensaje se presiona crtl+z para terminar con el envío. [55]

Una vez configurado el modem se desarrolla un script destinado al envío de un SMS dentro del proceso de apertura de puerta, el cual se basa en el diagrama de flujo de la Figura 2.14. En la Figura 2.50 se muestra el código del script para enviar el sms.

envioSMS_AT.py

```
import serial
import time
from curses import ascii

archivo=open("/home/pi/scripts/BaseDatos/base_datos_numeros.txt","r") #abrir el arch
fo=archivo.readlines()
archivo.close
print len(fo)

tiempo=time.strftime("El dia %x a la hora %X")

for i in range(len(fo)):
        fl=str(fo[i])
        fl=fl.lstrip('0')
        var="+593"+fl
        print var

        modem=serial.Serial('/dev/ttyUSB0',115200,timeout=5)

        try:
                modem.write("AT\r\n")
                print modem.readline()
                modem.write("AT+CMGF=1\r\n")
                print modem.readline()
                modem.write("AT+CMGS=\""+var+"\"\r\n");
                print modem.readline()
                modem.write("ALERTA!! Puerta Abierta: "+tiempo+ascii.ctrl('z'))
                time.sleep(5)
        except ValueError:
                print "Error de envio notificacion SMS"
```

Figura 2.50 Script para enviar notificación SMS

Finalmente, para complementar, el proceso de apertura de puerta principal y envío de notificaciones push, SMS se desarrolla un script basado en la Figura 2.12 mediante el cual se controla el sensor de contacto ubicado en la puerta una vez conectado a la Raspberry Pi.

El desarrollo del script empieza con la inclusión de la librería para uso de GPIO, mediante la línea `import RPi.GPIO as io`, seguido de la declaración del pin GPIO con su numeración BCM. Además, se obtiene el ID del proceso que se genera al ejecutar este script mediante el módulo `"os.getpid"` y se almacena en un archivo de texto que posteriormente será utilizado.

Mediante el proceso de iteraciones *While* se ejecuta el lazo solamente si la declaración

es *"True",* se considera un comparador *"if"* para determinar el estado del sensor (abierto o cerrado). Si el sensor se encuentra abierto se muestra en el shell de Python un mensaje donde indica la hora y el día en que la puerta fue abierta, seguido de la ejecución mediante el comando `commands.getoutput("python /directorio del script")`, también de los subprocesos de envío de notificaciones push y SMS a cargo de los scripts `"pushNotification.py" y "envioSms.sh"` respectivamente. En el caso contrario si el sensor está cerrado se muestra un mensaje `"Door Close"`. El código completo se lo puede apreciar en el ANEXO V.

IMPLEMENTACIÓN DEL MÓDULO LECTOR CÓDIGO QR

En esta sección se describe el proceso de implementación del módulo Lector de Código QR. Este módulo considera dos procesos o fases, la primera se encarga de la generación de un código QR en base a información de usuario, por otro lado, la segunda fase abarca la lectura de este código QR mediante una cámara USB dando como resultado la apertura de la cerradura magnética.

Generación código QR

La información que será almacenada en el código QR propuesto se forma a partir de una palabra base, a la cual se le agrega la hora de creación y el nombre del usuario.

Para la generación de un código QR mediante python es necesario la instalación de dos paquetes: `pycodeqr y pypng`, que permiten almacenar la información a ser codificada en una variable, añadir la información complementaria y generar el código, posteriormente se obtiene la imagen del código de acuerdo con la estructura mencionada en la sección anterior.

Pycodeqr [56]**.** Paquete para python que genera un código QR en base a la información ingresada. Dispone de una configuración abierta de opciones tales como, nivel de corrección de errores, versión, caracteres. etc. En Raspbian se instala con el comando `pip install pyqrcode` y su forma de uso se detalla en la Figura 2.51 donde se muestra un segmento de código del script de generación y como resultado un código QR versión 15.

Figura 2.51 Ejemplo de generación de código QR

Pypng. Es un paquete de Python que permite que los archivos con extensión .png puedan ser leídos y utilizados por este lenguaje de programación. Descargar el archivo pypng.tar.gz de la página de módulos para Python y descomprimirlo en la Raspberry Pi. Instalar mediante el comando sudo `python setup.py`, archivo incluido en el paquete.

La creación de scripts para generar códigos QR se basa en la Figura 2.16. Se agrega los módulos pyqrcode y pypng, seguido de la lectura de información de Base_Datos_Admin.txt para obtener el nombre de usuario, además se declara una palabra base y se configura la obtención del día/hora.

Mediante el comando `pyqrcode.create( información, opciones)` se obtiene un símbolo QR con los datos de usuario, hora y palabra base. Este comando permite el ingreso de la corrección de errores, la versión del código, el tamaño, entre otros. El código del script se muestra en el ANEXO VI.

Lectura código QR y Apertura cerradura magnética

La implementación de lector de código QR se basa en la conexión de una cámara USB y un sensor de proximidad, de esta forma cuando el sensor se activa se ejecuta el script lectorqr.py que se desarrolla en base al diagrama de la Figura 2. 18 dando lugar a dos opciones, una validación exitosa o una fallida.

Conexión cámara USB [57]

Una vez conectada la cámara USB se instala el paquete fswebcam a través del comando `sudo apt-get install fswebcam` el cual permite capturar imágenes de diferentes fuentes (cámaras) y guardarlas como archivo .png o jpg. La captura de imágenes se realiza mediante `fswebcam nombre_imagen.jpg`.

Desarrollo Script lectorqr.py

Se agrega el módulo para la lectura de códigos de barras y QR que es *zbarlight* [58] para

la decodificación del símbolo mediante el código `zbarlight.scan_codes('nombre de la imagen', qr)`, dicha información se almacena en una variable para su posterior comparación. Además, con la incorporación de LEDS se utiliza nuevamente la librería GPIO.

El proceso de validación empieza con la comparación del valor obtenido de *zbarlight* y la información de generación, de esta forma se imprime un mensaje en pantalla dependiendo del caso, si la validación es exitosa se procede a realizar el subproceso de apertura de cerradura magnética y acceso a la sala. La Figura 2.52 muestra un segmento de código donde se realiza la validación todo el código se muestra en el ANEXO VI.

```
#aqui toca leer el txt
    fa = open('codigoValidacion.txt', 'r')
    codigoValidacion= fa.read()
    print 'Codigo para validacion:'
    print codigoValidacion
    if(codes==codigoValidacion):
        print 'Validacion Exitosa Iniciar Proceso Apertura Puerta'
    else:
        print 'Validacion Fallida'
```

Figura 2.52 Segmento de código para realizar la Validación de símbolo QR

Conexión sensor de proximidad infrarrojo

Conectado a los pines GPIO (18) y manipulado mediante un script de python permite ejecutar el subproceso de lectura a cargo del script `lectorqr.py`. Proceso que se ejecuta con el módulo `os.system` en el momento que detecte un objeto acercándose.

Conexión sistema de alimentación alterna

De acuerdo a la tabla presentada en el ANEXO II el sistema de alimentación alterna se puede conectar de dos maneras mediante el cable micro USB o por medio de los pines GPIO, una vez conectado, si la Raspberry Pi por cualquier motivo quede sin energía actúa el sistema teniendo en cuenta un retraso aproximado de uno pocos milisegundos en retornar las funciones del prototipo.

2.6.3 Implementación subsitema de monitoreo

El subsistema de monitoreo permite encender y apagar de forma remota los PCs de la sala para lo cual se debe instalar diversos paquetes en la Raspberry Pi, realizar configuraciones previas en los computadores del laboratorio y configurar el servidor donde se ejecutarán los scripts.

Configuración de computadores

En esta sección se describe la configuración realizada en los computadores para solventar el requerimiento de encendido y apagado remotos.

- **Encendido Remoto**

El estándar *Wake On LAN* [59] permite que un computador se active o encienda de forma remota mediante una conexión a Internet y la previa activación en el BIOS, este proceso se detalla a continuación. Se ingresa a la configuración BIOS del computador durante el arranque al presionar varias veces F2, posteriormente se selecciona la opción consumo de energía y se habilita la opción *Wake On Lan*, Finalmente se guarda los cambios realizados y se sale de la pantalla de configuración.

Una vez activado el estándar para el encendido remoto de PC (WoL) en el BIOS, se realiza la activación del *magic packet* en el adaptador de red de la siguiente manera. Dirigirse al Administrador de Dispositivos de Windows e ingresar en la sección *Adaptadores de Red*, una vez dentro se selecciona las propiedades del adaptador y se habilita la opción "Activar Magic Packet", adicionalmente se deshabilita el *Inicio Rápido* del PC en *Opciones de Energía.*

En la Figura 2.53 se observa la habilitación del *magic packet* en la tarjeta de red de un computador, el proceso completo de configuración del encendido remoto de un pc se detalla en el ANEXO VIII.

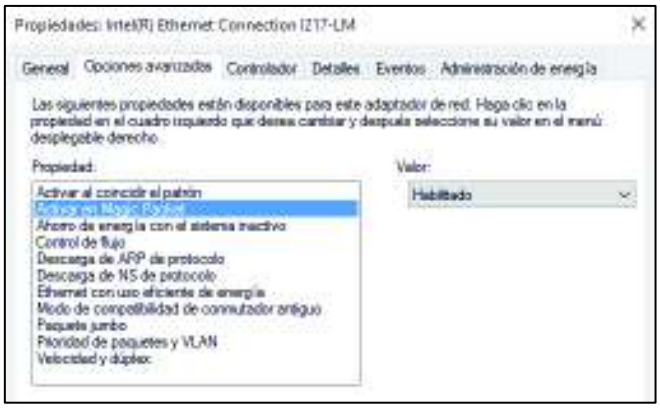

Figura 2.53 Habilitación *magic packet* en el adaptador de red

- **Apagado Remoto**

Para el apagado remoto se configura en el **Editor de Registro** el *UAC (User Account*

Control), mientras que en **Panel de Control** se configura los derechos de Usuario para evitar problemas de acceso. A continuación, se detalla el procedimiento de ambos casos.

Para evitar errores de Acceso [60] al intentar apagar remotamente un computador se ingresa a Panel de Control, dentro del directorio ***Herramientas Administrativas*** ingresamos a ***Directiva de Seguridad Local*.** Navegando en el interior de este directorio se selecciona la carpeta Directivas Locales, finalmente en la Asiganción de derechos de usuario se elige la Opción *"Forzar apagado desde un sistema remoto".*

De este modo se evita problemas de acceso al tratar de apagar el computador de forma remota.

La configuración del UAC se realiza en el **Editor de Registro** al cual se ingresa mediante la combinación de teclas `Windows+R` y con la palabra `regedit`, seguido se explora los directorios hasta encontrar "`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System`" en el cual se crea un D_WORD de 32 bits con el nombre "*LocalAccountTokenFilterPolicy"* y con un valor de 1.

En este punto ya se puede realizar diversas tareas de administración remota, tomando en cuenta que se debe tener una conexión a Internet estable y configurar el Firewall para poder acceder. Se deben crear dos reglas que habiliten los puertos necesarios para trabajar con el programa samba de la Raspberry Pi. La primera regla será nombrada SAMBA_TCP la cual permite conexiones entrantes hacia los puertos TCP: 139 y 445. La segunda regla será denominada SAMBA_UDP la cual permite conexiones entrantes hacia los puertos UDP: 137 y 138.

El proceso completo de configuración del apagado remoto se detalla en el ANEXO VIII.

Instalación y Configuración de paquetes

- **Configuración Apache2 para ejecutar scripts de Python** [61]

Posterior a la instalación del servidor apache se trabaja en los archivos de configuración en este caso para la ejecución de scripts, se manipula el archivo `serve-cgi-bin.conf` ubicado en el directorio `/etc/apache2/conf-enabled/`. La Figura 2.54 muestra la línea de código que se agrega dentro de la opción Directory, habilitando de esta forma la ejecución de scripts de Python desde el servidor.

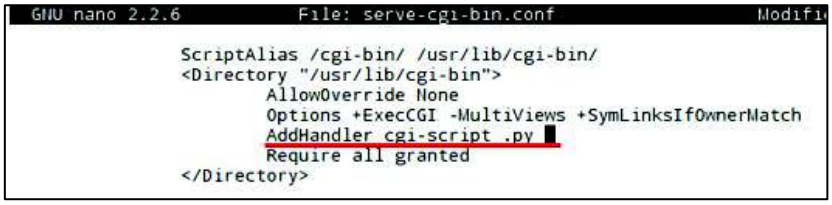

Figura 2.54 Archivo de configuración directorio */usr/lib/cgi-bin*

Se desarrolla un script de prueba al cual se le da permisos con el siguiente comando `sudo chmod +x /usr/lib/cgi.bin/prueba.py`, de esta forma el script se torna ejecutable. Los scripts deben ubicarse en el mismo directorio del archivo de configuración siendo este /`usr/lib/cgi-bin`. Finalmente se reinicia el servicio apache2 mediante `sudo service apache2 restart`.

