

Universidad Cenfotec

Maestría en Ciberseguridad

Documento final de Proyecto de Investigación Aplicada 2

Creación de un sistema de control de accesos seguro, basado en internet de las cosas (IoT), utilizando tecnologías de *hardware* y *software* libre

Zamora Madriz, Mario Andrés

Mayo, 2018

Declaratoria de derechos de autor

Queda prohibida la reproducción y distribución de esta obra para fines lucrativos. Este documento es confidencial por un plazo de 1 año a partir de su presentación.

Posterior a este plazo autorizo a la Universidad Cenfotec y a sus estudiantes activos a consultar el documento única y exclusivamente con fines académicos, de investigación o mejoras al trabajo realizado.

©2018, Sistema de control de accesos seguro basado en internet de las cosas (IoT), utilizando tecnologías de hardware y software libre, Mario Andrés Zamora Madriz.

Dedicatoria

A mi novia Shirley Pereira por todo el apoyo y comprensión brindado a través de todo el proceso, y darme la motivación y fuerzas necesarias para salir adelante.

A mi amigo Francisco Picado por su todo apoyo durante el proyecto, y ayudarme a perder el tiempo con distracciones necesarias.

A mi mamá por darme ideas creativas de cómo resolver los problemas y ayudarme cuando lo he necesitado.

Agradecimientos

Quiero agradecer al Ministerio de Ciencia y Tecnología por la oportunidad de cursar esta maestría, ya que con su apoyo logré adquirir muchos conocimientos para contribuir al mejoramiento de la seguridad en Costa Rica.

A “La Inventoría” de la Fundación Costa Rica para la innovación, pues me abrió las puertas para crear, inspirar e innovar en un entorno creativo y sacar adelante el proyecto.

A Johnny Romero, por su colaboración como mi tutor y ayuda durante el proceso de desarrollo de la investigación.

A Miguel Pérez Montero por su apoyo como lector.

Tabla de contenidos

Índice de figuras 6

Índice de tablas 8

Capítulo 1. Introducción 9
- 1.1 Antecedentes del problema 9
- 1.2 Definición y descripción del problema 9
- 1.3 Justificación 11
  - 1.3.1 Beneficios técnicos 11
  - 1.3.2 Beneficios operativos 12
  - 1.3.3 Beneficios económicos 12
- 1.4 Objetivos 12
  - 1.4.1 Objetivo general 13
  - 1.4.2 Objetivos específicos 13
- 1.5 Alcances y limitaciones 13
  - 1.5.1 Alcances 13
  - 1.5.2 Limitaciones 14
- 1.6 Estado de la cuestión 14

Capítulo 2. Marco teórico 25
- 2.1 Control de acceso 25
  - 2.1.1.1 Código de barras 26
  - 2.1.1.2 Código QR 27
  - 2.1.1.3 Teclado numérico 27
  - 2.1.1.4 RFID 28
  - 2.1.1.5 Sistemas biométricos 30
  - 2.1.1.6 *One Time Password* 31
- 2.2 *Hardware* libre 32
  - 2.2.1 Arduino 32
  - 2.2.2 Intel Galileo 34
  - 2.2.3 Raspberry Pi 35
    - 2.2.3.1 GPIO 37
    - 2.2.3.2 Raspbian 40
- 2.3 *Software* libre 41
- 2.4 Internet de las cosas (IoT) 42
  - 2.4.1 Definición 42
  - 2.4.2 Plataformas IoT en la nube 43

2.4.3 Protocolo MQTT 44
2.4.4 Seguridad en IoT 45
Capítulo 3. Marco metodológico 49
Capítulo 4. Propuesta de solución 51
4.1 Arquitectura de la solución 51
4.1.1 Aplicación web en la nube para administración del sistema 52
4.1.2 Servidor MQTT Broker en Amazon EC2 53
4.1.3 Controlador de hardware para el control de acceso 53
4.1.4 Dispositivos de control de acceso físico 53
4.1.5 Aplicación para usuarios finales 53
4.2 Servidor MQTT Broker 54
4.3 Dispositivo controlador del sistema de control de accesos 56
4.3.1 Elección del *hardware* libre para el controlador 56
4.3.2 Diseño y ensamblaje del *hardware* libre 58
4.3.3 Dispositivos de control de acceso compatibles 61
4.3.4 Configuración del sistema operativo y *software* 62
4.3.4.1 Reforzamiento de seguridad en Raspbian 62
4.3.4.2 Node-RED 65
4.4 Desarrollo de aplicación web en la nube para administración del sistema 71
4.4.1 Requerimientos 71
4.4.2 Diseño de la aplicación 74
4.4.3 Base de datos 76
4.4.4 Desarrollo de la aplicación 81
4.4.5 Buenas prácticas de seguridad 88
4.4.5.1 *OWASP IoT Vulnerabilities Project* 89
4.5 Modelo a escala 94
Capítulo 5. Conclusiones y recomendaciones 95
5.1 Conclusiones 95
5.2 Recomendaciones 96
Capítulo 6. Trabajos futuros 98
Referencias 99
Glosario 105
Anexo 1 107

Índice de figuras

Figura 1. Código de Barras EAN-13.

Figura 2. Código QR.

Figura 3. Teclado numérico.

Figura 4. Proceso de lectura de RFID.

Figura 5. Tarjetas RFID.

Figura 6. Arduino Uno Rev 3.

Figura 7. Intel Galileo Gen 2.

Figura 8. Raspberry Pi 3 Model B.

Figura 9. Raspberry Pi Zero W.

Figura 10. Pines GPIO.

Figura 11. Descripción de pines GPIO.

Figura 12. Sistema Operativo Raspbian.

Figura 13. Arquitectura de la Solución del Sistema control de accesos propuesto.

Figura 14. Módulo de 8 *relays*.

Figura 15. Conexión de pines GPIO del Raspberry Pi con Módulo de Relay.

Figura 18. Desactivar el SSH en Raspbian.

Figura 19. Nodos de salida Node-RED.

Figura 20. Nodos de entrada para mensajes en Node-RED.

Figura 21. Estructura de tópicos en mensajes del controlador.

Figura 22. Total de tópicos del controlador.

Figura 23. Panel de control de indicadores de acceso local.

Figura 24. Estructura de carpetas de la aplicación.

Figura 25. Diagrama entidad-relación de la base de datos de la aplicación.

Figura 26. Inicio de sesión en la aplicación.

Figura 27. Panel de control de la aplicación.

Figura 28. Administración de usuarios de la aplicación.

Figura 29. Administración de invitados de la aplicación.

Figura 30. Bitácora de la aplicación.

Figura 31. Administración de controladores de la aplicación.

Figura 32. Administración de roles de la aplicación.

Figura 33. Reportes de la aplicación.

Figura 34. Protección contra enumeración de usuarios.

Índice de tablas

Tabla 1. *Características del sistema de control de accesos NetAXS-123.*

Tabla 2. *Características del sistema de control de accesos UT0814-S4110V.*

Tabla 3. *Características del sistema de control de accesos Suprema Biostation T2.*

Tabla 4. *Características del sistema de control de accesos Keri Systems NXT-4D-MSC.*

Tabla 5. *Características del sistema de control de accesos Paxton PROXIMITY.*

Tabla 6. *Características del sistema de control de accesos EverFocus NAV-04-1ª.*

Tabla 7. *Características del sistema de control de accesos Keyscan CA4500.*

Tabla 8. *Características del sistema de control de accesos MountainOne Video Door Bell.*

Tabla 9. *Características del sistema de intercomunicación HikVision Video Intercom Door Station.*

Tabla 10. *Características del sistema de control de accesos INT-UNC100.*

Tabla 11. *Especificaciones técnicas de Arduino.*

Tabla 12. *Especificaciones técnicas de los modelos de Raspberry Pi.*

Tabla 13. *Comparación de dispositivos de hardware.*

Tabla 14. *Conexión de pines GPIO del Raspberry Pi con Módulo de Relay.*

Tabla 15. *Complementos de terceros de código abierto utilizados en la aplicación.*

Capítulo 1
Introducción

1.1 Antecedentes del problema

La seguridad de acceso físico forma parte importante en la protección de la información. Sin embargo, muchos profesionales en tecnologías de información delegan las tareas de administración y monitoreo al personal encargado de la vigilancia de las instalaciones o simplemente lo pasan por alto, para enfocarse en la seguridad lógica.

Las empresas poseen activos relevantes que se deben proteger contra el acceso o uso no autorizado para evitar daños, robos o cualquier tipo de problema que ponga en riesgo la seguridad de la compañía y la integridad de las personas. Por esta razón, se emplean mecanismos de defensa para protegerse de las posibles amenazas y reducir las vulnerabilidades que puedan afectar su funcionamiento normal.

Con el tiempo, los mecanismos de control de acceso han pasado de utilizar el recurso humano y los mecanismos mecánicos (llaves), a emplear diferentes sistemas automatizados y tecnologías, los cuales presentan un nivel de seguridad más alto con respecto a los métodos tradicionales. Es por esta razón que los sistemas de control de acceso tienen una gran demanda en el mercado actual.

Un sistema de control de accesos físico tiene como objetivo impedir que personas no autorizadas utilicen un recurso o un activo. Es capaz de gestionar el acceso físico para determinar quién está autorizado a entrar o salir, a qué área se permite ingresar, o cuando se permite el acceso. Con esto se asegura que solamente los usuarios autorizados puedan acceder al recurso cuando sea necesario.

1.2 Definición y descripción del problema

Los sistemas de control de acceso (SCA) utilizan dispositivos de *hardware* físicos para gestionar puntos de acceso como puertas, agujas vehiculares, portones, entre otros. Sin embargo, estos mecanismos presentan las siguientes desventajas que pueden disminuir la seguridad y eficacia del sistema:

- **Tarjetas de identificación inteligente:** Pueden ser copiadas mediante el uso de dispositivos especializados, inclusive sin que el usuario se dé cuenta; o pueden ser robadas o extraviadas, permitiendo ingresar a cualquier persona que la posea.
- **Dígitos numéricos PIN:** Generalmente los usuarios eligen identificaciones sencillas o predecibles. También se pueden olvidar fácilmente.
- **Sistemas biométricos:** Tienen un alto costo y bajo nivel de confiabilidad. Se corre un alto riesgo si la base de datos es alterada o robada, ya que la información biométrica, al contrario de las contraseñas, no se puede cambiar.

Con respecto al *software* que utilizan los sistemas, muchas veces las interfaces de usuario en la aplicación de administración son poco amigables, y son desarrolladas con tecnologías obsoletas o diseñadas para versiones de sistemas operativos específicos. Además, los fabricantes publican pocas o ninguna actualización, lo que lleva a que el *software* pueda presentar posibles vulnerabilidades en el código fuente y sea un objetivo de ataque. Aunado a esto, se debe pagar un costo por licenciamiento del *software*, el cual suele ser costoso. A nivel de las comunicaciones de red, los datos no siempre viajan mediante protocolos seguros, por lo que podrían ser escuchados por terceros dentro de la red o se utilizan certificados autofirmados que pueden ser falsificados.

Los SCA actuales no incluyen una forma automatizada para controlar el ingreso y salida de los visitantes. En el caso de los vehículos, el encargado de vigilancia debe llenar un documento con los datos personales de los ocupantes del automóvil, el modelo, número de placa, hora de ingreso y salida, proceso que suele ser lento y perjudica el tiempo de atención.

Para administrar y gestionar todos los dispositivos conectados a un SCA, se requiere de un controlador que integre y almacene toda la información, y ejecute acciones. Estos controladores pueden ser dispositivos autónomos, los cuales tienen memoria, capacidad de procesamiento y almacenamiento limitado; o pueden ser servidores completos con altas capacidades que involucran un costo muy elevado de implementación.

El alto costo de las implementaciones de SCA de calidad impiden a las pequeñas y medianas empresas adquirir las soluciones para su uso. Por esta razón, optan por seguir utilizando los métodos tradicionales o soluciones de SCA inseguras y de mala calidad, exponiéndose a los riesgos inherentes de su uso.

1.3 Justificación

En el mercado actual es difícil encontrar empresas que ofrezcan un sistema de control de acceso seguro y accesible, que pueda ser utilizado desde un dispositivo móvil, por medio de una aplicación.

Este proyecto se elaboró para satisfacer la necesidad de las empresas que requieren un sistema de control seguro para el acceso personal, vehicular y de invitados. Debe ser de fácil configuración, según los requerimientos y características de los distintos clientes, que permita el monitoreo de los accesos de los usuarios, y que tenga una interfaz gráfica amigable y moderna. Además, se pretende que sea accesible para pequeñas y medianas empresas, debido a su precio razonable.

Para esto se van a desarrollar alternativas que permitan resolver el problema de control de acceso haciendo uso de la tecnología actual, aplicándola a los dispositivos que actualmente no están conectados a internet, para que puedan transferir datos a la nube, donde se puedan administrar, analizar y transformar en aplicaciones prácticas.

Una adecuada implementación del sistema permitirá a las empresas reducir los costos indirectos, disminuir el tiempo de registro de visitantes, administrar los dispositivos de control de acceso desde una interfaz web, y obtener un rápido retorno de la inversión.

A continuación, se muestran los beneficios que se tendrán al utilizar el sistema de control de accesos propuesto.

1.3.1 Beneficios técnicos

- Se pueden desarrollar otros sistemas de información que utilicen los datos generados por el sistema de control de accesos, por medio de un API (*Application Programming Interface*).

- Se utiliza una plataforma en la nube de internet de las cosas (IoT), que asegura una alta disponibilidad del servicio.
- Al utilizar componentes de *hardware* libre se facilita el mantenimiento y reparación del dispositivo.
- El sistema operativo es *software* libre, el cual presenta constantes actualizaciones y parches de seguridad.

1.3.2 Beneficios operativos

- Disminución del tiempo utilizado en el registro de visitantes, ya que tendrán un acceso especial mediante una aplicación.
- El sistema se utiliza por medio de una aplicación web disponible desde internet.
- No se necesita realizar respaldos locales de la información, ya que los datos se almacenan en la nube, lo cual reduce los costos de administración del servicio.

1.3.3 Beneficios económicos

- No es necesario adquirir servidores o PC adicionales para el funcionamiento del sistema.
- El proyecto utiliza licencias de *software* libre que no tienen costo.
- El *hardware* libre utilizado permite un bajo costo del dispositivo.
- No se requieren mecanismos lectores (tarjetas, llaves, entre otros) para abrir y cerrar los dispositivos como puertas o agujas, lo cual disminuye el costo de implementación.

1.4 Objetivos

Para la definición de los objetivos del proyecto se utiliza la taxonomía de Bloom (Bloom, 1990), por su enfoque holístico en 6 niveles jerárquicos para el aprendizaje cognitivo.

1.4.1 Objetivo general

Crear un sistema de control de accesos seguro, basado en internet de las cosas (IoT), utilizando tecnologías de *hardware* y *software* libre.

1.4.2 Objetivos específicos

- Contrastar los distintos tipos de *hardware* libre para la implementación de la solución.
- Diseñar la arquitectura de componentes de *software* y *hardware* de la solución.
- Seleccionar el mejor método de control de acceso a utilizar.
- Configurar el *hardware* necesario para la comunicación entre la aplicación y los dispositivos de control de acceso.
- Desarrollar el *software* necesario para la configuración, utilización y monitoreo del sistema, utilizando buenas prácticas de ciberseguridad.
- Integrar el *software* y *hardware* de la solución.
- Crear un modelo a escala de la solución, con el propósito de demostrar la viabilidad de su implementación.

1.5 Alcances y limitaciones

1.5.1 Alcances

En este proyecto desarrolló un dispositivo seguro para el control de accesos, utilizando componentes de *hardware* libre, basado en una plataforma en la nube para su configuración, utilización y monitoreo.

Se desarrolló una aplicación web que permite la configuración y administración del dispositivo, así como un módulo para el ingreso de visitantes, quienes la pueden utilizar desde un teléfono celular con internet.

Adicionalmente, se creó un modelo a escala de una implementación de la solución completamente funcional, el cual permite demostrar las funcionalidades del sistema.

1.5.2 Limitaciones

- El desarrollo de las aplicaciones realizó mediante desarrollo web. Por limitantes de tiempo del proyecto no se realizaron aplicaciones nativas para móviles.
- No se crearon manuales de usuario o manuales técnicos del uso del sistema.
- Debido a las diferentes variables de entorno de cada empresa, el proyecto se limita a la creación del dispositivo, pero no a la implementación del sistema de control de accesos.

1.6 Estado de la cuestión

El tema de control de accesos físicos es sumamente importante en el área de ciberseguridad. No obstante, estos sistemas no han evolucionado en los últimos años y la mayoría están basados en soluciones con vulnerabilidades conocidas, como lo es RFID, códigos numéricos, entre otros. A pesar de que existen otras tecnologías de punta, como los dispositivos biométricos, estos son poco utilizados, debido a su alto costo de implementación y su bajo nivel de confiabilidad. A continuación, se muestra un análisis de los sistemas actuales de control de acceso disponibles en el mercado, incluyendo sus ventajas, desventajas y características básicas.

Tabla 1

Características del sistema de control de accesos NetAXS-123

Solución 1: NetAXS-123	
Modelo	NetAXS-123 
Fabricante	Honeywell
Descripción	Sistema de control de acceso basado en red, que permite a los administradores gestionar el sistema de acceso físico a puertas, en cualquier lugar donde haya una conexión a red, sin un servidor dedicado.
Ventajas	● Acceso web para administración y configuración. ● Tiene una batería de 12V en caso de que se corte el suministro eléctrico. ● No requiere un servidor adicional para su funcionamiento.
Desventajas	● Este sistema solamente es capaz de controlar puertas con lector RFID. ● El dispositivo solo soporta una puerta. Si se requieren más puertas, se deben conseguir módulos adicionales con un costo adicional elevado. Si se quiere conectar más de 3 puertas, se necesita adquirir otro dispositivo NetAXS-123. ● Interfaz de usuario web obsoleta. Utiliza certificados digitales autofirmados y se necesita de Internet Explorer 9 para funcionar. Esto lo convierte en una plataforma insegura y vulnerable a ataques. ● Código privativo. No se pueden conocer los mecanismos de seguridad que utiliza.
Costo Estimado	Control de accesos para 2 puertas (solo del dispositivo) Panel de control NetAXS-123: $985 Módulo 1 puerta: $415 Total: $1400

Fuente: elaboración propia con los datos de Honeywell (2017).

Tabla 2

Características del sistema de control de accesos UT0814-S4110V

Solución 2: UT0814-S4110V	
Modelo	UT0814-S4110V 
Fabricante	UHPPOTE
Descripción	Sistema de control de accesos en red, que permite el manejo de 4 puertas. Los lectores y tarjetas de acceso son de tecnología RFID. Soporta la creación de hasta 20.000 usuarios.
Ventajas	● Incluye soporte para 4 puertas. ● Incluye el lector y las tarjetas de acceso.
Desventajas	● Se necesita instalar *software* adicional en una computadora para su configuración y administración. ● El *software* de administración no cuenta con características de seguridad. ● No tiene interfaz web. ● El *software* no está en español ● Los lectores utilizan tecnología RFID 125Khz barata, con lo cual las tarjetas pueden ser clonadas fácilmente, y con esto desvirtúa el objetivo del dispositivo. ● El fabricante no tiene página web y el soporte al usuario y los manuales son muy limitados, lo cual no permite conocer el nivel real de seguridad del sistema.
Costo Estimado	Control de accesos para 4 puertas Incluye panel de control, 4 lectores, 4 tarjetas y fuente de poder. Total: $435.15

Fuente: elaboración propia con los datos de Amazon (2017b).

Tabla 3

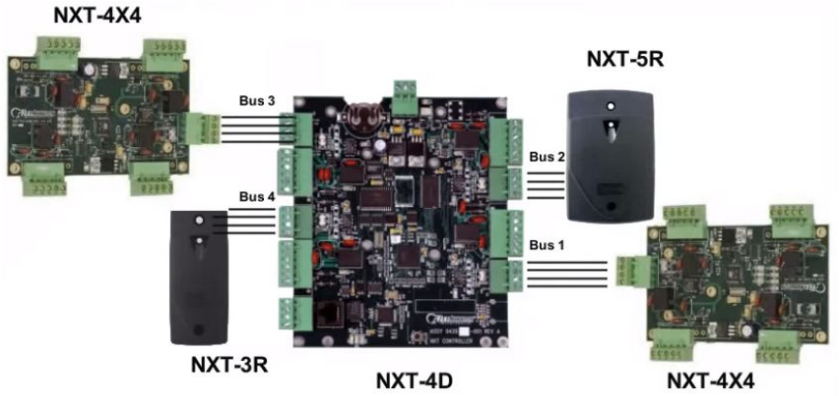
Características del sistema de control de accesos Suprema Biostation T2

Solución 3: Suprema Biostation T2	
Modelo	Biostar 
Fabricante	Suprema
Descripción	Sistema de control de acceso basado en conectividad IP y seguridad biométrica.
Ventajas	● Puede manejar hasta 512 puertas. ● Se pueden manejar zonas de acceso y alarmas. ● Permite definir tiempos, turnos, horarios por días y feriados para el control de accesos.
Desventajas	● Cada puerta necesita un lector inteligente especializado para utilizarse con el sistema. ● Cada lector es muy costoso. ● Se requiere un servidor adicional como controlador del sistema, lo cual incrementa el costo del sistema. ● Se basa en tecnología biométrica de huellas digitales. Se requiere tomar los datos biométricos de cada usuario y almacenarlo en el dispositivo. ● Se necesita instalar y configurar *software* adicional en los equipos que se quiere administrar. ● El *software* no está en español
Costo Estimado	Costo de 1 lector: $2281.56 Costo del servidor: $1600 Total: $3881.56

Fuente: elaboración propia con los datos de Suprema Solution (2017).

Tabla 4

Características del sistema de control de accesos Keri Systems NXT-4D-MSC

Solución 4: Keri Systems NXT-4D-MSC	
Modelo	NXT-4D-MSC 
Fabricante	Keri Systems
Descripción	Sistema de control de acceso basado en conectividad IP. Utiliza cables CAT5 para conectar los dispositivos.
Ventajas	● Permite el descubrimiento de *hardware* confiable y verdadero a través de identificación por MAC Address. ● Permite controlar hasta 4 puertas, 8 lectores. ● Es expandible con el uso de otras tarjetas expansoras.
Desventajas	● Requiere una aplicación cliente servidor construida en .NET para su administración. Su desventaja es que solamente es compatible con Windows. ● El *software* no está en español
Costo Estimado	Panel de control NXT-4D $1815.51 Fuente de poder KPS-5-1 $128.26 Carcasa: $253.00 Total: $2197.77

Fuente: elaboración propia con los datos de Keri Systems Incorporated (2017).

