UNIVERSIDAD POLITÉCNICA SALESIANA DEL ECUADOR

SEDE GUAYAQUIL

CARRERA DE INGENIERÍA ELECTRÓNICA

PROYECTO TÉCNICO DE TITULACIÓN

TÍTULO:

"Diseño e Implementación de una Plataforma de Control de Acceso Magnético/Biométrico y Supervisión Remota basado en Raspberry Pi"

AUTORES:

Dennys Andres Cedeño Ramos

Luis Eduardo Moncayo Solís

TUTORA:

Ing. Mónica Miranda R.

GUAYAQUIL - 2019

INFORMACION DE LOS AUTORES

Dennys Andres Cedeño Ramos

Estudiante de la Carrera de Ingeniería Electrónica de la Universidad Politécnica Salesiana

Email: dennys_9415@hotmail.com, dcedenor@est.ups.edu.ec

Luis Eduardo Moncayo Solís

Estudiante de la Carrera de Ingeniería Electrónica de la Universidad Politécnica Salesiana

Email: luis_mon24@hotmail.com, lmoncayos@est.ups.edu.ec

DECLARACIÓN DE RESPONSABILIDAD Y AUTORÍA

Yo, Dennys Andres Cedeño Ramos, y, Luis Eduardo Moncayo Solis, declaramos que somos los únicos autores de este trabajo de titulación titulado “Diseño e Implementación de una Plataforma de Control de Acceso Magnético/Biométrico y Supervisión Remota Basado en Raspberry Pi”. Los conceptos aquí desarrollados, análisis realizados y las conclusiones del presente trabajo, son de exclusiva responsabilidad de los autores.

_____	_____
Dennys Andres Cedeño Ramos	Luis Eduardo Moncayo Solis
C.I.:0952457869	C.I.:0931242069

DECLARACIÓN DE CESIÓN DE DERECHOS DE AUTOR

Quien suscribe, en calidad de autora del trabajo de titulación titulado “Diseño e Implementación de una Plataforma de Control de Acceso Magnético/Biométrico y Supervisión Remota Basado en Raspberry Pi”, por medio de la presente, autorizo a la UNIVERSIDAD POLITÉCNICA SALESIANA DEL ECUADOR a que haga uso parcial o total de esta obra con fines académicos o de investigación.

_____	_____
Dennys Andres Cedeño Ramos	Luis Eduardo Moncayo Solís
C.I.:0952457869	C.I.:0931242069

DECLARACIÓN DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Quien suscribe, en calidad de director del trabajo de titulación titulado "Diseño e Implementación de una Plataforma de Control de Acceso Magnético/Biométrico y Supervisión Remota Basado en Raspberry Pi", desarrollado por los estudiantes Dennys Andres Cedeño Ramos y Luis Eduardo Moncayo Solis previo a la obtención del Título de Ingeniería Electrónica, por medio de la presente certifico que el documento cumple con los requisitos establecidos en el Instructivo para la Estructura y Desarrollo de trabajos de Titulación para pregrado de la Universidad Politécnica Salesiana. En virtud de lo anterior, autorizo su presentación y aceptación como una obra autentica y de alto valor académico.

Dado en la Ciudad de Guayaquil, a los 25 días del mes de Julio del 2018

_____

Ing. Mónica Miranda R.

Docente Directora del Proyecto Técnico

DEDICATORIA

Este Proyecto de titulación está dedicado a mi madre Cecilia Ramos Celleri, que siempre me ha apoyado en todo momento inculcándome valores y enseñándome a no rendirme. A mi hermano menor Henry Cedeño Ramos y a mi padre Guido Cedeño García por estar siempre a mi lado. Ya que han sido mi pilar fundamental en mi vida para seguir siempre adelante y lograr terminar con éxito mi carrera.

Dennys Cedeño Ramos

DEDICATORIA

Durante mi recorrido de estudios, esta dedicatoria es en especial a Dios quien me ayudó a llegar a estas instancias, a mi familia que con su apoyo brindado lograron que culmine mis estudios universitarios, gracias a mi Señor quien me brindo fuerzas en todo momento para lograr mis objetivos.

Luis Moncayo Solis

AGRADECIMIENTOS

Mi agradecimiento principal es para Dios por guiarme y darme fortaleza y sobre todo paciencia.

A mi madre Cecilia Ramos Celleri que es el pilar fundamental, sin ella nada sería posible, por su apoyo, comprensión y cariño; a mi hermano Henry Cedeño Ramos y a mi padre Guido Cedeño García, y a todos los que creyeron en mí en todo momento y que podía terminar mi carrera.

Dennys Cedeño Ramos

AGRADECIMIENTO

Un profundo agradecimiento a Dios por las bendiciones recibidas en todo momento de mi vida. A mis padres y hermanos que son el pilar fundamental de apoyo para conseguir mis objetivos. A los docentes y compañeros de la carrera por compartir sus enseñanzas y conocimientos durante mis estudios, a la empresa Credilit por permitir desarrollar este proyecto de titulación para poder culminar mi carrera universitaria.

Luis Moncayo Solis

RESUMEN

En la presente propuesta de una plataforma de control de acceso magnético/biométrico y supervisión remota basada en Raspberry Pi, se realizó en la empresa CREDILIT, ubicada en la ciudad de Guayaquil; para determinar la aplicación de este proyecto técnico se ha basado en referencias del historial de seguridad de la empresa, por medio de una auditoría se pudo evidenciar la falta de control de seguridad dentro del almacén, además de tener en cuenta el área en donde se encuentra localizado que tiene cierta inseguridad.

Este trabajo presenta el diseño, construcción e implementación de una plataforma de control de acceso y supervisión remota, donde se concentra gran parte en los dispositivos Raspberry Pi 3 y Pi Zero, y en el Arduino Pro Mini permitiendo el control de entrada y de salida, y la gestión de la misma agregando y/o eliminando usuarios en la base de datos, el sistema está conformado por una cámara conectado directamente a la Raspberry Pi Zero, un dispositivos de acceso Biométrico mediante huellas dactilares, y otro dispositivo de acceso Magnético mediante tarjetas identificadoras, adicional la parte de seguridad compuesta por un sistema basado en mensajes de aviso.

En la parte del control se basa en el desarrollo de una aplicación mediante VueJS y NodeJS para la gestión de usuarios y permisos de acceso a la bodega; el sistema de supervisión remota puede establecer conexión de internet desde cualquier parte del mundo, así mismo para la gestión de las tarjetas y huellas dactilares; mediante Python se desarrolla el control de la cámara para la obtención automática de fotos en un determinado tiempo.

En la parte de la seguridad se encuentra establecida en mensajes de advertencia de diversos tipos; el primer mensaje trata de mensajes de texto GSM desarrollado en el dispositivo Arduino Uno y una placa GSM/GRP en la que indica que se intentó un ingreso no identificado, y el segundo mensaje desarrollado por Python para enviar un mensaje email en la que indica el mismo mensaje adicionando una foto de la escena obtenida de una cámara conectada a la Raspberry pi 3, los dos mensajes son enviados bajo la misma condición.

La plataforma de control de acceso posee una base de datos MySQL en la Raspberry con el registro de entradas y salidas de la empresa, además se implementan protocolos de seguridad por medio de una clave de acceso a las bases de datos para que sea gestionada por personas autorizadas.

Los autores del trabajo de titulación eligen apropiadamente los lugares específicos para la posición e implementación de los dispositivos de control de acceso y gestión pertenecientes a la plataforma para la mejora el control global del esquema. El

presente trabajo establece una optimización en los niveles de control y seguridad de la propiedad comercial.

CREDILIT, se caracteriza por ser un local comercial de ventas de artículos mobiliarios domésticos, donde brindan atención al cliente y se almacena una gran cantidad de bienes de un gran valor monetario. Pero no posee un buen sistema de seguridad, además de estar localizado en un área insegura por la noche.

ABSTRACT

The presentation proposes a platform for magnetic access control / biometric and remote monitoring based on Raspberry Pi, was carried out in the company CREDILIT, located in the city of Guayaquil; to determine the application of this technical project has been based on references in the safety record of the company, in the middle of an audit can be evidenced the lack of control of security in the warehouse, in addition to taking into account the area where it is located.

This paper presents the design, construction and implementation of an access control and remote supervision platform, where a large part is concentrated in the Raspberry Pi 3 and Pi Zero devices, and in the Arduino Pro Mini allowing the control of entry and exit, and the management of the same adding and / or eliminating users in the database, the system is conformed by a camera connected directly to the Raspberry Pi Zero, a biometric fingerprint access devices, and another magnetic access device using cards Identifiers, additional security part composed of a system based on warning messages.

In the control part, it is based on the development of an application through VueJS and NodeJS for the management of users and access permits to the winery; the remote monitoring system can establish an internet connection from anywhere in the world, as well as for the management of cards and fingerprints; Through Python, the control of the camera is developed to automatically obtain photos in a certain time.