- **Instalación de wakeonlan**

Es un paquete que permite encender remotamente un computador mediante línea de comandos o un script de Python, se instala con el comando `sudo apt-get install wakeonlan`. Para trabajar en Python se importa la librería del paquete mientras que para la acción de envío del paquete mágico a la MAC se utiliza el comando `wol.send_magic_packet(´MAC ADDRESS´)`, de esta forma se enciende un computador o varios.

- **Instalación de samba-common**

El uso de este paquete en este proyecto se realiza solamente para el reconocimiento de ciertos comandos como net, rpc, entre otros en el uso del apagado remoto más no para que trabaje como un servidor. Mediante el comando `sudo apt-get install samba-common` se instala el paquete descrito.

Desarrollo de scripts

Considerando los diagramas presentados anteriormente en la Figura 2.20, Figura 2.21,y Figura 2.22 se describe el proceso de implementación.

Encendido Remoto de Computador

Este script se desarrolló en base al diagrama de la Figura 2.20 , se agrega los módulos tanto de wakeonlan como el cgi para ejecución de scripts,, el comando que permite el envío del *magic packet* tiene la siguiente sintaxis `wol.send_magic_packet(´Dirección MAC´)`. Un mensaje en pantalla es una

opción para demostrar que el script funciona correctamente por lo tanto se utiliza "`print`" para mostrar un texto de confirmación en el navegador. La Figura 2.55 ilustra el segmento de código para encender un computador.

```
#!/usr/bin/python
from awake import wol  #Importar el modulo de wakeonlan
import cgi      # Importando modulo de soporte para scripts CGI
import cgitb    #Se ingresa parametros si desea visualizar los errores
cgitb.enable()
wol.send_magic_packet('64:00:6A:8C:B4:38') # Encender un computador con su MAC
print 'Content-type: text/html\n\n'
print '<h1>ENCENDIDO EXITOSO</h1>'# Mostrar un mensaje en pantalla
```

Figura 2.55 Código script wake27.py

Apagado Remoto de Computador

En el script se dispone de una nueva librería que permite la ejecución de comandos de linux en Python mediante el comando `commands.getoutput("comando del intérprete de Linux")`. El comando `net rpc shutdown` considera parámetros como el nombre de usuario y contraseña del computador además de la dirección IP. Ejecutado el script muestra un mensaje en la pantalla del computador y se apaga. En la Figura 2.56 se aprecia un segmento de código del script de apagado de un computador. Cabe mencionar que el script de apagado se creó en base al diagrama de la Figura 2.21

```
#!/usr/bin/python
import cgi     # Importar modulo de soporte para scripts CGI
import cgitb
import commands # Importando el modulo commands (Para usar comandos bash)
cgitb.enable() # Se ingresa parametros si desea visualizar errores
commands.getoutput('net rpc shutdown -U Intercon15%labcon2017 -I 172.31.45.17')
# Utilizacion modulo commands para ejecutar un comando Bash,
#net rpc shutdown permite el apagado del computador mediante la direccion IP asignada
print 'Content-type: text/html\n\n'
print '<h1>APAGADO EXITOSO</h1>' # Mostrar mensaje en pantalla
```

Figura 2.56 Script "shut17.py" permite apagar el computador con IP termina en .17

Obtención información sobre estado del PC

Es un script que permite determinar el estado de los computadores del laboratorio en base a su ejecución continua, crea un archivo de texto donde se almacenan los estados. Utiliza el comando *"ping"* propio de Linux. El resultado obtenido "activado" o "desactivado" se almacena en un archivo de texto para posteriormente ser utilizado desde el servidor apache y en la aplicación. Dependido del estado de los computadores el proceso de verificación se demora aproximadamente 45s si todos están apagados y 10s si todos están encendidos. La Figura 2.57 ilustra el código del script de monitoreo de

estados.

```
#!/usr/bin/python
import os
import cgi
import cgitb
cgitb.enable()

os.system('sudo rm /var/www/estadoPCs.txt')
archivo = ""
#print '<h1>ARCHIVO DE ESTADOS CREADO</h1>' # Mostrar mensaje en pantalla

for i in range(2,30):
    hostname = "172.31.45." + str(i) #Declarar las IPs de los computadores
    response = os.system("ping -c 1 -W 1 "+ hostname)# Comando ping
    if response == 0: #Compraracion
        print hostname + " Encendido"
        archivo+="Encendido " # Almacenamiento en el archivo de texto

    else:
        print hostname + " Apagado"

        archivo+="Apagado "

archivo2= archivo.rstrip() #Eliminaci[on de carateres
text_file = open('estadoPCs.txt', 'w')
text_file.write(archivo2)  #escritura en el archivo de texto
text_file.close()
os.system("sudo mv estadoPCs.txt /var/www/")
```

Figura 2.57 Script “estadoPCs.py”

Dado que la interacción Usuario-prototipo se lleva cabo mediante una interfaz que monitorea las funciones del sistema a continuación se detalla la implementación de la aplicación móvil considerando el diseño previo.

2.6.4 Desarrollo de aplicación movil

En esta sección se describe el proceso de desarrollo de la aplicación móvil partiendo de la instalación de programas y paquetes complementarios.

De forma general la utilización de *Ionic Framework* para el desarrollo de aplicaciones móviles se basa en módulos mas no en clases, además con la compatibilidad de Firebase se vuelve un Framework de uso cotidiano para los desarrolladores.

Instalación de recursos necesarios

El desarrollo de la aplicación móvil con Ionic empieza con la instalación y configuración de algunas herramientas necesarias para su operación, por ejemplo, para la gestión de paquetes necesarios para el desarrollo con Javascript (Node.js), para el uso de funciones propias del smartphone (Apache Cordova), para editar código con las facilidades necesarias (Visual Code Studio), para testear las apk (Android studio SDK), entre otros.

Nodejs. Es un ambiente de ejecución de JavaScript, trabaja conjuntamente con npm que

es el administrador de paquetes de Node.js, es decir mediante npm se instala los paquetes. Para su instalación existen dos formas: mediante un gestor de paquetes (usuarios sistemas Linux) o el método universal para diferentes usuarios (Linux,Windows), mediante la descarga del sitio oficial [62].

Apache Cordova: Permite la creación de proyectos (Apps) y construirlas para diversas plataformas (Android, iOS), de esta forma se puedan emular o reproducir en dispositivos reales. Su instalación [63] se realiza mediante manejo de paquetes en base a los siguientes pasos:

1. Instalar Node.js y comprobar el funcionamiento de node y npm en el CLI.
2. Descargar e instalar un cliente git, para que el CLI pueda usarlo de fondo en la descarga de algunos paquetes.
3. Instalar el paquete de cordova mediante el manejador de paquetes de Node.js con el siguiente comando `npm install -g cordova`

Visual Code Studio. Editor de código provisto para desarrolladores, su agradable presentación en base a colores y complementación de código facilita el desarrollo de aplicaciones híbridas o páginas web. El proceso de instalación se detalla a continuación:

1. Descargar el instalador para el sistema operativo deseado del enlace provisto https://code.visualstudio.com/download
2. Ejecutar el .exe descargado para iniciar la instalación.
3. Seguir los pasos de configuración e instalación.

Una vez instalados los recursos necesarios se procede con la instalación del framework en el cual se desarrollará la aplicación móvil híbrida denominado IONIC.

El software detallado para el desarrollo de la aplicación móvil se instala en un computador que posea el sistema operativo Windows, se utiliza Node.js como un gestor de paquetes para poder instalar Ionic como se detalla a continuación. Se optó por el editor de código Visual porque posee soporte para varios lenguajes, es decir al agregar nuevas librerías reconocía de forma sencilla la sintaxis de los lenguajes.

Ionic. Es un *framework* destinado al desarrollo de aplicaciones móviles hibridas, es *open source*, provee componentes y herramientas basadas en HTML, JS y CSS. Su proceso de instalación [64] y uso se describe a continuación:

1. Utilizar el manejador de paquetes de Node.js e instalar Ionic mediante el comando

```
npm install -g ionic
```

Finalmente, para comprobar el funcionamiento de los recursos instalados se crea una aplicación móvil básica de la siguiente manera:

1. Crear un directorio y una app en blanco provista por *Ionic* digitando el siguiente comando `ionic start [directorio] blank –type [nombre de la app]`
2. La edición de los archivos creados se la realiza en Visual Code Studio, trabajando con HTML, JS y CSS
3. Añadir las plataformas (Android, iOS) para las cuales se desarrolla la aplicación: `ionic cordova platform add ios` y `ionic cordova platform add android`. En este caso se utiliza como base cordova anteriormente instalado.
4. Probar la app en un emulador, en la propia máquina `ionic cordova emulate android`, o en un dispositivo real i`onic cordova build android` (este último utiliza las herramientas del Android SDK previamente instalado para formar el apk). Otra forma de probar la app es mediante `ionic serve` el cual nos permite emularla en un navegador web, tomando en cuenta que no se podrán comprobar todas las funcionalidades debido a que no se tiene todos los complementos necesarios en el navegador.

Desarrollo de Interfaces

Cada una de las interfaces de la aplicación móvil se crean en base a 3 archivos, un "*archivo .html*", un "*archivo .ts*" y por último un "*archivo. scss".* Esta estructura de archivos se genera una vez creada una aplicación en blanco mediante Ionic, además se utiliza componentes y funciones propias de este *framework.*

- **CREACION DE UNA APP EN BLANCO [65]**

El editor de código instalado anteriormente (Visual Studio Code) permite utilizar un terminal de Node.js, mediante `ionic start –type blank` se crea una aplicación basada en una plantilla de *Ionic*, consiste en un directorio provisto de todos los archivos necesarios para empezar a desarrollar la app.

Una vez creada la plantilla de la aplicación móvil se genera una a una las páginas con el comando `ionic generate page [nombre de la página]`, se añade componentes como botones, pies de página, íconos, entre otros, los cuales pueden

realizar funciones como por ejemplo abrir un enlace, redirigir a otra interfaz, cambiar de color, etc. El funcionamiento de cada componente se comprueba en un navegador mediante `ionic serve`, teniendo la posibilidad de observar los cambios del producto final cada vez que se realicen cambios en el código. Dichos comandos se ingresan en el *comand prompt* de Node.js

En la Figura 2.58 se puede observar los directorios y archivos que se crean con cada proyecto de Ionic, además de la visualización de una interfaz en el navegador web una vez ejecutado el comando ionic serve.

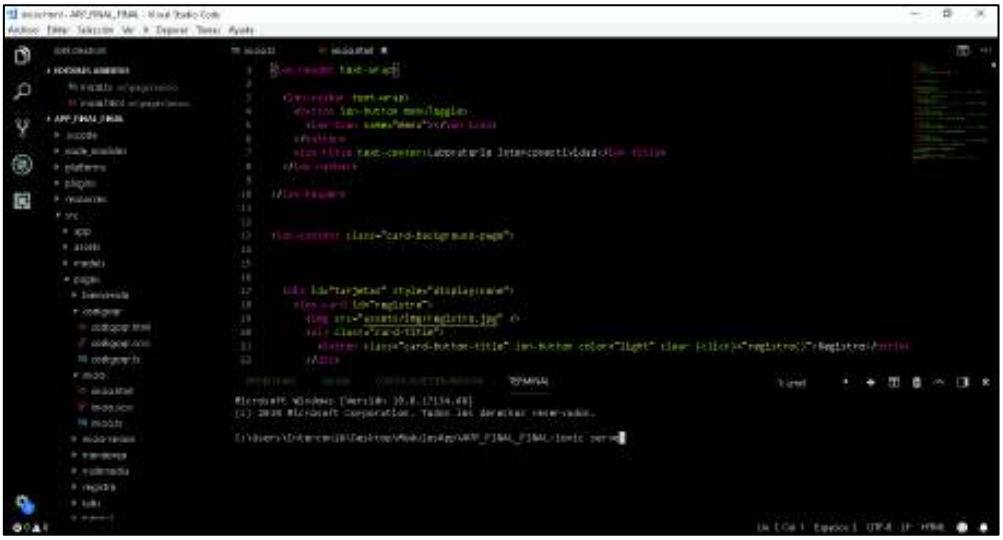

Figura 2.58 Directorios generados al crear una app en Visual Studio Code

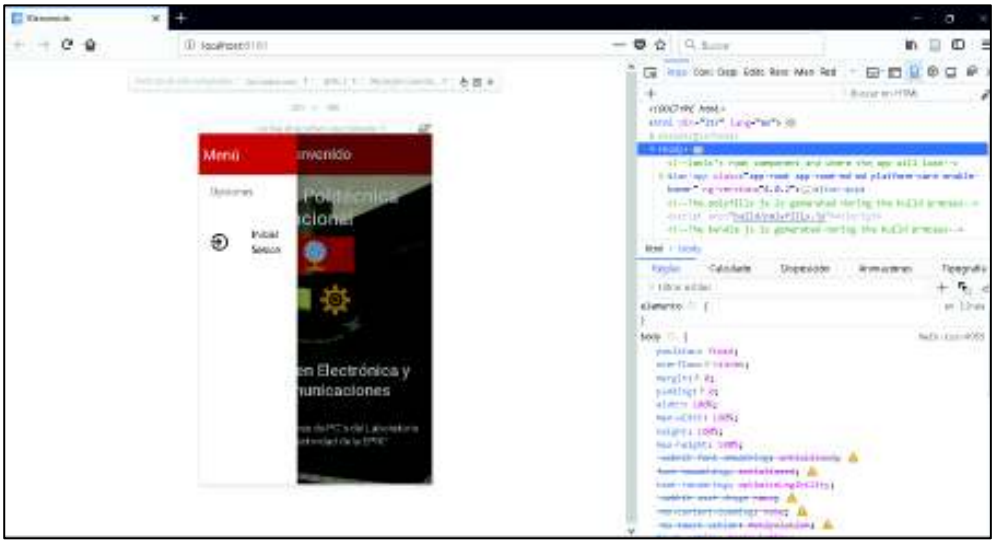

Figura 2. 59 Simulación de la aplicación en navegador mediante comando ionic serve

- **INTERFAZ *BIENVENIDA***

De acuerdo con el diseño de la interfaz descrita en la Figura 2.27, son necesarios componentes de *Ionic* como botones, imágenes y texto [66]**.** Dichos componentes se los encuentra en la documentación propia de Ionic los cuales se describen a continuación.