Tabla 5

Características del sistema de control de accesos Paxton PROXIMITY

Solución 5: Paxton PROXIMITY	
Modelo	Paxton PROXIMITY 
Fabricante	Paxton
Descripción	Es un dispositivo autónomo de control de accesos que utiliza un código pin o una tarjeta magnética.
Ventajas	● Es un sistema compacto. ● No requiere servidores o controladores adicionales.
Desventajas	● Toda la electrónica de control se encuentra en el interior del lector o teclado. Este tipo de sistema no se recomienda para las puertas exteriores o interiores de alta seguridad. ● No tiene control por usuarios, solamente es un código de acceso universal para todos. ● No mantiene un registro de las entradas y salidas. ● Se programa directamente con el teclado numérico. Si no se cambia la contraseña por defecto, cualquier persona que la conozca puede tener acceso a la puerta, lo cual es un grave problema de seguridad.
Costo Estimado	Lector Paxton Proximity: $293.97 Total: $293.97

Fuente: elaboración propia con los datos de Paxton (2017).

Tabla 6

Características del sistema de control de accesos EverFocus NAV-04-1A

Solución 6: EverFocus NAV-04-1A	
Modelo	EverFocus NAV-04-1A 
Fabricante	EverFocus
Descripción	Sistema de control de accesos basado en Linux, con una interfaz web para administración.
Ventajas	● Requiere una llave para abrir la carcasa contenedora. ● Permite el control de 2 puertas.
Desventajas	● No se tiene claro la versión de Linux utilizada. El sistema operativo puede tener vulnerabilidades conocidas y el sistema no recibe actualizaciones periódicas. ● Tiene un almacenamiento compartido de 512MB, el cual se puede llenar con los registros de la base de datos. ● Requiere 2 lectores para el funcionamiento de cada puerta: un lector afuera de la puerta y otro adentro (*Anti-Passback*). Esto eleva los costos en adquisición de lectores adicionales.
Costo Estimado	Panel de control $2314.15 Total: $2314.15

Fuente: elaboración propia con los datos de Everfocus Electronics Corp (2017).

Tabla 7

Características del sistema de control de accesos Keyscan CA4500

Solución 7: Keyscan CA4500	
Modelo	Keyscan CA4500 
Fabricante	Keyscan
Descripción	Sistema de control de acceso diseñado para una sola puerta o un sistema complejo de muchas puertas.
Ventajas	● Almacena 45.000 credenciales en la memoria. ● Múltiples entradas y salidas para expansión. ● Permite abrir 4 puertas. ● Batería de respaldo opcional. ● Mantiene un sistema de alarmas que monitorea el acceso a las puertas.
Desventajas	● La conectividad TCP/IP no viene incluida y se tiene que adquirir un módulo separado. ● Su proceso de configuración es complejo, ya que involucra la configuración de puertos seriales en un equipo, y su configuración es mediante consola. ● Requiere un *software* adicional para su administración. ● No se conoce la seguridad en la comunicación de las conexiones de red que realiza.
Costo Estimado	Panel de control $2453.4 Total: $2453.4

Fuente: elaboración propia con los datos de Keyscan Access Control Systems (2017).

Tabla 8

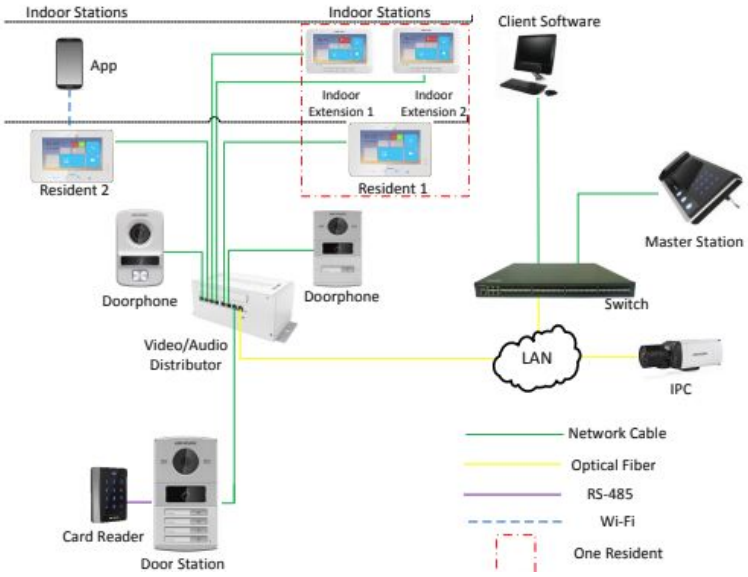
Características del sistema de control de accesos Mountain One Video Door Bell

Solución 8: Mountain One Video Door Bell	
Modelo	
Fabricante	Mountainone
Descripción	Dispositivo de control de acceso con intercomunicador incorporado.
Ventajas	● Posee un intercomunicador de audio y video incorporado. ● Incluye un control remoto con 20 metros de rango.
Desventajas	● No se pueden configurar diferentes pines para el acceso a usuarios. ● Se necesita instalar *software* adicional en una computadora para su configuración y administración. ● No tiene ninguna opción de personalización o configuración. ● Los lectores utilizan tecnología RFID 125Khz barata, con lo cual las tarjetas pueden ser clonadas fácilmente, lo que desvirtúa el objetivo del dispositivo. ● El fabricante no tiene página web. El soporte al usuario y los manuales son muy limitados, lo cual no permite conocer el nivel real de seguridad del sistema. ● La calidad de los componentes es muy baja. ● No es expandible a más puertas. Solo es compatible con una puerta.
Costo Estimado	Incluye panel intercomunicador, 4 tags RFID, cierre magnético, control remoto. Panel de control $180 Total: $180

Fuente: elaboración propia con los datos de Amazon (2017c).

Tabla 9

Características del sistema de intercomunicación HikVision Video Intercom Door Station

Solución 9: HikVision Video Intercom Door Station	
Modelo	DS-KV8102-IM 
Fabricante	HikVision
Descripción	Dispositivo de intercomunicación de video y control de acceso.
Ventajas	● Posee un intercomunicador de audio y video incorporado. ● Toma una fotografía cuando se abre la puerta. ● Posee una aplicación móvil para el control de acceso a puertas. ● Compatibilidad para agregar un lector de tarjetas.
Desventajas	● La aplicación solo funciona dentro de la misma red. ● Se requiere de un dispositivo “Doorphone” por cada puerta. ● El producto está muy enfocado al entorno del hogar. ● No se especifica cuál es la tecnología del lector de tarjetas. ● La interfaz de administración utiliza contraseñas inseguras: 12345. ● Se requiere un disco duro adicional para guardar la información.
Costo Estimado	Incluye panel intercomunicador, comunicador interno pantalla 7”, distribuidor de audio y video y montura de pared. Total: $388.57

Fuente: elaboración propia con los datos de HikVision (s.f).

Tabla 10

Características del sistema de control de acceso INT-UNC100

Solución 10: Controlador Universal INT-UNC100	
Modelo	INT-UNC100 
Fabricante	RBH Access
Descripción	Sistema de control de acceso basado en conectividad IP.
Ventajas	● Permite controlar hasta 2 puertas. ● Utiliza Power over Ethernet (PoE). ● El canal abordo de RS485 hace que sea fácil proporcionar conectividad a otros dispositivos RBH y a *hardware* de terceros, tales como: lectores, paneles de alarma antirrobo, sistemas de automatización, etc.
Desventajas	● Máxima cantidad de registro de eventos: 3000. ● Utiliza el protocolo inseguro Telnet con el *password* por defecto: "Password" para administración. ● Requiere un *software* de administración que se vende por aparte para su administración. Solamente es compatible con Windows.
Costo Estimado	Panel de control INT-UNC100, puente de Poder 12V, carcasa Total: $3,127

Fuente: elaboración propia con los datos de RBH Access Technologies Inc (2014).

Capítulo 2
Marco teórico

2.1 Control de acceso

The Internet Engineering Task Force (2007) define los controles de acceso como "un proceso mediante el cual el uso de los recursos del sistema está regulado de acuerdo con una política de seguridad y sólo está permitido por entidades autorizadas (usuarios, programas, procesos u otros sistemas) de acuerdo con esa política" (p.11).

Un control de acceso previene el acceso no autorizado a un activo. Además, permite definir los usuarios que acceden a los recursos, qué pueden hacer, a cuáles zonas tienen permiso de acceso y qué procesos puede realizar.

Kim & Solomon (2010) proponen que un control de acceso se compone de cuatro partes:

- **Autorización:** quién está autorizado para acceder y cuáles recursos puede utilizar.
- **Identificación:** cómo los usuarios son identificados.
- **Autenticación:** cómo se puede verificar la identidad de un usuario.
- **Responsabilidad:** cómo se rastrean las acciones de un usuario para identificar los cambios realizados y poder ejecutar reportes posteriormente.

Los controles de acceso se dividen en dos tipos:

- **Controles de acceso físicos:** puntos de acceso a edificios, parqueos, áreas protegidas. Ejemplo: agujas para parqueo y puertas con control de acceso.
- **Controles de acceso lógicos:** controles de acceso a un sistema informático o una red. Ejemplo: usuario y contraseña.

2.1.1 Mecanismos de control de acceso

Un sistema de control de acceso utiliza un mecanismo que verifica la identificación del usuario que solicita acceso a un recurso, con el objetivo de permitir o denegar el acceso basado en sus credenciales. A continuación, se describen algunos mecanismos que utilizan los sistemas control de acceso.

2.1.1.1 Código de barras

Se trata de un código con una serie de líneas paralelas negras de distinto grosor y espacios en blanco que representan caracteres como letras, números y símbolos, el cual se imprime y se adhiere a un artículo que se quiera identificar. Fue creado en el año 1952 por los inventores Joseph Woodland y Bernard Silver en Estados Unidos (Estados Unidos, patente nº US 2612994, 1952).



Figura 1. Código de barras EAN-13 (VaGla, 2007).

Son muy utilizados en supermercados, hospitales, equipaje de avión, tiquetes, paquetería, inventarios, entre otros. Esto es debido a que son una forma rápida de leer un código para luego procesarlo en un sistema informático.

Para leer un código de barras se necesita un lector. El lector es un dispositivo que, por medio de un láser, es capaz de leer el código de barras y decodificar la información incluida para transmitirla vía USB, bluetooth o wifi a la computadora o dispositivo al cual se encuentra conectado para su procesamiento.

2.1.1.2 Código QR

El código QR (*Quick Response*) es un tipo de código de barras que almacena su información en una matriz de barras bidimensional. Se compone de tres cuadrados grandes en las esquinas para ubicar el lector, seguido de cuadrados negros sobre un fondo blanco. Pueden almacenar hasta 4296 caracteres alfanuméricos (Denso Wave Incorporated, s.f.).



Figura 2. Código QR.
(Denso Wave Incorporated, 2011).

Tienen una ventaja con respecto a los códigos de barras tradicionales, y es que no se requiere de un lector especializado, porque están diseñados para ser leídos por cámaras de forma muy rápida. Se puede utilizar una cámara web o una cámara de un teléfono celular para leer estos códigos.

2.1.1.3 Teclado numérico

Es uno de los mecanismos de control de acceso más utilizados, ya que suele ser económico. Está formado por un teclado numérico en la entrada, el cual se comunica con un sistema electrónico para habilitar el paso una vez digitado el código correcto.



Figura 3. Teclado numérico.
(Assa Abloy, 2017).

2.1.1.4 RFID

La identificación por radiofrecuencia o RFID es una tecnología que comunica un lector y una etiqueta unida a un producto, por medio de una conexión inalámbrica de campos electromagnéticos por medio de ondas de radio.

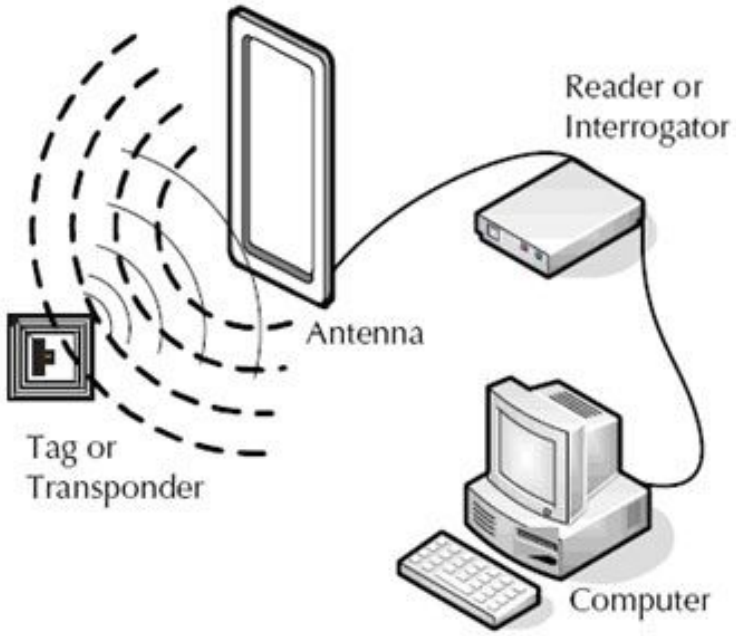


Figura 4. Proceso de lectura de RFID.
(Barcodes Inc., s.f).

Las etiquetas contienen información almacenada electrónicamente en un chip. A diferencia de los códigos de barras, las etiquetas no tienen que estar dentro de la línea de visión del lector, por lo que son muy utilizadas en dispositivos industriales en fábricas en procesos automatizados o en grandes almacenes.

Existen 2 tipos de etiquetas RFID:

- **Etiquetas activas:** este tipo de etiqueta posee una fuente de alimentación propia (usualmente una batería) y pueden operar a cientos de metros del lector RFID.
- **Etiquetas pasivas:** las etiquetas pasivas recogen energía a partir de las ondas de radio del lector RFID cercano. Tiene un alcance de pocos centímetros. Pueden operar en diferentes frecuencias, las cuales le aumentan o disminuyen el rango:
    - 125-134 Khz: Baja frecuencia (LF): de 1 a 10 centímetros de rango.
    - 13.56 Mhz: Alta frecuencia (HF)(NFC): de 1 centímetro a 1 metro.
    - 865-960 Mhz: Ultra alta frecuencia (UHF): 5 a 6 metros.

Estas etiquetas vienen en distintos factores de forma y tamaños. El más utilizado en sistemas de control de acceso son las tarjetas inteligentes RFID, las cuales son un tipo de etiqueta pasiva.



Figura 5. Tarjetas RFID.
(Amazon, 2017).

2.1.1.5 Sistemas biométricos

Los sistemas biométricos son sistemas capaces de identificar mediante una característica única a una persona. Usan, por ejemplo, huellas, ojos, reconocimiento facial, voz, entre otros. Entre los sistemas biométricos más comunes se encuentran:

Huellas digitales: se registra el patrón de crestas y valles en la punta de un dedo. Una huella digital es única.

Reconocimiento facial: identifica a la persona mediante el análisis de los rasgos faciales obtenidos de un video o una imagen. Los sistemas más modernos utilizan tecnologías de modelado en 3D de la cara para detectar rasgos faciales como ojos abiertos y movimiento.

Además, utilizan "machine learning" para aprender sobre las diferencias que se puedan dar en un rostro cuando, por ejemplo, tiene barba o anteojos.

Iris del ojo: los ojos tienen pequeñas fibras o pecas únicos, este sistema utiliza un pequeño grabador de vídeo para capturar esos patrones. Los escaneos de iris son muy precisos para la identificación y autenticación. El tiempo de respuesta es de 1-2 segundos.

Voz: se capturan parámetros únicos de la voz para identificar a la persona. La precisión se puede reducir por el ruido de ambiente. El tiempo de respuesta puede llegar hasta 14 segundos. Debido a este largo tiempo de respuesta, además de que podría ser fácilmente reproducido por cualquier dispositivo (teléfono, computadora), no es popular para el uso diario.

Firmas: se colocan sensores en una tableta de escritura y se usan para registrar la velocidad, la dirección y la presión del movimiento del lápiz. Este tipo de mecanismo biométrico es bastante preciso, aunque es poco usado.

La biometría hace innecesario portar un dispositivo para autenticarse, además de que elimina la tarea de recordar contraseñas. Los costos de implementación para una empresa son muy altos, debido a sus lectores especializados y mantenimiento. Por tales razones, esta tecnología no es muy utilizada.

2.1.1.6 *One Time Password*

Un *One-Time Password* (OTP) es una contraseña válida una única vez para una sesión o transacción de un usuario. Se utilizan mucho en autenticación de doble factor, ya que solucionan las deficiencias con las contraseñas tradicionales. Son resistentes a ataques de fuerza bruta porque son de solo un uso y esto frustra los intentos posteriores.

Existen varios acercamientos de cómo implementar un OTP. The Internet Engineering Task Force (2005) creó un algoritmo para crear OTP basado en el código de

autenticación de mensajes hash (HMAC) y SHA-1 al que llamaron HOTP (*HMAC-based One Time Password*). Este algoritmo se propone para utilizar en aplicaciones de red como VPN y Wi-Fi.

2.2 *Hardware* libre

El *hardware* libre nos da la libertad de estudiar y acceder a recursos como esquemas, diagramas, código, archivos. Esta autonomía nos da la posibilidad de remezclar, modificar y vender un *hardware* creado (Pearce, 2014).

Con los dispositivos de *hardware* privativo, no es posible entender el *hardware* y muchas veces se toman medidas para dificultar el acceso a sus componentes, como utilización de herramientas especiales para su desensamble.

Con el uso de *hardware* libre, los usuarios pueden construir sus propios dispositivos y muchas veces a un costo mucho menor de otros dispositivos en el mercado. A continuación, se especifican algunos dispositivos de *hardware* libre.

2.2.1 Arduino

Arduino es una placa electrónica de código abierto basada en *hardware* y *software* fáciles de usar, que tienen la funcionalidad de leer entradas y convertirlas en salidas. Utiliza el lenguaje de programación Arduino, el cual, a su vez, se encuentra basado en el lenguaje de programación Wiring y utiliza un entorno de desarrollo integrado llamado también Arduino.

Arduino nació en el año 2005 en el *Ivrea Interaction Design Institute* de Italia, como una herramienta fácil para el prototipado rápido, dirigido a estudiantes sin experiencia en electrónica ni programación.

Desde su creación se ha utilizado en miles de proyectos, como productos de electrónica simple, portátiles, impresión 3D, aplicaciones IoT y hasta complejos instrumentos científicos. Todo esto es debido a que Arduino es de código abierto, lo que le permite a los usuarios construir los proyectos y adaptarlo a sus necesidades particulares.

Han existido varias versiones de Arduino, en este momento, el Arduino Uno Rev 3 es la versión actual.



Figura 6. Arduino Uno Rev 3.
(Arduino, 2017).

Arduino Uno es una placa electrónica basada en el chip ATmega328P. Cuenta con 14 pines digitales de entrada/salida, 6 entradas analógicas, un cristal de cuarzo de 16MHz, una conexión USB, un conector de alimentación y un botón de reinicio.

Tabla 11
Especificaciones técnicas de Arduino

Especificaciones	Arduino Uno Rev 3
Microcontrolador	ATmega328P
Voltaje operativo	5 V
Voltaje de entrada límite	6-20 V
Entradas y salidas digitales	14
Entradas analógicas	6
Memoria	32KB Flash, SRAM 2KB, EEPROM 1KB
Frecuencia de reloj	16 MHz

Peso	25 gramos
Costo	$22

Fuente: elaboración propia con los datos de Arduino (2017).

2.2.2 Intel Galileo

El Intel Galileo es una tarjeta microcontroladora basada en el procesador de aplicaciones Intel Quark SoC X1000. Es la primera placa basada en la arquitectura Intel, diseñada para ser compatible con el *hardware* y *software* de Arduino Uno R3 (Intel, 2014).



Figura 7. Intel Galileo Gen 2.
(Intel, 2014).

Los pines del Galileo se encuentran en la misma ubicación que en el Arduino Uno R3 y son compatibles con *software* del Entorno de Desarrollo de *Software* Arduino (IDE). Como complemento al *hardware* del Arduino, Galileo tiene una ranura mini-PCI Express, un puerto Ethernet 100Mb, una ranura MicroSD, un puerto serie RS-232 y 2 puertos USB.

Fue diseñado para ser *hardware* libre, por lo que se pueden descargar los esquemas, archivos y componentes.

Cabe agregar que, el 5 de julio del 2017, Intel anunció que va a descontinuar el producto y no hay planes de sacar nuevas versiones. Sin embargo, todos los recursos *online* se mantendrán en el sitio web de Intel hasta el 15 de junio del 2020. Los archivos de código fuente y *software* libre se almacenarán en *Github*. (Intel, 2017).

2.2.3 Raspberry Pi

El Raspberry Pi es una computadora de bajo costo del tamaño de una tarjeta de crédito, al cual se le puede conectar un monitor y dispositivos USB como teclado y *mouse*, que permite la navegación en internet, juegos y reproducción de video, entre otras tareas (Raspberry Pi Foundation, 2017a).

Fue inventado en el año 2011 por Eben Upton y desarrollado en Reino Unido por la Fundación Raspberry PI en la Universidad de Cambridge. Se creó con el objetivo de ayudar a las personas a aprender, resolver problemas y divertirse mediante la enseñanza de la informática. Es muy utilizado en proyectos de electrónica, debido a su bajo consumo y puertos GPIO (*General Purpose Input/Output*), los cuales permiten conectar todo tipo de dispositivos de electrónica digital, como LED, LCD, relay, entre otros. Existen varios modelos de Raspberry Pi, a continuación, se muestran los modelos disponibles para la venta actualmente:

Tabla 12
Especificaciones técnicas de los modelos de Raspberry Pi

Modelo	Velocidad Procesador	Memoria RAM	GPIO	Puertos USB	Puerto de Red	Wireless /Bluetooth	Costo
Raspberry Pi Model A+	700Mhz	512MB	40	1	No	No	$20
Raspberry Pi Model B+	700Mhz	512MB	40	4	Si	No	$25
Raspberry Pi 2 Model B	900Mhz	1GB	40	4	Si	No	$35
Raspberry Pi 3 Model B	1200Mhz	1GB	40	4	Si	Si	$35
Raspberry Pi Zero	1000Mhz	512MB	16	1	No	No	$5

Raspberry Pi Zero W	1000Mhz	512MB	16	1	No	Si	$10

Fuente: elaboración propia con los datos de Raspberry Pi Foundation (2017a).

El modelo recomendado es el Raspberry Pi 3 Model B el cual se lanzó en febrero de 2016. Entre sus especificaciones técnicas se encuentra:

- CPU ARM Cortex-A53 de cuatro núcleos de 64 bits (SoC).
- 1 GB de memoria RAM.
- Tarjeta LAN inalámbrica integrada 802.11n y Bluetooth 4.1.
- Salida analógica de audio estéreo por Jack de 3.5 mm.
- Pines de entrada y salida de propósito general (GPIO).
- Conector de alimentación microUSB.
- Lector de tarjetas SD.