In the security part, it is established in warning messages of various types; the first message deals with GSM text messages developed in the Arduino Uno device and a GSM / GRP board in which it indicates that an unidentified entry was attempted, and the second message developed by Python to send an email message indicating the same message adding a photo of the scene obtained from a camera connected to the Raspberry Pi 3, the two messages are sent under the same condition.

The access control platform has a MySQL database in the Raspberry with the entry and exit records of the company, in addition security protocols are implemented by means of a key to access the databases so that it is managed by people authorized.

The authors of the titling work appropriately choose the specific places for the position and implementation of the access control and management devices belonging to the platform for the improvement of the overall control of the scheme. The present work establishes an optimization in the levels of control and security of commercial property.

CREDILIT, is characterized by being a commercial place for sales of domestic furniture, where they provide customer service and are stored a large amount of goods consisting of a large monetary value. But it does not have a good security system, besides being located in an unsafe area at night.

ÍNDICES GENERAL

Contenido

ÍNDICES FIGURA

Contenido

ÍNDICES DE TABLAS

Contenido

ÍNDICES DE ANEXOS

Contenido

ABREVIATURAS

GPU: Graphics processing unit “Unidad de procesamiento gráfico”

SBC: Session border controller “Controlador de borde de sesión”

SFTP: Secure File Transfer Protocol “Protocolo de transferencia segura de archivos”

ISO: International Organization for Standardization “Organización Internacional de Normalización”

PHP: Hypertext Preprocessor “Preprocesador de Hipertexto”

SQL: Structured Query Language “Lenguaje de Consulta Estructurado”

JS: JavaScript

CSS: Cascading Style Sheets “Hojas de estilo en cascada”

HTML: Hyper Text Markup Language “Lenguaje de marcado de hipertexto”

DSL: Domain-Specific Language “Lenguaje específico de dominio”

ADSL: Asymmetric Digital Subscriber Line “Línea de Abonado Digital Asimétrica”

DBMS: Database Management System “Sistema de Administración de Base de Datos”

GPIO: General Purpose Input Output “Salida de entrada de propósito general”

ETHERNET: Es la tecnología de red de área local (LAN) más ampliamente instalada.

VCN: Virtual Network Computing “Computación en red virtual”

VPN: Virtual Private Network “Red privada virtual”

DHCP: Dynamic Host Configuration Protocol “El protocolo de configuración de host dinámico”

DMZ: Demilitarized Zone “zona desmilitarizada (a veces denominada red perimetral)”

PROTOCOLO IEEE: Protocol Institute of Electrical and Electronics Engineers "Protocolo Instituto de Ingenieros Eléctricos y Electrónicos"

SOC: System on a chip "Sistema en un chip"

W3C: Consortium World Wide Web " El consorcio World Wide Web "

PLUGINS: es un componente de software que agrega una característica específica a un programa de computadora existente.

HDMI: High-Definition Multimedia Interface "Multimedia de interfaces en alta definición"

IDE: Integrated development environment "Entorno de desarrollo integrado"

BLE: Bluetooth Low Energy "Bluetooth de baja energía"

LAN: Local Area Network "Red de área Local"

WLAN: Wireless LAN "LAN inalámbrico"

PBC: print boart "Placa de circuito impreso"

DNC: do no conected "No conectar"

AP: Access point "Punto de acceso"

PWM: Pulse Width Modulation "Modulación de ancho de pulso"

SPA: single page applications "Aplicaciones de una sola página"

NPM: node package manager "Gestión de paquete node"

INTRODUCCION

En el mundo en el que vivimos la inseguridad se encuentra en todos lados, lo que hace que las personas busquen maneras de protegerse a ellos mismos y sus bienes; las organizaciones deben anticiparse a aquellas amenazas que podrían existir, tratando evitar riesgos.

A nivel nacional se registran al menos 7520 asaltos a locales comerciales. Por otro lado, en el mercado existen sistemas de seguridad que son muy costosos, difíciles de usar y a veces ineficientes, poniendo como ejemplo que un sistema de seguridad con control de acceso, cámaras y la parte domótica abordaría a un valor mayor a los $3000.

Por este motivo, en este trabajo de titulación, se ha decidido diseñar una plataforma de control de acceso y gestión remota para un local comercial que sea eficiente y versátil para el administrador abarcando los riesgos de seguridad que puedan existir, además de poder gestionar el local desde cualquier parte del mundo con vía internet.

El objetivo de este trabajo de titulación es mejorar tecnológicamente la seguridad de los locales comerciales, protegiéndolo de cualquier tipo de amenazas y a su vez realizar una introducción de un sinnúmero de proyectos que se podrían realizar con la Raspberry Pi y el Arduino.

El funcionamiento de la plataforma se basa en la gestión remota, cámara y dispositivos de control de acceso magnético y biométrico. Con la Raspberry pi, de manera remota con varios tipos de programaciones como lo son VueJS, NodeJS, MySQL y Python, y dispositivos electrónicos logran realizar un mayor control de seguridad y acceso dentro del local; ya que desde cualquier tipo de dispositivo electrónico móvil u ordenador se podrá observar los registros de acceso con sus respectivas imágenes.

Para el desarrollo y cumplimiento de lo mencionado, el presente proyecto técnico se ha fragmentado en cuatro capítulos, en los que trata los siguientes temas:

Capítulo I: Se narra una breve reseña histórica de la empresa, su visión, misión, los tipos de servicios que ofrece y las marcas que comercializa sus clientes, se describen los puntos importantes que tiene todo lo relacionado al planteamiento del problema, delimitaciones, se plantea el objetivo general y los objetivos específicos, metodología, beneficiarios del trabajo de titulación.

Capítulo II: Se realiza el marco teórico en el cual se desarrolla un enfoque más profundo sobre los temas principales del proyecto de titulación, dando énfasis a los elementos tanto en software como en hardware que son aplicados a través del diseño y el desarrollo de la plataforma.

Capítulo III: Se desarrolla una síntesis de manera técnica del proceso que se realiza para el desarrollo del proyecto de titulación, dando enfoque en las principales características del proceso, el análisis de la plataforma y estudios reales realizados en la implementación y prueba del proyecto adjuntos con otros detalles de plataformas similares

Capítulo IV: Se realiza un estudio de campo revisando y analizando de manera rigurosa los resultados acerca de las pruebas experimentales enfocadas en cada área importante de la plataforma tomando a consideración los inconvenientes, fracasos y metas alcanzadas a lo largo del proyecto.

CAPITULO 1

1 EL PROBLEMA

1.1 La Empresa

1.1.1 Reseña Histórica

En el 14 de Julio de 1994 fue fundada la empresa ecuatoriano conocida como CREDILIT (Créditos del Litoral), la cual se formó cuando el país pasaba por el mayor auge del comercio el cual gracias a su esfuerzo y trabajo fue uno de los mayores proveedores del estado, más concentrándose en Guayaquil y Manabí, Su fundador el Economista Luis Absalón Moncayo Sánchez, en un inicio comenzó con las ventas por catálogo, debido a la gran acogida de clientes, la empresa formalizo su local que hasta hoy es la matriz.

1.1.2 Misión

La empresa Credilit promueve en nuestros clientes el acceso a productos financieros, bienes y servicios, aportando crecimiento y calidad de vida.

1.1.3 Visión

La empresa Credilit, será en el 2020 una empresa rentable, responsable y sostenible para sus colaboradores, clientes, accionistas y gobierno; brindando acceso a través de bienes, servicios y productos financieros, en el marco de una promesa de valor, cubriendo las más exigentes demandas del mercado.

1.1.4 Objetivos de la Empresa

Ofrecer a sus clientes asesoramiento de calidad y confiabilidad en los diferentes productos que ofrece la empresa.

Brindar la mejor calidad en productos relacionado a la exigencia del cliente.

Entregar oportunamente los productos comercializados por la empresa en óptimas condiciones en el domicilio de nuestros clientes.

Garantizar una atención personalizada, respetuosa y óptima que permita una buena comunicación y efectividad en el cierre de negocios.

Mantener contacto con nuestros clientes y disminuir de manera sustancial las peticiones quejas reclamos y sugerencias en beneficio de un servicio de excelencia.

1.1.5 Función de la Empresa

La función principal de la empresa CREDILIT, es la de brindar servicios de asesoramiento de calidad en cuanto a sus productos, atención al cliente y de comercialización de artículos mobiliarios domésticos de múltiples tipos de marcas asegurando calidad y facilidades de pagos ajustes a su estabilidad económica.