Buttons. Un botón de menú se define mediante el componente "`menuToggle`", la ubicación está determinada en la esquina superior izquierda de la página dentro de una cabecera. Dentro de las etiquetas `<button></button>` se ingresa el tipo de botón, `ion-button menuToggle`, de ser necesario a los botones se les puede agregar iconos de la siguiente forma `<ion-icon> name="nombre del icono"</ion-icon>`.

Text: Un texto en HTML se agrega dentro de etiquetas que determinan el formato de texto, puede ser el caso de título, subtítulo o párrafo. Por ejemplo `<h1>INGRESE AQUÍ SU TEXTO</h1>`, en pantalla se imprime como un título.

Imágenes: Una imagen se añade mediante la etiqueta `<img src=" [directorio de la imagen]">`, no necesita etiqueta de cierre. Para imágenes de fondo necesariamente se necesita una clase que se la configura en el archivo. scss

Clases: Una clase se define mediante `class ="Nombre"`, por ejemplo, definida una clase para un componente, la misma puede ser utilizada en otro componente del mismo tipo. Generalmente se utiliza dentro de un `<div></div>` el cual está destinado a crear secciones o agrupar contenido

La Figura 2.60 indica la interfaz bienvenida una vez terminada.

- **INTERFAZ *INICIO SESIÓN***

***Floating Label*:** Permite el ingreso de texto mediante teclado, los casos más comunes de uso es el ingreso de información para registro de usuarios y el ingreso de contraseñas en texto no visible al usuario. La incorporación de este componente en el código fuente se realiza mediante las etiquetas `<ion-label> </ion-label>` que contiene en su interior otra etiqueta denominada `<ion-input></ion-input>` la cual permite seleccionar el tipo de texto a ser ingresado.

La etiqueta `ion-input` permite el ingreso del tipo de texto, por ejemplo, el estilo `type=text`, permite al usuario ingresar texto de forma normal, sin embargo para el ingreso de información más delicada como datos personales o contraseñas se utiliza el

estilo `type=password` la cual permite ocultar el texto mediante símbolos visibles al usuario. Una vez ingresada las credenciales del usuario mediante un *button* se realiza la validación. La Figura 2.61 indica la interfaz Inicio de Sesión una vez terminada.

Figura 2.60 Interfaz Bienvenida terminada

Figura 2.61 Interfaz Inicio Sesión Terminada

- **INTERFAZ *INICIO:***

Pantalla visualizada por el usuario una vez autenticado, provista de una cabecera con el botón de menú y un pie de página con un icono. Contiene *button cards* con una pequeña explicación y que dirige hacia páginas como vigilancia, monitoreo, mediante la pulsación de estas.

Header: Es una región al inicio de la pantalla que puede contener un título, botones de navegación ya sea en el lado izquierdo o derecho y puede ser configurada con varios colores**.** Su utilización se basa en etiquetas de la siguiente forma: `<ion-header><ion-title>TITULO DE LA CABECERA</ion-title></ion-header>`

Footer: Región ubicada en la parte inferior de la pantalla, al igual que la cabecera dispone de botones, un texto, adicionalmente se pude agregar iconos en el botón. Dispone los mismos parámetros de configuración de colores que la cabecera. Se usa mediante etiquetas**:** `<ion-footer><ion-title>TITULO DEL PIE DE PÁGINA</ion-title></ion-footer>`

Cards*:* Componente de Ionic donde se puede visualizar imágenes, agregar texto e incluso un button que realice una función. La sintaxis para agregar una *card* en una página es la siguiente: `<ion-card><ion-card-content>Texto a mostrarse en la card</ion-card-content></ion-card>`**.** Es decir, dentro de la etiqueta `<ion-card>` se puede tener funciones como `<img>`, `<div>`, `<button>`, considerando el requerimiento a solventar.

Iconos: Son imágenes que tienen como objetivo un objeto, se agregan en diferentes componentes de Ionic como cabeceras, pie de página, entre otros, siendo el más común en el interior de un botón para indicar una acción. Para añadir un icono se emplea `<ion-icon> name="nombre del icono"</ion-icon>`.

La Figura 2.62 indica la interfaz Inicio una vez terminada.

- **INTERFAZ *VIGILANCIA*:**

Interfaz diseñada para la visualización de la sala a través de video producido por las cámaras IP. Mediante el parámetro `<img src="directorio de la imagen">`, se obtiene la imagen en un recuadro. Esto se logra mediante el ingreso de la dirección IP de la Raspberry Pi y el puerto configurado.

Image viewer: Componente destinado a mejorar la visualización de imágenes dentro de una galería creada con Ionic, de tal forma que se pueda realizar un zoom. Se instala en base al comando `npm install –save ionic-img-viewer` y se lo utiliza colocando la palabra "`imageViewer`" justo después del directorio de la imagen: `<img src="directorio de la imagen" imageViewer>`. La Figura 2. 63 indica la interfaz Inicio una vez terminada.

Figura 2.62 Interfaz Inicio terminada

Figura 2. 63 Interfaz Vigilancia terminada

- **INTERFAZ *MULTIMEDIA*:**

Interfaz diseñada para la verificación visual de sucesos de alarma en el interior del laboratorio fuera de un horario laboral. Consta de una galería en la cual se aprecia imágenes y videos. A continuación, se describe los elementos y componentes a utilizar en la implementación de esta interfaz.

Galería dinámica: Se utiliza una Plantilla Open Source denominada *folliogallery*, permite visualizar los recursos multimedia almacenados. Una vez descargados los archivos incluyen en el directorio principal del servidor apache configurado en Raspberry.

Object: Componente de HTML que permite la inserción de un documento HTML, permite la configuración de parámetros para hacer más agradable la visualización del archivo HTML. Se lo incrusta mediante las etiquetas `<Object> </Object>`. En la Figura 2.64 se observa la Interfaz Multimedia finalizada.

Figura 2.64 Interfaz Multimedia terminada

- **INTERFAZ *MONITOREO PC's***

Interfaz implementada para la visualización del estado de los computadores, además permite el encendido y apagado remoto de los mismos. A continuación, se describen los componentes necesarios para el desarrollo de esta interfaz.

JQuery: Es una biblioteca adicional de JavaScript que tiene como función principal simplificar la forma de interacción con componentes de HTML, además de agregar

animaciones y permitir una interacción con AJAX en los documentos HTML. En este proyecto se utiliza para obtener los valores de un archivo.txt almacenado en la Raspberry Pi para posteriormente realizar comparaciones y poder realizar un cambio de color en botones[67].

BackgroundColor: Comando utilizado para dar color de fondo a los componentes de *Ionic* en base a clases. Por ejemplo para el boton1 se declara una variable y se obtiene la clase con el comando `document.getElementById ("clase del componente")`, mientras que para darle un color de fondo se utiliza el comando `nombre_variable.style.BackgroundColor ="#000000"`. El valor del color se lo puede obtener de una paleta de colores para HTML el cual proporciona el código en dígitos hexadecimales.

Alert: Son cuadros de diálogos que se presentan al usuario con algún tipo de información, además permite la interacción con otras funciones mediante una agregación de botones.

Window.open: Permite abrir un navegador web dentro de la aplicación. La sintaxis del comando es `var ref= window.open(url , opciones)` de esta forma se ejecuta los scripts del servidor.

Refresher: Provee la función de actualizar un componente o página dentro de una aplicación desarrollada en Ionic, dentro del `<ion-content></ion-content>` se hace referencia a la etiqueta `<ion-refresher></ion-refresher>` para su posterior utilización. Parámetros como el tiempo de actualización, la figura mientras se actualiza se puede cambiar de acuerdo con la documentación de Ionic. En la Figura 2.65 se muestra la Interfaz Monitoreo PC's terminada.

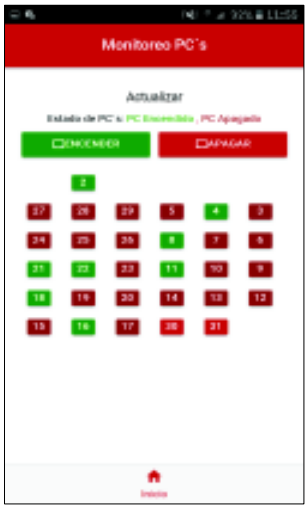

Figura 2.65 Interfaz Monitoreo Pc's terminada

- **INTERFAZ *AGREGAR USUARIOS***

Interfaz que permite que el administrador del sistema añada usuarios, a continuación, se detalla los componentes utilizados para la implementación de la misma. Se utiliza *"Floating Labels"* para el ingreso del Nombre, Correo electrónico, contraseña, celular, asignatura, para el rol de usuario se utiliza una selección de elementos de la misma forma para la selección del día. La hora de Inicio y Fin se añaden mediante el componente *DateTime* los mismos que se detallan a continuación.

DateTime. Componente utilizado para hacer más agradable al usuario la selección de fechas y horas en la aplicación, mediante la etiqueta `<ion-datetime>` se despliega una ventana de selección de horas.

Ion select. Permite al usuario seleccionar un elemento de una lista provista por una ventana que se genera mediante las etiquetas `<ion-select></ion-select>` complementada en su interior con `<ion-option> Opcion </ion-option>` donde se ingresa las distintas opciones a ser escogidas. En la Figura 2.66 se observa la interfaz Agregar Usuarios finalizada.

Figura 2.66 Interfaz Agregar Usuarios terminada

- **INTERFAZ *GENERAR CODIGO QR***

Interfaz que provee un código QR para la autenticación del usuario e ingreso al laboratorio, comprende un button que al presionarlo ejecuta un script en el controlador, espera alrededor de 3 seg. para obtener la respuesta del servidor y muestra el Código QR durante 15 seg. suficientes para realizar la validación, al mismo tiempo oculta el botón de generación para evitar problemas con las peticiones al servidor. Para mostrar la imagen en pantalla de la aplicación móvil se considera el uso de la etiqueta `<img src="directorio de la imagen">`. En Figura 2.67 se muestra la interfaz código QR finalizada.

Figura 2.67 Interfaz Generar Código QR terminada

- **INTERFAZ AYUDA**

Conformada por un conjunto de diapositivas donde se muestra la forma de uso de la aplicación móvil mediante el usuario, se utiliza el componente "slides" de Ionic a través de etiquetas <ion-slides></ion-slides>. En la Figura 2. 68 se indica la primera página de la Interfaz ayuda.

Terminado el desarrollo de las interfaces se comprueba su funcionamiento mediante un sistema de emulación para la posterior creación del archivo **.apk** e instalación en el dispositivo real.

Figura 2. 68 Primera página de Interfaz Ayuda

2.7 CONSTRUCCION DEL APK y EMULACIÓN EN AVD

Para el presente proyecto se trabaja en el terminal de comandos de Node.js se añade la opción de construir el apk para la plataforma Android mediante los siguientes comandos ingresados por teclado: `Ionic cordova platform android` y `Ionic cordova build android`. El proceso aproximadamente dura un minuto dependiendo de imágenes y módulos utilizados en la construcción de la app. Finalizado el proceso el archivo .apk se almacena en el directorio de la aplicación y se procede con la instalación en un emulador o en un dispositivo real. A continuación, se detalla la emulación de la apk en un *Android Virtual Device* AVD.

Se crea un emulador virtual en Android Studio mediante el AVD con características de un Smartphone, seguido de configuraciones en la interfaz, nombre del simulador, posición horizontal o vertical, entre otros. Una vez creado el emulador se procede a la instalación del archivo .apk siguiendo los pasos que se detallan a continuación.

Para la instalación del apk en el simulador se ingresa al directorio del SDK Manager provisto por Android Studio, seguido de copiar el archivo .apk al directorio "platform-tools". Una vez dentro mediante línea de comandos digitar `adb install nombre_archivo.apk`. Se recibe un mensaje de `Success` terminado el proceso. Finalmente se visualiza el ícono de la aplicación instalada y se procede con la ejecución.