Este modelo tiene unas dimensiones de 85.60x56x21 milímetros y pesa 45g. Comparado con los modelos anteriores, el Raspberry Pi 3 Model B reemplazó el *slot* de memoria SD por un *slot* de memoria micro-SD, y tiene un consumo de energía menor a los modelos anteriores. Una característica muy llamativa es su costo, el cual ronda los $35.



Figura 8. Raspberry Pi 3 Model B.
(Raspberry Pi Foundation, 2017b).

El Raspberry Pi Zero tiene la mitad del tamaño de un modelo A+, con un CPU de un solo núcleo de 1Ghz, 512MB de RAM, puertos mini-HDMI, USB y un conector de cámara. Tiene unas dimensiones de 65 mm x 30 mm x 5.4 mm y pesa 9g.



.

Figura 9. Raspberry Pi Zero W.
(Raspberry Pi Foundation, 2017c).

El Raspberry Pi Zero W tiene las mismas especificaciones que el modelo Raspberry Pi Zero, solo que además incluye una tarjeta LAN inalámbrica integrada 802.11n y Bluetooth 4.1.

2.2.3.1 GPIO

Una característica de mucha importancia en el Raspberry Pi son los puertos GPIO. Estos son unos pines que se encuentran en la tarjeta principal del Raspberry Pi y son

interruptores que se pueden encender o apagar para controlar dispositivos de *hardware*.

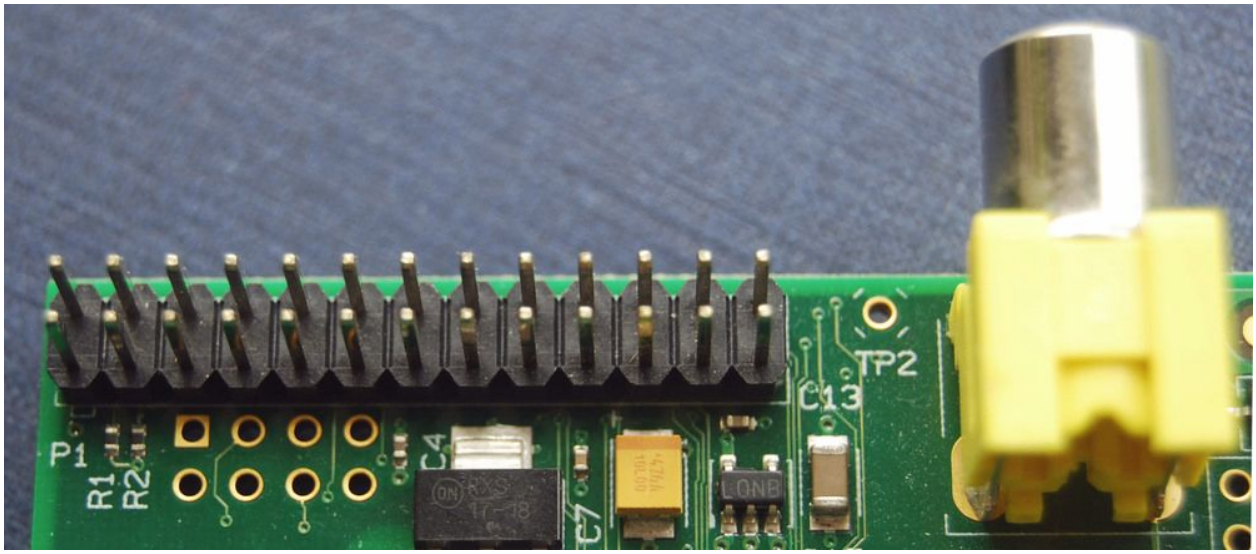


Figura 10. Pines GPIO.
(Raspberry Pi Foundation, s.f.).

Estos pines pueden ser programados como entrada o salida. Por ejemplo, un pin configurado como entrada podría leer las señales de un sensor de luz o un sensor ultrasónico. Por otra parte, un pin configurado como salida podría encender un LED, un *relay* o enviar una señal a otro dispositivo.

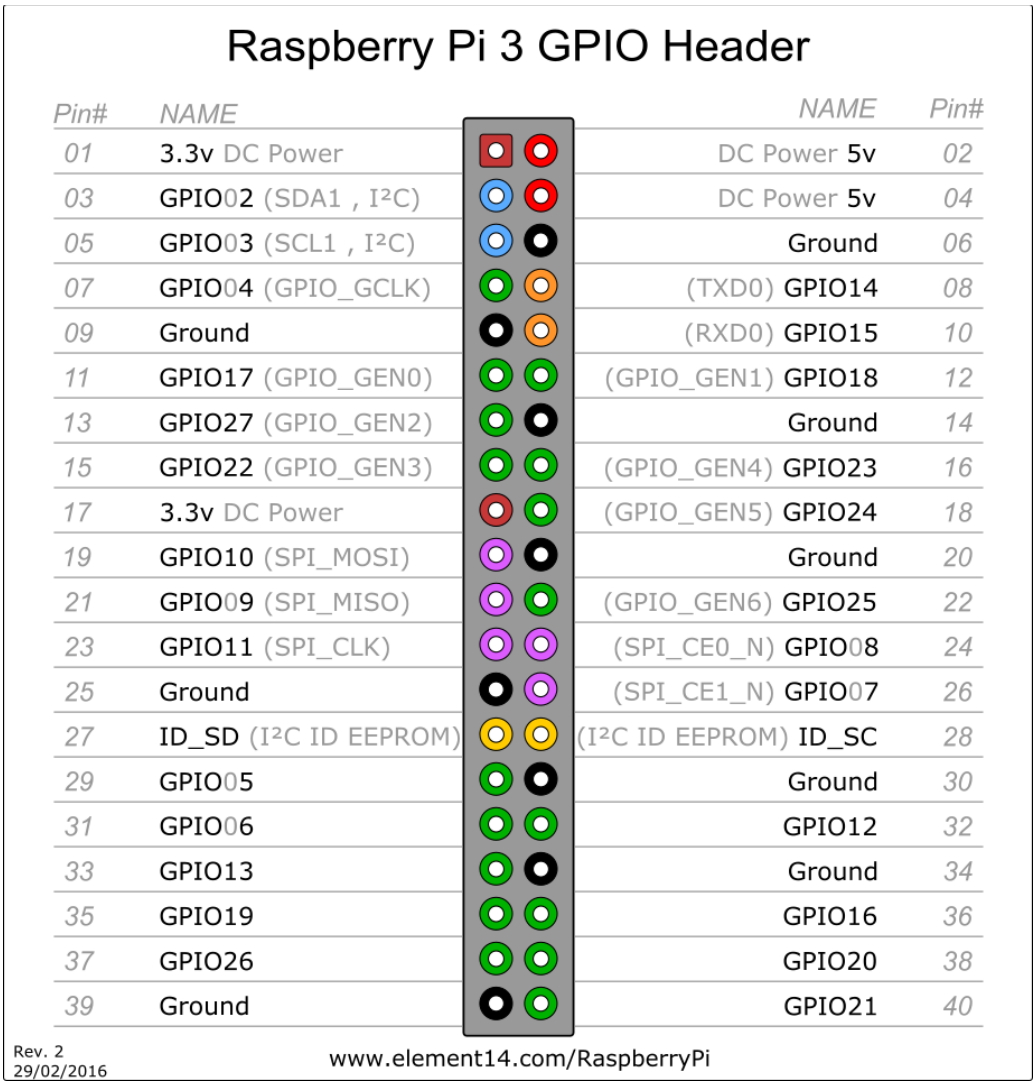


Figura 11. Descripción de pines GPIO (Element14, 2015).

Solamente 17 de los 26 pines son GPIO, los demás son de 3.3 voltios o 5 voltios, y 8 pines de tierra. Los GPIO pueden ser configurados desde una red local, o inclusive mediante internet, por lo que son muy utilizados en proyectos de domótica o proyectos de electrónica en general.

Con respecto al sistema operativo utilizado, existen varias distribuciones oficiales disponibles como Ubuntu Snappy Core, Ubuntu Mate, Risc OS, Windows 10 IoT Core o la distribución recomendada: Raspbian. Al ser un sistema abierto, se crean distribuciones no

oficiales creadas por la comunidad de usuarios para labores distintas como Kali Linux, LibreELEC, Pidora o Pi-topOS.

2.2.3.2 Raspbian

Es un sistema operativo de código abierto creado por la Raspberry Foundation. Está optimizado para trabajar junto con el hardware del Raspberry Pi. Es una distribución derivada del sistema operativo GNU/Linux llamado Debian 8.0 y se lanzó durante julio de 2012.

Se instala por medio de la ranura de memoria SD y ocupa 4Gb de espacio para su instalación. Sin embargo, necesita una memoria de mínimo 8GB para su correcto funcionamiento (Raspberry Pi Foundation, 2017a).

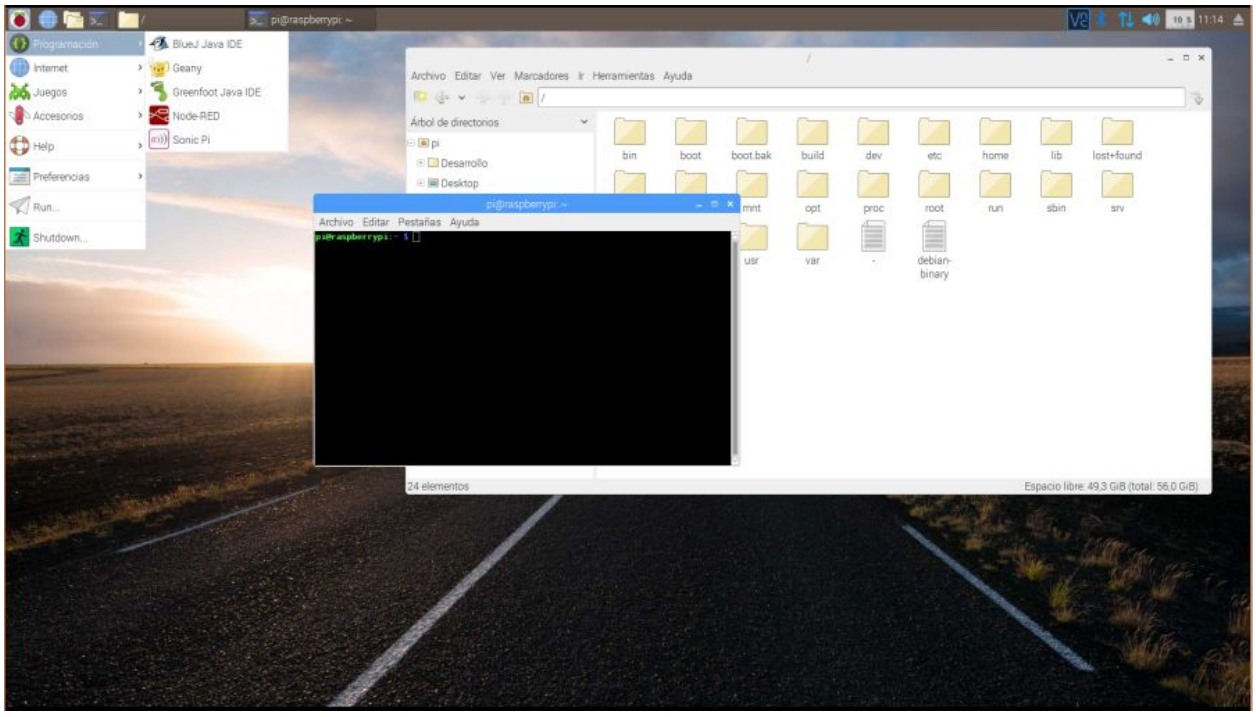


Figura 12. Sistema operativo Raspbian.
Fuente: elaboración propia.

Además, Raspbian utiliza un entorno de escritorio liviano llamado PIXEL (*Pi Improved X Windows Environment*), que es una versión modificada del entorno de escritorio LXDE (*Lightweight X11 Desktop Environment*).

Viene preinstalado con paquetes de *software* como Mathematica, entornos de desarrollo para aplicaciones en Python, PHP, Java, Ruby como Idle, GreenFoot o BlueJ, los cuales permiten la interacción del *software* con el *hardware* por medio de los GPIO.

2.3 Software libre

Según la Free Software Foundation (2001, párr 3) "el *Software* [*sic*] libre es el software que respeta la libertad de los usuarios y la comunidad. A grandes rasgos, significa que los usuarios tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el *software*".

El *software* libre es muy ventajoso, debido a que no hay que pagar ningún costo de licencia y, aunque los trabajos derivados pueden ser vendidos comercialmente, es un factor propio de este tipo de licenciamiento que brinda gran libertad.

En 1983, Richard Stallman anunció el proyecto para desarrollar el sistema operativo GNU (GNU No es Unix), un sistema operativo similar a Unix, destinado a ser totalmente *software* libre. Con este proyecto se lanzó el movimiento de *software* libre y la creación de la Free Software Foundation (Stallman, 1983).

La diferencia entre el *software* gratuito y el *software* libre es que un *software* gratuito puede distribuirse sin costo al usuario, pero no necesariamente se comparte su código fuente y no se permite la modificación o redistribución de versiones modificadas del programa, acciones posibles de hacer con el *software* libre.

En cuanto a la naturaleza del *software* libre, la Free Software Foundation (2001) define cuatro libertades básicas:

- **Libertad 0:** ejecutar el programa como se desea, con cualquier propósito.
- **Libertad 1:** estudiar cómo funciona el programa, y cambiarlo para que haga lo que usted quiera.
- **Libertad 2:** redistribuir copias para ayudar a su prójimo.

- **Libertad 3:** distribuir copias de sus versiones modificadas a terceros. Esto le permite ofrecer a toda la comunidad la oportunidad de beneficiarse de las modificaciones.

Estas libertades nos dictan las leyes básicas que deben seguir los desarrollos de *software* libre y deben ser respetadas.

2.4 Internet de las cosas (IoT)

2.4.1 Definición

El internet de las cosas o IoT, por sus siglas en inglés, se refiere a una conexión por parte de dispositivos cotidianos a internet, con el objetivo de convertirlos en objetos inteligentes que permitan utilizar el dispositivo remotamente y analizar sus datos de uso.

Jammes (2016, p2) define IoT como "la red de objetos físicos que contienen tecnología incorporada para comunicarse y detectar o interactuar con sus estados internos o con el entorno externo".

El término fue usado por primera vez en 1999 por Ashton, de Procter & Gamble, mientras realizaba investigaciones con tecnologías RFID en la cadena de suministros y sensores de la empresa (Ashton, 2009).

IoT puede ser utilizado en múltiples industrias, ya que cualquier dispositivo puede llegar a adaptarse para utilizar sensores, almacenamiento y análisis de datos en la nube. Con la integración de esas tecnologías se busca mejorar la eficiencia o facilidad de uso de los dispositivos, hacer que las empresas puedan resolver sus problemas de forma más eficiente, competitiva y rentable.

De acuerdo con Gartner (2017), para finales del año 2017, se conectaron alrededor de 8400 millones de dispositivos, lo que significa un crecimiento del 31% con respecto al año 2016, que puede llegar hasta a 20,400 millones de dispositivos en el 2020, lo que significa que las empresas se están interesando en esta tecnología y que ven sus grandes beneficios en múltiples sectores de la industria.

2.4.2 Plataformas IoT en la nube

Las aplicaciones de IoT son desarrolladas y alojadas en plataformas en la nube. La conectividad se da mediante protocolos web, que soportan conexiones múltiples. Los usuarios no necesitan instalar *software* en sus equipos, ya que, al estar en la nube, consumen servicios tecnológicos como "Software as a Service" (SaaS).

Existen plataformas ya listas para trabajar con IoT, las cuales almacenan y transfieren datos entre la nube y los dispositivos. Entre las plataformas más utilizadas se encuentran las siguientes:

Amazon Web Services IoT: es una plataforma para registrar dispositivos IoT, que cuenta con un SDK para desarrollo de aplicaciones. Se puede conectar una gran variedad de dispositivos: desde automóviles hasta bombillos inteligentes. Amazon realizó alianzas con empresas como Intel para crear kits compatibles con la plataforma. Tiene una capa gratuita que soporta hasta 250.000 mensajes al mes.

Microsoft Azure IoT: plataforma IoT de Microsoft que incluye un motor de reglas, monitoreo, identificación de dispositivos y posee "Azure Stream Analytics" para analizar grandes volúmenes de datos. Es un servicio de pago, en su capa básica soporta 400.000 por $50.

Google Cloud: Google desarrolló una plataforma única para analizar grandes cantidades de datos utilizando "Google IoT Core". Actualmente, es una de las plataformas mejor valoradas con respecto al análisis y procesamiento de datos. Es un servicio de pago.

IBM Watson: Es una plataforma basada en la nube de IBM Cloud, la cual permite la creación de aplicaciones rápidas de IoT con comunicaciones seguras. Es un servicio de pago.

Todas estas plataformas integran y procesan los mensajes provenientes de los dispositivos IoT utilizando protocolos livianos. El protocolo más utilizado para la comunicación con dispositivos IoT se llama MQTT.

2.4.3 Protocolo MQTT

El estándar internacional ISO/IEC 20922 (ISO/IEC, 2016) describe el protocolo MQTT (*Message Queue Telemetry Transport*) como un protocolo de transporte de mensajes cliente-servidor de suscripción/publicación. Es un protocolo liviano, abierto, simple y diseñado para ser fácil de implementar.

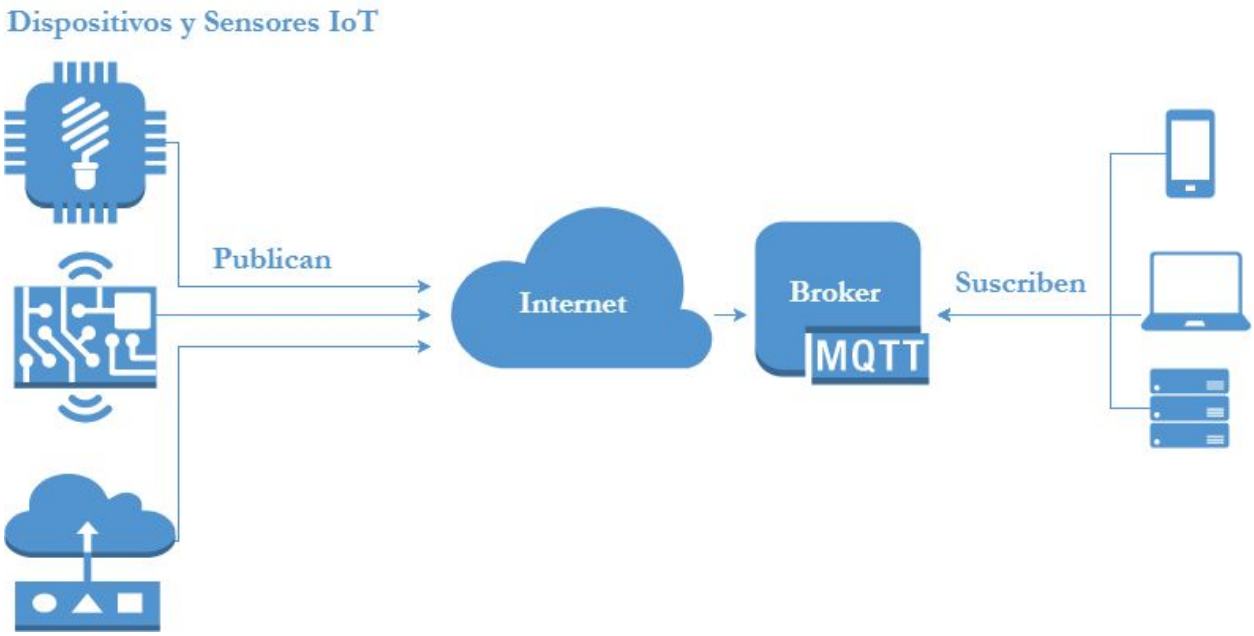


Figura 13. Implementación del protocolo MQTT mediante la nube.
Fuente: elaboración propia.

Como se muestra en la figura 13, para la comunicación entre dispositivos se envían mensajes mediante un servicio web intermediario llamado "broker", el cual gestiona el origen y destino de los datos por medio de un "tema", que es una cadena de texto jerárquica que contiene el mensaje. Los dispositivos que quieran recibir el mensaje se suscriben a estos temas y el *broker* crea una conexión TCP siempre conectada.

El protocolo se ejecuta a través de TCP/IP mediante el puerto estándar 1883 o el puerto 8883 utilizando SSL/TLS. Las características propias del protocolo hacen que sea ideal para el establecimiento de proyectos de IoT, por su bajo consumo de ancho de banda y su fácil implementación.

Existen implementaciones de *broker* de código abierto como Eclipse Mosquitto, desarrollado por un grupo de empresas que invierten y promueven comunidades para IoT llamado "The Eclipse IoT Working Group". Eclipse Mosquitto implementa las versiones del protocolo MQTT 3.1 y 3.1.1. MQTT y tiene módulos para desarrollo con Python (Eclipse IoT Working Group, s.f).

2.4.4 Seguridad en IoT

El IoT es una tecnología innovadora que está revolucionando los mercados de negocio por todo el mundo, pero, como en toda tecnología emergente, se presentan situaciones que perjudican o limitan su implementación.

Los dispositivos IoT enfocados en consumidores, como televisores, bombillos y cámaras, son fabricados en grandes cantidades; sin embargo, no se construyen pensando en la seguridad. Usualmente, su *software* no es actualizado e inclusive, en algunos casos, no se puede del todo actualizar. Esto lleva a que el *software*, eventualmente, presente alguna vulnerabilidad y pueda ser atacada por un ciberdelincuente.

Precisamente, eso sucedió en el 2016 cuando un virus llamado Mirai atacó el proveedor de servicios DNS Dyn. Este virus consistía en una *botnet* de dispositivos IoT como cámaras IP, impresoras y sensores mal configurados o con serias vulnerabilidades, cuyo objetivo fue realizar un ataque DDoS (*distributed denial of service attack)* a gran escala de aproximadamente 1.2 Tbps (The Guardian, 2016).

Según IBM (2017), el desarrollo de aplicaciones IoT requiere un enfoque de "Seguridad por diseño", en el cual los requisitos de seguridad del dispositivo deben darse en la parte de diseño, en lugar de hacer asunciones y aplicar la seguridad en un proceso posterior al desarrollo, para evitar problemas de seguridad y altos costos en procesos posteriores.

Es muy importante analizar previamente los requerimientos de seguridad y diseñar soluciones que los incorporen, siguiendo recomendaciones y buenas prácticas e implementando aplicaciones con código seguro.

La interoperabilidad es fundamental para implementar un dispositivo IOT y permitir que se comunique con otros dispositivos. Esta comunicación se debería dar por protocolos estándar que permitan el cifrado de la información, como lo hace el protocolo MQTT.