1.1.6 Tipos de Servicios

La empresa CREDILIT realiza diferentes tipos de servicios como son:

- Ventas al por mayor y menor de ventas de productos de dormitorio
- Asesoramiento sobre productos de calidad
- Entrega a domicilio sobre sus productos
- Ofrecer atención al cliente informándoles de los mejores productos
- Ofrecer facilidades de pagos de acuerdo con la estabilidad económica del cliente

1.1.7 Presentación de las Marcas

CREDILIT comercializa y distribuye productos de calidad y marcas de prestigio nacional e internacional, como se aprecia en la figura 1.

Figura 1.1.7. Marcas que comercializa CREDILIT

Fuente: Los Autores

1.1.8 Información Legal

CREDILIT tiene como finalidad realizar todas las actividades de comercialización de artículos mobiliarios domésticos, cumpliendo con todas las obligaciones establecidas por las leyes gubernamentales, como:

- IESS
- SRI
- Ministerio de Trabajo
- Permisos Municipales
- Permisos del Cuerpo de Bomberos

1.2 Importancia y alcances

La implementación del presente trabajo de titulación es importante y conveniente, porque busca crear una plataforma (software y hardware) la cual sirve para gestionar

los horarios y permisos de acceso a la empresa “CREDILIT” así como tener en una base de datos un registro de accesos y las respectivas fotografías.

Este proyecto de titulación fue pensado como un proyecto de suma importancia ya que solventa los problemas de control de acceso y de seguridad que se tenía con el personal para la mejora del ambiente laboral, siendo sus principales beneficiados los empleadores, empleados y clientes de la empresa.

1.3 Delimitación

El desarrollo de este trabajo de titulación fue realizado en la empresa CREDILIT la cual se encuentra ubicada en la provincia del Guayas, Cantón Guayaquil, en Rumichaca y Manabí, numeración 2208.

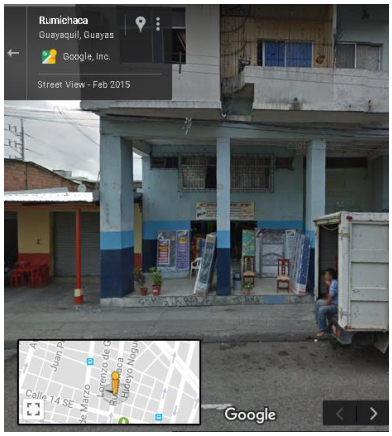


Figura 1.3.1 Vision frontal de la empresa Credilit

Fuente: Los Autores

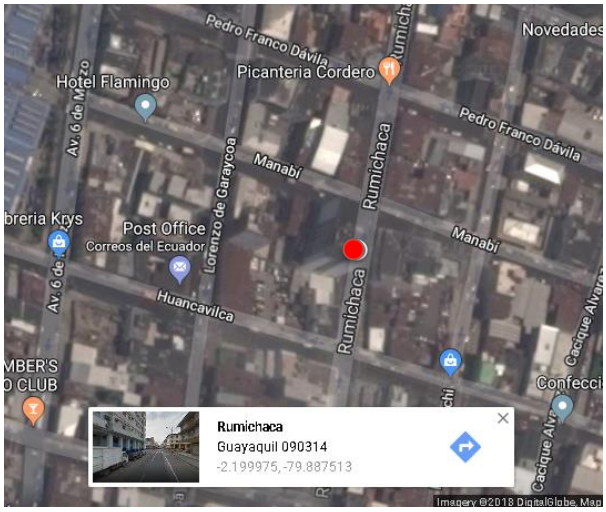


Figura 1.3.2 Ubicación geografica de la empresa Credilit

Fuente: Los Autores

1.4 Formulación del Problema

En la provincia del Guayas, en la zona central del Cantón Guayaquil, ubicado en Rumichaca 2208 y Manabí se encuentra las instalaciones de CREDILIT, que cuenta con una bodega en la cual se almacenan los productos de venta entre otros artículos varios, cada uno de los empleados del local pueden ingresar libremente a la bodega sin ningún tipo de control.

Debido a este inexistente control de acceso a la bodega se han reportado perdidas de artículos de la bodega y no se ha podido determinar los responsables, esto genera pérdidas económicas para la empresa además de un ambiente de desconfianza lo cual perjudica a la empresa.

1.5 Delimitación del Problema

El Proyecto se desarrolló para la empresa CREDILIT el cual permite gestionar permisos de acceso a la bodega de la empresa desde una aplicación web que es ser visible desde un navegador en cualquier dispositivo con acceso a internet.

El administrador puede crear, editar y eliminar las tarjetas y usuarios desde la aplicación web y asignarles las tarjetas a los usuarios para que puedan acceder a la bodega de CREDILIT, el sistema también cuenta con un control de acceso biométrico, que mediante la huella dactilar del empleado se registra la hora de acceso a la bodega, adicional se le incorpora un sistema de mensajes de aviso.

Se instaló una cámara dentro de la bodega para que cuando el usuario ingrese a la bodega (con su huella dactilar o con su tarjeta de acceso asignada) la cámara toma varias fotos durante pequeños intervalos de tiempo para verificar quien o quienes son los que ingresan a la bodega, de esta forma podemos verificar el listado de accesos con las fotografías tomadas y una cámara fuera de la bodega el cual toma fotos cuando un sujeto que no se encuentra en a base de datos intenta entrar.

1.6 Objetivos

1.6.1 Objetivo General

Diseñar e implementar una plataforma de control de acceso mediante tarjetas de identificación y sensor biométrico basado en Arduino pro mini, Raspberry Pi Zero, NodeJS, VueJS y MySQL.

1.6.2 Objetivos Específicos

- Desarrollar una aplicación en VueJS y NodeJS para gestión de usuarios y permisos de acceso de la bodega.
- Diseñar un sistema de control de acceso con tarjetas identificadoras basado en Arduino pro mini.
- Diseñar un sistema de control de acceso biométrico basado en Arduino pro mini.
- Crear una base de datos MySQL en la Raspberry con el registro de entradas y salidas de la bodega para su seguridad.
- Desarrollar un software web en la Raspberry Pi para la gestión de las tarjetas y huellas dactilares.
- Diseñar un sistema de supervisión remota para el establecimiento mediante conexión de internet desde cualquier parte del mundo.
- Implementar protocolos de seguridad por medio de una clave de acceso a las bases de datos para que sea gestionada por personas autorizadas.

1.7 Metodología

1.7.1 Métodos

La metodología es el procedimiento y la técnica usada para resolver un problema, para el desarrollo del proyecto de titulación se llegó a la conclusión del uso de los siguientes métodos:

Método Deductivo

El método deductivo consiste en hacer uso de todos los conocimientos adquiridos a lo largo de nuestra etapa universitaria.

Se hace uso de esta metodología durante el diseño e implementación de la parte electrónica de la plataforma de acceso, ya que se necesitarán de los conocimientos tales como:

- Programación de microcontroladores.
- Electrónica digital.
- Electrónica analógica.

Método Científico

El método científico consiste en la investigación y adquisición de nuevos conocimientos y destrezas para poder hacer frente a los problemas.

Se necesita de esta metodología durante la etapa de desarrollo del software de administración ya que debemos adquirir nuevos conocimientos tales como:

- Diseño de aplicaciones web.
- Diseño y administración de bases de datos.

Método Experimental

El método experimental consiste en el control deliberado y el registro de las variables que afectan al objeto de estudio.

Se hace uso de esta metodología durante las programaciones y la forma del empleo de los equipos que se realizarán a lo largo de la elaboración del proyecto y está enfocado en:

- Programación en Raspberry.
- Programación en Arduino.

1.7.2 Técnicas

Las técnicas son el conjunto de acciones usadas para lograr un objetivo, las técnicas que se aplicaron en este proyecto fueron:

Técnica de Investigación

La técnica de investigación consiste en la búsqueda de información necesaria respecto a uno o varios temas con el fin de poder obtener los suficientes conocimientos para la toma de decisiones y alcanzar el resultado deseado.

Técnica de Experimentación

La técnica de experimentación consiste en la prueba e intento de diversas hipótesis para probar cual es la que más se acerca al resultado deseado

1.8 Descripción de la propuesta

Diseñar e implementar una plataforma de control de acceso magnético/biométrico y gestión remota que pueda gestionar las entradas y salidas de la bodega, donde se instaló un dispositivo basado en Arduino Pro Mini el cual tiene conectado un lector de tarjetas RFID y un sensor biométrico, de estos dos sensores se lee información del usuario para enviarla a la Raspberry Pi 3.

La Raspberry Pi 3 recibe los datos con la información del usuario y mediante un script programado en NodeJS, consulta la base de datos MySQL para determinar si el usuario puede o no acceder a la bodega.

Si el usuario puede ingresar a la bodega se le envía un "ok" al Arduino para que habilite el acceso de la cerradura electromagnética y se registra en la base de datos al usuario que ingresó, si el usuario no está autorizado para ingresar a la bodega, la puerta no se abrirá y además se enviara un mensaje GSM indicando el intento no autorizado y un mensaje email con el mismo texto y una foto tomada en el instante del intento de intrusión.