En la Figura 2.69 se observa la aplicación en ejecución y el ícono de esta en la pantalla.

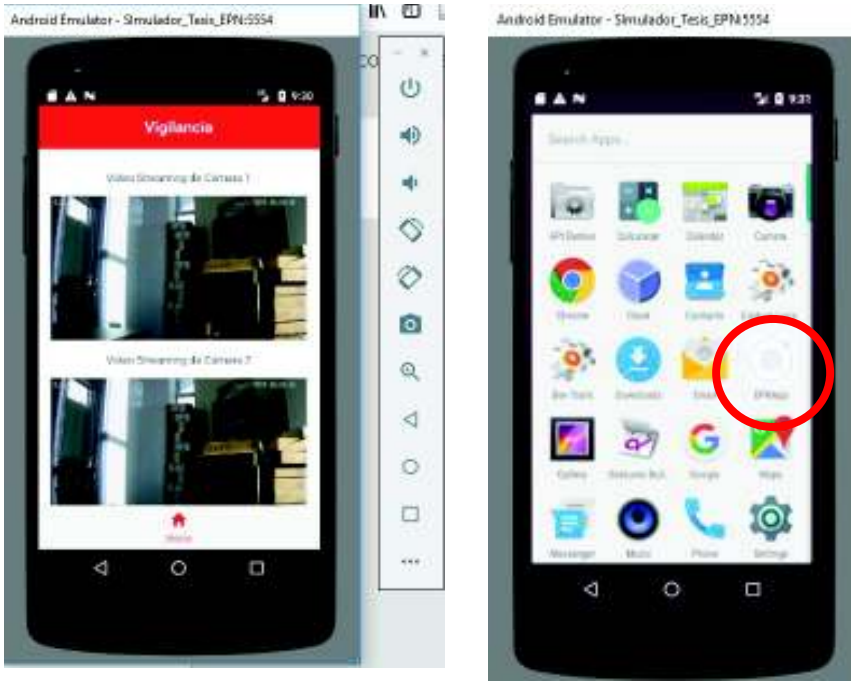

Figura 2.69 Emulador con AVD "Emulador_Tesis_EPN"

Adicionalmente al momento de instalar la aplicación en dispositivos móviles se obtiene un TOKEN conocido como `device_registration id`, el cual se lo utiliza en el proceso de envío de la "*push notification*" descrita en la sección de implementación. El proceso de obtención del Token se detalla a continuación.

2.8 Obtención REGISTRATION_ID

Para obtener el device_id (token) [68] al instalar una aplicación en un dispositivo móvil es necesario adicionar dos módulos al proyecto de Ionic, mediante línea de comandos en el terminal de Node.js se ingresa por teclado `npm install firebase angularfire2 –save` y `ionic plugin add cordova-plugin-fcm`.

Una vez creada las interfaces de la aplicación móvil se selecciona la principal para realizar las configuraciones del plugin FCM. Dentro de la función `ionViewDidLoad` la cual ejecuta procesos al ingresar a la página, se toma parte de la documentación del *plugin fcm* y se ingresa en dicho apartado. Con esta porción de código se tiene dos casos, primero en el caso que la aplicación esté cerrada llega la notificación y se puede ingresar desde la misma, segundo si la aplicación está abierta entonces al recibir el token se muestra los datos de desarrollador.

Dos funciones necesarias para la obtención y almacenamiento del token: `tokensetup` y `storetoken`.

tokensetup: Mediante el comando `FCMPlugin.getToken` se obtiene información acerca del token generado al instalar e iniciar la aplicación móvil, de ser necesario mostrar esta información se utiliza un `alert(token)`.

Storetoken: Esta función requiere del importe de `AngularFireDatabase` desde el módulo de `angularfire`, se inicia creando un nodo en la base de datos en este caso denominado `pushtokens`. Mediante el comando `"push"` se añade información adicional como el UID del usuario que se autenticó a la aplicación y el valor del token.

2.9 INSTALACIÓN FÍSICA DE DISPOSITIVOS

En esta sección se detalla el proceso de instalación de los dispositivos que conforman el prototipo además de la ubicación de los mismos en el interior de la sala.

CÁMARAS: Previstas para generar una trasmisión de video en tiempo real para los usuarios se ubican de forma opuesta en las esquinas de la sala, de esta forma la cámara EPN1 cubre 6 mesas de computadores incluyendo la parte frontal con la puerta. Por otro lado, mediante la cámara EPN2 se aprecia en conjunto las mesas de computadores, la parte posterior de la sala donde se ubican los racks, y armario de cables. La Figura 2.70 ilustra las dos cámaras en sus posiciones actuales.

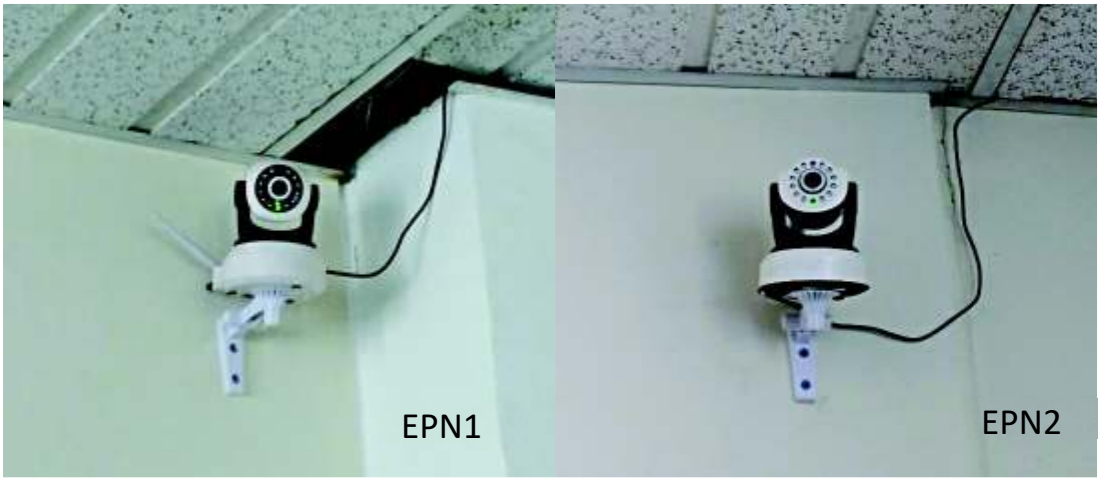

Figura 2.70 Ubicación cámaras interior de la sala

CONJUNTO SENSOR DE CONTACTO/CHAPA MAGNETICA: Tomando como elemento principal la puerta de la sala, ambos dispositivos se encuentran ubicados en la frontera entre la puerta y el marco superior donde, el sensor de contacto ubica la parte cableada en el marco y la otra simple en la puerta, de la misma forma se dispuso la chapa magnética. En la Figura 2.71 se ilustra el sensor y la chapa magnética montada y funcional. Además, para permitir la apertura de la chapa magnética desde el interior de la sala, se instala un pulsador NC junto al interruptor de iluminación.

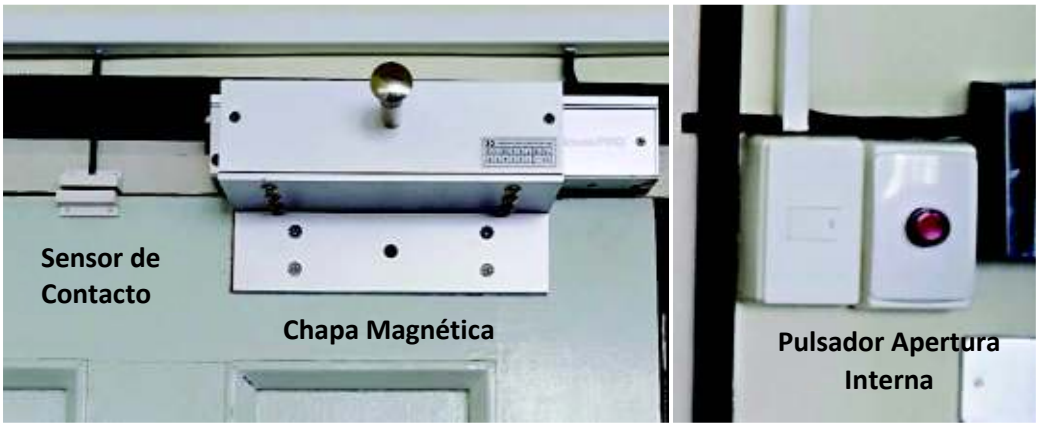

Figura 2.71 a) Ubicación del sensor de contacto y chapa magnética (b) Pulsador de apertura

LECTOR QR: Se encuentra conformado por una cámara USB, un sensor de proximidad, 3 diodos led y se ubica en la parte posterior de la puerta en una caja negra. Ahora bien, para que el usuario pueda ingresar a la sala debe mostrar el celular frente al lente de la cámara y con su dedo activar el sensor tomando así una imagen para ser decodificada y posterior a la validación, desactivar la chapa magnética. La Figura 2.72 ilustra la ubicación del lector QR fuera de la sala.

CONTROLADOR (Raspberry Pi 3): Al igual que el lector QR la Raspberry Pi se encuentra en el interior de una caja negra con todos sus componentes, *flash memory,* modem 3G, relé para la activación de la chapa, sistema de alimentación alterna y conexiones entrantes de los sensores utilizados. Está ubicada a lado derecho del pulsador de la chapa magnética como se ilustra en la Figura 2.73

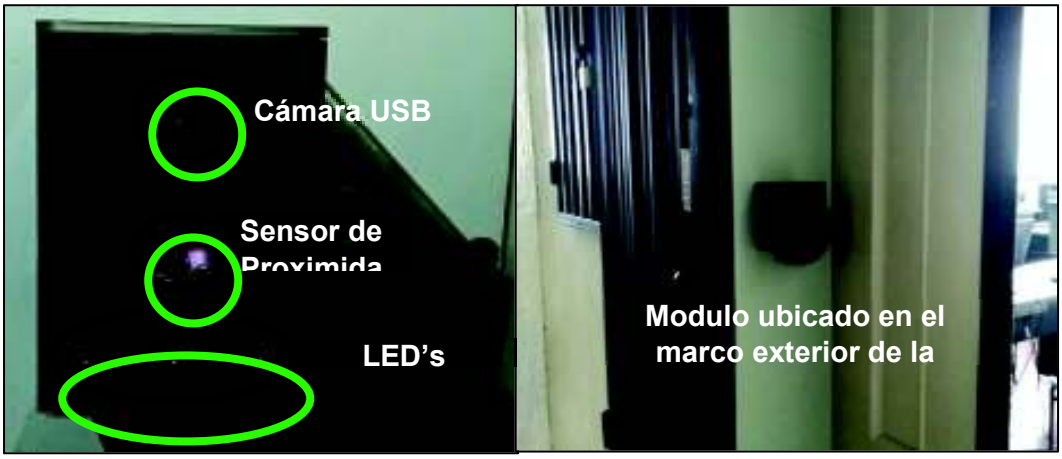

Figura 2.72 Lector QR para ingreso a la sala

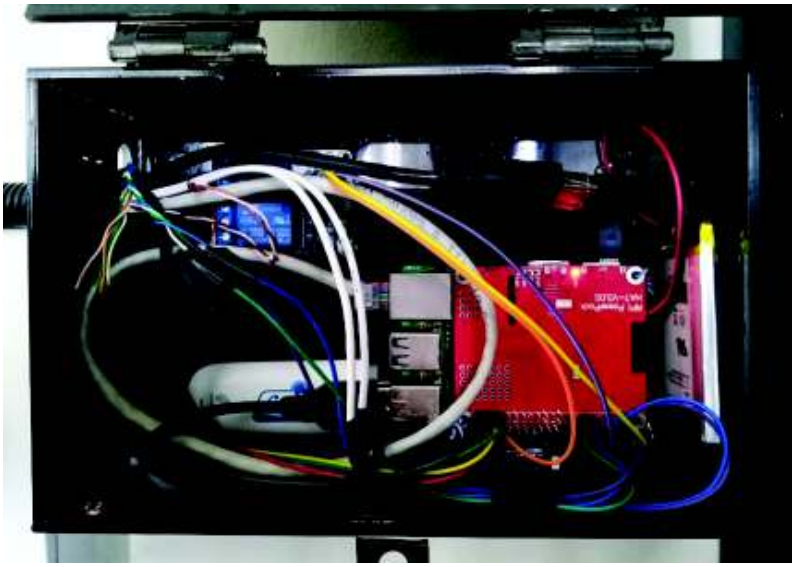

Figura 2.73 Contenedeor de Raspberry Pi y periféricos

En el ANEXO X se presenta un diagrama como guía de conexión, identificando los pines GPIO que han sido utilizados con los sensores presentados.