Al utilizar estos protocolos abiertos, se reducen los costes de implementación, ya que permiten que los fabricantes se enfoquen en un lenguaje en común para su comunicación y les ayuda a que no incurran en soluciones propietarias que dificulten la comunicación con servidores u otros dispositivos de IoT. Si los datos se encuentran en un formato abierto, posibilita que otras aplicaciones o dispositivos que no son IoT utilicen esa información.

Por otra parte, el IoT presenta nuevos desafíos de privacidad, ya que se tiene que definir cuáles datos son privados, o si el propietario de esos datos es el fabricante o el propietario del dispositivo. Por esta razón, los usuarios tienen que tener un nivel de confianza con los proveedores para utilizar los dispositivos IoT.

Un ejemplo de esto es el caso ocurrido en marzo del 2017, cuando se presentó una demanda en la Corte del Distrito de los Estados Unidos para el Distrito Norte de Illinois contra la empresa Standard Innovation Corp por $3.75 millones de dólares, ya que la empresa desarrolló un juguete sexual IoT llamado *We-Vibe*, que recolectaba información privada del usuario sin su consentimiento, como la fecha y hora de uso, intensidad de vibración y dirección de correo electrónico del usuario (Law 360, 2017). Este caso evidencia la importancia de establecer las medidas necesarias para garantizar la privacidad de los usuarios y que ese tipo de datos se encuentren claramente establecidos en el contrato.

Por todos estos incidentes de seguridad relacionados con IoT, surgen iniciativas como el proyecto libre “OWASP Internet of Things”, el cual está diseñado para ayudar a los fabricantes, desarrolladores y consumidores a comprender mejor los problemas de seguridad relacionados con el *Internet of Things* y permitir a los usuarios en cualquier contexto tomar mejores decisiones de seguridad al construir, implementar o evaluar las

tecnologías IoT.

El proyecto busca definir una estructura para la seguridad en IoT, como áreas de superficie de ataque, guías de pruebas y vulnerabilidades conocidas. OWASP (The Open Web Application Security Project OWASP, 2018) define las principales vulnerabilidades en IoT como:

1. **Enumeración de usuarios:** capacidad de recopilar un conjunto de nombres de usuario válidos mediante la interacción con el mecanismo de autenticación.
2. **Contraseñas débiles:** posibilidad de establecer contraseñas de cuenta como "1234" o "123456".
3. **Bloqueo de cuenta:** posibilidad de seguir enviando intentos de autenticación después de 3-5 intentos fallidos de inicio de sesión.
4. **Servicios sin cifrar:** los servicios de red no están encriptados correctamente para evitar escuchas o manipulación por parte de atacantes.
5. **Autenticación de dos factores:** la falta de mecanismos de autenticación de dos factores, como un token de seguridad o un escáner de huellas dactilares.
6. **Cifrado mal implementado:** el cifrado se implementa, no obstante, está configurado incorrectamente o no se está actualizando correctamente.
7. **Actualización enviada sin encriptación:** las actualizaciones se transmiten a través de la red sin usar TLS o encriptando el archivo de actualización.
8. **Ubicación de actualización es modificable:** la ubicación de almacenamiento para los archivos de actualización es mundialmente modificable, lo que permite modificar y distribuir el *firmware* entre todos los usuarios.
9. **Denegación de servicio:** el servicio puede ser atacado de manera que niegue el servicio a otro en específico, o a todo el dispositivo.

10. **Mecanismo de actualización faltante:** no se puede actualizar el dispositivo.

11. **Componentes de terceros inseguros:** versiones desactualizadas de *software* de terceros como busybox, openssl, ssh, servidores web, etc.

Capítulo 3

Marco metodológico

La investigación aplicada es el método más adecuado para resolver el problema planteado. Consiste en la búsqueda de información en diversos temas para ser empleada en un prototipo, por medio de recursos tecnológicos. La principal ventaja de esta técnica es la posibilidad de utilizar el modelo experimental para obtener resultados reales que sustenten la teoría y responda a la problemática.

En este caso en específico, se toman en consideración estudios realizados sobre control de acceso, internet de las cosas, *software* libre, lenguajes de programación y *hardware* de uso libre, para proponer el diseño, construcción y configuración de un prototipo gestor de control de acceso que llegue a resolver los problemas de ciberseguridad y costo de los productos actuales.

El alcance investigativo de este proyecto es de carácter exploratorio. La investigación exploratoria se efectúa sobre un tema poco estudiado o que no ha sido abordado antes, en donde los alcances se limitan a la observación y al registro objetivo de lo encontrado (Barrantes, 1999).

Para iniciar la creación del dispositivo, se debe decidir la plataforma de *hardware* libre a utilizar, así como identificar los componentes electrónicos que se ajusten a las necesidades del proyecto. Este proceso inicia con la adquisición del *hardware* necesario, lectura de documentación, pasar por etapas de prueba y error, para la construcción de un *hardware* que permita el control de accesos.

Posteriormente, se debe diseñar el modelo arquitectura de *software* que integre todos los requisitos de un sistema de control de accesos y definir la plataforma en la nube IoT que va a utilizar en el dispositivo. Además, se debe desarrollar el *software* para la configuración, administración y uso del dispositivo. Para realizar esto, es necesario elegir los lenguajes de programación a utilizar y el sistema operativo por el cual va a correr el sistema.

Una vez creado el dispositivo e integrado el *software* en la plataforma en la nube, se procede a realizar pruebas de funcionamiento y robustecer la seguridad del sistema para integrar el sistema de control de accesos a un modelo a escala con el propósito de demostrar su viabilidad de implementación.

Capítulo 4

Propuesta de solución

A continuación, se presenta la propuesta de un sistema de control de accesos seguro, en el cual se tomaron como base los resultados obtenidos en la presente investigación sobre IoT, *hardware* y *software* libre.

El sistema propuesto se compone de un dispositivo controlador ensamblado con componentes de *hardware* libre, que se comunica por medio de un protocolo de IoT a una plataforma en la nube, con el fin de controlar el ingreso y salida de personas residentes o visitantes en empresas, condominios, locales y/o casas de habitación, por medio de una aplicación conectada a internet. El principal beneficio es brindarles a los usuarios un sistema de fácil uso, que sea oportuno, económico y seguro para control del acceso a sus inmuebles.

Seguidamente, se presenta la propuesta para el sistema de control de accesos seguro. Este sistema de control se divide en capítulos, los cuales contienen el detalle del proceso de creación llevado a cabo.

4.1 Arquitectura de la solución

La elección de la arquitectura tecnológica fue una de las decisiones más importantes para el desarrollo del proyecto, ya que es el momento en el que se definen los equipos, infraestructura, dispositivos y *software* que va a utilizar todo el sistema de control de accesos.

Como se puede apreciar en la figura 13, la arquitectura del sistema propuesto se compone de:

1. Aplicación web en la nube para administración del sistema.
2. Servidor MQTT Broker en Amazon EC2.
3. Controlador de *hardware* para el control de acceso.
4. Dispositivos de control de acceso físico.

5. Aplicación para usuarios finales.

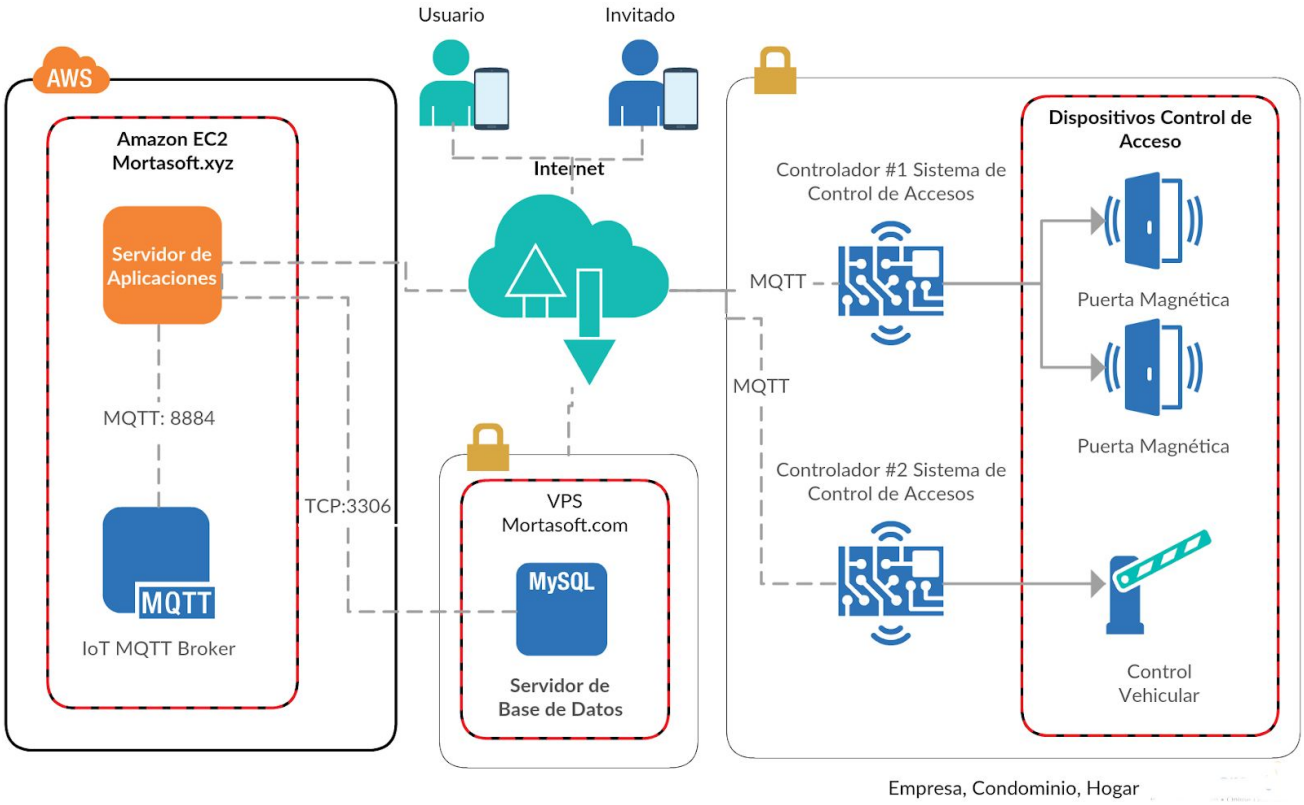


Figura 13. Arquitectura de la solución del sistema control de accesos propuesto.
Fuente: elaboración propia.

Para detallar la arquitectura del sistema, a continuación, se explica cada uno de los componentes que lo integran.

4.1.1 Aplicación web en la nube para administración del sistema

Primeramente, el sistema está compuesto por una instancia de servidor virtual en la plataforma de servicios en la nube de Amazon Elastic Cloud Compute (Amazon EC2) de tipo t2.micro, el cual posee un procesador virtual Intel Xeon y 1GB de memoria RAM y utiliza como sistema operativo Ubuntu Server. Una de las razones principales para la elección de esta plataforma es la seguridad, capacidad de cómputo, el soporte, la redundancia, la resistencia a desastres y la buena capacidad de recuperación ante fallos.

Además, el bajo costo, ya que por su escalabilidad y flexibilidad se pagan solo los recursos que se utilizan.

En dicha instancia, se aloja una aplicación desarrollada a la medida, que administra y configura todo el sistema, la cual va a ser utilizada por los usuarios como un servicio SaaS, de manera que puedan acceder a ella desde diferentes dispositivos y pueda ser utilizada por varios clientes a la vez. Para almacenar los datos, la aplicación utiliza un motor de base de datos MySQL Community Edition, que se encuentra en un VPS privado.

4.1.2 Servidor MQTT Broker en Amazon EC2

A la instancia creada en Amazon EC2 se le instaló Eclipse Mosquitto IoT MQTT Broker, para servir como un servicio intermediario entre el controlador y la aplicación de administración.

4.1.3 Controlador de *hardware* para el control de acceso

El controlador es un dispositivo de *hardware* basado en el microcomputador Raspberry Pi 3, el cual se encuentra siempre conectado a internet y realiza una conexión entre los dispositivos de control de acceso y la plataforma creada en Amazon EC2 de la aplicación y el *broker*.

4.1.4 Dispositivos de control de acceso físico

Los dispositivos de control de acceso físico son los que permiten el acceso a recursos físicos como puertas magnéticas, motores eléctricos o agujas vehiculares. Cabe indicar que el sistema inclusive es capaz de controlar dispositivos de iluminación que, aunque no forman parte de un control de accesos, se pueden utilizar como un método disuasivo de seguridad.

4.1.5 Aplicación para usuarios finales

El control de los dispositivos físicos por parte de los usuarios finales se realiza por medio de una aplicación que se puede acceder desde dispositivos móviles, la cual utiliza

usuarios y roles para determinar los accesos a los recursos, además de brindar acceso a los visitantes.

Una vez definida la arquitectura a utilizar, se va a explicar en detalle cada uno de los componentes que integran el sistema de control de accesos.

4.2 Servidor MQTT Broker

Como servidor de IoT MQTT Broker se instaló Eclipse Mosquitto. Este servidor es el encargado de la comunicación entre el dispositivo controlador y la aplicación administrativa, que se encuentra en la nube. Los datos se transfieren de forma encriptada por medio de un certificado digital SSL.

Este servicio fue instalado en un servidor Ubuntu Server en Amazon EC2 y para su instalación y configuración se siguieron los siguientes pasos:

1. Se instaló del paquete:

```
sudo apt-get install mosquitto mosquitto-clients
```

2. Para usar el encriptado SSL se requiere un certificado digital. En este caso, se eligió el servicio gratuito de Let's Encrypt:

```
sudo add-apt-repository ppa:certbot/certbot
sudo apt-get update
sudo apt-get install certbot
sudo ufw allow http
      sudo certbot certonly --standalone
      --standalone-supported-challenges http-01 -d midominio.xyz
```

3. Se activó la renovación automática de certificados a las 4:00 a.m. todos los días:

```
sudo crontab -e
      0 4 * * * certbot renew --noninteractive --post-hook "systemctl
      restart mosquitto"
```

4. Para reforzar la seguridad, fue necesario configurar un usuario y *password* para conectarse al servicio de Mosquitto:

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd usuario
sudo nano /etc/mosquitto/conf.d/default.conf
      #Añadir:
      allow_anonymous false
      password_file /etc/mosquitto/passwd
```

5. Se reinició el servicio:

```
sudo systemctl restart mosquitto
```

6. Para verificar correcto funcionamiento del servicio, se suscribió al tópico “Prueba” y luego se publicó un mensaje:

```
mosquitto_sub -h localhost -t Prueba -u "usuario" -P "password"
mosquitto_pub -h localhost -t "Prueba" -m "Mensaje" -u "usuario" -P
"password"
```

7. Se habilitó MQTT SSL:

```
sudo nano /etc/mosquitto/conf.d/default.conf
# Editar el contenido del archivo y añadir:
listener 1883 localhost
listener 8888
certfile /etc/letsencrypt/live/midominio.xyz/cert.pem
cafile /etc/letsencrypt/live/midominio.xyz/chain.pem
keyfile /etc/letsencrypt/live/midominio.xyz/privkey.pem
# Luego ejecutar el comando
sudo systemctl restart mosquitto
```

8. Se hizo la prueba de suscribirse a un tópico mediante SSL:

```
mosquitto_pub -h midominio.xyz -t test -m "Mensaje" -p 8888 --capath
/etc/ssl/certs/ -u "usuario" -P "password"
```

9. Para que la aplicación web pudiera comunicarse con el servidor, fue necesario habilitar WebSockets:

```
sudo nano /etc/mosquitto/conf.d/default.conf
# Editar el contenido del archivo y añadir:
listener 8083
certfile /etc/letsencrypt/live/midominio.xyz/cert.pem
cafile /etc/letsencrypt/live/midominio.xyz/chain.pem
keyfile /etc/letsencrypt/live/midominio.xyz/privkey.pem

# Luego ejecutar el comando
sudo systemctl restart mosquitto
```

Una vez realizados estos pasos, el servidor estaba listo para recibir tópicos. Ahora, el servidor de aplicaciones crea un token único de 48 bits, y lo asigna a un dispositivo controlador. Por ejemplo:

a606400b7935eea43de41d60b96ac542d297dd7a1df76781.

Sobre este tópico, se van a configurar cada uno de los dispositivos de control de acceso, asignados a un subtópico que referencia al tipo de dispositivo y el puerto en el cual está conectado. Por ejemplo:

a606400b7935eea43de41d60b96ac542d297dd7a1df76781/acceso-1
a606400b7935eea43de41d60b96ac542d297dd7a1df76781/puerta-2

4.3 Dispositivo controlador del sistema de control de accesos

La función principal del controlador de *hardware* es controlar a los dispositivos que se conecten a él y tomar acciones basado en eventos que se accionen desde una aplicación central que almacena la información de configuración y mantenimiento.

4.3.1 Elección del *hardware* libre para el controlador

En el mercado del *hardware* de uso libre, existen diferentes alternativas que se pueden utilizar para comunicar los distintos mecanismos de control de acceso. Por esta razón, es necesario realizar un análisis de las diferentes opciones, basado en criterios de mercado y técnicos, que faciliten escoger el dispositivo más adecuado para las necesidades del proyecto. Cada criterio está compuesto de una característica que comparten todos los aparatos y que pueden ser medibles y comparables, de manera que se seleccione un dispositivo que se utilice como un controlador del sistema de control de accesos.

Como posibles opciones para basar dicho controlador, se consideraron 5 placas de dispositivos de *hardware* libre: Arduino, Intel Galileo, Raspberry Pi Model B+, Raspberry Pi Zero W, Raspberry Pi 3 Model B. Los dispositivos anteriores se escogieron debido a que son muy utilizados en ambientes educativos para proyectos de electrónica, además de que existe mucha documentación sobre su manejo y comunidades de usuarios alrededor del mundo, así como el constante desarrollo y mejora en el que se encuentran.

Se requería que el dispositivo a escoger fuera capaz de manejar entradas y salidas para el control de señales digitales, que tuviera un puerto de conexión de red para su

administración, que pudiera ejecutar una página web y que tuviera un costo accesible para su adquisición.

En la tabla 13 se puede apreciar una comparación de las características de cada uno de los dispositivos, como frecuencia de procesador, memoria RAM, puertos GPIO, puertos USB, HDMI y puerto de red, así como su costo. Este costo es un estimado basado en el precio en Estados Unidos más el costo de envío a Costa Rica.

Tabla 13

Comparación de dispositivos de hardware

Componente	Arduino	Intel Galileo	Raspberry Pi Model B+	Raspberry Pi Zero W	Raspberry Pi 3 Model B
Frecuencia de reloj del procesador	16Mhz	400Mhz	700Mhz	1000Mhz	1200Mhz
Memoria RAM	*	256MB	512MB	512MB	1GB
Puertos GPIO	14	14	40	16	40
Puertos USB	1	3	4	4	1
HDMI	No	Sí	Sí	No	Sí
Puerto de red	No tiene	Ethernet	Ethernet	Wireless	Ethernet + Wireless
Costo*	**$38**	**$113.26**	**$103.75**	**$63.50**	**$101.70**

Fuente: elaboración propia.

Al contrastar las características de cada dispositivo contra los requerimientos del controlador, se determinó que la mejor opción para basar el sistema de control de accesos es el Raspberry Pi Model B.

Este dispositivo posee mayores características técnicas, en las que destaca la frecuencia de reloj del procesador, la cantidad de memoria RAM y la cantidad de puertos

GPIO, los cuales son requeridos para comunicarse con otros dispositivos electrónicos del sistema. Adicionalmente, cuenta con conexión Ethernet y un módulo de wifi incorporado, lo cual tiene una ventaja ante los demás dispositivos, porque puede utilizarse de forma inalámbrica o alámbrica.

Si bien el costo no es el más bajo, tiene una mejor relación de características versus precio y mantiene una amplia comunidad de usuarios alrededor del mundo y manuales en línea de su configuración y funcionamiento.

4.3.2 Diseño y ensamblaje del *hardware* libre

Los dispositivos de control de acceso físico para el ingreso y salida que puede administrar el controlador utilizan voltajes de entre 24V a 250V, los cuales no pueden ser manejados solo por el Raspberry Pi. Por eso, es necesario utilizar *relays* en el diseño del controlador.

La figura 14 que se muestra a continuación, presenta un módulo *relay* de 8 canales que se utilizará como parte del controlador. Este módulo opera a 5V y los *relays* pueden manejar voltajes de hasta 250V-10A y ser configurados para funcionar en modo "Normalmente Abierto" (NO) o "Normalmente Cerrado" (NC), de acuerdo con la necesidad de los dispositivos a conectar. Estos *relays* pueden ser controlados individualmente por medio de los puertos GPIO del Raspberry Pi y poseen un circuito integrado de protección contra corrientes, para evitar que se puedan dañar componentes internos del microcomputador.

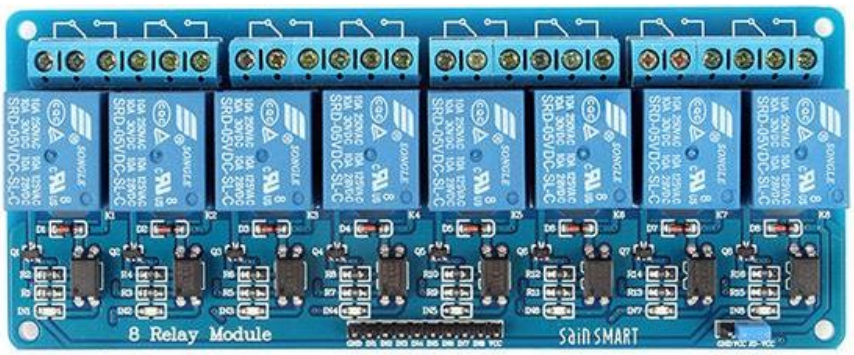


Figura 14. Módulo de 8 Relays.
(Amazon, 2017d).

Para conectar dicho módulo de *relays* con el Raspberry Pi se utilizaron cables tipo "jumper hembra-hembra", como se muestra en la tabla 14 y la figura 15.

Tabla 14
Conexión de pines GPIO del Raspberry Pi con Módulo de Relay

Puerto GPIO	Pin 2	Pin 11	Pin 13	Pin 15	Pin 29	Pin 31	Pin 33	Pin 37	Pin 36	Pin 6
# Relay	5V	1	2	3	4	5	6	7	8	GND

Fuente: elaboración propia.

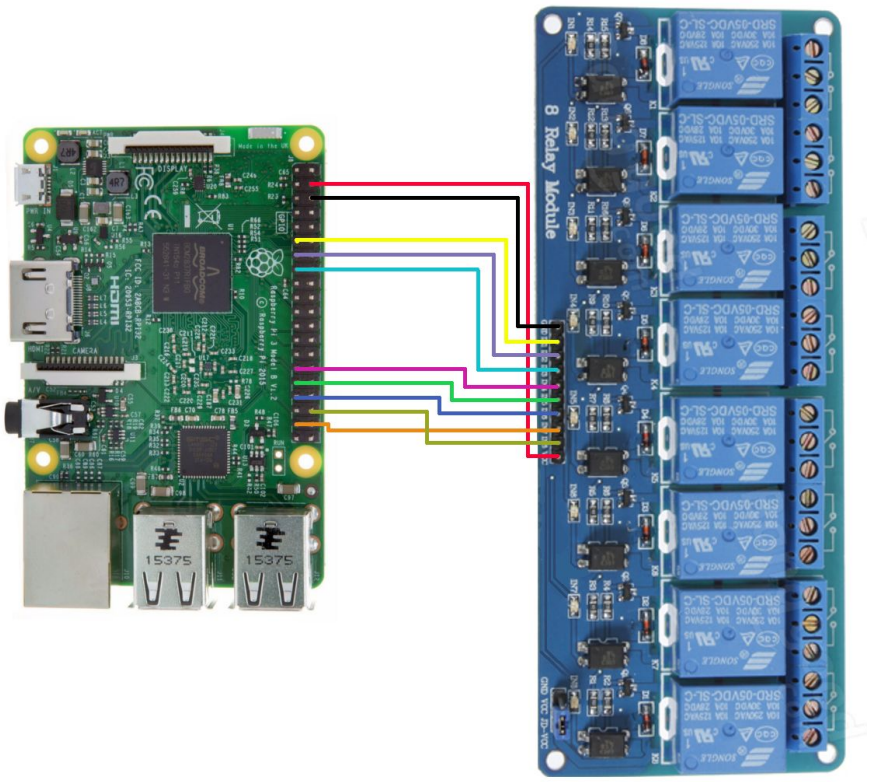


Figura 15. Conexión de pines GPIO del Raspberry Pi con Módulo de Relay.
Fuente: elaboración propia.

En cada uno de los *relays* se puede conectar un dispositivo de control de acceso físico, como puertas magnéticas, agujas vehiculares, motores eléctricos, dispositivos de iluminación como bombillos o lámparas.

Para el almacenamiento de archivos, configuraciones y el sistema operativo se instaló una memoria tipo microSD de 64GB clase 10, la cual tiene una tasa de transferencia de archivos de 98 MB/s; sin embargo, el controlador puede utilizar memorias de menor tamaño.

Con el propósito de almacenar y proteger los componentes electrónicos dentro del controlador, se realizó un diseño en 3D de un almacenamiento en plástico según las necesidades, el cual fue impreso en 3D mediante una impresora *Printrbot Simple Metal* utilizando una fibra tipo PLA.

En la figura 16 y 17 se muestra el diseño, el cual consiste en una base de 168.20mm x 149.99mm x 30.02mm, una tapa con aberturas para ventilación y otra tapa con un logo. Cuenta con cuatro pestañas en cada esquina para permitir el uso de tornillos para instalarlo sobre una superficie vertical.

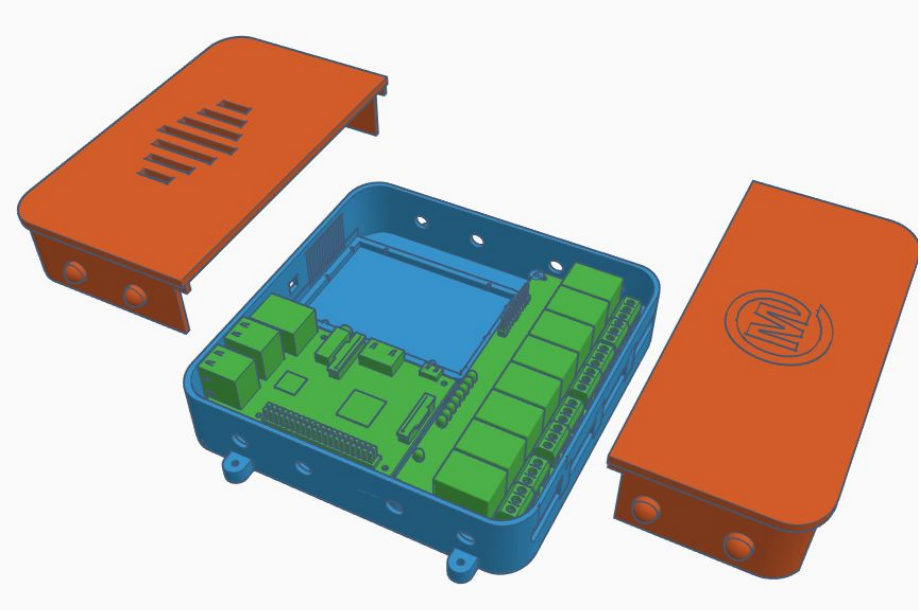


Figura 16. Diseño de la caja del controlador tapas abiertas.
Fuente: elaboración propia.



Figura 17. Diseño de la caja del controlador tapas cerradas.
Fuente: elaboración propia.

4.3.3 Dispositivos de control de acceso compatibles

Por medio del controlador de *hardware* del sistema, se pueden controlar distintos dispositivos de control de accesos físicos como:

- **Portones eléctricos:** se puede adaptar a cualquier tipo de motor eléctrico residencial o comercial.
- **Puertas magnéticas:** puertas con cerradura magnética.
- **Agujas vehiculares:** todo tipo de agujas vehiculares.
- **Dispositivos de iluminación:** si bien la iluminación no es un dispositivo de control de acceso, se puede utilizar como un método de seguridad disuasiva, que permita desviar la atención o intimidar a un posible delincuente, para que desista de intentar alguna clase de medida que pongan en riesgo la seguridad física de la empresa.

Por la forma en la que se diseñó el sistema de control de accesos, es posible configurar dispositivos diferentes a los mencionados, realizando configuraciones a nivel de *software*.

4.3.4 Configuración del sistema operativo y *software*

El sistema operativo que utiliza el controlador es la distribución Linux para procesadores ARM Raspbian, ya que es un sistema robusto soportado ampliamente por Raspberry Pi. Sobre este sistema operativo, se instala el *software* necesario para su correcto funcionamiento.

Con el propósito de reforzar la seguridad del sistema operativo, disminuir las vulnerabilidades y reducir los posibles ataques, se deben realizar ajustes a nivel de la configuración por defecto. Dichos ajustes están relacionados con la eliminación de *software* no utilizado, servicios o usuarios innecesarios en el sistema, así como el cierre de puertos que no estén en funcionamiento.

Como parte de este reforzamiento al sistema operativo, a continuación, se muestra la lista de ajustes realizados, además de sus respectivos comandos y configuraciones.

4.3.4.1 Reforzamiento de seguridad en Raspbian

1. **Cambiar el usuario por defecto.** Como medida de seguridad en cualquier sistema operativo, se debería cambiar el usuario por defecto o administrador.

    a. Añadir un nuevo usuario:

```
sudo useradd -m usuario -s /bin/bash -m -G adm,sudo
```

    b. Cambiar el *password*:

```
sudo passwd usuario
```

    c. Eliminar el usuario y la carpeta “Home” que trae por defecto:

```
sudo deluser -remove-home pi
```

2. **Desactivar SSH.** Para efectos de la controladora no se requiere utilizar el protocolo SSH, por lo tanto, se debe desactivar.
    a. Desactivar en la aplicación de configuración de Raspberry Pi con `raspi-config.`

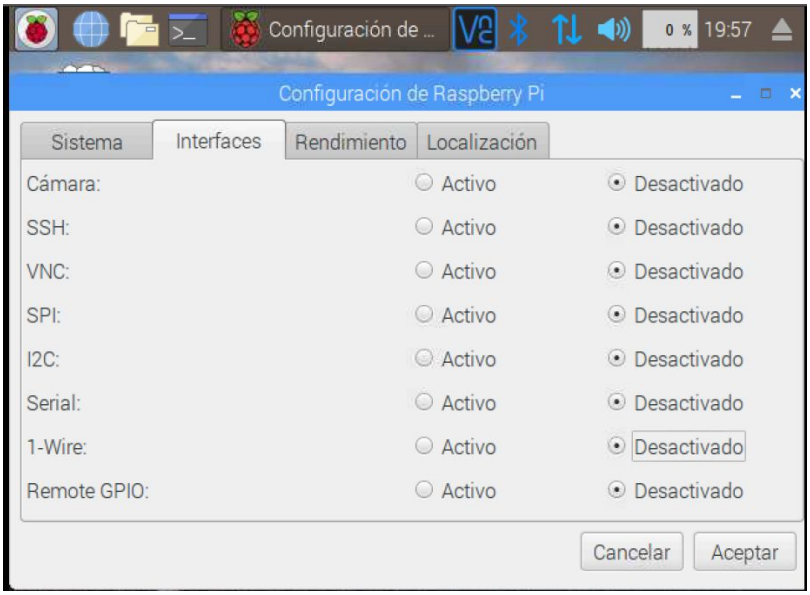


Figura 18. Desactivar el SSH en Raspbian.
Fuente: elaboración propia.

3. **Instalar un *firewall*.** Es esencial para filtrar las conexiones que entran y salen del controlador, por esta razón solo se deben permitir los puertos necesarios.

   a. Instalar Uncomplicated Firewall:

```
sudo apt-get install ufw
```

   b. Activar el *Firewall:*

```
sudo ufw enable
```

   c. Permitir los puertos utilizados por MQTT solo a la IP del servidor Mosquitto Broker:

```
sudo ufw allow from 35.167.19.43 to any port 1883
sudo ufw allow from 35.167.19.43 to any port 1884
sudo ufw allow from 35.167.19.43 to any port 8083
sudo ufw allow from 35.167.19.43 to any port 8888
```

   d. Permitir a un equipo en la red el acceso al puerto 80. Este equipo va a ser el administrador del sistema:

```
sudo ufw allow from 192.168.1.10 to any port 80
```

   e. Habilitar el registro completo de eventos:

```
sudo ufw logging on
```

4. **Actualización automática de paquetes.** Es de vital importancia mantener todos los paquetes actualizados, ya que los parches eliminan vulnerabilidades y corrigen errores. Por defecto, no se realiza una actualización automática, por lo que se creó un *script* que soluciona este inconveniente:

   a. *Script* de actualización del sistema operativo update.sh:

```
#!/bin/bash
sudo apt-get update && sudo apt-get upgrade
```

   b. Hacer el *script* ejecutable:

```
chmod +x /home/usuario/update.sh
```

   c. Agregarlo a *cron* para que se ejecute todos los días a las 12:0 a.m.:

```
crontab -e
0 0 * * * /home/usuario/update.sh
```

5. **Eliminar paquetes innecesarios**. Se eliminan paquetes que no se van a utilizar.

   a. Samba

```
sudo apt-get remove --purge smbclient samba-common
```

   b. Soporte de Audio

```
sudo apt-get remove --purge alsa-base alsa-utils
```

   c. Git

```
sudo apt-get remove --purge git git-core git-man
```

   d. Otros

```
sudo apt-get remove --purge smartsim minecraft-pi freepats ed
sudo apt-get remove --purge penguinspuzzle udisks vim-common vim-tiny
sudo apt-get autoremove –-purge
sudo rm -R /home/pi/python_games
```

Una vez realizada la configuración y mejora en la seguridad del controlador, se procede a instalar y configurar el *software* necesario para el control de los dispositivos de acceso llamado Node Red.

4.3.4.2 Node-RED

Para poder conectar el controlador con la plataforma en la nube de Amazon y el servidor MQTT Broker, se utilizó la herramienta de desarrollo visual Node-RED, en la cual los datos se estructuran por medio de flujos, para controlar los pines GPIO, enviar alertas o generar paneles con información.

Esta herramienta ya viene instalada por defecto en Raspbian, solamente hay que habilitar el servicio para que inicie automáticamente al arrancar el sistema operativo con el comando:

```
sudo systemctl enable nodered.service
```

Una vez iniciado el servicio, se debe iniciar el programa:

```
sudo node-red-start
```

Para la creación de un panel de control, es necesario instalar el paquete *node-red-dashboard* y *wiring-pi* con el comando:

```
npm install node-red-dashboard wiring-pi
```

Para el manejo de los mensajes por medio del protocolo MQTT, se configuraron los nodos para conectarse al puerto 8888 mediante certificados digitales. Estos nodos se dividen
en dos grupos:

- **Entrada:** contiene los nodos que manejan el comportamiento del controlador hacia los mensajes que recibe en un tópico específico para encender o apagar un *relay*.
- **Salida:** contiene los nodos que envían datos sobre estadísticas y *logs* del sistema.

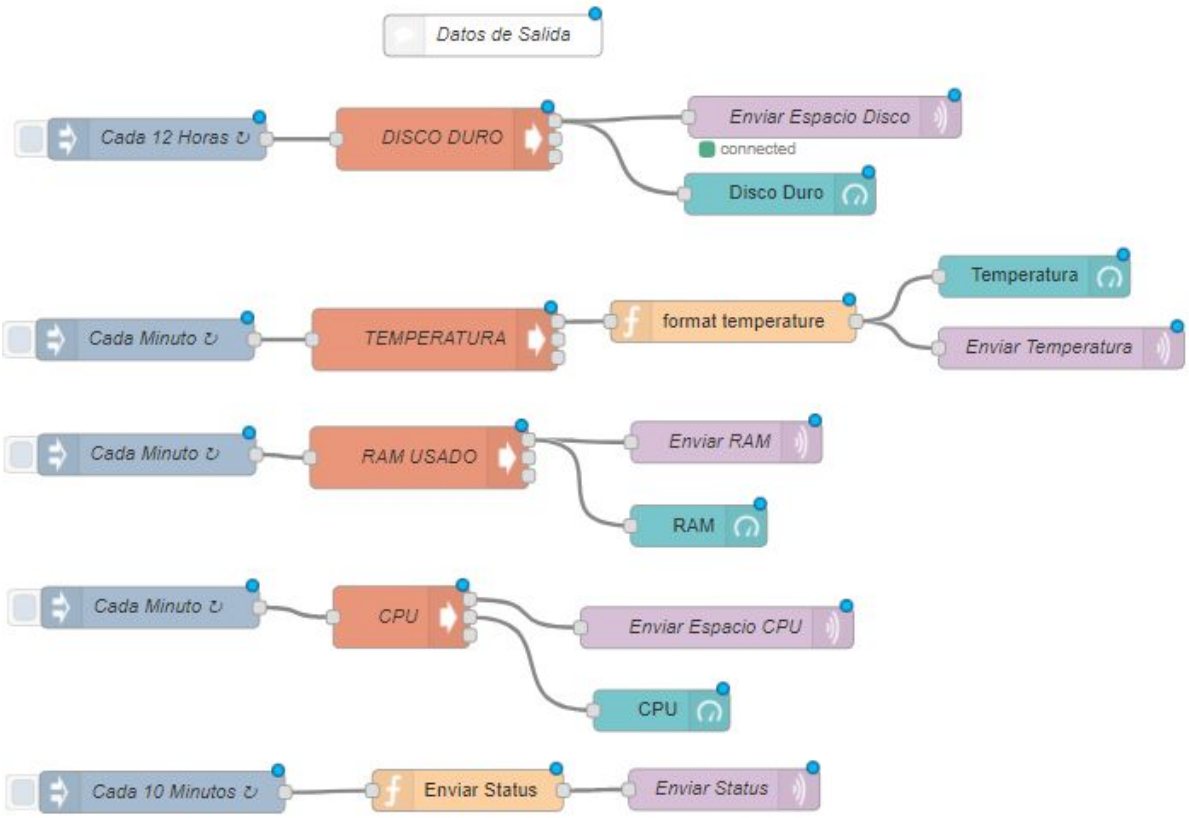


Figura 19. Nodos de salida Node-RED.
Fuente: elaboración propia.

Como se puede observar en la figura 19, los nodos de salida se componen de métricas con datos específicos del controlador como:

- **Disco duro**: envía al *broker* la cantidad de espacio en GB utilizado en la memoria microSD cada 12 horas y lo publica con el tópico #controlador/disco. El nodo utiliza una función que llama al comando:

  ```
  df -h | awk 'FNR == 2 {print $3}'
  ```

- **Temperatura:** envía al *broker* la temperatura del CPU en grados centígrados cada minuto y lo publica con el tópico #controlador/temperatura. El nodo utiliza una función que llama al comando:

```
vcgencmd measure_temp
msg.payload = Number(msg.payload.slice(5,-3));
return msg;
```

- **RAM utilizada:** envía al *broker* la cantidad de memoria RAM utilizada cada minuto y la publica con el tópico #controlador/ram. El nodo utiliza una función que llama al comando:

```
free | awk 'FNR == 3 {print $3/1024}'
```

- **CPU:** envía al *broker* el porcentaje de uso del procesador utilizado actualmente cada minuto y lo publica con el tópico #controlador/cpu. El nodo utiliza una función que llama al comando:

```
echo $[100-$(vmstat 1 2|tail -1|awk '{print $15}')]
```

- ***Status*:** envía al *broker* la fecha actual del sistema cada 10 minutos, para saber la última fecha de conexión del controlador. Se publica con el tópico #controlador/status. El nodo utiliza la función:

```
var date = new Date();
msg.payload = date.toString()
return msg;
```

- ***Log*:** envía al *broker* el tamaño del archivo de *log* cada 12 horas y lo publica con el tópico #controlador/log. El nodo utiliza una función que llama al comando:

```
du -h log_mqtt.txt | awk '{print $1}'
```

Cada uno de los flujos anteriores envían la información al *broker* de forma periódica, de acuerdo con el tiempo establecido dentro de cada uno de los flujos. Este tiempo establecido se podría modificar, si se requiere.

El grupo de flujos de entrada son los encargados de manejar los mensajes recibidos del *broker* y de realizar acciones de acuerdo con su contenido. En la figura 20, se describe todo el flujo que atraviesa un mensaje.

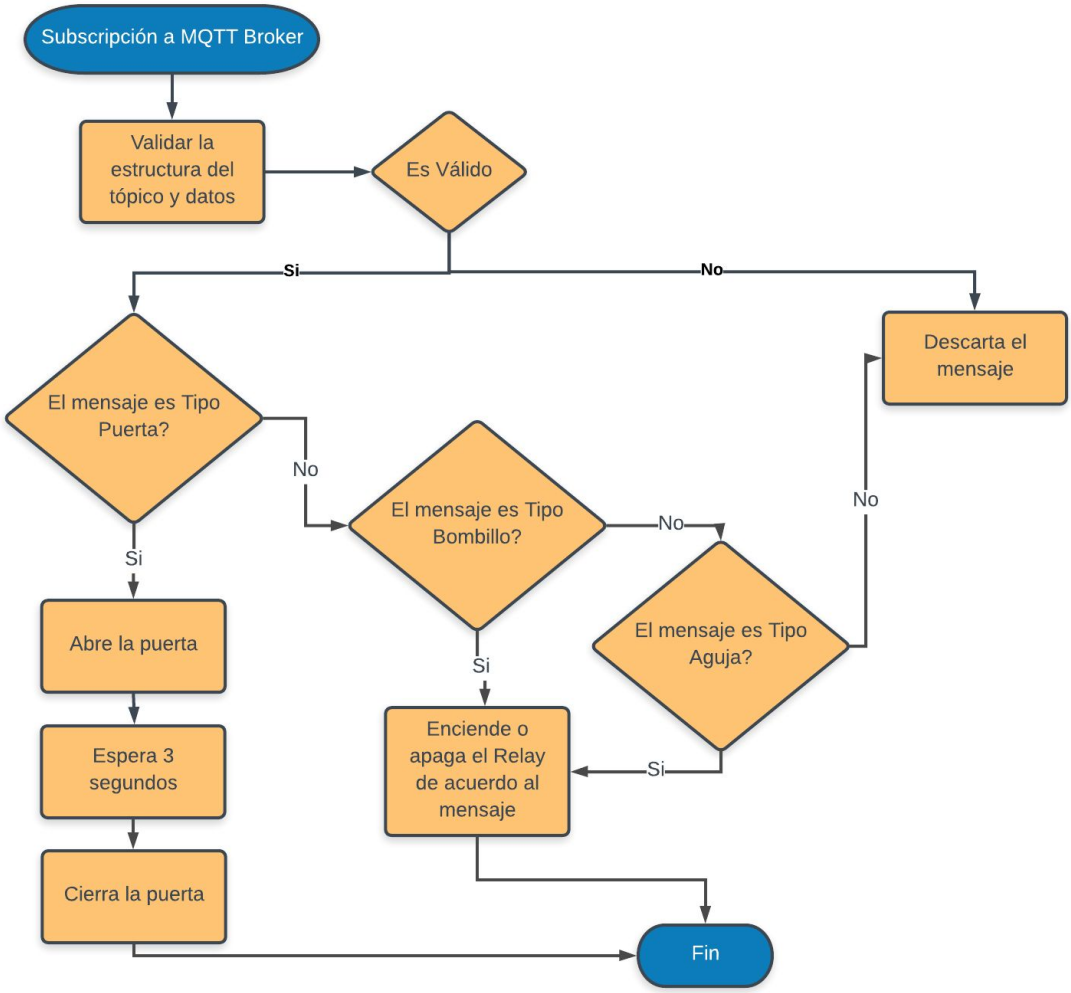


Figura 20. Nodos de entrada para mensajes en Node-RED.
Fuente: elaboración propia.

Cada uno de los mensajes que recibe el controlador por parte del *broker* se compone de la siguiente información:

- **Tópico:** jerarquía de mensajes descriptivos estructurados separados por un "/", que corresponde a un identificador del controlador, un tipo de dispositivo y un número de dispositivo.
- ***Payload*:** corresponde a la acción que se desea ejecutar. Puede ser "0" o "1", lo cual corresponde a un alto o bajo en un puerto digital del controlador.

- ***Qos*:** se refiere a la calidad de servicio, se envía un 1, que corresponde a que el mensaje se envíe por lo menos una vez, y si no se recibe respuesta, lo envía de nuevo hasta tener una confirmación.

- ***Msgid*:** consecutivo que identifica a cada mensaje,

- ***Relay*:** número de conector en el módulo de *relay*.

Como se puede observar en la figura 21, el tópico es una cadena de texto con un token único de 48 bits que identifica al controlador. Dicho tópico separa al token por un "/", seguido del tipo de dispositivo que se desea controlar, así como un número que identifica a un *relay* conectado en el puerto GPIO.

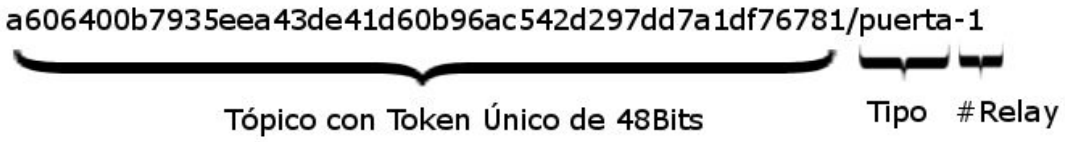


Figura 21. Estructura de tópicos en mensajes del controlador.
Fuente: elaboración propia.

Cada vez que ingresa un mensaje, el flujo se encarga de interpretar la información que contiene el mismo y validar que los datos sean correctos. De acuerdo con el tipo de dispositivo que se quiera controlar, el comportamiento cambia. El controlador es muy flexible con respecto a los dispositivos que se pueden utilizar, se adapta a las necesidades de los usuarios del sistema. Sin embargo, se establecieron únicamente 3 tipos de dispositivos de forma predeterminada:

- **Puerta:** cuando se recibe un mensaje de tipo "puerta", el flujo valida el token, obtiene el número de *relay* que se quiere controlar, envía una señal de 0 al puerto GPIO correspondiente al *relay* deseado y abre la puerta. Luego de 3 segundos el *relay* vuelve al estado 1 y se cierra la puerta.

- **Aguja:** cuando se recibe un mensaje de tipo “aguja”, el flujo valida el token, obtiene el número de *relay* que se quiere controlar y envía una señal de 1 al puerto GPIO para abrir y una señal de 0 para cerrar.

- **Bombillo:** cuando se recibe un mensaje de tipo “bombillo”, el flujo valida el token, obtiene el número de *relay* que se quiere controlar y envía una señal de 0 al puerto GPIO para abrir y una señal de 1 para cerrar.

En síntesis, tomando en cuenta los flujos de entrada y salida, el controlador utiliza los siguientes tópicos, mostrados en la figura 22.

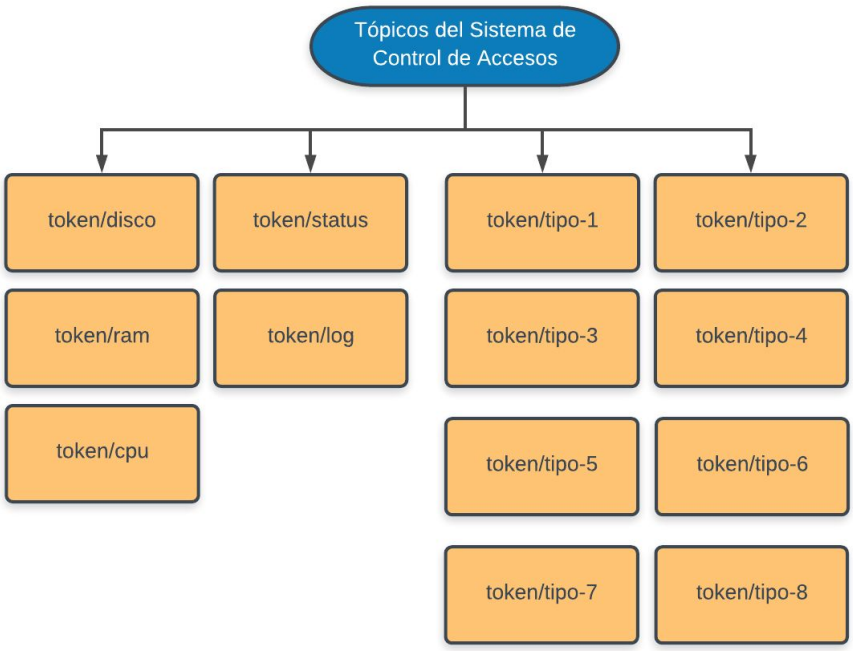


Figura 22. Total de tópicos del controlador.
Fuente: elaboración propia.

El controlador de *hardware* será controlado por una aplicación web en la nube; no obstante, como mecanismo de contingencia, en caso de que el controlador pierda la conexión a internet, se creó un mecanismo para administrar el controlador por medio de la red local por parte de un administrador del sistema.

Dicho mecanismo es una página web de acceso local, a la cual se puede acceder desde la dirección **http://direccion_ip:1880/ui** y consta de un panel de control creado con

el paquete *node-red-dashboard*. En este panel, se tiene acceso a gráficos con la información de los nodos de salida, así como dominio de cada uno de los dispositivos del controlador, como se observa en la figura 23.

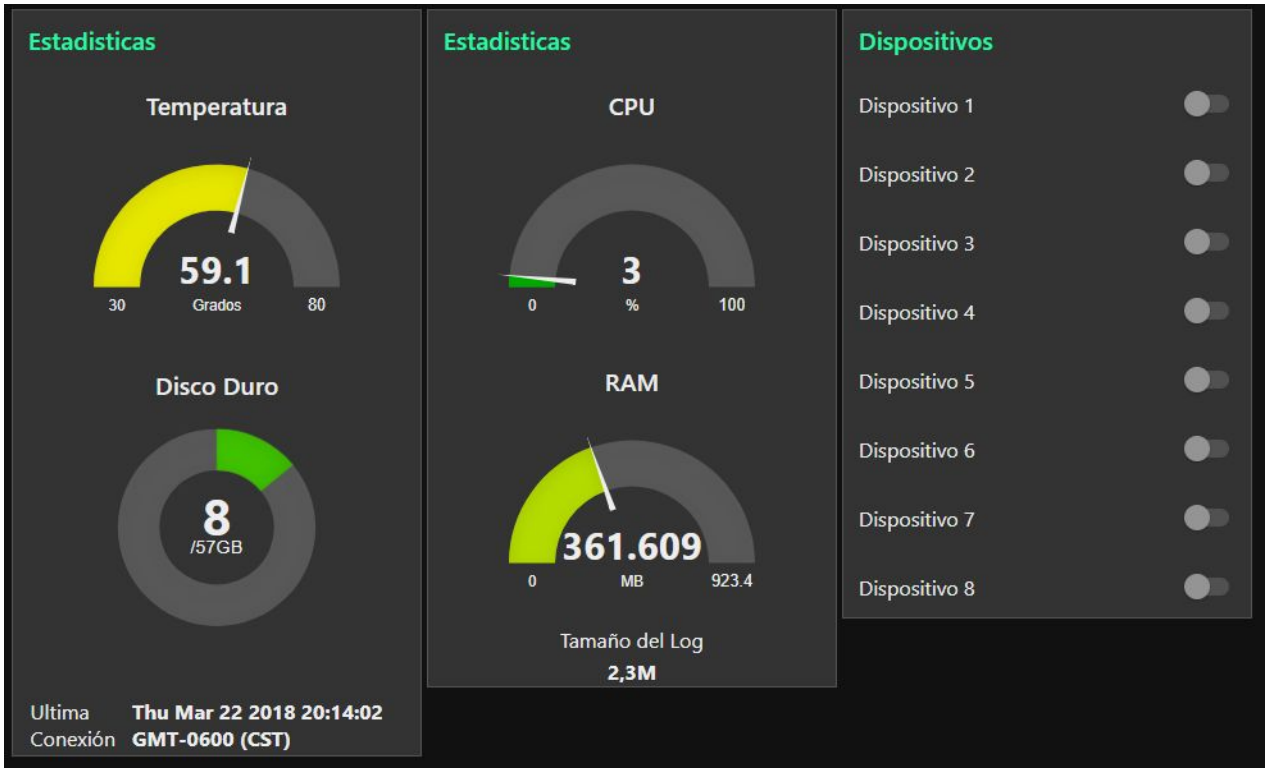


Figura 23. Panel de control de indicadores de acceso local.
Fuente: elaboración propia.

Este panel de control puede ser configurado y modificado de acuerdo con las necesidades del usuario del sistema de control de accesos una vez implementado el sistema.

4.4 Desarrollo de aplicación web en la nube para administración del sistema

4.4.1 Requerimientos

Con el propósito de detallar de una manera clara cada una de las funcionalidades del sistema y facilitar su desarrollo, es necesario definir los requerimientos de la aplicación web. A continuación, se detalla cada uno de ellos:

Requerimiento 1: inicio de sesión. El acceso al sistema debe estar restringido por el uso de claves asignadas a cada uno de los usuarios y transmitir los datos de sesión de forma segura.

Requerimiento 2: mantenimiento de roles. El sistema debe permitir crear, asignar y mantener roles de acceso. Además, debe ser posible asignar los derechos de acceso a los diferentes componentes dispositivos.

Requerimiento 3: mantenimiento de usuarios. El sistema debe hacer posible que se le dé mantenimiento a los usuarios y que se consulten, creen, modifiquen, borren los registros relacionados.

Requerimiento 4: mantenimiento de invitados. Cada usuario podrá consultar, crear, modificar, y borrar invitados, los cuales quedarán asociados a una identificación del usuario. A cada invitado se le debe generar un token único para identificarlos ante el sistema y que con ese código puedan utilizar un dispositivo.

Requerimiento 5: mantenimiento de controladores. El sistema debe permitir el mantenimiento de controladores, así como consultar, crear, modificar y borrar los registros relacionados y asignarle un token único que sirva como un token para la conexión con el servidor de MQTT Broker.

Requerimiento 6: mantenimiento de dispositivos. El sistema debe facilitar el mantenimiento de dispositivos y la consulta, creación, modificación y eliminación de los registros relacionados. Cada uno de los dispositivos se encuentra asociado a un controlador y a un número de controlador. Estos dispositivos pueden ser puertas magnéticas, motores eléctricos, agujas vehiculares o dispositivos de iluminación, los cuales pueden ser activados o desactivados desde el sistema.

Requerimiento 7: mantenimiento de empresas. El sistema debe hacer posible que se le dé mantenimiento a empresas y que se consulten, creen, modifiquen,

borren los registros relacionados. Las empresas van a estar relacionadas con cada uno de los controladores, dispositivos y usuarios.

Requerimiento 8: bitácora de eventos. El sistema deberá llevar una bitácora de eventos que registre las acciones que se dan a nivel de todos los componentes del sistema.

Requerimiento 9: bitácora de registro de acciones. El sistema deberá llevar un registro con todos los eventos que registren los dispositivos asociados a cada controlador.

Requerimiento 10: reportes. El sistema deberá presentar reportes con datos estadísticos sobre la utilización del sistema en cuanto al uso de dispositivos y acciones de los usuarios.

Requerimiento 11: interconexión con el MQTT Broker. El sistema debe permitir que se realice una conexión con el servidor de MQTT Broker para tener dominio de los dispositivos que se encuentren conectados físicamente al controlador.

Requerimiento 12: panel de control. El sistema deberá presentar un panel de control donde se muestre información del controlador y se pueda interactuar con cada uno de los dispositivos asociados a uno de ellos. Este panel sólo es accesible desde el rol administrador.

Requerimiento 13: idioma. El sistema deberá estar completamente en español.

4.4.2 Diseño de la aplicación

La aplicación fue diseñada para que los usuarios del sistema la utilicen como un servicio SaaS, y pueda ser accesible desde cualquier lugar o dispositivo. Se ejecuta sobre una máquina virtual con Ubuntu Server 16.04 en Amazon EC2 y utiliza el servidor web HTTP de código abierto Apache.

Como plataforma de desarrollo de la aplicación, se seleccionó el lenguaje de programación PHP versión 7.0 como *backend*, un diseño *responsive* con HTML5, CSS3 y Javascript como *frontend* y MySQL como gestor de base de datos.

Las herramientas de desarrollo mencionadas anteriormente son de código abierto y sus licencias permiten su uso para aplicaciones comerciales o no comerciales. Adicionalmente, se utilizaron complementos web de terceros, también de código abierto para mejorar el diseño, fluidez y la funcionalidad del sistema. En la tabla 15 se muestran los complementos utilizados.

Tabla 15
Complementos de terceros de código abierto utilizados en la aplicación

Complemento	Licencia
Bootstrap	MIT License
Chart.js	MIT License
Font Awesome	Creative Commons 4.0 / MIT License
InputMask	MIT License
Jquery	MIT License
Pnotify	MIT License
Select2	MIT License
Gentellela Alela!	MIT License
Eclipse Paho JavaScript client	ISC License

DataTables	MIT License

Fuente: elaboración propia.

La figura 24 muestra la estructura de carpetas de la aplicación, donde se organiza el código fuente y todos los archivos requeridos. La estructura es la siguiente:

api: contiene todas las funciones para interactuar con todos los objetos de la aplicación.

build: contiene los archivos CSS, imágenes y archivos Javascript necesarios para toda la aplicación.

login: es el directorio con el módulo de ingreso de los usuarios a la aplicación.

pages: contiene el *frontend* con todas las páginas y formularios.

template: contiene componentes y plantillas para el uso del *frontend*.

update: contiene las funciones necesarias para la actualización del *software* de las controladoras.

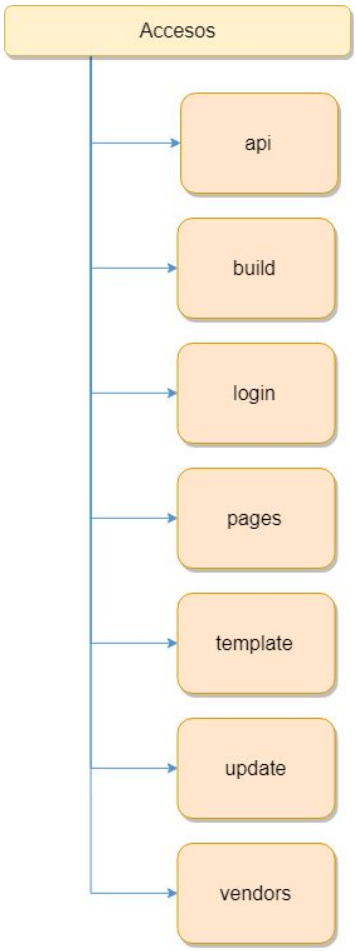


Figura 24. Estructura de carpetas de la aplicación.
Fuente: elaboración propia.

4.4.3 Base de datos

Como motor de base de datos relacional se seleccionó MySQL, que posee una licencia GNU GPL y puede ser utilizado para fines privados y comerciales. Este motor permite una fácil conexión con el lenguaje PHP. La base de datos se encuentra instalada en un VPS, el cual posee una alta disponibilidad y escalado, y posee mecanismos integrados para incrementar la seguridad, como segmentación de usuarios con mínimos permisos y listas blancas de IP de los clientes que se pueden conectar para evitar accesos remotos no autorizados.

En el anexo 1, se muestra el diagrama de entidad relación de la base de datos y, a continuación, se especifica la utilidad de cada una de las tablas.

Usuario: esta tabla almacena la información de cada uno de los usuarios que inician sesión en el sistema. Contiene los campos:

- **idUsuario:** identificador único del usuario
- **idEmpresa:** empresa a la cual se encuentra relacionado el usuario.
- **idRol:** rol que tiene asignado el usuario en el sistema.
- **nombre de usuario:** usuario con el cual se va a identificar en el sistema
- **contraseña:** se utiliza el algoritmo CRYPT_BLOWFISH para crear un *hash* y el resultado siempre será un *string* de 60 caracteres.
- **cédula:** cédula de la persona.
- **correo:** correo electrónico ligado al usuario.
- **teléfono:** número de teléfono de la persona.
- **otros_Datos:** campo que puede ser utilizado para almacenar información adicional del usuario.
- **activo:** identifica si el usuario se encuentra activado o desactivado. Si se encuentra desactivado, no podrá iniciar sesión en el sistema.
- **fechaCreacion:** campo que identifica la fecha de creación del usuario.
- **fechaUltimoAcceso:** campo que identifica la última fecha de acceso del usuario al sistema.

Controlador: esta tabla almacena la información de cada uno de los controladores de *hardware* del sistema. Contiene los campos:

- **idControlador:** identificador único del controlador.
- **idEmpresa:** empresa a la cual se encuentra relacionado el controlador.
- **idModelo:** campo que identifica el modelo del controlador.
- **serie_controlador:** campo que almacena el número de serie del controlador.
- **nombre_controlador:** nombre por el cual se puede identificar al controlador.
- **tópico**: campo que almacena el tópico con el cual se va a identificar el controlador para comunicarse con el servidor MQTT Broker.

Modelo: esta tabla es un catálogo que almacena la información de cada uno de los modelos de controladores. Contiene los campos:

- **idModelo:** identificador único del modelo.
- **nombreModelo:** nombre por el cual se identifica el modelo.
- **ram:** cantidad de RAM que posee el modelo.
- **disco:** tamaño del dispositivo de almacenamiento del modelo.
- **temperatura:** umbral máximo de temperatura que soporta el dispositivo.

Dispositivo: esta tabla almacena cada uno de los dispositivos de acceso físico que se conectan a cada controladora de *hardware* y se van a controlar por medio de la aplicación. Contiene los campos:

- **id_dispositivo:** identificador único del dispositivo.
- **idControlador:** controlador al cual se encuentra relacionado.
- **tipo_dispositivo:** especifica qué tipo de dispositivo es.
- **nombre_dispositivo:** nombre por el cual se identifica al dispositivo
- **descripcion:** almacena una pequeña descripción sobre el dispositivo.
- **conector:** especifica en cuál conector del controlador de *hardware* se encuentra conectado.
- **activado:** identifica si el usuario se encuentra activado o desactivado. Si se encuentra desactivado, no se podrá utilizar.
- **latitud:** especifica junto con la longitud la ubicación geográfica del dispositivo.
- **longitud:** especifica junto con la latitud la ubicación geográfica del dispositivo.

Tipo dispositivo: esta tabla es un catálogo que contiene los tipos de dispositivos de acceso físico que tiene la aplicación, y, con base en cada uno de los tipos, se tiene un comportamiento diferente de acuerdo con el tópico asignado. Contiene los campos:

- **idTipo_dispositivo:** identificador único del tipo de dispositivo.
- **nombre:** nombre por el cual se identifica al tipo de dispositivo
- **topico:** especifica el tópico específico por medio del cual se va a identificar a este dispositivo. El controlador de *hardware*, de acuerdo a cada uno de los tópicos,

tendrá un comportamiento distinto en cada uno de los dispositivos de control de acceso físico.

Empresa: esta tabla es muy importante, ya que almacena la información de empresas o clientes, las cuales son las propietarias de cada uno de los controladores y de los usuarios del sistema. La aplicación está diseñada para que a cada usuario le muestre únicamente la información y dispositivos de su empresa. Contiene los campos:

- **idEmpresa:** identificador único de la empresa.
- **nombre:** nombre de la empresa.
- **direccion:** dirección de la empresa.
- **telefono:** teléfono de la empresa.
- **representante:** nombre del representante de la empresa.
- **user:** usuario creado en el servidor MQTT, el cual es el único que puede publicar a los tópicos del controlador de *hardware*.
- **password:** *password* del usuario del servidor MQTT.

Rol: esta tabla almacena el rol que tiene un usuario en el sistema, puede ser un rol de administrador del sistema, un usuario normal u otro rol. Contiene los campos:

- **idRol:** identificador único del rol.
- **idEmpresa:** identificador de la empresa a la cual se encuentra relacionado.
- **nombreRol:** nombre del rol.

Invitado: esta tabla almacena a invitados, que pueden ser creados por cada usuario, a los cuales se les puede dar acceso temporal a los dispositivos a los cuales tengan permisos de utilizar. Contiene los campos:

- **idVisitante:** identificador único del invitado
- **idUsuario:** usuario que creó el invitado.
- **identificacion:** número de identificación del invitado, puede almacenar cédulas o número de pasaporte.
- **idPermiso:** identificador del permiso asociado.

- **nombre:** nombre del invitado.
- **telefono:** teléfono del invitado.
- **correo:** correo electrónico del invitado.
- **placa:** número de placa del vehículo del invitado.
- **fechaInicio:** fecha de inicio a la cual va a tener acceso al dispositivo.
- **fechaFinal:** fecha final a la cual se le retira el permiso de acceso al dispositivo.
- **token:** token único que sirve para validar la identidad de un invitado en el sistema.
- **activo:** campo que identifica si el invitado se encuentra activado o desactivado. Si se encuentra desactivado no se podrá utilizar.
- **uuid:** identificador único del celular o tableta desde el cual va a acceder a la aplicación. Este campo se utiliza para brindar una mayor seguridad y asociar un token a un solo dispositivo y asegurarse que no sea utilizado en otro dispositivo.

Permiso: esta tabla contiene los permisos a cada uno de los dispositivos configurados para cada uno de los invitados. Contiene los campos:

- **idPermiso:** identificador único del permiso.
- **idEmpresa:** empresa a la cual se encuentra relacionado el permiso.
- **descripción:** descripción del permiso.

Permiso_Invitados: esta tabla asocia cada permiso con cada uno de los dispositivos. Contiene los campos:

- **idPermisoInvitado:** identificador único del invitado.
- **idPermiso:** identificador único del permiso.
- **idDispositivo:** identificador único del dispositivo.

Registro: esta tabla almacena un registro de todas las acciones que se dan en los dispositivos. Contiene los campos:

- **idRegistro:** identificador único del registro.
- **idDispositivo:** identificador único del dispositivo al cual se encuentra relacionado el registro.

- **idUsuario:** identificador único del usuario. Si el registro fue accionado por un usuario, se llena este campo.
- **idVisitante:** identificador único del visitante. Si el registro fue accionado por un invitado, se llena este campo.
- **accion:** es la acción que se registra en el dispositivo.
- **fecha:** este campo es la fecha en la que se creó este registro.
- **estado:** este campo indica el estado de la acción registrada por el usuario, para establecer si fue autorizada o denegada.

Evento: esta tabla almacena un registro de eventos del sistema, como la creación de usuarios, creación de invitados, creación de dispositivos, entre otros. Contiene los campos:

- **idEvento:** identificador único del evento.
- **idEmpresa:** identificador único de la empresa a la cual se encuentra relacionado el evento.
- **fecha:** fecha en la que se registró el evento.
- **accion:** campo que indica cuál fue la acción que se realizó en la aplicación.

4.4.4 Desarrollo de la aplicación

Para que la aplicación web se pueda conectar con el controlador de *hardware* y pueda administrar los dispositivos de control de acceso físico, se utilizó una librería de Javascript llamada Paho MQTT JavaScript Client, la cual implementa la versión 3.1.1 del protocolo MQTT. Esta librería permite la conexión a un *broker* desde cualquier navegador de internet, por medio del uso de *websockets* y certificados SSL seguros; con lo cual genera una conexión constante a través del tiempo e intercambio de paquetes livianos.

La figura 25 describe el flujo para el envío de mensajes a través de la aplicación, para la utilización de un dispositivo de control de acceso físico.

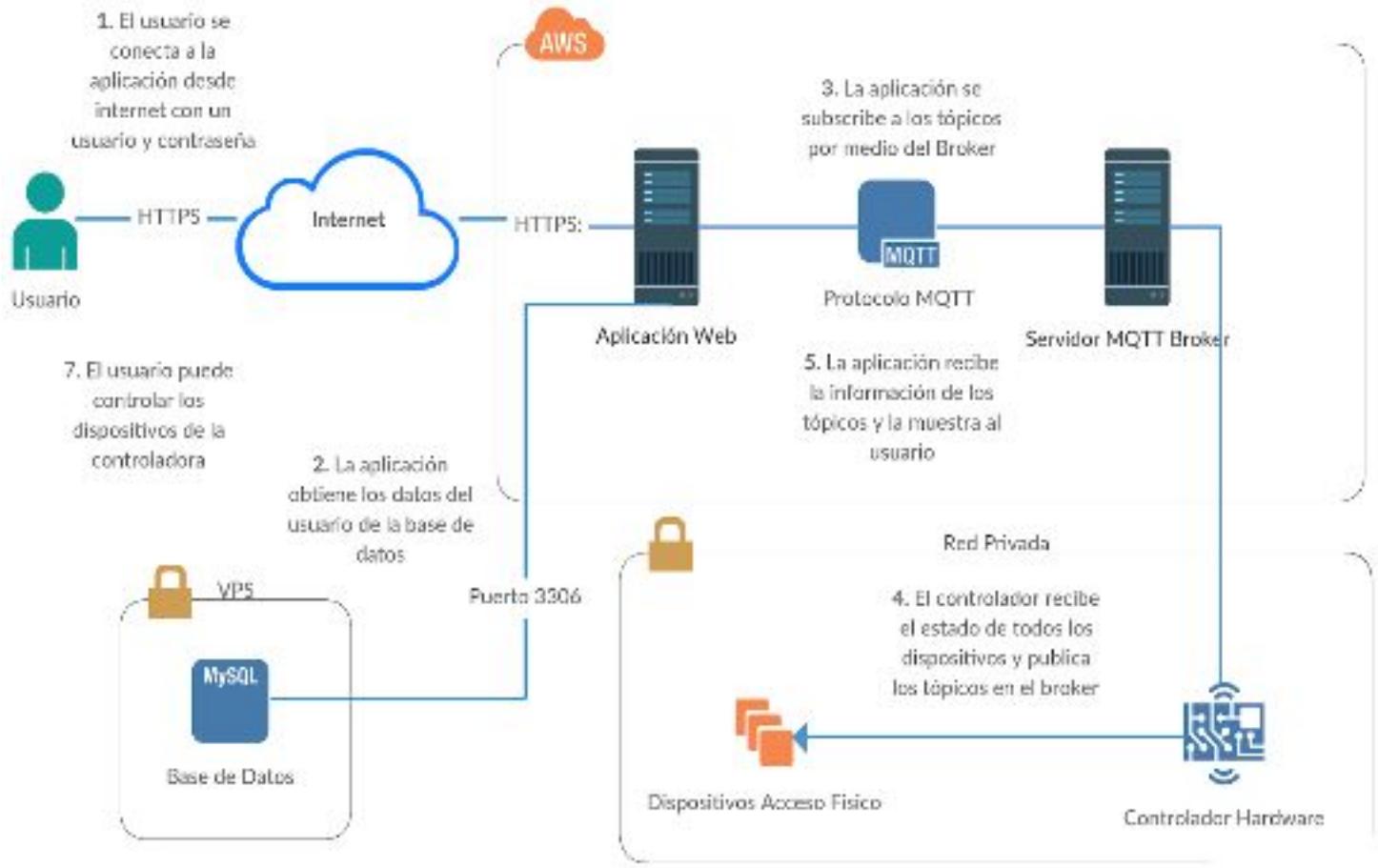


Figura 25. Flujo de la aplicación para la utilización de un dispositivo.
Fuente: elaboración propia.

La aplicación web cuenta con una ventana de inicio de sesión única, en la cual los usuarios ingresan independientemente de su rol, como se puede ver en la figura 26.

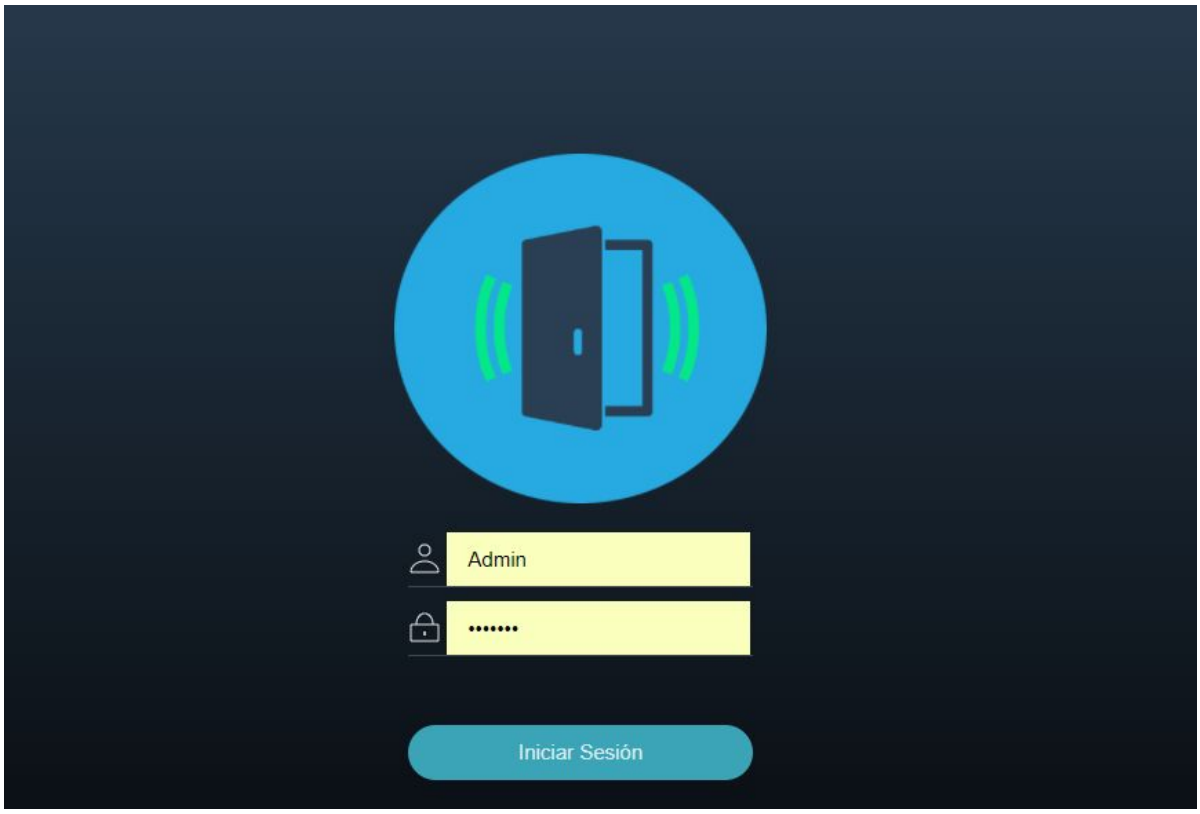


Figura 26. Inicio de sesión en la aplicación.
Fuente: elaboración propia.

Una vez iniciada la sesión de un usuario con el rol de administrador, se presenta un panel de control que muestra todos los controladores configurados, cada uno con gráficos de los datos actuales de temperatura, cantidad de memoria RAM en uso, espacio en disco, porcentaje CPU utilizado y si el controlador se encuentra en línea; así como sus respectivos dispositivos y las acciones que se pueden tomar de acuerdo con el tipo de dispositivo.

La figura 27 muestra el panel de control con un controlador configurado y sus gráficos y en la parte inferior de la pantalla se muestran tres dispositivos configurados: un dispositivo de iluminación, un acceso vehicular y una puerta magnética.

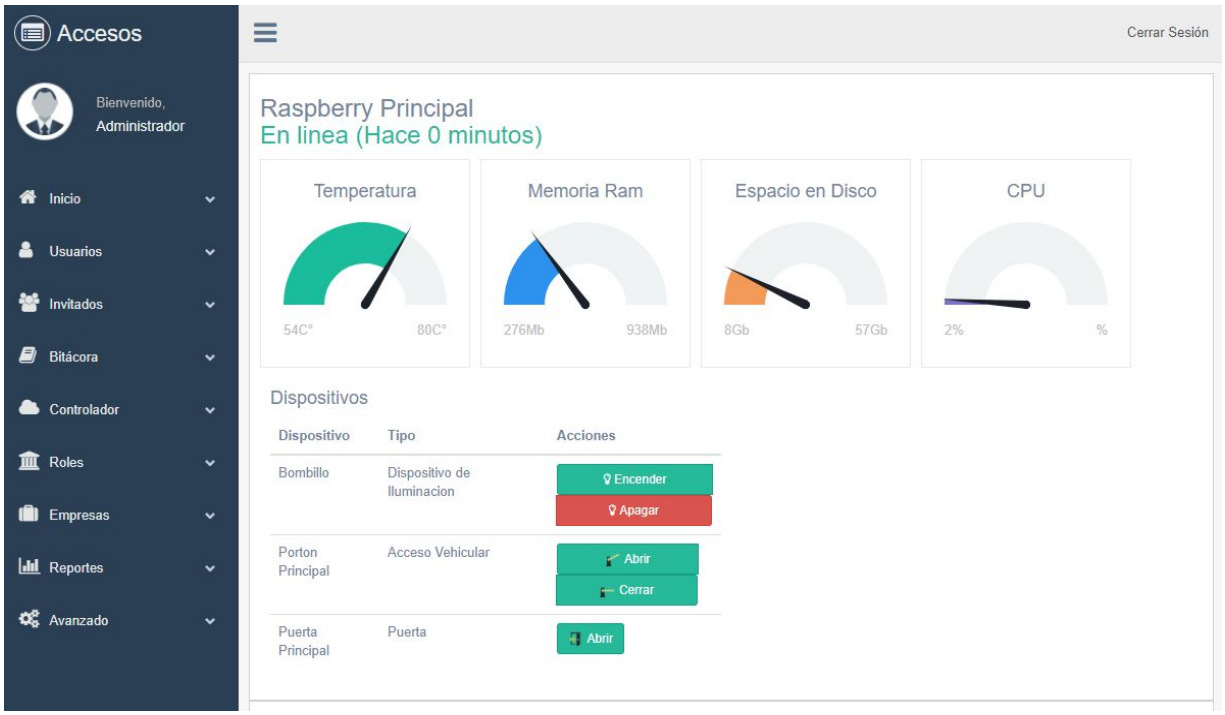


Figura 27. Panel de control de la aplicación.
Fuente: elaboración propia.

La aplicación permite al administrador dar mantenimiento a cada uno de los módulos del sistema:

Administración de usuarios: en este módulo se puede agregar, modificar y borrar los usuarios del sistema, así como generar nuevas contraseñas. La página se muestra en la figura 28.

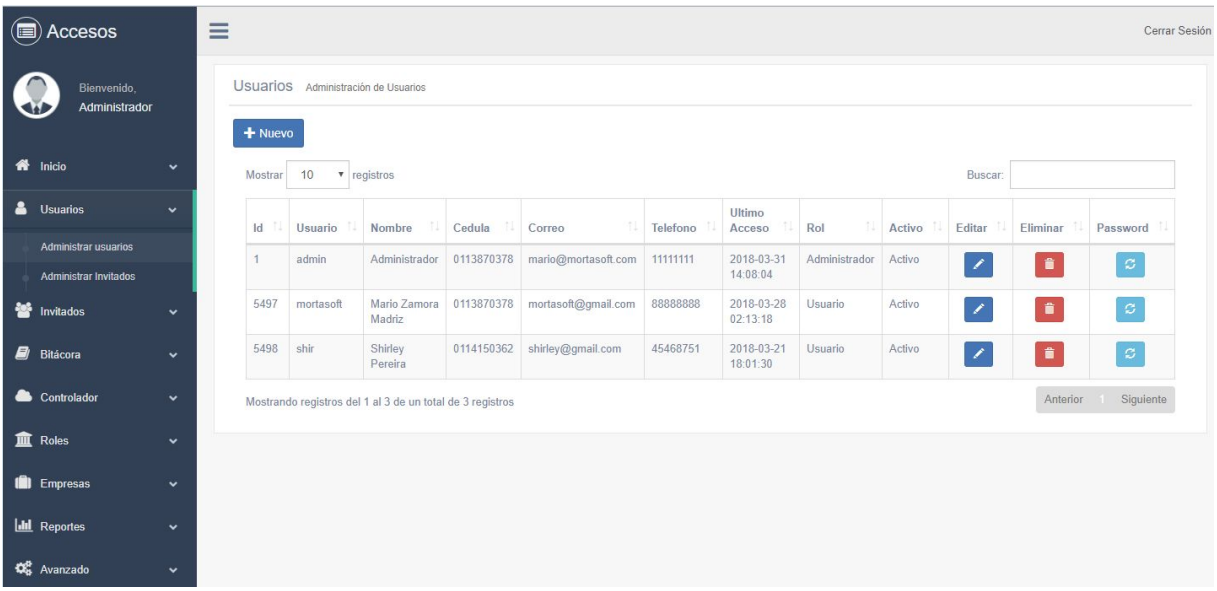


Figura 28. Administración de usuarios de la aplicación.
Fuente: elaboración propia.

Administración de invitados: en este módulo se puede agregar, modificar y borrar los invitados, además permite el envío por correo de una invitación personalizada a los invitados para el control de los dispositivos. La página se muestra en la figura 29.

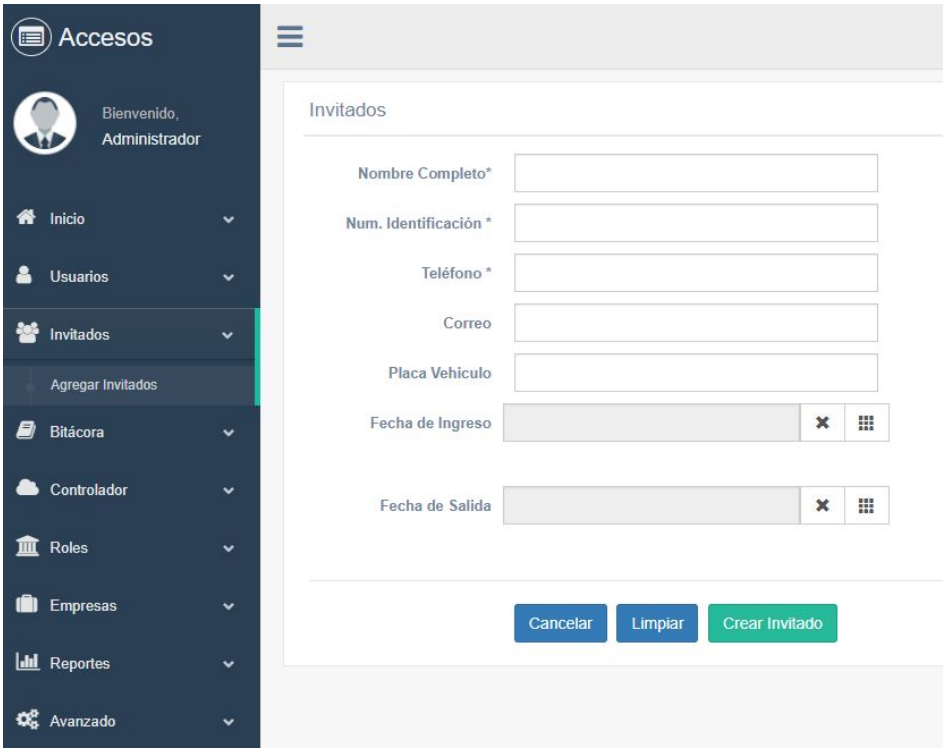


Figura 29. Administración de invitados de la aplicación.
Fuente: elaboración propia.

Bitácora: muestra el registro de eventos de los dispositivos y el registro de eventos del sistema en cuanto a la creación, eliminación y modificación de usuarios e invitados. Ver figura 30.



Figura 30. Bitácora de la aplicación.
Fuente: elaboración propia.

Administración de controladores: en este módulo se puede agregar, modificar y borrar los controladores del sistema y los dispositivos de control de acceso físico. Ver figura 31.



Figura 31. Administración de controladores de la aplicación.
Fuente: elaboración propia.

Administración de roles: se da mantenimiento a los roles de usuarios y se asignan grupos de permisos a los usuarios para el uso de cada dispositivo. Ver figura 32.

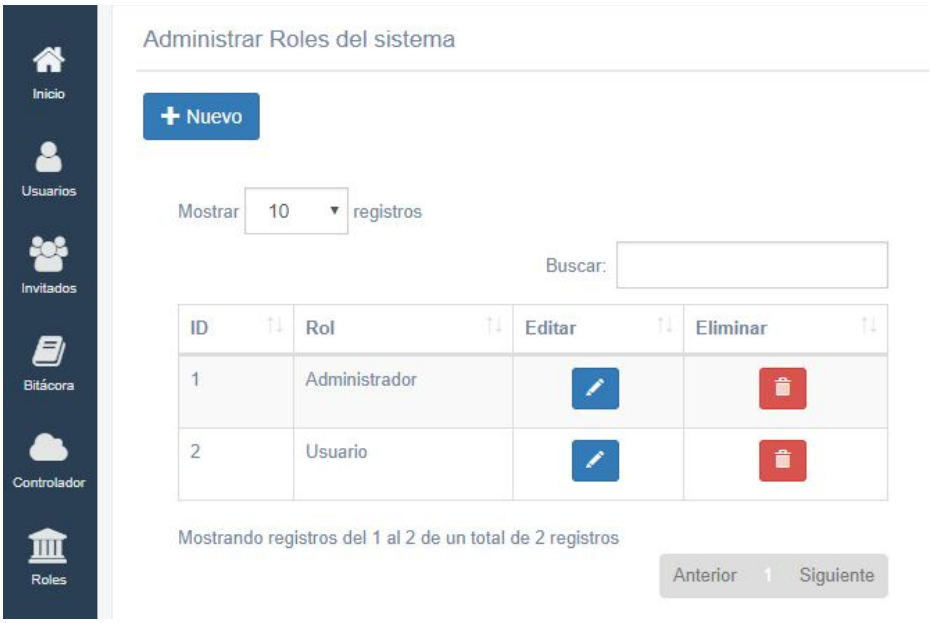


*Figura 32***.** Administración de roles de la aplicación.
Fuente: elaboración propia.

Reportes: muestra reportes y gráficos sobre los distintos componentes del sistema. Ver figura 33.



Figura 33. Reportes de la aplicación.
Fuente: elaboración propia.

4.4.5 Buenas prácticas de seguridad

Como medidas de seguridad para proteger la aplicación y los datos que se transmiten por parte del sistema, se implementaron controles para mitigar los riesgos de seguridad que se puedan presentar:

Principio de mínimo privilegio. Se creó un usuario en la base de datos con los mínimos privilegios sobre la misma y solo con permiso de modificar las tablas del sistema, para que ante cualquier fallo o vulnerabilidad se reduzca el impacto que pueda causar.

Protección contra SQL Injection. Se aplica un proceso de sanidad a todos los parámetros recibidos por GET o POST para eliminar todos los caracteres especiales, además la aplicación utiliza el controlador de base de datos PDO y consultas mediante sentencias preparadas para evitar ataques de SQL Injection.