En el momento que se identifica un intento de ingreso no autorizado la foto es tomada a través de una script en Python por medio de una cámara conectada a la Raspberry pi 3.

En el momento que se abre la puerta, la Raspberry Pi Zero ejecuta un script en Python para capturar fotos de los que acceden a la bodega para poder tener un registro fotográfico de los accesos y salidas de la misma.

La gestión de estos usuarios la realiza el administrador del sistema desde un navegador mediante una aplicación web escrita en VueJS, esta aplicación permite ver, crear, editar y eliminar los usuarios que tienen acceso a la bodega, así como ver el historial de accesos y fotografías tomadas.

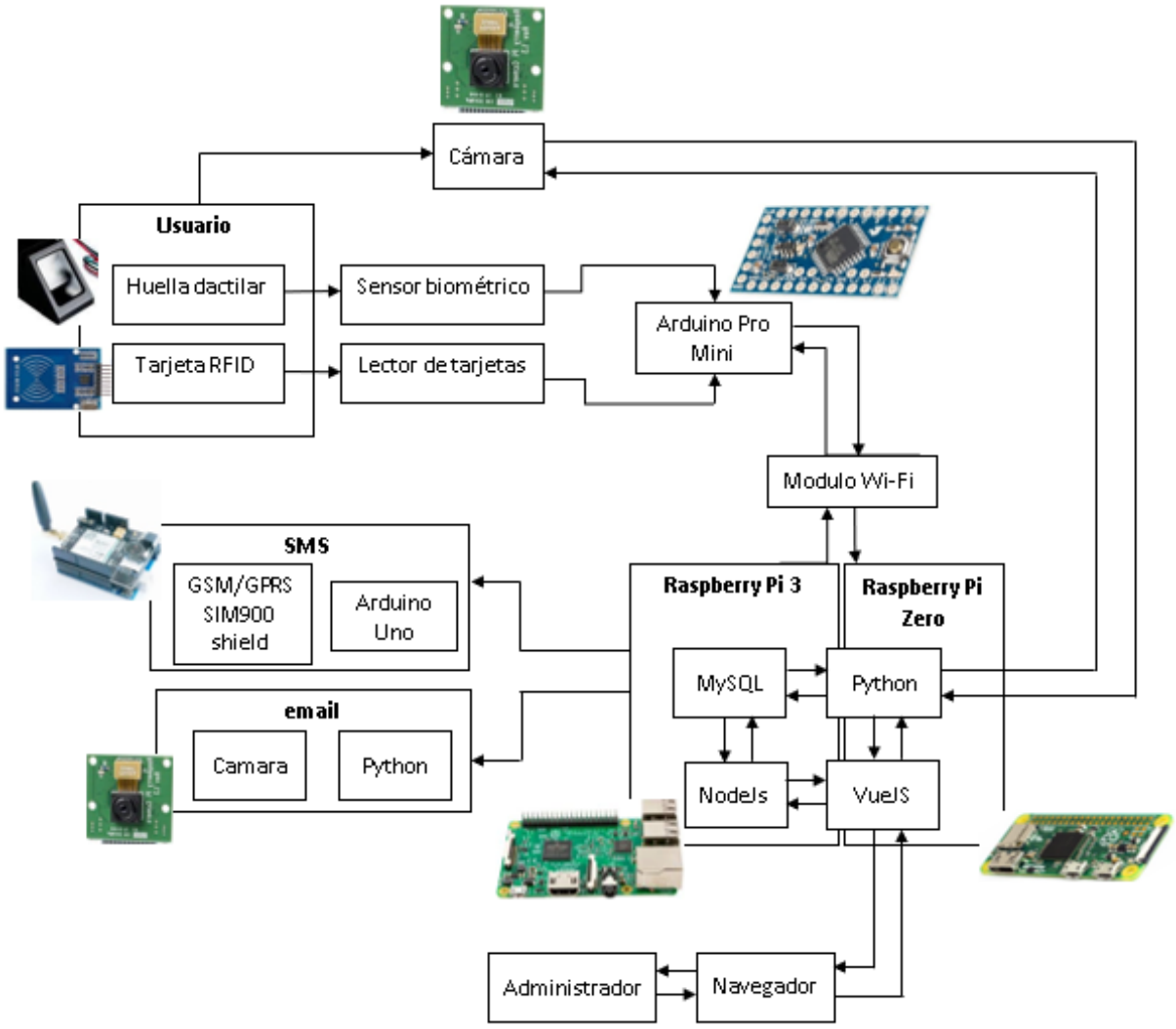


Fig. 1.8. Diagrama de bloques del sistema.

Fuente: Los Autores

1.9 Beneficiarios

Los Beneficiarios directos de la plataforma de control de acceso y gestión remota son las personas que trabajan dentro del local comercial, las personas encargadas del local y el dueño.

1.10 Impacto

Al realizar una plataforma de control de acceso y gestión remota con los distintos dispositivos mostrados y al interconectarlos y entrelazarlos con el miniordenador

Raspberry Pi se incrementó la seguridad de la empresa debido a su tecnología, además de su facilidad de uso para la interacción con el usuario.

Por otra parte, tener una confortable acogida al uso de nuevas tecnologías para la contribución y desarrollo de la sociedad, así como su uso en áreas de seguridad y de control.

CAPITULO 2

2 MARCO TEORICO

2.1 Raspberry Pi

La Raspberry Pi es una minicomputadora formada por una placa de tamaño de una tarjeta de crédito, con suficiente potencia para realizar funciones como juegos, administrar un servidor de archivos, varias páginas webs dinámicas, una base de datos, o un punto de acceso inalámbrico.

La arquitectura de la Raspberry Pi está basada en un sistema de Broadcom System on a chip (SoC) que incluye un potente procesador de gráficos (GPU), fue desarrollado en el Reino Unido por la Fundación Raspberry PI.

La Raspberry Pi fue diseñada con el objetivo de promover la enseñanza de la informática básica en las escuelas, el diseño del miniordenador incluye ocho puertos de entradas/salidas de uso general (GPIO), bus de datos I2C y SPI además de dos puertos USB, un adaptador para la cámara diseñada especialmente para cada versión, un puerto ethernet, conexión HDMI, una ranura para agregar una tarjeta de memoria SD y 512 MB de memoria.

Con sus conexiones periféricas estándar, la Raspberry Pi posee el potencial de ser más que una herramienta para la educación o un juguete.

El diseño de la Raspberry Pi incluye:

- SoC Broadcom BCM2835.
- Un procesador gráfico de Video Core IV.
- Buses USB 2.0.
- Una salida analógica de audio estéreo por Jack de 3.5 mm.
- Salida digital de video y audio HDMI.
- Salida analógica de video RCA.
- Pines de entrada y salida de propósito general.
- Puerto de alimentación microUSB.
- Interfaz de lector de tarjetas SD.

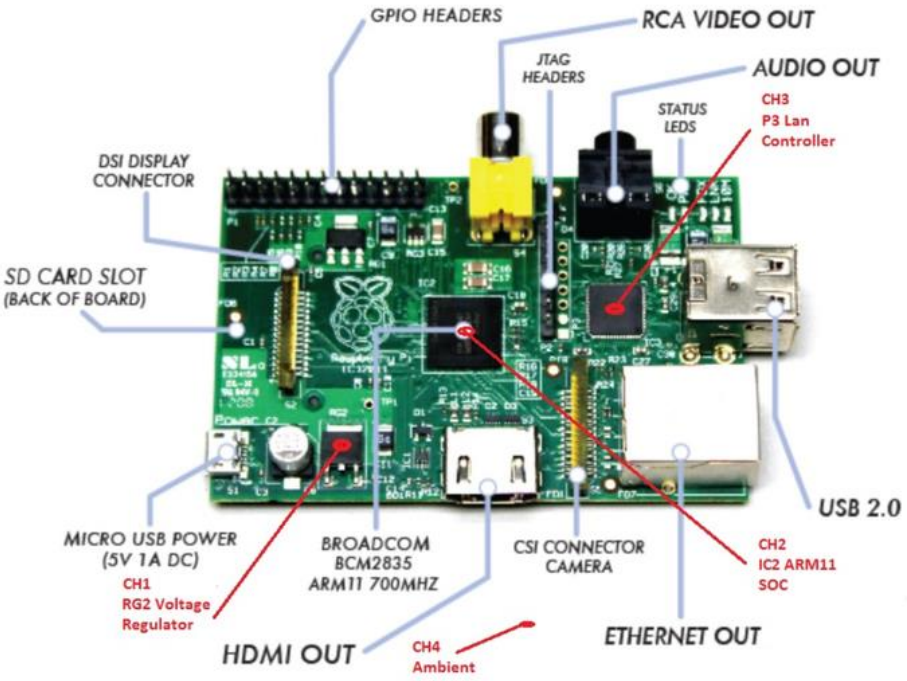


Fig. 2.1. Tarjeta Raspberry pi.