3. RESULTADOS Y DISCUSIÓN

En esta sección se muestra los resultados obtenidos de las pruebas realizadas al prototipo, luego de la implementación y despliegue de este, para ello se toma en cuenta diversos factores en el proceso de funcionamiento los cuales puedan afectar en su desempeño. Posteriormente se detalla las pruebas realizadas desde un navegador web como desde la aplicación móvil, estas se realizan considerando como escenario de pruebas el Laboratorio de Interconectividad de la FIEE donde se encuentra instalado el prototipo y un entorno exterior para pruebas de manipulación remota desde la aplicación.

3.1 PRUEBAS DE FUNCIONAMIENTO VIDEOVIGILANCIA

Las pruebas se realizaron luego de la instalación del prototipo en el escenario, por consiguiente, se puede mencionar que las pruebas se basan principalmente en confirmar el cumplimiento de los requerimientos, respetando también las limitaciones que conllevan sus dispositivos físicos.

Las pruebas de funcionamiento de este subsistema se dividen en 4 áreas:

- Obtención del Stream de video.
- Visualización de recursos multimedia en la galería dinámica.
- Recepción de “Push Notification” y SMS cuando el sensor de puerta es activado.
- Lectura de código QR y apertura de chapa magnética.

3.1.1 Obtención de transmisión de Video en tiempo real

El propósito de esta etapa es comprobar el funcionamiento del controlador como un AP. Posterior a la asignación de direcciones provista por el AP una vez que se enciende el controlador, se comprueba la configuración del archivo “motion.conf” donde se establecen parámetros de las cámaras. Con el empleo de un navegador web, la dirección IP y el puerto adecuado se observa la transmisión de video correspondiente a cada cámara como lo ilustra la Figura 3.1 justo después de iniciado *motion* mediante el comando `sudo motion -n`. La posición de las cámaras se ajusta para no tener puntos ciegos, dicho proceso se lo realiza desde la configuración propia de la cámara.

Figura 3.1 Transmisión de video en tiempo real cámara 1 y cámara 2

3.1.2 Visualización recursos multimedia

Esta etapa presenta un mecanismo de verificación visual para el usuario donde puede apreciar tanto imágenes como videos almacenados debido a que existió movimiento en los interiores del laboratorio, de esta forma el administrador puede constatar dicho suceso y tomar las medidas respectivas. Para acceder a los recursos multimedia se ingresa la dirección del servidor en un navegador web como se muestra en la Figura 3.2 se aprecia tanto imágenes como videos.

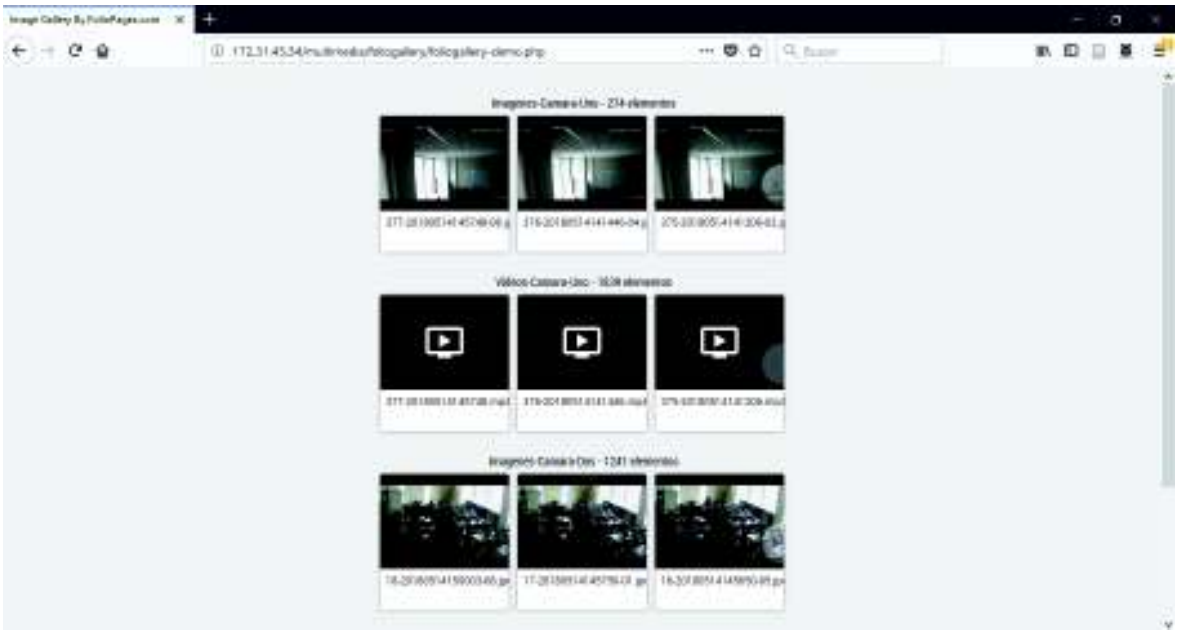

Figura 3.2 Galería Dinámica como herramienta de verificación visual

Las imágenes almacenadas en la galería presentan la hora y fecha del momento en que sucedió el movimiento, estos datos se encuentran ubicados en la esquina inferior derecha, mientras que en la esquina superior izquierda ilustran el nombre de la cámara. Una característica importante de las fotografías es que se presentan en una calidad suficiente para el usuario, mientras que los videos en formato mp4 se reproducen sin ningún problema. La Figura 3.3 ilustra un ejemplo de imagen obtenida del interior de la sala.

Figura 3.3 Imagen vista en la galería dinámica

3.1.3 Recepción de *Push Notification* y SMS

Entre tanto, esta etapa es en la cual se verifica la recepción de la *push notification* y SMS de alerta al teléfono móvil. El usuario al presionar la *push notification* recibida puede ingresar directamente a la aplicación móvil y constatar mediante la interfaz vigilancia que está ocurriendo en ese momento. El SMS y la notificación de la aplicación contienen información acerca de la hora y el día en que ocurrió el suceso de alarma como se ilustra en la Figura 3.4

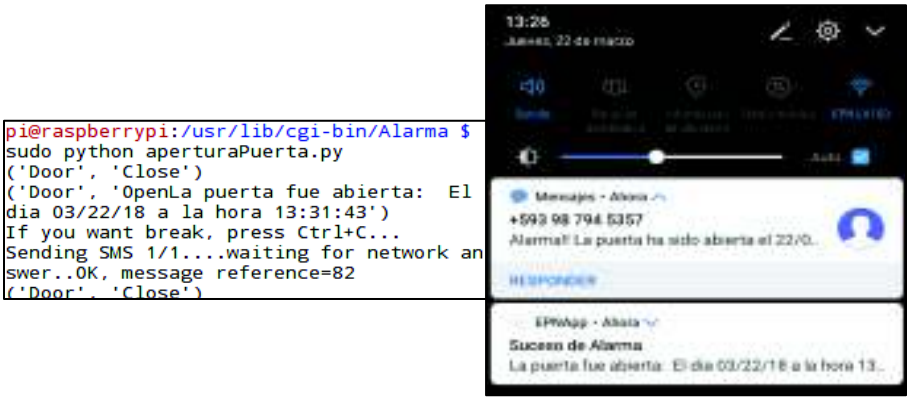

Figura 3.4 Recepción de SMS y "Push Notification"

3.1.4 Lectura de código QR y subproceso de apertura

Una vez accionado el sensor de proximidad la cámara USB se activa junto con el script de lectura (lectorqr.py), dicho proceso se verifica en el intérprete de Python. De esta forma se capta una imagen con la cámara, si la imagen posee un código QR se decodifica y muestra los datos almacenados como se observa en Figura 3.5, sin embargo, si no posee un símbolo se muestra el mensaje "Código QR no encontrado y se debe generar otro código QR. Finalizada la comparación de información se obtiene una validación exitosa o fallida para realizar los subprocesos concernientes.

```
pi@raspberrypi:/usr/lib/cgi-bin/codigoQR $ sudo python lectorqr1.py
Capturando Imagen..
Imagen capturada..

Escaneando imagen..
Codigo QR LEIDO:
Jefferson CordovaLaboratorio Interconectividad03/01/18 10:30:03
Codigo QR PARA VALIDAR:
Jefferson CordovaLaboratorio Interconectividad03/01/18 10:30:03
RESULTADO DE COMPARACION
Validacion Exitosa
pi@raspberrypi:/usr/lib/cgi-bin/codigoQR $
```

Figura 3.5 Resultados mediante mensajes en Python Shell

A continuación, se ilustra las pruebas realizadas con el lector QR, en primera instancia se ejecuta el script lectorqr1.py mediante el sensor de proximidad, en este caso el LED color amarillo se enciende hasta obtener una respuesta, sea afirmativa o negativa, tal como se ilustra en la Figura 3.6.

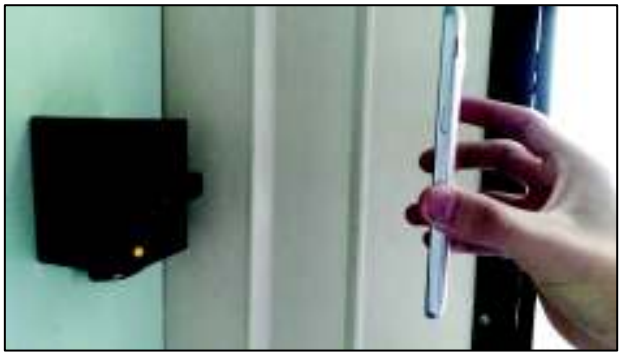

Figura 3.6 LED indicando el inicio del proceso

Una segunda prueba se realiza de la misma forma activar el sensor y mostrar a la cámara el código QR generado por la aplicación móvil, la Figura 3.7 ilustra el encendido del LED color verde que indica una autenticación válida. Adicionalmente para esta prueba la cerradura magnética se abre.

Figura 3.7 LED indicando validación exitosa

Finalmente, al mostrar en el lector QR un código QR no registrado, o si existe un mensaje de error se encenderá el LED color rojo como indica la Figura 3.8, de esta forma el usuario interpreta eso como una señal para generar nuevamente el código QR.

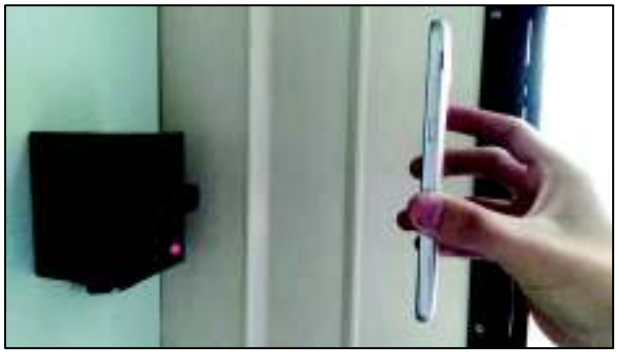

Figura 3.8 LED indicando mensaje de error o validación fallida

Cabe aclarar que, dentro del proceso de reconocimiento y decodificación de la fotografía, la librería PIL propia de Python abre la imagen captada por la cámara USB, seguidamente el paquete *zbarlight* destinado a la lectura de códigos se centra en encontrar un símbolo QR y discriminar los diversos objetos obtenidos en la imagen en este caso puede ser el usuario con el teléfono móvil en frente de la cámara.

3.1.5 Proceso obtención de información

Esta etapa permite obtener información de la base de datos del Firebase para su uso en Python, esto se realiza mediante la ejecución automática de scripts para descargar la información mencionada en secciones anteriores, la Figura 3.9 ilustra la base de datos del Administrador y el Ayudante generada en el controlador para de esta forma utilizar esta información en scripts como el envío de las notificaciones y de SMS además del uso en la generación del código QR para el ingreso a la sala. De la misma forma se realiza la obtención de información de los otros parámetros en la base de datos.

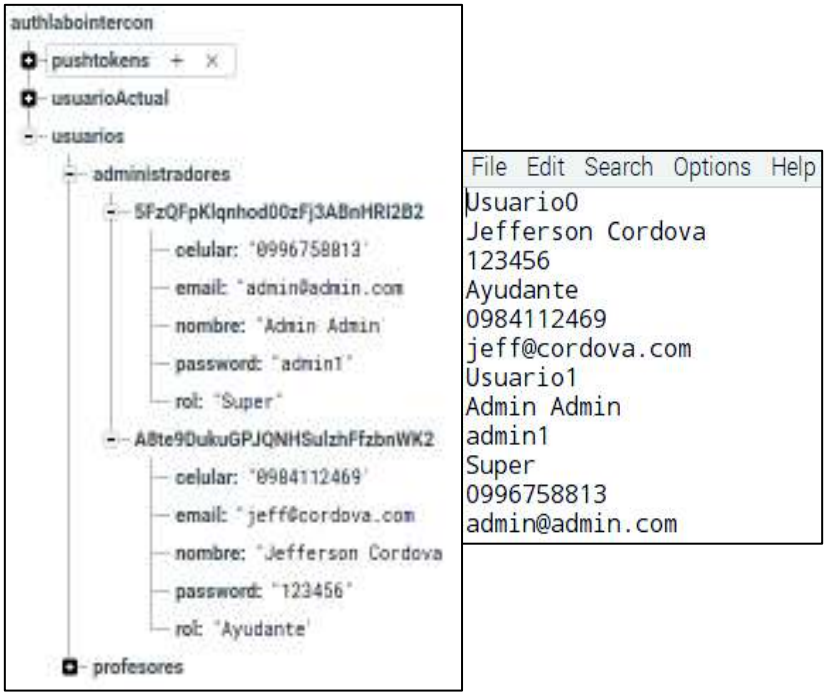

Figura 3.9 Base de datos alojada en: (a) Firebase (b) Raspberry Pi

3.1.6 Pruebas de funcionamiento desde el controlador

Cabe mencionar que en la Tabla 3.1 presentada se realiza las pruebas sin tomar en cuenta a la aplicación móvil debido a que antes de acoplar con este debe estar

funcionando el prototipo correctamente, en una sección posterior se ilustra las pruebas realizadas desde la aplicación móvil.