```
    //PHP:
    filter_var_array($_POST, FILTER_SANITIZE_SPECIAL_CHARS);
$dbConnection = new PDO('mysql:dbname=db;host=ip; 'user', 'pass');
    $dbConnection->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
    $dbConnection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $dbConnection->select * from Usuario where usuario=:idusuario;
    $dbConnection->bind(':idusuario, $_POST[‘idusuario’]);
    $salida = $dbConnection->fetch();
```

Protección contra XSS. Cada uno de los campos de ingreso del sistema se validan mediante expresiones regulares:

```
    protected function validarNombre($nombre){
    if (!empty($nombre)) {
    $regex = '/^[a-z\s]{' . $this->nombreMin . ',' . $this->nombreMax .
'}$/i';
    if (!preg_match($regex, $nombre)) {
  array_push($this->mensajes, ['Componente' => "Nombre", 'Mensaje' => "El
nombre solo puede contener letras y números. De 2 a 32 caracteres "]);
          return False;
    }
    return True;
    }else{
    array_push($this->mensajes, ['Componente' => "Nombre", 'Mensaje' =>
"Nombre vacío"]);return False;
    }}
```

Cross-Site Request Forgery (CSRF). Cada vez que un usuario abre la aplicación, se crea una sesión en la aplicación web y se le asocia un token el cual es una cadena hexadecimal de caracteres aleatorios de 32 bits y cada vez que la aplicación realiza una solicitud a una página se valida que el token sea válido.

```
if (empty($_SESSION['token'])) {
 if (function_exists('mcrypt_create_iv')) {
  $_SESSION['token'] = bin2hex(mcrypt_create_iv(32, MCRYPT_DEV_URANDOM));
 } else {
  $_SESSION['token'] = bin2hex(openssl_random_pseudo_bytes(32));
 }
}
$token = $_SESSION['token'];
```

Certificados digitales. La aplicación utiliza un certificado digital SSL de 2048 Bits emitido por la entidad certificadora gratuita Let's Encrypt, lo que hace que la información se transporte de forma encriptada mediante el protocolo seguro https.

4.4.5.1 OWASP IoT Vulnerabilities Project

Para proteger la aplicación en un entorno de IoT, se siguen las recomendaciones dadas por el proyecto de OWASP IoT Vulnerabilities, además de implementarse una medida contra cada una de las vulnerabilidades en las superficies de ataque de aplicaciones web o interfaces en la nube. A continuación, se describen cada una de las amenazas y su contramedida.

Vulnerabilidad 1: enumeración de usuarios. El sistema está protegido contra enumeración de usuarios, ya que al iniciar sesión los mensajes de error no revelan si el usuario es válido dentro del sistema o si la contraseña es incorrecta. Ver figura 34.

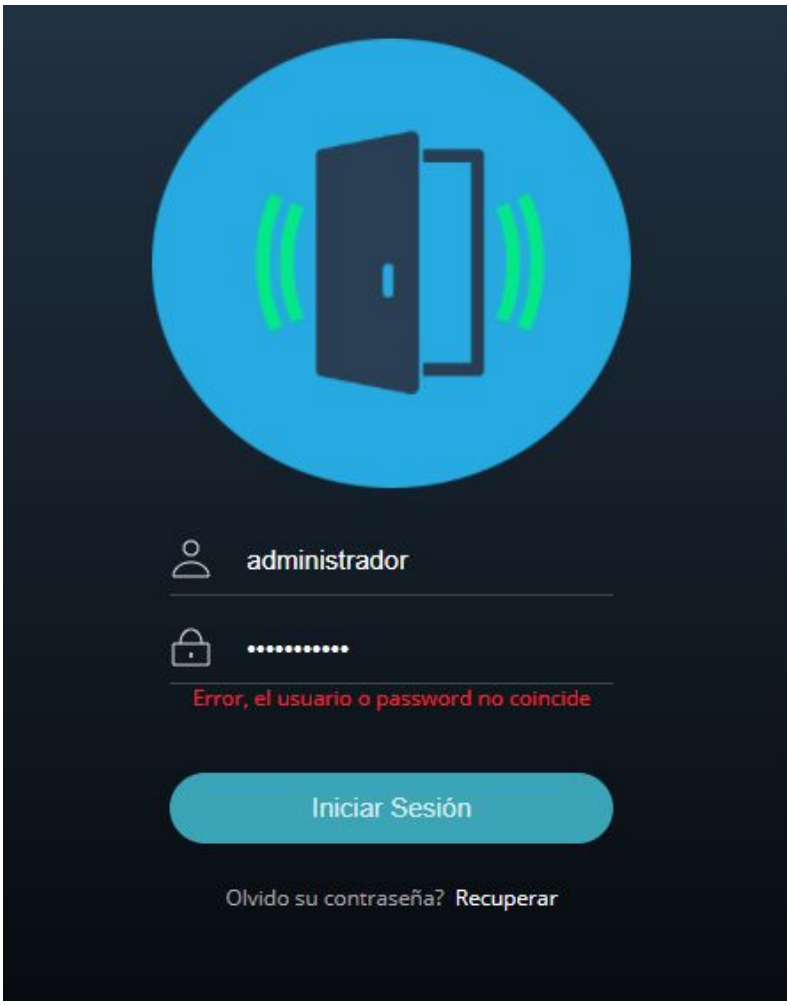


Figura 34. Protección contra enumeración de usuarios.
Fuente: elaboración propia.

Vulnerabilidad 2: contraseñas débiles. El sistema valida el *password* ingresado por el usuario por medio de una función y verifica que la contraseña cumpla con las siguientes reglas:

- El *password* debe tener más de 8 caracteres.
- El *password* debe incluir un número.
- El *password* debe incluir una letra.
- El *password* debe incluir un carácter especial.