Fuente: http://lhermie.net/raspberry-pi-cheat-sheet/

2.1.1 Raspberry Pi 3

Es un miniordenador con un procesador BCM2837 siendo el primero en ofrecer soporte de 64 bits, integrado un soporte inalámbrico de doble banda con un radio capaz de conectarse a redes Wifi de 2,4 GHz y dispositivos Bluetooth, además de un encabezado de 40 pines GPIO.

Figura 2.1.1. Raspberry Pi 3

Fuente: raspberrypi.org

2.1.2 Raspberry Pi Zero

Raspberry PI Zero es un miniordenador con la ventaja de ser más pequeño y de un valor económico, de placa única (SBC), posee 40 puertos GPIO de uso general además de conectores periféricos y todos los componentes necesarios para su funcionamiento.

Las características más relevantes son:

- Conexión WIFI LAN inalámbrica 802.11
- Conexion Bluetooth 4.1
- Puertos Mini HDMI y micro USB
- Cabezal de 40 pines compatible con HAT
- Puerto para cámara Raspicam

Figura 2.1.2. Raspberry Pi Zero

Fuente: raspberrypi.org

2.2 Editor de texto

Es un software desarrollado con el objetivo de escribir código fuente de aplicaciones en general en diversos lenguajes de programación. Generalmente los editores de código pueden trabajan con varios lenguajes a la vez y son capaces de abrir varios archivos en el mismo tiempo, los editores de texto actuales vienen desarrollados con herramientas claves para resaltar su sintaxis y ofrecer ayudas contextuales en el desarrollo, la escribir o visualización del código de programación de las aplicaciones.

2.2.1 Visual Studio Code

Visual Studio Code es una herramienta que conforma la función de un editor de código fuente ligera pero potente con lo que los desarrolladores necesitan para el ciclo de edición, construcción y depuración, además de ser ejecutable desde el escritorio y está disponible para diversos sistemas operativos. Viene con soporte integrado para JavaScript, TypeScript y Node.js y provee extensiones para otros lenguajes (como C ++, C #, Java, Python, PHP) y tiempos de ejecución.

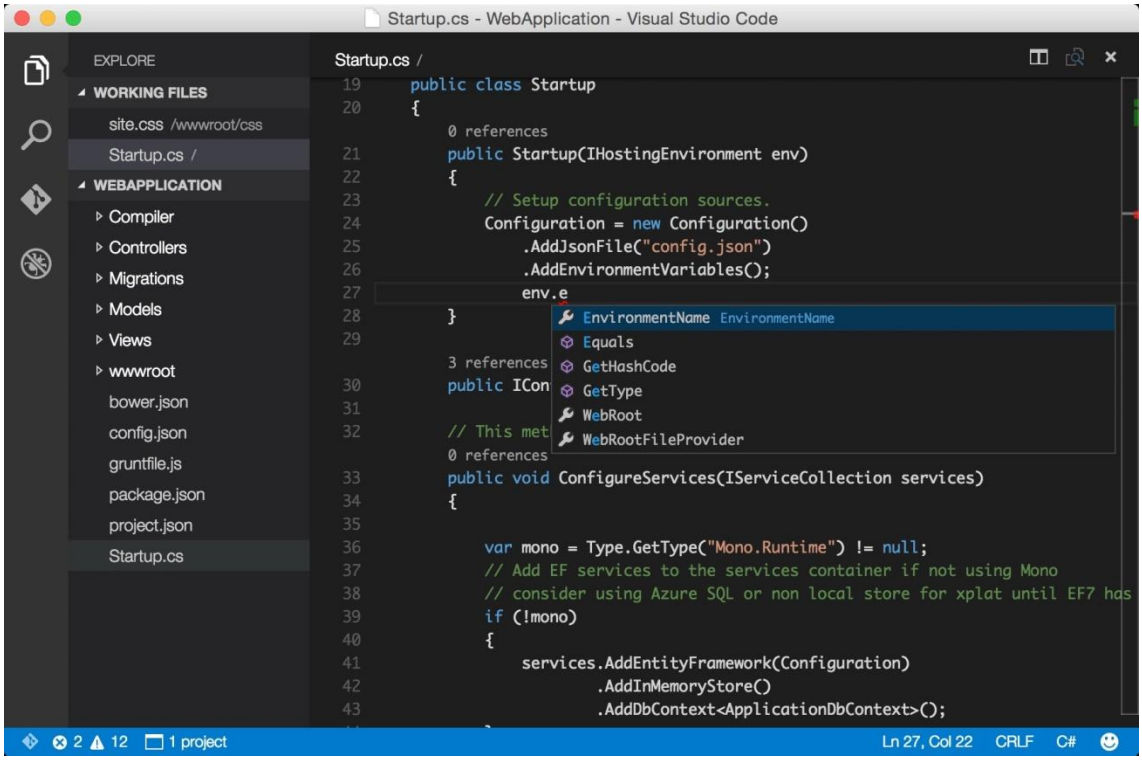


Fig. 2.2.1. Visual Studio Code.

Fuente: https://alternativeto.net/software/visual-studio-code/

2.2.2 Arduino IDE

Arduino es un tablero diseñado para la elaboración de proyectos de electrónica de código libre basada en hardware y software adaptable de utilizar.

2.2.2.1 Plataforma de Arduino

Arduino se compone de hardware y software. La plataforma de Arduino está compuesta por un software de desarrollo de código libre, la cual permite al usuario la facilidad de crear líneas de programación, compilarlo y subirlo a la placa Arduino, este software se encuentra habilitada para ser ejecutada en varios tipos de sistemas operativos como Windows, Mac OS y Linux. El software está basado en Java y en Processing, además cuenta con un editor de texto, área de mensajes y barra de herramientas la cual te permite subir el código creado a varias versiones de placas Arduino existentes y una serie de herramientas adicionales.

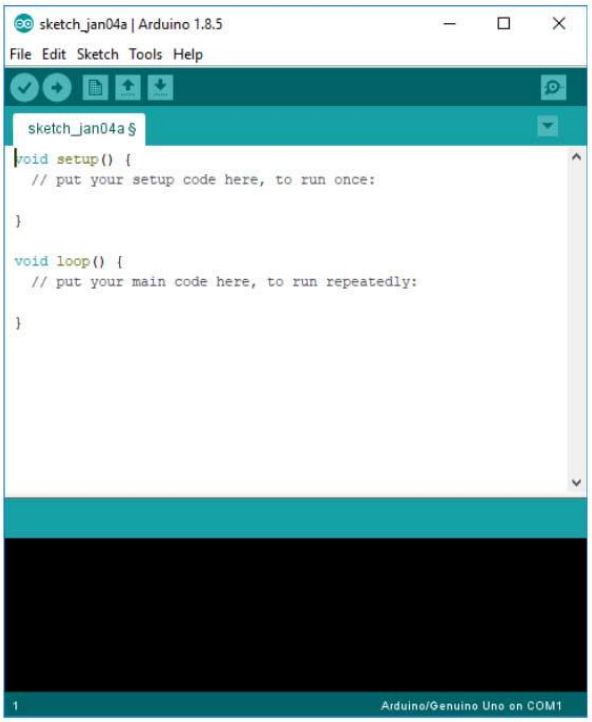


Figura 2.2.2.1. Arduino IDE

Fuente: arduino.cc

2.2.2.2 Bibliotecas de Arduino

Dentro del software de programación de Arduino existen bibliotecas de códigos y son usadas para un desarrollo más fácil y rápido de códigos de programación, estas bibliotecas funcionan de trabajar con hardware adicionales como sensores o manipular datos obtenidos por diferentes equipos adicionales.

Las bibliotecas que pueden ser usados en Arduino son desarrollados por empresas o individuos que aportan al entorno experimental electrónico, al importar una

biblioteca dentro de la programación del prototipo se la realiza por medio de las declaraciones #include agregando el nombre de la biblioteca a utilizar, una programación en Arduino tiene la capacidad de poder agregar más de una biblioteca para poder manejar diversos tipos de datos. Debido al adjunto de la biblioteca en el código se aumenta el espacio que ocupa.

2.3 Sistema Operativo

El miniordenador Raspberry Pi se diseñó para poder adoptar varios sistemas operativos, uno de los más populares GNU/Linux. Linux a diferencia de otros sistemas operativos es de código abierto la cual tiene la facilidad de realizar cambios personalizados y volverlos adaptable al usuario, debido a su característica de poder ser configurable ha permitido el desarrollo de una versión ejecutable sobre a Raspberry Pi.