Tabla 3.1 Resultados de pruebas subsistemas vigilancia y monitoreo PC's

Pruebas realizadas	**Observación**
SUBSISTEMA DE VIGILANCIA	
Obtención transmisión de video cámara EPN1	Correcto
Obtención transmisión de video cámara EPN2	Correcto
Navegación por galería de recursos multimedia	Correcto
Visualización de imágenes (Navegador Web)	Correcto
Reproducción de videos (Navegador web)	Correcto
Recepción notificación SMS	Correcto
Recepción *Push notification*	Correcto
Activación sensor proximidad, ejecución lectorqr1.py	Correcto
Encendido Led color amarillo (Empieza proceso lectura QR)	Correcto
Captura de imagen mediante cámara USB	Correcto
Encendido LED color verde (Validación exitosa)	Correcto
Apertura de cerradura magnética	Correcto
Encendido LED rojo (Validación Fallida o error)	Correcto
Mover imágenes y videos a dispositivo almacenamiento	Correcto
Eliminar imágenes y videos de dispositivo almacenamiento	Correcto
SUBSISTEMA MONITOREO DE PC's	
Encender un computador	Correcto
Apagar un computador	Correcto
Encender todos los computadores a la vez	Correcto
Apagar todos los computadores a la vez	Correcto

3.2 PRUEBAS DE FUNCIONAMIENTO MONITOREO de PCs

Las pruebas de funcionamiento del subsistema de monitoreo consideran los 28 computadores del laboratorio, la ejecución de los scripts se la realiza mediante el servidor Apache. Para verificar el funcionamiento se muestra un mensaje en el navegador dependiendo del caso como muestra la Figura 3.10. El apagado remoto se verifica de la misma forma que el proceso de encendido.

Figura 3.10 Encendido exitoso PC 5, apagado exitoso PC5

Un elemento adicional es el script de “estadosPCs.py” el cual proporciona información de si el computador se encuentra activo o inactivo guardando en un archivo de texto dentro del controlador como se ilustra en la Figura 3.11. Tanto el estado “encendido” como el “apagado” son almacenados en forma de un *array*. Este archivo .txt se utiliza posteriormente para una función de la aplicación móvil.

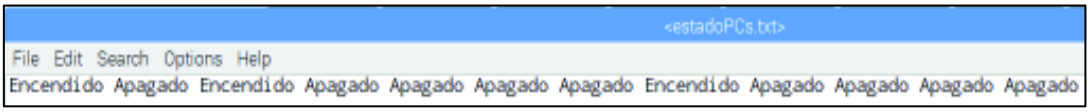

Figura 3.11 Array de estados de los PC’s, ejemplo 12 PCs

Cuando un computador se encuentra encendido el ping demora en promedio 0.6 milisegundos, sin embargo, cuando un computador está apagado se demora entre 1 a 2 segundos en devolver una respuesta. De esta forma se determina que el peor de los casos se produce cuando todos los equipos se encuentran apagados siendo así el tiempo umbral determinado de un minuto para crear el archivo de texto.

3.3 INSTALACION DE LA APLICACIÓN MOVIL

Una ventaja de Ionic es que con el mismo código fuente permite emular la aplicación en un navegador web o en un emulador de Smartphone. El escenario de pruebas de funcionamiento de la aplicación móvil considera 3 usuarios el Administrador, Ayudante y Profesor, de esta manera se puede verificar cada una de las funciones dependiendo del rol de usuario.

3.3.1 Instalación en un dispositivo android

En esta sección se detalla la instalación y pruebas de funcionamiento de la aplicación EPN en un dispositivo móvil.

Instalación de la aplicación móvil en dispositivo

Para realizar la instalación de la aplicación móvil en un dispositivo real, primero se debe copiar la aplicación EPN.apk en la tarjeta microSD o en el almacenamiento interno del dispositivo móvil, después debe seleccionar el archivo apk. Dependiendo de la configuración del celular, este puede mostrar un mensaje de orígenes desconocidos al iniciar la instalación, se confirma y listo. Posterior a esto se instala la aplicación y finalmente se puede ejecutar la aplicación en el dispositivo móvil.

3.4 RESULTADOS DE LAS PRUEBAS DE FUNCIONAMIENTO DE LA APLICACIÓN EN EL DISPOSITIVO REAL

A continuación, se detalla el análisis de resultados para cada uno de los usuarios propuestos y con cada Interfaz diseñada de la aplicación.

Tabla 3.2 Análisis de resultados: **Usuario:** Administrador **Interfaz:** Bienvenida

Pruebas realizadas	**Observación**
Permite abrir menú lateral	Correcto
Permite redirigir a la interfaz Inicio Sesión desde el menú	Correcto
Muestra mensaje de guía al usuario	Correcto

Tabla 3.3 Análisis de resultados: **Usuario**: Administrador **Interfaz**: Inicio Sesión

Pruebas realizadas	**Observación**
Permite ingresar texto en: Correo Electrónico	Correcto
Permite ingresar texto no visible al usuario en: Contraseña	Correcto
Permite redirigir a Interfaz Inicio	Correcto
Muestra mensaje de guía al usuario	Correcto

Tabla 3.4 Análisis de resultados**: Usuario:** Administrador **Interfaz:** Inicio

Pruebas realizadas	**Observación**
Permite abrir menú lateral	Correcto
Permite Cerrar Sesión y Visualizar el manual de Ayuda	Correcto
Permite direccionar a cada una de las interfaces propuestas	Correcto
Muestra botón de apertura de emergencia de la sala	Correcto
Permite activar alarma o desactivar por un tiempo	Correcto

Tabla 3.5 Análisis de resultados: **Usuario:** Administrador **Interfaz:** Registro Usuarios

Pruebas realizadas	**Observación**
Permite completar los campos: Nombre y Apellido, correo electrónico, contraseña, teléfono celular mediante ingreso de texto	Correcto
Permite seleccionar el Cargo del Usuario (Ayudante, Profesor) mediante un selector de opciones	Correcto
Permite agregar el Nombre de la Asignatura, el día, la hora de inicio y la hora de fin	Correcto
Permite eliminar una asignatura de ser el caso	Correcto
Muestra mensaje de confirmación al registrar usuario	Correcto
Permite regresar a la Interfaz Inicio (Home)	Correcto

Tabla 3.6 Análisis de resultados: **Usuario:** Administrador **Interfaz:** Generar Código QR

Pruebas realizadas	**Observación**
Permitir mostrar el código QR generado	Correcto
Oculta el botón de generar el código	Correcto
Permite redirigir a Interfaz Inicio	Correcto
Muestra mensaje de tiempo de autenticación	Correcto

Tabla 3.7 Análisis de resultados: **Usuario:** Administrador **Interfaz:** Vigilancia

Pruebas realizadas	**Observación**
Permitir visualizar el Stream de video EPN1	Correcto
Permitir visualizar el Stream de video EPN2	Correcto
Permite realizar un pequeño zoom al video (ambas cámaras)	Correcto
Permite redirigir a Interfaz Inicio	Correcto

Tabla 3.8 Análisis de resultados**: Usuario:** Administrador **Interfaz:** Monitoreo

Pruebas realizadas	**Observación**
Permite actualizar la página	Correcto
Muestra estados de los PC's según colores	Correcto
Permite apagar el computador seleccionado	Correcto
Permite encender el computador seleccionado	Correcto
Permite apagar todos los computadores a la vez	Correcto
Permite encender todos los computadores a la vez	Correcto
Permite redirigir a Interfaz Inicio	Correcto
Muestra alertas de confirmación para cada botón	Correcto

Tabla 3.9 Análisis de resultados: **Usuario:** Administrador **Interfaz:** Multimedia

Pruebas realizadas	**Observación**
Perite visualizar imágenes de ambas cámaras	Correcto
Permite reproducir los videos de ambas cámaras	Correcto
Permite navegar entre las imágenes o videos	Correcto
Permite redirigir a Interfaz Inicio	Correcto

De la misma forma para verificar el funcionamiento de la aplicación con el usuario AYUDANTE se dispone de tablas.

Tabla 3.10 Análisis de resultados**: Usuario:** Ayudante **Interfaz:** Bienvenida

Pruebas realizadas	**Observación**
Permite abrir menú lateral	Correcto
Permite redirigir a la interfaz Inicio Sesión desde el menú	Correcto
Muestra mensaje de guía al usuario	Correcto

Tabla 3.11 Análisis de resultados: **Usuario:** Ayudante **Interfaz:** Iniciar Sesión

Pruebas realizadas	**Observación**
Permite ingresar texto en: Correo Electrónico	Correcto
Permite ingresar texto no visible al usuario en: Contraseña	Correcto
Permite redirigir a Interfaz Inicio	Correcto
Muestra mensaje de guía al usuario	Correcto

Tabla 3.12 Análisis de resultados: **Usuario:** Ayudante **Interfaz:** Inicio

Pruebas realizadas	**Observación**
Permite abrir menú lateral	Correcto
Permite Cerrar Sesión y Visualizar el manual de Ayuda	Correcto
Permite direccionar a cada una de las interfaces propuestas	Correcto
Muestra botón de apertura de emergencia de la sala	Correcto

Tabla 3.13 Análisis de resultados: **Usuario:** Ayudante **Interfaz:** Generar Código QR

Pruebas realizadas	**Observación**
Permitir mostrar el código QR generado	Correcto
Oculta el botón de generar el código	Correcto
Permite redirigir a Interfaz Inicio	Correcto
Muestra mensaje de tiempo de autenticación	Correcto

Tabla 3.14 Análisis de resultados: **Usuario:** Ayudante **Interfaz:** Vigilancia

Pruebas realizadas	**Observación**
Permitir visualizar el Stream de video EPN1	Correcto
Permitir visualizar el Stream de video EPN2	Correcto
Permite realizar un pequeño zoom al video (ambas cámaras)	Correcto
Permite redirigir a Interfaz Inicio	Correcto

Tabla 3.15 Análisis de resultados: **Usuario:** Ayudante **Interfaz:** Monitoreo

Pruebas realizadas	**Observación**
Perite visualizar imágenes de ambas cámaras	Correcto
Permite reproducir los videos de ambas cámaras	Correcto
Permite navegar entre las imágenes o videos	Correcto
Permite redirigir a Interfaz Inicio	Correcto

Finalmente, el usuario PROFESOR dispone de una restricción de interfaces para navegar, los resultados obtenidos en la fase de pruebas se ilustran en las siguientes tablas

Tabla 3.16 Análisis de resultados: **Usuario:** Profesor **Interfaz:** Bienvenida

Pruebas realizadas	**Observación**
Permite abrir menú lateral	Correcto
Permite redirigir a la interfaz Inicio Sesión desde el menú	Correcto
Muestra mensaje de guía al usuario	Correcto

Tabla 3.17 Análisis de resultados: **Usuario:** Profesor **Interfaz:** Iniciar Sesión

Pruebas realizadas	**Observación**
Permite ingresar texto en: Correo Electrónico	Correcto
Permite ingresar texto no visible al usuario en: Contraseña	Correcto
Permite redirigir a Interfaz Inicio	Correcto
Muestra mensaje de guía al usuario	Correcto

Tabla 3.18 Análisis de resultados: **Usuario:** Profesor **Interfaz:** Inicio

Pruebas realizadas	**Observación**
Permite abrir menú lateral	Correcto
Permite Cerrar Sesión y Visualizar el manual de Ayuda	Correcto
Permite direccionar a cada una de las interfaces propuestas	Correcto

Tabla 3.19 Análisis de resultados**: Usuario:** Profesor **Interfaz:** Generar Código QR

Pruebas realizadas	**Observación**
Permitir mostrar el código QR generado	Correcto
Oculta el botón de generar el código	Correcto
Permite redirigir a Interfaz Inicio	Correcto
Muestra mensaje de tiempo de autenticación	Correcto

Tabla 3.20 Análisis de resultados: **Usuario:** Profesor **Interfaz:** Monitoreo

Pruebas realizadas	**Observación**
Permite actualizar la página	Correcto
Muestra estados de los PC's según colores	Correcto
Permite apagar el computador seleccionado	Correcto
Permite encender el computador seleccionado	Correcto
Permite apagar todos los computadores a la vez	Correcto
Permite encender todos los computadores a la vez	Correcto
Permite redirigir a Interfaz Inicio	Correcto
Muestra alertas de confirmación para cada botón	Correcto

3.5 PRESUPUESTO REFERENCIAL

En esta sección se presenta en modo de tablas los precios referenciales requeridos para la implementación del proyecto, además se considera un aproximado de costos en base a horas de diseño y desarrollo de la aplicación, además de la depuración de errores debido a que el Framework utilizado es de software libre.