```
protected function validarPassword($password){
  if (strlen($password) < 8){
```

```
   array_push($this->mensajes, ['Componente' => "Password", 'Mensaje' => "El
password debe tener más de 8 caracteres"]);
   return False;}
  if (!preg_match("#[0-9]+#", $password)) {
   array_push($this->mensajes, ['Componente' => "Password", 'Mensaje' => "El
password debe incluir un número"]);
   return False;}
  if( !preg_match("#[a-z]+#", $password) ) {
   array_push($this->mensajes, ['Componente' => "Password", 'Mensaje' => "El
password debe incluir una letra"]);
   return False;}
  if( !preg_match("#\W+#", $password) ) {
   array_push($this->mensajes, ['Componente' => "Password", 'Mensaje' => "El
password debe incluir un carácter especial"]);
   return False;}
  else { //Validado
   return True;
  }
 }
```

Vulnerabilidad 3: bloqueo de cuenta. Posibilidad de seguir enviando intentos de autenticación después de 3-5 intentos fallidos de inicio de sesión.

```
 $db->query('SELECT idusuario, usuario, nombre, password, activo, idempresa,
idrol, obtenerNombreEmpresa(idusuario)
nombreEmpresa,obtenerDireccionEmpresa(idusuario) direccion FROM Usuario WHERE
usuario=:usuario' and AND failed_login_attempts >= 3
AND last_failed_login > DATE_SUB(NOW(), INTERVAL 10 MINUTE)" or die
(mysql_error());
);
$db->bind(':usuario', $usuario');
$salida = $db->fetch();

// Si hay más de 1 registro crea la tabla
if ($db->numrows() >= 1) {
 //Usuario puede loguearse al sistema
}else{
 // El usuario superó
}
```

Vulnerabilidades 4 y 5: servicios sin cifrar y cifrado mal implementado. El servidor hace uso de certificados digitales SSL V2 para utilizar la aplicación mediante el protocolo seguro HTTPS y el protocolo MQTT para la comunicación entre la aplicación web y el MQTT Broker. Adicionalmente, se cambiaron los puertos por defecto del servicio del *broker* Mosquitto por el puerto 8888.

Vulnerabilidad 6: autenticación de dos factores. La aplicación que utiliza el visitante incluye una autenticación de dos factores, pues utiliza el número de serie UUID único del teléfono o tableta para identificar cada uno de los dispositivos, lo asocia a un número de token único en la aplicación y con esa verificación autentica al usuario y muestra las opciones correspondientes para el rol del usuario y los permisos sobre los dispositivos.

Vulnerabilidad 7: actualización enviada sin encriptación. Las actualizaciones del *software* del controlador se dan mediante una conexión SSL cifrada.

Vulnerabilidades 8 y 9: ubicación de actualización es modificable y mecanismo de actualización faltante. La aplicación incluye un mecanismo para actualizar el *software* del controlador remotamente, por medio de un *script* que se ejecuta mediante *cron* en el controlador. La ubicación del archivo solamente es editable por el usuario configurado en el controlador:

```
    #!/bin/bash
    dispositivo=$(<dispositivo.txt)
    version=$(<version.txt)
    usuario=$(whoami)
    #echo "Tipo de controlador: $dispositivo"
    #echo "Version del software: $version"
    #echo "Usuario: $usuario"
    rm /tmp/controlador-$dispositivo.sh
    wget -P /tmp/
https://mortasoft.xyz/Accesos/update/controlador-$dispositivo.sh
    chmod +x /tmp/controlador-$dispositivo.sh
    /tmp/controlador-$dispositivo.sh
```

Este *script* se comunica con el servidor y descarga otro *script* a la controladora:

```
    #!/bin/bash
    echo 'Actualizador del controlador'
    versionRemota=1.0.1
    usuario=$(whoami)
    dispositivo=$(</home/$usuario/update/dispositivo.txt)
    versionLocal=$(</home/$usuario/update/version.txt)
    echo "Tipo de controlador: $dispositivo"
    echo "Version actual del software: $versionLocal"
    echo "Usuario: $usuario"
    if [ "$versionRemota" == "$versionLocal" ]
    then
           echo "Tiene el software actualizado"
    else
```

```
        wget -P /tmp/
https://mortasoft.xyz/Accesos/update/controlador-$dispositivo.json
        cp /tmp/controlador-$dispositivo.json
/home/$usuario/.node-red/flows_raspberrypi.json
    node-red-stop
    node-red-start
        echo $versionRemota > /home/$usuario/update/version.txt
        echo "Se actualizo a la version $versionRemota"
    fi
```

Dicho *script* descargado valida la versión del *software* instalado en la controladora y, si la versión remota es mayor, entonces inicia el proceso de instalación de los nuevos archivos actualizados. Toda la nueva configuración se descarga en formato JSON.
La actualización del *software* se realiza automáticamente todos los días a las 2 a.m. y se programa con el comando *cron*:

```
crontab -e
2 0 * * * /home/usuario/update/script
```

Vulnerabilidad 10: denegación de servicio. La aplicación web se encuentra alojada en la nube de Amazon, por lo tanto, cuenta con el servicio de AWS Shield[1], que es un servicio de protección contra ataques de denegación de servicio distribuidos. La protección se da de forma automática y no tiene cargo adicional. Si se requiere, se puede adquirir contrademanda el servicio de AWS Shield Advanced, que además incluye un *firewall* para aplicaciones web AWS WAF.

Vulnerabilidad 11: componentes de terceros inseguros. Se creó un *script* de actualización de los paquetes del sistema operativo Raspbian que se ejecuta automáticamente. Se puede ver el *script* en el capítulo "4.3.4.1 Reforzamiento de seguridad en Raspbian" de este documento.

[1] Para más detalle, ir a https://aws.amazon.com/es/shield/

4.5 Modelo a escala

Con el propósito de poner a prueba el sistema de control de accesos completo y con el fin de determinar su factibilidad en una implementación real, se diseñó un modelo a escala de una puerta de aluminio con unas dimensiones de 50 cm x 30 cm, en la cual se instaló un cierre magnético para puerta de 280 kg de resistencia, que opera a 12V en corriente directa y 3 amperios, para lo cual se incluye una fuente de poder que transforma 110V a 12V.

Capítulo 5

Conclusiones y recomendaciones

En el siguiente capítulo, se presentan las conclusiones y recomendaciones sobre el tema de investigación una vez finalizado el análisis de resultados, los cuales se relacionan con los objetivos específicos planteados al inicio del proyecto.

5.1 Conclusiones

- Las empresas anualmente gastan grandes cantidades de dinero para la protección de sus bienes, ya que es a través de ellos que se puede cumplir con la misión de la compañía.
- Se evidencia que el mercado actual no cuenta con dispositivos de control de acceso de fácil uso, bajo costo y que sean robustos en términos de seguridad informática; muchos de los dispositivos actuales de media o baja gama presentan vulnerabilidades explotables o se desconocen por completo los mecanismos de seguridad que utilizan.
- Contar con una aplicación tipo SaaS y que utilice IoT, permite a las empresas hacer una menor inversión inicial, al no tener que adquirir servidores o *software* adicional para el funcionamiento del sistema; además de que se pueden acceder a actualizaciones de seguridad, nuevas funcionalidades, y soporte de forma inmediata.
- Aunado a esto, la solución propuesta ofrece el beneficio de requerir poco mantenimiento por parte de la empresa que lo implemente, al no tener que generar planes de contingencia para el respaldo y recuperación de los datos, lo que se traduce en una menor dependencia al departamento de TI y le permite a la compañía enfocar sus esfuerzos en su negocio.
- Al utilizar componentes de *hardware* y *software* libre y estándares abiertos, se crea un controlador para el dominio de accesos de buena calidad y de bajo costo, independiente de fabricantes y con una rápida corrección de errores y nuevas actualizaciones por parte de la comunidad de usuarios.

- El controlador diseñado es altamente personalizable y permite el control de gran cantidad de dispositivos diferentes, como puertas magnéticas, agujas vehiculares, motores eléctricos, dispositivos de iluminación por medio de una aplicación en internet. Esas funcionalidades no las posee ningún controlador del mercado.
- Si bien es cierto, el IoT es una tecnología que se encuentra en auge en los últimos años que, al ser tan nueva, ha sido blanco de los ciberdelincuentes y se han encontrado muchas vulnerabilidades en gran cantidad de dispositivos. Sin embargo, esto no es un problema de la tecnología en sí, sino de los fabricantes, que no se han preocupado por la seguridad en estos dispositivos y han dejado agujeros de seguridad inocentes y mecanismos de seguridad pobres que se quedan cortos en las implementaciones finales de los productos.
- Con el uso de buenas prácticas de seguridad y una actualización y revisión constante del *software*, se puede hacer del IoT una tecnología viable para una solución de control de accesos.

5.2 Recomendaciones

- Se recomienda la creación de otros modelos de controladores que posean distintas capacidades de *hardware* y utilicen la misma plataforma tecnológica, para adaptarse, de una mejor manera, a las necesidades de todos los clientes.
- Esta solución podría ser utilizada por empresas de pequeño o mediano tamaño o particulares; sin embargo, se recomienda que, para la implementación de este sistema de control de accesos, se realice una evaluación de riesgos para determinar si se adapta a sus necesidades, capacidad e infraestructura tecnológica.
- Para la implementación del sistema se recomienda firmar un contrato de confidencialidad y un acuerdo de nivel de servicio entre el proveedor del servicio y el cliente que adquiera el sistema, para asegurar que la información que contenga el mismo no vaya a ser utilizada para otros fines y que el servicio posea un nivel aceptable de disponibilidad, confidencialidad e integridad de los datos.

- Es aconsejable hacer pruebas de penetración periódicas a la aplicación web y actualizaciones constantes a las controladoras, para encontrar vulnerabilidades que puedan ser explotadas y evitar posibles ataques.

Capítulo 6

Trabajos futuros

Como parte de los trabajos futuros que se pueden realizar para mejorar el sistema de control de accesos e incluir nuevas funcionalidades se pueden mencionar:

- Implementación de herramientas cognitivas como reconocimiento de voz, imagen o rostros para la identificación de usuarios y la utilización de los dispositivos de acceso.
- Control de dispositivos por medio de mensajería instantánea, mediante el uso de *chatbot*s, por ejemplo, Slack, Telegram, Facebook, entre otros.
- Incluir sensores inteligentes al sistema como sensores de movimiento, entrada por reconocimiento facial, videovigilancia.
- Crear una aplicación móvil nativa para Android y iOS para el control de los dispositivos por medio de teléfonos.
- Implementar dentro de los controladores mecanismos alternos en caso de fallo, como un módulo 4G alternativo, en caso de que se suspenda la conexión a internet por parte del proveedor de servicios y un módulo de baterías en caso que se corte el suministro eléctrico.
- Agregar un módulo 4G a la controladora en caso de pérdida de internet por parte del proveedor de servicios.

Referencias

Amazon. (2017a). *Proximity RFID Card 125KHZ EM4100 Read Only ID Smart Door Entry Access Control Plastic Card (pack of 10) -YARONGTECH*. Recuperado el 05 de setiembre de 2017 de https://www.amazon.com/Proximity-125KHZ-Control-Plastic-YARONGTECH/dp/B01M1L7NHI/

Amazon. (2017b). *UHPPOTE UT0814-S4110V Security Network RFID Access Control Board Kit Metal AC110V Power Box For 4 Doors*. Recuperado el 24 de setiembre de 2017 de https://www.amazon.com/UHPPOTE-UT0814-S4110V-Security-Network-Control/dp/B00VBRUYNY

Amazon. (2017c). *Touch Key 7" Lcd RFID Password Video Door Phone Intercom System Kit+ Electric Strike Lock+ Wireless Remote Control unlock.* Recuperado el 08 de octubre de 2017 de https://www.amazon.com/MOUNTAINONE-Doorbell-Intercom-1-camera-1-monitor/dp/B017GVSIAY

Amazon. (2017d). *JBtek 8 Channel Module Arduino Raspberry.* Recuperado el 07 de enero de 2018 de https://www.amazon.com/JBtek-Channel-Module-Arduino-Raspberry/dp/B00KTELP3I

Arduino. (2017). *Arduino Uno Rev 3.* Recuperado el 08 de setiembre de 2017 de https://store.arduino.cc/usa/arduino-uno-rev3

Assa Abloy. (2017). *Securitron DK-16.* Recuperado el 25 de agosto de 2017 de http://www.intelligentopenings.com/en/site/intelligentopenings/products/keypad-access-control/securitron-dk-16/

Bar Codes Inc. (s.f). *RFID Buying Guide*. Recuperado el 01 de setiembre de 2017 de https://www.barcodesinc.com/info/buying-guides/rfid.htm

Bloom, B. (1990). Taxonomía de los objetivos de la educación. Buenos Aires: El Ateneo

Denso Wave Incorporated. (2011). *QR Code® Essentials*. Recuperado el 01 de setiembre de 2017 de http://denso-adc.com/download/whitepaper/1

Denso Wave Incorporated. (s.f). *QR Code Standardization*. Recuperado el 01 de setiembre de 2017 de http://www.qrcode.com/en/about/standards.html

Element14. (2015). *Raspberry Pi 3 Model B GPIO 40 Pin Block Pinout*. Recuperado el 05 de agosto de 2017 de https://www.element14.com/community/docs/DOC-73950/l/raspberry-pi-3-model-b-gpio-40-pin-block-pinout

Eclipse IoT Working Group. (s.f). *Mosquitto: An Open Source MQTT v3.1/v3.1.1 Broker.* Recuperado el 06 de setiembre de 2017 de https://mosquitto.org/

Everfocus Electronics Corp. (2017). *Access Control.* Recuperado el 08 de octubre de 2017 de http://www.everfocus.com/_product/access-control_controllers/#accordion

Free Software Foundation. (2001). *Qué es el software libre.* Recuperado el 15 de setiembre de 2017 de https://www.gnu.org/philosophy/free-sw.html

Gartner. (2017). *Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016.* Recuperado el 28 de agosto de 2017 de http://www.gartner.com/newsroom/id/3598917

Hikvision. (s.f). *Video Intercom Water Proof Metal Door Station*. Recuperado el 12 de 03 de 2018 de http://www.hikvision.com/en/Products_accessries_700_i5989.html

Honeywell. (2017). *Honeywell Access Control System*. Recuperado el 24 de setiembre de 2017 de https://www.honeywellaccess.com/products/access-control-systems/web-based/294386.html

IBM. (2017). *Combating IoT Cyber Threats: Security best practices for IoT applications.* Recuperado el 15 de 09 de 2017 de https://www.ibm.com/developerworks/library/iot-security-best-practices-iot-apps/iot-security-best-practices-iot-apps-pdf.pdf

IEEE Spectrum. (2011). *The Making of Arduino.* Recuperado el 25 de agosto de 2017 de https://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino

Intel. (2014). *Intel Galileo Gen 2 Development Board.* Recuperado el 05 de setiembre de 2017 de https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/galileo-g2-datasheet.pdf

Intel. (2017). *Intel® Galileo Discontinuation Notice.* Recuperado el 05 de setiembre de 2017 de https://communities.intel.com/docs/DOC-112094

ISO/IEC. (2016). *ISO/IEC 20922:2016 Information technology -- Message Queuing Telemetry Transport (MQTT) v3.1.1.* Recuperado el 06 de setiembre de 2017 de http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

Jammes, F. (2016) *Internet of Things in Energy Efficiency*. Ubiquity. 2016, p.1-p.8

Keri Systems Incorporated. (2017). NXT-4D – FOUR DOOR/EIGHT READER CONTROLLER. Recuperado el 08 de octubre de 2017 de https://www.kerisys.com/pages/products/nxt/nxt-4d/

Keyscan Access Control Systems. (2017). *Keyscan CA4500 4-Reader Access Control Unit.* Recuperado el 08 de 10 de 2017, de http://www.keyscan.ca/English/PDFs/Products/CA4500-0317-E.pdf

Kim, D., & Solomon, M. (2010). *Fundamentals of Information Systems Security.* Estados Unidos: Jones & Barlett Learning.

Law 360. (2017). *N.P. v. Standard Innovation (US), Corp.* Recuperado el 15 de setiembre de 2017 de https://www.law360.com/cases/57c9efc14184df6b35000004

Paxton. (2017). *Paxton Standalone Systems.* Recuperado el 08 de octubre de 2017 de https://www.paxton-access.com/products.asp?strStage=stand_alone

Pearce, J. M. (2014). *Open-Source Lab: How to Build Your Own Hardware and Reduce Research Costs.* Elsevier Science

Raspberry Pi Foundation. (2017a). *Raspberry Pi FAQs*. [Preguntas Frecuentes de Raspberry Pi]. Recuperado el 25 de octubre de 2017 de https://www.raspberrypi.org/help/faqs/

Raspberry Pi Foundation. (2017b). *Raspberry Pi 3 Model B*. Recuperado el 31 de agosto de 2017de https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Raspberry Pi Foundation. (2017c). *Raspberry Pi Zero W*. Recuperado el 31 de agosto de 2017 de https://www.raspberrypi.org/products/raspberry-pi-zero-w/

Raspberry Pi Foundation. (2017d). *GPIO: Raspberry Models A and B*. Recuperado el 31 de agosto de 2017 de https://www.raspberrypi.org/documentation/usage/gpio/

RBH Access Technologies. (2014). *Int-unc100.* Recuperado el 12 de marzo de 2018 de http://www.rbh-access.com/products/integra32/hardware/int-unc100

Rfid Journal. (2009). *That 'Internet of Things' Thing.* Recuperado el 07 de setiembre de 2017 de http://www.rfidjournal.com/articles/view?4986

Suprema Solution. (2017). *BioStation*. Recuperado el 08 de octubre de 2017 de http://www.supremasolution.com/BioStation.php

Stallman, R. (1983). *A Serious Bio.* Recuperado el 15 de setiembre de 2017 de http://www.stallman.org/cgi-bin/showpage.cgi?path=/common/bio.html

The Guardian. (2016). *DDoS attack that disrupted internet was largest of its kind in history, experts say.* Recuperado el 14 de diciembre de 2017 de https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet

The Internet Engineering Task Force (IETF). (2005). *HOTP: An HMAC-Based One-Time Password Algorithm.* Recuperado el 17 de enero de 2018 de https://tools.ietf.org/html/rfc4226

The Open Web Application Security Project OWASP (2018). *OWASP Internet of Things (IoT) Project*. Recuperado el 12 de febrero de 2018 de https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project

VaGla (2007). Polski: kod EAN-13. Recuperado el 02 de setiembre de 2017de https://upload.wikimedia.org/wikipedia/commons/8/84/EAN13.svg

Woodland, N. J., & Bernard, S. (1952). *Patente de Estados Unidos No. US 2612994*. Washington, DC: U.S. Patent and Trademark Office.

Glosario

API: *Application Programming Interface,* es un conjunto de funciones de un *software* para comunicarse o interactuar con otro.

ARM: *Advanced RISC Machine*, es una arquitectura de procesadores que requieren menos transistores, lo cual mejora su costo, consumo de energía, y disipación del calor. Es ampliamente utilizado en teléfonos y tabletas.

Backend: en el diseño de *software*, el *backend* es el componente de *software* que procesa los datos ingresados por el usuario en el *frontend* y produce una respuesta para mostrarla al usuario.

BOTNET: es un término que hace referencia a un conjunto o red de robots informáticos o *bots*, que se ejecutan de manera autónoma y automática. Suelen ser utilizados por atacantes para realizar ataques informáticos masivos.

Frontend: en el diseño de *software*, el *frontend* es el componente que interactúa directamente con los usuarios para ingresar datos de entrada y mostrar los resultados.

GPIO: *General Purpose Input/Output*, es un pin que puede ser programado para realizar una amplia gama de tareas de entrada y salida.

IETF: *Internet Engineering Task Force*, es una organización internacional abierta de normalización en ingeniería de internet.

LED: *Light Emitting Diode*, es un diodo semiconductor que produce una fuente de luz al alimentar de energía sus dos terminales.

LXDE: *Lightweight X11 Desktop Environment*, es un entorno de escritorio ligero y libre para entornos Unix.

Payload: en computación y telecomunicaciones, el *payload* es la parte de los datos transmitidos, que es en realidad el mensaje enviado.

PLA: poliácido láctico, fibra utilizada para la impresión en 3D.

PIN: *Personal Identification Number*, un número de identificación personal.

QR: *Quick Response Code*, es un tipo de código de barras bidimensional utilizado para contener información y ser leído fácilmente por una máquina lectora.

Relay: es un dispositivo electromagnético que funciona como un interruptor que controla por medio de una bobina y un electroimán, el paso de por electrodos de otros circuitos electrónicos externos.

RFID: *Radio Frequency Identification*, es una tecnología que utiliza campos electromagnéticos para identificar y seguir etiquetas adheridas a un objeto.

SaaS: *Software as a Service*, es un modelo en el cual el *software* es licenciado por medio de una suscripción a un servicio central por medio de internet y usualmente es accedido por medio de un navegador de internet.

SoC: *System on a Chip*, es una tendencia de fabricación que integra todos o gran parte de los módulos que componen un computador o sistema electrónico en un único circuito integrado.

TI: tecnologías de información.

USB: *Universal Serial Bus*, es un estándar que define cables, conectores y protocolos para la comunicación y alimentación entre computadoras dispositivos periféricos.

VPS: *Virtual Private Server*, es una máquina virtual alquilada como servicio por parte de un servicio de *hosting* en internet.

Anexo 1

A continuación, se muestra el diagrama de entidad relación de base de datos, de la aplicación web del sistema de control de accesos.