Existen varias distribuciones del sistema operativo Linux que pueden ser adoptadas al chip BCM2835, propia de la Raspberry Pi, las cuales están Debian, Arch Linux y Fedira. Cada versión diferente de Linux posee la característica de atender distintas necesidades y basado en diferentes capacidades del ordenador en donde son ejecutadas y todas las versiones de Linux comparten la propiedad de ser de código libre además de poseer una compatibilidad con las demás versiones.

2.3.1 Raspbian

Raspbian es el sistema operativo más utilizado en el desarrollo de la Raspberry Pi. Raspbian es una versión no oficial basado en Debian ajustado para correr un código de ¨flotación dura¨ optimizado para ser ejecutable para Raspberry Pi.

Raspbian ofrece capacidades más altas que un simple sistema operativo, ya que viene con más de 35000 paquetes adicionales que funcionan con el propósito de mejorar el rendimiento del miniordenador, además de tener la característica de ser recompilado en un formato sencillo para una instalación rápida y fácil.

El sistema operativo Raspbian fue desarrollado por grupos aficionados a Debian, sistemas operativos, desarrollo de software y hardware raspberry con fines educativos, Raspbian no posee ninguna afiliación con la fundación Raspberry Pi.

2.4 Servidor Web

Un servidor web o también llamado HTTP es un programa diseñado para procesar un aplicativo web o página web dinámica con todos sus elementos, realiza comunicaciones de entrada y salida con el usuario o cliente generando o cediendo una respuesta en cualquier tipo de lenguaje, para la comunicación se suele usar protocolos de internet, generalmente HTTP.

Los servidores web se almacén dentro de un ordenador que tiene conexión directa con una salida a Internet, la cual está abierta a peticiones de cliente-servidor devolviendo como respuesta servidor-cliente un aplicativo web basado en HTML.

2.4.1 Apache Server

Un servidor Apache es un tipo de servidor web o servidor HTTP de código libre, con la capacidad de poder ser instalada y ejecutada en diversos tipos diferentes de sistemas operativos, además de ser muy complejo que destaca por ser seguro y posee un gran rendimiento.

El servidor Apache HTTP puede almacenar tanto páginas web estáticas basado en HTML como aplicativos web o web dinámicas, tiene la capacidad de poder adicionar módulos especiales para utilizar diversos tipos de lenguajes como PHP.

Apache Software Foundation es el encargado del desarrollo del servidor Apache como un proyecto HTTP server.

Ventajas

- Software de código abierto para instalación y/o configuración.
- El servidor web Apache HTTP es de coste gratuito.
- Funcional y Soporte por parte de individuos que aportan con mejoras que son disponibles.
- Puede ser instalados y ejecutados en muchos sistemas operativos informáticos.
- Posee un rendimiento alto para la gestión de más de un millón de visitas por día.
- Posee un soporte de seguridad SSL y TLS.

Fig. 2.4.1. APACHE HTTP SERVER.

Fuente: Acción Network Global SL

2.5 Software Python

2.5.1 Introducción

Python es un lenguaje de programación basado en una sintaxis clara y sencilla con el fin de ser un lenguaje de alto nivel, además de ser comendado y acogido como lenguaje principal de programación que utiliza la Raspberry Pi para el desarrollo de proyectos electrónicos, tiene la capacidad de ser un lenguaje de programación flexible, dinámico y su facilidad de aprendizaje.

Python tiene la propiedad de poder ser instalable y ejecutable en cualquier tipo de sistema operativo y es un lenguaje orientado a objetos. Python es un lenguaje de programación que consta de muchos de librerías para poder desarrollar cualquier tipo de proyecto, además de ser relacionados como un lenguaje robusto y potente.

El sistema operativo Raspbian que generalmente se corre en los dispositivos Raspberry Pi tienen la plataforma Python instalado como aplicativo por defecto.

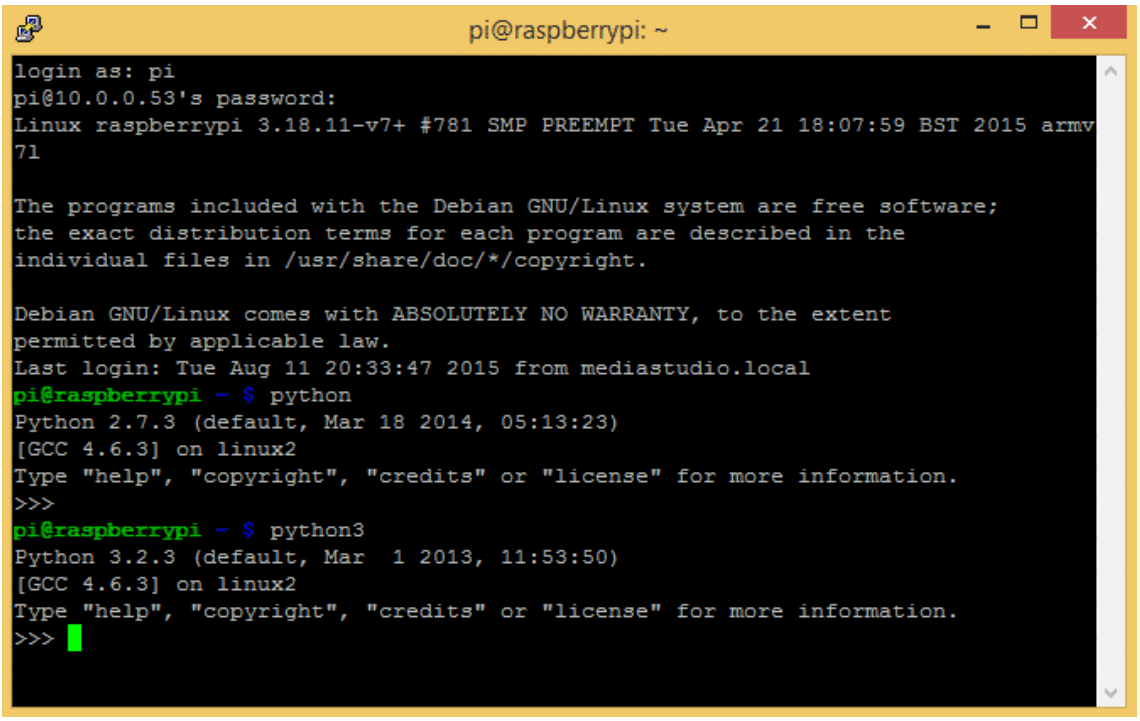


Fig. 2.5.1. Python REPL.

Fuente: http://www.circuitbasics.com

2.5.2 Características

- Lenguaje de programación creado para realizar todo tipo de programas.
- Lenguaje adaptable a muchos sistemas informáticos.
- Interpretado, no se necesita una compilación compleja del programa para poder ejecutarlo, su compilación es transparente para el programador.
- Interactivo al producir un resultado visible al momento de ejecutar la programación.
- Orientado a Objetos.
- Posee la propiedad de poder realizar funciones además de tener muchas librerías que se pueden importar para tratar temas específicos de una programación.
- Posee una sintaxis sencilla y clara basado en la tabulación como margen de programación.

2.6 Estándares Web

2.6.1 Introducción

Los estándares web son lenguajes web, protocolos y tecnologías que fueron diseñadas y son utilizadas con el objetivo de sacar el máximo potencial de la Web;

emplean tecnologías Inter operativas e internacionales con la finalidad de ser compatibles entre ellas y permitir un acceso multiplataforma.

La comunidad internacional World Wide Web consorciem mejor conocida como W3C es la encargada del funcionamiento y rendimiento de la web, realizando la interoperabilidad Web.

Los estándares web son conjuntos de reglas reglas de cumplimiento para servicios y productos, que funcionan como base para que distintos hardware o software puedan hacer uso de ellas estableciendo una gran compatibilidad.

El W3C es la encargada de desarrollar estándares que sirven como referencia para la creación de una web habilitarle con un rendimiento alto y una buena interoperabilidad, en la que se puedan desarrollar páginas web o aplicativos webs cada vez más complejas.

Los estándares Web que son los más conocidos y utilizados son:

- HTML: es un lenguaje de marcado que se utiliza para la creación de paginas web y aplicativos webs, es usada para dar estructura por medio de etiquetas propias del lenguaje.
- CSS: es un lenguaje de reglas en cascada que se utiliza para brindar estilo al contenido HTML para las páginas web y aplicativos webs para desarrollar un entorno amigable con la vista.
- JavaScript: es un lenguaje de programación que se utiliza para desarrollar páginas web dinámicas o también llamadas aplicativos web, permitiendo manejar archivos multimedia, desarrollar imágenes animadas y muchas más aplicaciones adicionales.

Fig. 2.6.1. Estándares W3C

Fuente: disenowebakus.net

2.7 JavaScript

2.7.1 Introducción

JavaScript es un lenguaje de programación que desarrolla páginas web dinámicas o también llamados aplicativos web, que son capaces de interactuar en el lado del cliente para poder realizar funciones dentro de una página web.