Tabla 3.21 Presupuesto referencial de materiales

Ítem	**Cantidad**	**Valor Unitario ($)**	**Valor total ($)**
Raspberry pi 3 Model B	1	70	70
MicroSD 8GB	1	10	10
Flash Memory 8GB	1	8	8
Cámara IP	2	70	140
Cámara USB	1	15	15
Sensor de contacto	1	3	3
Sensor infrarrojo de Presencia	1	7	7
Modem 3G	1	30	30
Cargador o fuente de alimentación	2	5	10
Fuente de alimentación alterna	1	15	15
Chapa magnética	1	42	42
Contenedor del controlador(RPi)	1	18	18
Contenedor lector código QR	1	7	7
Relé de 2 canales	1	10	10
Total Material			385

Tabla 3. 22 Presupuesto referencial de la mano de obra

Ítem	Cantidad	Valor Unitario ($)	Valor total ($)
Tiempo de diseño(horas)	40	10	400
Tiempo de desarrollo(horas)	120	10	1200
Tiempo de instalación(horas)	24	10	240
Tiempo de pruebas(horas)	24	10	240
Total Mano de Obra			2080

Tabla 3.23 Presupuesto referencial del prototipo

Ítem	Valor total ($)
Total Material	385
Total Mano de Obra	2080
Total Prototipo	2465

Se ha estimado que el costo necesario para el desarrollo del prototipo sería de $ 2465,00. Se debe resaltar el hecho de que en una segunda ocasión en la que se replique el prototipo, el costo se reduciría drásticamente dado que los tiempos de diseño y desarrollo se reducirían considerablemente.

Cabe mencionar que Firebase permite comenzar a usar sus productos gratis, dependiendo de la demanda se puede ajustar a la escala y así pagar solo por lo que realmente se usa. Con esta idea se promocionan 3 planes: el plan *Spark* que es gratis, el plan *Flame* que tiene un valor de $25 al mes, y el plan *Blaze* que es pago por uso con ventajas que no incluyen el resto de planes como: copias de seguridad automáticas en la base de datos, y usar *BigQuery* junto con otras funciones con modalidad *Infraestructure as a Service* (IaaS).

4. CONCLUSIONES

4.1 Conclusiones

- En la implementación del prototipo una parte muy importante fue la capacidad que brindó Python para el desarrollo de casi cualquier tipo de rutinas o tareas que permiten controlar y leer datos de los periféricos utilizados, esto es gracias a la gran cantidad de librerías de las que dispone. A esto lo favorece en gran medida la capacidad de conexión a Internet ya que de esta manera se pudo integrar varios servicios de la plataforma de Firebase y en consecuencia esto facilitó ampliamente la gestión de la información, la medición de parámetros, el procesamiento de datos, etc.

- Al utilizar la Raspberry Pi 3 Model B como elemento controlador del prototipo, se logró concentrar todos los sensores y periféricos en un solo punto sin necesidad de ninguna clase de hub o multiplexor adicional, debido a que ofrece un conjunto muy útil de pines GPIO y un diverso grupo de conectores que permiten la incorporación de dispositivos como cámaras IP, componentes USB, sensores, entre otros, dando lugar a un sistema practico y accesible.

- En este proyecto la implementación del lector de códigos QR como un método de control de acceso, fue factible gracias a la librería de Python zbarlight que permite la decodificación del código QR leído de una manera sencilla. En consecuencia, permite obtener la información y crear un registro de acceso de usuarios de manera simple, para una posterior revisión y respaldo por parte del administrador.

- Mediante alertas vía SMS y del tipo “Push” el usuario administrador y ayudante del sistema pueden constatar sucesos de alarma fuera del horario laboral específicamente el caso de una apertura de puerta. En nuestro proyecto, el SMS es considerado como una notificación de respaldo debido a que si existe un problema en la red la notificación “Push” no llegaría al teléfono móvil del usuario, y se aseguraría de esta forma que el usuario reciba la notificación del evento de alarma.

- La plataforma Firebase ayudo en las funciones de: almacenaje de información de usuario, ordenamiento y clasificación de usuarios por roles, además almacenamiento de los tokens generados por cada dispositivo móvil. De esta manera, y siendo un servicio en la nube, permitió que la Raspberry Pi se conecte a través de Internet con la aplicación móvil haciendo las veces de una pequeña interfaz de comunicación bidireccional clave en este prototipo.

- Para la gestión remota de los computadores del laboratorio se aprovechó una característica muy útil de las tarjetas de red, disponibles en los equipos del laboratorio, denominada *wake on lan*, la cual permite reemplazar el proceso de encendido y apagado manual de las máquinas por parte del encargado, dando lugar a minimizar el tiempo en el proceso, ya que mediante un botón de la aplicación es posible encender/apagar todos los computadores a la vez.

- Respecto a la elaboración de la aplicación móvil el *framework* utilizado contribuyó en la reducción del tiempo de desarrollo ya que de cierta forma orienta al programador en primera instancia elaborar la estructura de la presentación a través de un archivo HTML, posteriormente se añaden funciones necesarias utilizando operaciones que brinda JavaScript y en ciertos casos se modificó los estilos predeterminados usando el archivo SCSS. Estos lenguajes de programación permiten al programador modificar, añadir y eliminar bloques de código, así como características y paquetes adicionales con mucha facilidad.

- Se optó por el desarrollo de una aplicación móvil como mecanismo de interacción con el prototipo debido a que actualmente el smartphone es uno de los dispositivos más utilizados. El usuario se encuentra muy familiarizado con él al utilizarlo diariamente en sus actividades cotidianas, de esta manera está al día en instalación y manipulación de aplicaciones de mensajería, juegos, fotos, etc. En consecuencia, una aplicación para el monitoreo de un sistema de vigilancia no representa ningún impedimento para el usuario sino más bien permite la movilidad y manejo remoto del sistema en circunstancias que este lo amerite.

- La instalación de los dispositivos que conforman el prototipo se realizó en base a un boceto en el cual se definió la ubicación de estos considerando la función a desempeñar y el tamaño de estos. En base a este principio se justifica el lugar de emplazamiento de los dispositivos. La ejecución de la instalación donde comprende el paso del cableado, perforación de paredes y sujeción de los elementos, se facilitó enormemente con la planificación mencionada.

- Se realizaron pruebas de funcionamiento en el escenario específico donde la red eléctrica no se encuentra suministrando energía, con el propósito de comprobar el funcionamiento de la fuente de alimentación alterna y se confirmó con éxito que el prototipo cumple las funciones propuestas.

- Acorde a las pruebas realizadas desde un lugar externo a la sala del laboratorio se obtuvo un retardo adicional en la respuesta del prototipo, comparando con el caso en que se utilice la aplicación desde la misma red local de la Universidad. Esto es debido al mayor número de saltos que debe atravesar las peticiones desde la aplicación al prototipo

4.2 Recomendaciones

- Se recomienda realizar pruebas de funcionamiento de la aplicación móvil en primer lugar con un emulador y varios AVD's, para que a continuación se pruebe en diversos tipos de celulares. De esta forma se pueden depurar varios errores en la primera etapa antes de pasar a las pruebas en dispositivos físicos.

- Se recomienda ser cuidadoso al instalar los diferentes paquetes en la Raspberry PI para prevenir que la memoria interna se llene de forma innecesaria, y de esta manera evitar el uso excesivo del CPU por parte de algunas tareas que se ejecuten en base a estos paquetes.

- Es recomendable realizar una aplicación móvil inicial de tal forma que se compruebe solamente funcionalidad, de esta manera una vez comprobado las funciones del prototipo se puede añadir los detalles para que sea una aplicación óptima y de fácil manejo para el usuario.

- En una nueva versión del prototipo es recomendable realizar algunos cambios además de una ampliación de funciones, de esta forma se podrá brindar otros servicios como pueden ser:
    - El control remoto de las luminarias en el interior de la sala.
    - Dar la posibilidad al encargado de visualizar al usuario que desea entrar a la sala y que tenga la opción de abrir directamente la puerta desde la aplicación.
    - Crear una opción para gestionar las bases de datos con el objetivo de que presente la información de una forma más accesible al usuario no técnico.
    - Generar automáticamente reportes de ingresos al salón, en un formato pdf para que sea más fácil de manipular.
    - Incluir en la aplicación móvil interfaces que controlen los servicios antes mencionados que podrían ser agregados en el prototipo.

- Este proyecto se podría expandir a los laboratorios aledaños replicando el prototipo y definiendo a un Raspberry Pi como central para que recabe y gestione toda la información que se manejaría en ese caso. Además, se debería tomar en cuenta el nuevo modelo de la Raspberry Pi, que promete mejoras en su desempeño gracias a su CPU de mayor velocidad, soporte de WiFi doble banda y puerto Gigabit ethernet.

- Un tema interesante que se podría considerar es la implementación de un método de control de acceso mediante el análisis de imagen para una detección de rostros, de esta manera con una base de datos se podría obtener la información del sujeto en caso de encontrar alguna coincidencia.

- Sería recomendable que se continúe desarrollando este proyecto a futuro para que se logre mejorar el rendimiento del sistema intentando explotar mucho más la capacidad de GPU que posee la Raspberry Pi.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] «Video Surveillance Trends for 2018 - IHS Markit.pdf». [Online]. Disponible en: https://goo.gl/TQCYJk. [Accedido: 15-may-2018]

[2] Anastasios Dimou, «Video Surveillance Systems: Current Systems and Future Trends» [Online]. Disponible en: https://www.ies.be/files/ADVISE-Aosta_12_DIMOU_video_surveillance_systems.pdf. [Accedido: 15-may-2018]

[3] «Video camaras de vigilancia conceptos y debate etico - Camaras de seguridad, camaras IP, KIT de camaras, camaras espia, CCTV, camaras de vigilancia.» [Online]. Disponible en: http://topsecurityperu.com/video-camaras-de-vigilancia-conceptos-y-debate-etico/. [Accedido: 19-sep-2017]

[4] Raúl Alfredo Faicán Pila, «Diseño e implementación de un prototipo inalámbrico de monitoreo a través de secuencias de imágenes utilizando hardware y software libre», Escuela Politécnica Nacional, Quito, 2016 [Online]. Disponible en: http://bibdigital.epn.edu.ec/bitstream/15000/16792/1/CD-7385.pdf. [Accedido: 16-may-2018]

[5] S. R. Montoro y D. M. Costilla, «Streaming de Audio/Video. Protocolo RTSP», *Enginy*, n.º 01, 2012 [Online]. Disponible en: http://enginy.uib.es/index.php/enginy/article/viewFile/74/53. [Accedido: 19-sep-2017]

[6] P. Sachdeva y S. Katchii, «A review paper on raspberry pi», *Dimensions (Wash.)*, vol. 85, p. x56mm, 2014 [Online]. Disponible en: http://inpressco.com/wp-content/uploads/2014/11/Paper53818-3819.pdf. [Accedido: 19-sep-2017]

[7] «Raspberry Pi Downloads - Software for the Raspberry Pi», *Raspberry Pi*. [Online]. Disponible en: https://www.raspberrypi.org/downloads/. [Accedido: 17-may-2018]

[8] M. Says, «Transition from Scratch to Python with FutureLearn», *Raspberry Pi*. 26-feb-2018 [Online]. Disponible en: https://www.raspberrypi.org/blog/futurelearn-scratch-to-python/. [Accedido: 17-may-2018]

[9] «Raspberry Pi Desktop», *Raspberry Pi*. [Online]. Disponible en: https://www.raspberrypi.org/downloads/raspberry-pi-desktop/. [Accedido: 17-may-2018]

[10] «Apache.pdf». [Online]. Disponible en: http://descargas.pntic.mec.es/mentor/visitas/Apache.pdf. [Accedido: 26-ene-2018]

[11] «Instalación y configuración de Apache | Redes Linux». [Online]. Disponible en: http://www.ite.educacion.es/formacion/materiales/85/cd/linux/m3/instalacin_y_config

uracin_de_apache.html. [Accedido: 26-ene-2018]

[12] «IR Sensor | What is an IR Sensor?» [Online]. Disponible en: http://education.rec.ri.cmu.edu/content/electronics/boe/ir_sensor/1.html. [Accedido: 17-may-2018]

[13] Sergio Herrero, «Elementos de detección». [Online]. Disponible en: http://www.rnds.com.ar/articulos/014/RNDS_096W.pdf. [Accedido: 17-oct-2017]

[14] «Interruptor Magnetico de Inteperie IMI-1 - Cetronic». [Online]. Disponible en: http://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=1&idTienda=93&codProducto=033072002&cPath=704. [Accedido: 17-may-2018]

[15] RNDS, «Cámaras de red / Cámaras IP», pp. 140-152 [Online]. Disponible en: http://www.rnds.com.ar/articulos/046/RNDS_140W.pdf. [Accedido: 17-oct-2017]

[16] «Códigos QR». [Online]. Disponible en: http://www.acta.es/medios/articulos/comunicacion_e_informacion/063009.pdf. [Accedido: 20-sep-2017]

[17] D. García Gadea, «Sistema autónomo y de bajo coste para reconocimiento de códigos QR», 2016.