Entre las funciones más reconocidas de los aplicativos web es la de presentar efectos que aparecen dependiendo de la acción que realices, permite animaciones y desarrollo de una acción a través de botones y ventanas emergentes o de aviso.

JavaScript es un lenguaje de código interpretado por lo que no existe necesidad de realizar la compilación de código para poder ejecutarlos, el código desarrollado por JavaScript se puede ejecutar a través de cualquier navegador sin procesos intermedios.

El lenguaje de programación JavaScript no tiene relación con Java, que es un lenguaje de programación totalmente diferente. JavaScript es reconocido como uno de los lenguajes más utilizables por ser practico y disponible dentro de cualquier navegador web.

Existen dos tipos de JavaScript que son usados: el primero es el JavaScript que se ejecute en el lado del cliente y denominado como navegador JavaScript, el segundo en el lado del servidor, frontend, estableciendo una mayor interactividad con la web, además de disponer de librerías y frameworkbasados en JavaScript como Vuejs, Nodejs, jQuery, angular, etc.

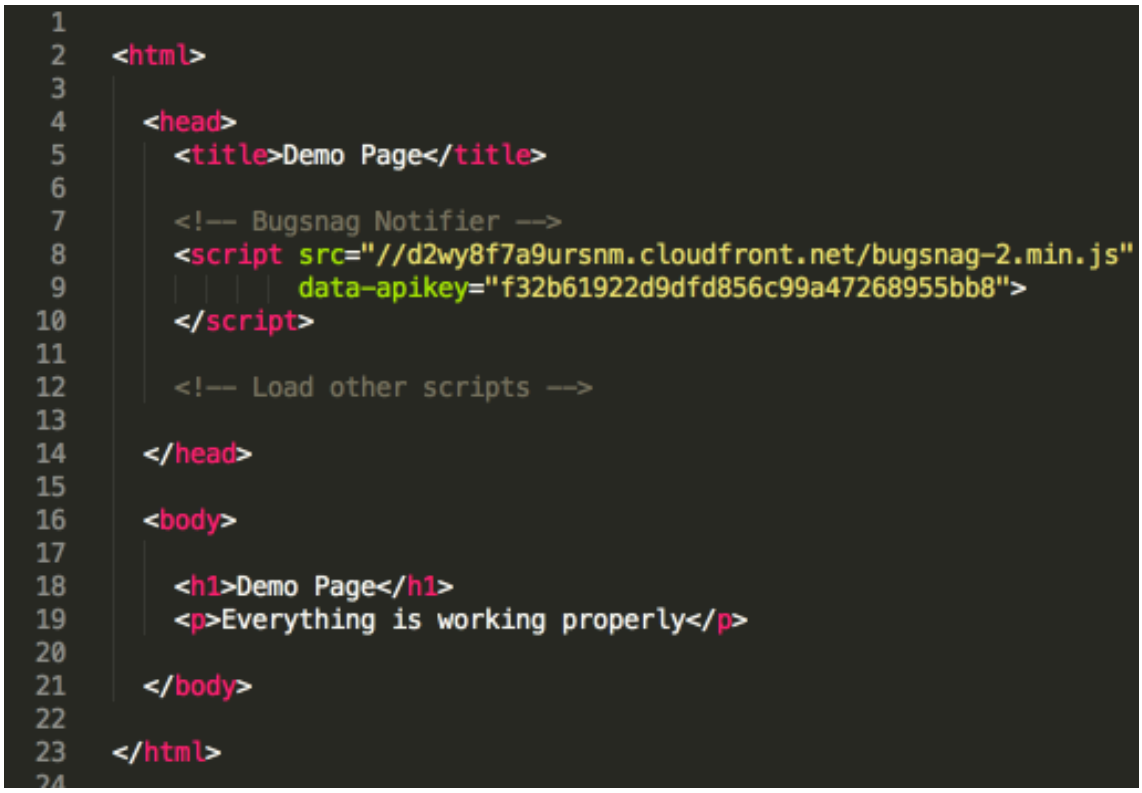


Fig. 2.7.1. Estructura de un Aplicativo Web.

Fuente: www.bugsnag.com

2.7.2 Características

- Es Liviano.
- Capacidad de poder ser ejecutable en diferentes sistemas operativos.
- Es Imperativo y estructurados través de un conjunto de instrucciones por código indicando al computador qué acción realizar.
- Prototipado para el uso de herencia.
- Lenguaje orientado a objetos y eventos.
- Puede ser ejecutado en el lado del cliente y del servidor.
- Lenguaje de programación que permite crear aplicaciones complejas y dinámicas e incluso capaz de ejecutarse en un servidor Web como con nodejs.

2.8 Software VueJS

2.8.1 Introducción

Vuejs es un marco de trabajo con la caracterisitcadeser progresivo en la construcción de interfaces graficas de usuario. A diferencia de varios marcos de trabajo, Vuejs tiene la capacidad de poder ser adoptable incrementalmente. La biblioteca principal

esta principalmente diseñada para la capa de visualización, además de poder ser adicionada a programaciones o proyectos ya existentes, vuejs es perfectamente ser capaz de impulsar complejas aplicaciones y poder agregar varias herramientas y bibliotecas que sirven para la mejora de la programación.

Vue.js, es un framework diseñada principalmente hacia la vista, y tiene la propiedad de poder ser implementado en el HTML de aplicativos web ya existentes sin la necesidad de realizar cambios drásticos, se puede utilizar para desarrollar pequeños widgets interactivos y desarrollar “Single Page Applications” (SPA) complejas.

```
<div class="container">
  <div class="row">
    <div class="col-md-6">
      <h3>Mi formulario</h3>

      <form id="mi-formulario" action="controlador.php" method="post" target="resultado">
        <div class="form-group">
          <input class="form-control" type="text" name="usuario" placeholder="usuario">
        </div>
        <div class="form-group">
          <input class="form-control" type="text" name="email" placeholder="email">
        </div>
        <div class="form-group">
          <input class="form-control" type="password" name="password" placeholder="contraseña">
        </div>
        <div class="form-group">
          <button class="btn btn-success btn-block" type="submit">OBTENER FORMULARIO CIFRADO</button>
        </div>
      </form>

    </div>
    <div class="col-md-6">
      <h3>Enigma devuelve</h3>
      <iframe id="resultado" name="resultado" width="100%" height="180px"></iframe>
    </div>
  </div>
</div>
```

Fig. 2.8.1.1. Programación Vue.js.

Fuente: http://enigma.itp.edu.co/docs/ejemplo-simple.html

Existen plugin muy importantes como:

- VUE-ROUTER: Para la gestión de rutas.
- Conectar a servicios externos: Como vue-axios o vuefire para conectarnos a una base de datos.
- TEST: Se pueden hacer test unitarios y funcionales.
- VUEX: Es una implementación de la arquitectura de aplicación FLUX basada en la arquitectura ELM y además creado un poco en REDUX, es una implementación para gestionar el flujo de datos en nuestra aplicación.

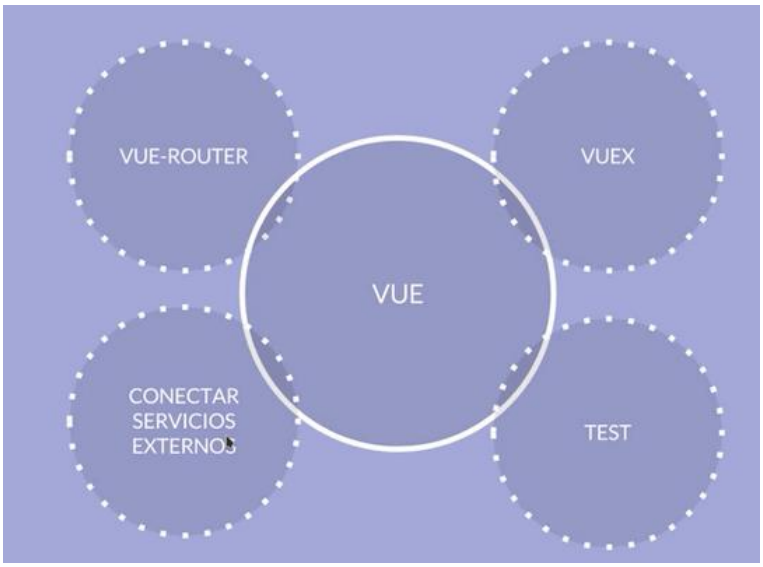


Fig. 2.8.1.2. Plugins Vue.js.

Fuente: https://styde.net/introduccion-a-vue-js/

2.8.2 Características

- Tiene la capacidad de poder ser añadido en proyectos ya existentes
- Escalable, se puede incorporar cosas a medida que se los necesite
- Open source, Accessible.
- Está enfocado en la vista.
- Versátil, se escala a través de plugin.
- Optimizado, su Core ocupa 74KB, bastante liviano.