[18] «Format and Version Information - QR Code Tutorial». [Online]. Disponible en: http://www.thonky.com/qr-code-tutorial/format-version-information. [Accedido: 18-oct-2017]

[19] «QR Code Basics». [Online]. Disponible en: http://www.qr-code-generator.com/qr-code-marketing/qr-codes-basics/. [Accedido: 17-oct-2017]

[20] *Proceedings of the XV International Scientific Conference on Industrial Systems (IS'11)*. FON, 2011.

[21] «Back-End Web Architecture», *Codecademy*. [Online]. Disponible en: https://www.codecademy.com/articles/back-end-architecture. [Accedido: 17-may-2018]

[22] «HTML», *Mozilla Developer Network*. [Online]. Disponible en: https://developer.mozilla.org/es/docs/Web/HTML. [Accedido: 17-oct-2017]

[23] «Introducción a JavaScript». [Online]. Disponible en: https://librosweb.es/libro/javascript/. [Accedido: 17-oct-2017]

[24] «JavaScript», *Mozilla Developer Network*. [Online]. Disponible en: https://developer.mozilla.org/es/docs/Web/JavaScript. [Accedido: 17-oct-2017]

[25] «CSS», *Mozilla Developer Network*. [Online]. Disponible en: https://developer.mozilla.org/es/docs/Web/CSS. [Accedido: 17-oct-2017]

[26] «Python 2.7.14 documentation». [Online]. Disponible en: https://docs.python.org/2/. [Accedido: 17-oct-2017]

[27] «Desarrollo de aplicaciones móviles», *Estudio WAM*. [Online]. Disponible en: http://estudiowam.com/desarrollo-de-aplicaciones-moviles/. [Accedido: 17-oct-2017]

[28] «Los 3 tipos de aplicaciones móviles: ventajas e inconvenientes», *LanceTalent*. 20-feb-2014 [Online]. Disponible en: https://www.lancetalent.com/blog/tipos-de-aplicaciones-moviles-ventajas-inconvenientes/. [Accedido: 17-oct-2017]

[29] «¿Qué es un archivo APK y cómo instalar? - AndroidPIT». [Online]. Disponible en: https://www.androidpit.es/android-para-principiantes-apk. [Accedido: 17-may-2018]

[30] «Capítulo 8: Diseño visual – Diseñando apps para móviles», *Designing mobile apps*. [Online]. Disponible en: http://appdesignbook.com/es/contenidos/diseno-visual-apps-nativas/. [Accedido: 26-ene-2018]

[31] J. J. Gutiérrez, «¿Qué es un framework web?», p. 4.

[32] Drifty, «Ionic Framework», *Ionic Framework*. [Online]. Disponible en: https://ionicframework.com/docs/. [Accedido: 17-oct-2017]

[33] «Angular - What is Angular?» [Online]. Disponible en: https://angular.io/docs. [Accedido: 17-oct-2017]

[34] «Architectural overview of Cordova platform - Apache Cordova». [Online]. Disponible en: https://cordova.apache.org/docs/en/latest/guide/overview/index.html. [Accedido: 17-oct-2017]

[35] «Study of Google Firebase API for Android». [Online]. Disponible en: https://www.ijircce.com/upload/2016/september/133_Study.pdf. [Accedido: 17-oct-2017]

[36] «Pidora: ¿Qué es “Pidora”? | Gustavo Pimentel’s GNU/Linux Blog». [Online]. Disponible en: http://gustavo.pimentel.eu/2013/05/pidora-que-es-pidora/. [Accedido: 17-may-2018]

[37] «Arch Linux». [Online]. Disponible en: https://www.archlinux.org/. [Accedido: 17-may-2018]

[38] nomore, «Kano OS. El sistema operativo perfecto para Raspberry pi.», *Nomoretechnology*. 11-jul-2016 [Online]. Disponible en: http://nomoretechnology.net/2016/07/11/kano-os-el-sistema-operativo-perfecto-para-raspberry-pi/. [Accedido: 17-may-2018]

[39] K. Cevallos, «UML: Casos de Uso», *INGENIERÍA DEL SOFTWARE*. 04-jun-2015 [Online]. Disponible en: https://ingsotfwarekarlacevallos.wordpress.com/2015/06/04/uml-casos-de-uso/.

[Accedido: 25-ene-2018]

[40] «SD cards - Raspberry Pi Documentation». [Online]. Disponible en: https://www.raspberrypi.org/documentation/installation/sd-cards.md. [Accedido: 17-may-2018]

[41] «adafruits-raspberry-pi-lesson-6-using-ssh.pdf». [Online]. Disponible en: https://cdn-learn.adafruit.com/downloads/pdf/adafruits-raspberry-pi-lesson-6-using-ssh.pdf. [Accedido: 26-ene-2018]

[42] «Tutorial Raspberry Pi – 7. Escritorio remoto VNC + NO-IP». [Online]. Disponible en: https://geekytheory.com/tutorial-raspberry-pi-7-escritorio-remoto-vnc-no-ip. [Accedido: 26-ene-2018]

[43] O. Vatansever, *python-firebase: Python interface to the Firebase's REST API.* [Online]. Disponible en: http://ozgur.github.com/python-firebase/. [Accedido: 26-ene-2018]

[44] «Wireless AP with udhcpd and NAT - Alpine Linux». [Online]. Disponible en: https://wiki.alpinelinux.org/wiki/Wireless_AP_with_udhcpd_and_NAT. [Accedido: 26-ene-2018]

[45] lxxxvi, «Use a Raspberry Pi 3 as an access point», *Simplificator*. 28-abr-2017 [Online]. Disponible en: https://blog.simplificator.com/2017/04/28/use-a-raspberry-pi-3-as-an-access-point/. [Accedido: 26-ene-2018]

[46] Alfredo Fiebig, «IPTABLES ¿Que es? y ¿Como Funciona?», 21:31:41 UTC [Online]. Disponible en: https://es.slideshare.net/afiebig/iptables-que-es-y-como-funciona. [Accedido: 26-ene-2018]

[47] «Manual básico de como usar Cron». [Online]. Disponible en: https://www.linuxtotal.com.mx/index.php?cont=info_admon_006. [Accedido: 20-feb-2018]

[48] «mount y fstab - montar particiones». [Online]. Disponible en: https://usuariodebian.blogspot.com/2007/08/fstab-montar-particiones-en-el-arranque.html. [Accedido: 26-ene-2018]

[49] «Mover todos los archivos del mismo tipo de un arbol de directorios a la vez». [Online]. Disponible en: http://joedicastro.com/mover-todos-los-archivos-del-mismo-tipo-de-un-arbol-de-directorios-a-la-vez.html. [Accedido: 26-ene-2018]

[50] «folioGallery is a Free PHP, jQuery, Ajax Photo Gallery, No Database Required - FolioPages.com». [Online]. Disponible en: http://www.foliopages.com/php-jquery-ajax-photo-gallery-no-database. [Accedido: 26-ene-2018]

[51] «Raspberry Pi GPIO Pins and Python | Make»:, *Make: DIY Projects and Ideas for*

Makers. [Online]. Disponible en: https://makezine.com/projects/tutorial-raspberry-pi-gpio-pins-and-python/. [Accedido: 26-ene-2018]

[52] «FCM - send push notifications using Python», *Lintel Technologies Blog*. 12-nov-2016 [Online]. Disponible en: http://howto.lintel.in/push-notifications-using-python-via-fcm/. [Accedido: 26-ene-2018]

[53] E. Adegbite, *pyfcm: Python client for FCM - Firebase Cloud Messaging (Android & iOS)..* [Online]. Disponible en: https://github.com/olucurious/pyfcm. [Accedido: 26-ene-2018]

[54] «Installing 3G modem | Raspberry-at-home.com – Your ultimate source of Raspberry Pi tutorials (WiFi, 3G, XBMC, Subtitles, VoD, TVN Player, IPLA, TVP, Squeezeslave, Logitech Media Server, Sickbeard, Webcam, Torrent, DLNA)». [Online]. Disponible en: http://raspberry-at-home.com/installing-3g-modem/. [Accedido: 26-ene-2018]

[55] «Send SMS using AT commands». [Online]. Disponible en: http://www.smssolutions.net/tutorials/gsm/sendsmsat/. [Accedido: 18-may-2018]

[56] M. Nooner, *PyQRCode: A QR code generator written purely in Python with SVG, EPS, PNG and terminal output.* [Online]. Disponible en: https://github.com/mnooner256/pyqrcode. [Accedido: 26-ene-2018]

[57] «Using a standard USB webcam - Raspberry Pi Documentation». [Online]. Disponible en: https://www.raspberrypi.org/documentation/usage/webcams/. [Accedido: 26-ene-2018]

[58] Polyconseil, *zbarlight: A simple zbar wrapper*. [Online]. Disponible en: https://github.com/Polyconseil/zbarlight. [Accedido: 20-feb-2018]

[59] «TeamViewer-Manual-Wake-on-LAN-es.pdf». [Online]. Disponible en: https://www.teamviewer.com/es/res/pdf/TeamViewer-Manual-Wake-on-LAN-es.pdf. [Accedido: 26-ene-2018]

[60] «Shutdown y acceso denegado – Apagar PC remotamente». [Online]. Disponible en: https://www.spamloco.net/2007/10/shutdown-y-el-acceso-denegado.html. [Accedido: 26-ene-2018]

[61] «Tutorial de Apache: Contenido Dinámico con CGI - Servidor HTTP Apache Versión 2.5». [Online]. Disponible en: https://httpd.apache.org/docs/trunk/es/howto/cgi.html. [Accedido: 26-ene-2018]

[62] «Cómo instalar Node.js y npm». [Online]. Disponible en: https://rukbottoland.com/blog/como-instalar-node-js-y-npm/. [Accedido: 26-ene-2018]

[63] «La interfaz de linea de comandos - Apache Cordova». [Online]. Disponible en: https://cordova.apache.org/docs/es/latest/guide/cli/. [Accedido: 26-ene-2018]

[64] «Cómo instalar Ionic en Ubuntu 16.04: Tutorial Completo», *OpenWebinars.net*, 13-sep-2016. [Online]. Disponible en: https://openwebinars.net/como-instalar-ionic-en-ubuntu-16/. [Accedido: 26-ene-2018]

[65] «Nuestro primer proyecto · Ionic v1». [Online]. Disponible en: https://ajgallego.gitbooks.io/ionic/content/basicos_primer_proyecto.html. [Accedido: 26-ene-2018]

[66] Drifty, «Ionic Framework», *Ionic Framework*. [Online]. Disponible en: https://ionicframework.com/docs/. [Accedido: 26-ene-2018]

[67] jQuery F.- jquery.org, «jQuery». [Online]. Disponible en: https://jquery.com/. [Accedido: 26-ene-2018]

[68] fechanique, *cordova-plugin-fcm: Google FCM Push Notifications Cordova Plugin*. 2018 [Online]. Disponible en: https://github.com/fechanique/cordova-plugin-fcm. [Accedido: 26-ene-2018]

[69] «Sensor Magnetico Alarma Puerta Ventana Porton Abertura Detec». [Online]. Disponible en: https://www.01seguridad.com.ar/sensor-magnetico-alarma-puerta-ventana-porton-abertura-detec-01SEGURIDAD-_380. [Accedido: 20-feb-2018]

[70] «Huawei E173 - 3G modem wiki». [Online]. Disponible en: http://www.3g-modem-wiki.com/page/Huawei+E173. [Accedido: 20-feb-2018]

[71] «X5Tech XW-360 Cámara web caracteristicas, opiniones y precios», *Estimamos.com*. [Online]. Disponible en: http://estimamos.com/webcam-x5tech/x5tech-xw-360/. [Accedido: 20-feb-2018]

5 ANEXOS

ANEXO I

El Anexo I se encuentra en el CD adjunto

ANEXO II

El Anexo II se encuentra en el CD adjunto

ANEXO III

El Anexo III se encuentra en el CD adjunto

ANEXO IV

El Anexo IV se encuentra en el CD adjunto

ANEXO V

El Anexo V se encuentra en el CD adjunto

ANEXO VI

El Anexo VI se encuentra en el CD adjunto

ANEXO VII

El Anexo VII se encuentra en el CD adjunto

ANEXO VIII

El Anexo VIII se encuentra en el CD adjunto

ANEXO IX

El Anexo IX se encuentra en el CD adjunto

ANEXO X

El Anexo X se encuentra en el CD adjunto

ANEXO XI

El Anexo XI se encuentra en el CD adjunto

ORDEN DE EMPASTADO