2.9 Software NodeJS

2.9.1 Introducción

Node.js es una plataforma de código abierto que le permite crear aplicaciones de red rápida y escalable con JavaScript. Node.js está construido sobre V8, una moderna máquina virtual de JavaScript que funciona con el navegador web Chrome de Google. Node.js utiliza un modelo de E / S sin bloqueo controlado por eventos que lo hace ligero y eficiente. Node.js puede manejar múltiples conexiones de red concurrentes con poca sobrecarga, lo que lo hace ideal para aplicaciones de tiempo real con uso intensivo de datos. Con Node.js, puede crear muchos tipos de aplicaciones en red.

Node.js tiene la propiedad de ser de multiplataforma lo que te permite instalarlo en diferentes sistemas operativos.

Al usar la transmisión para manejar los datos, puede usar Node.js para controlar las secuencias entrantes y salientes y permitir que su servicio sobreviva. Al usar Node Package Manager (NPM), puede instalar, administrar y usar fácilmente cualquiera de los varios módulos contenidos en un repositorio grande y en crecimiento. NPM también le permite administrar los módulos de los que depende su aplicación de manera aislada, lo que permite que diferentes aplicaciones instaladas en la misma máquina dependan de diferentes versiones del mismo módulo sin originar un conflicto.

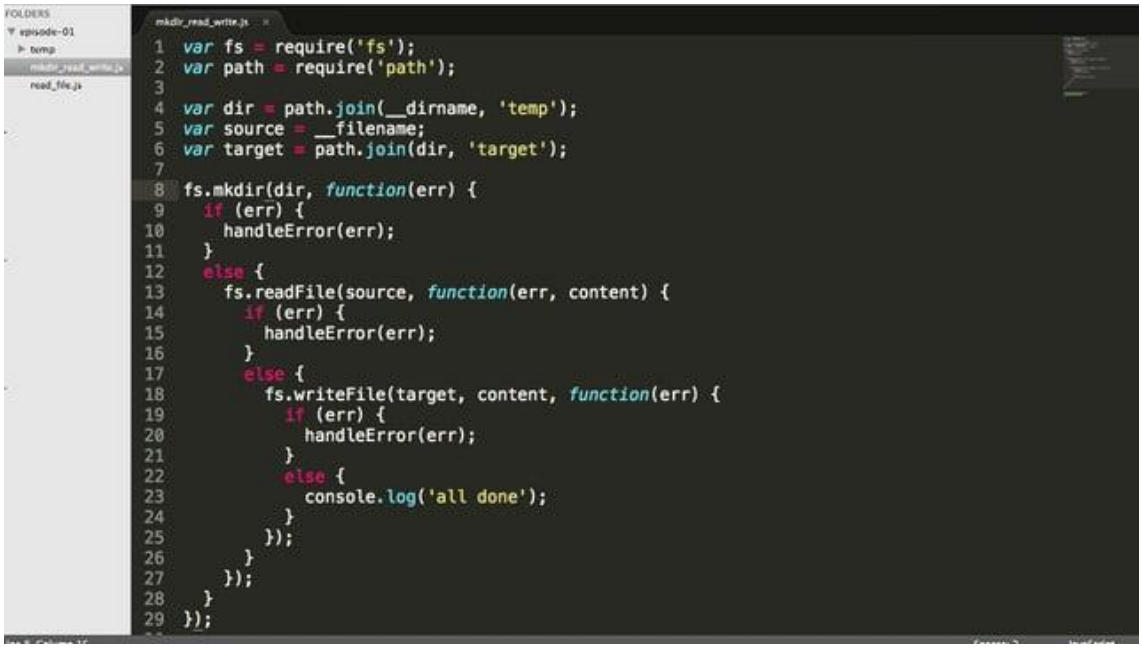


Fig. 2.9.1. Programación Node.js.

Fuente: http://nodetuts.com/series/asynchronous-programming.html

2.9.2 Características

- Basado en JavaScript en el lado del servidor
- Basado en la máquina virtual de Chrome V8
- Multiplataforma
- Es una marca registrada y es open source
- Asíncrono y orientado a eventos
- Ideal para aplicaciones que consumen datos en tiempo real y que se ejecutan a través de dispositivos distribuidos
- Muchas empresas lo usan

2.10 Software MySQL

2.10.1 Introducción

MySQL es una plataforma de gestión de bases de datos (Database Management System, DBMS) para bases de datos relacionales. MySQL es un aplicativo desarrollado para la administración de archivos.

MySQL es la opción perfecta para proporcionar datos a través de Internet debido a su capacidad para manejar cargas pesadas y sus medidas de seguridad avanzadas.

Existen varios tipos de bases de datos, como un ahivo de almacenamiento hasta sistemas orientados a objetos.

MySQL tiene la propiedad de ser de código abierto, MySQL es la construcción de base de datos que permite que PHP y Apache trabajen juntos para acceder y mostrar datos en un formato legible para un navegador. Es un servidor de lenguaje de consulta estructurado diseñado para cargas pesadas y procesamiento de consultas complejas. Como un sistema de base de datos relacional, MySQL permite que muchas tablas diferentes se unan para lograr la máxima eficiencia y velocidad.

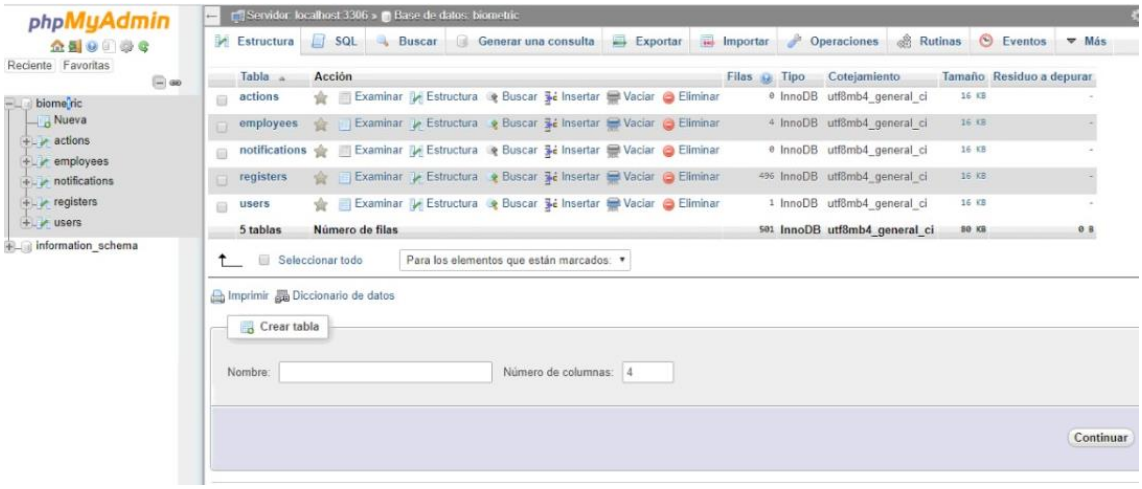


Fig. 2.10.1. Programación MySQL.

Fuente: Autor

2.10.2 Características

- Asociados con aplicativos web y la publicación en línea
- Es un administrador de base de datos en multihilo y multiusuario, lo que permite ser utilizado por múltiples usuarios a la vez y realizar varias consultas al mismo tiempo.
- Puede ser ejecutada en diferentes sistemas operativos.
- Es una marca registrada y es open source
- Ofrece aplicaciones que permiten acceder a las sentencias del gestor de base de datos

2.11 Conectores GPIO

GPIO, en un sistema compuesto de pines de entrada y salida de propósito general, para ser usado en cualquier proyecto, cada pin puede ser usado como pin de entrada o pin de salida para usos múltiples, además de estar presentes en todos los módulos de la Raspberry Pi.

Los GPIO es la interfaz de comunicación entre el miniordenador y el mundo exterior por medio de sensores y actuadores.

La programación que se usa para poder utilizar las interfaces de GPIO de la Raspberry Pi tienen la ventaja de ser flexible, ya que se lo puede desarrollar a través de diferentes lenguajes de programación entre ellos el más conocido Python, o desde la consola utilizando sencillos scripts y comandos.

Todos los pines GPIO son de tipo “unbuffered”, lo que significa que no disponen de buffers de protección, por lo que se debe tener cuidado con las magnitudes de voltaje o intensidad al momento de desarrollo de un proyecto.

La cabecera GPIO de la rasoberry este compuesto con diferentes pines que compone una función diferente como:

- **Pines de alimentación:** pines de 5v, 3v3 (limitados a 50mA) y tierra.
- **DNC (Do Not Connect):** son pines que no tienen función, en las placas actuales han sido marcados como GND.
- **GPIO normales:** son conexiones configurables que se pueden programar.
- **GPIO especiales:** compuestos por pines de interfaz UART.

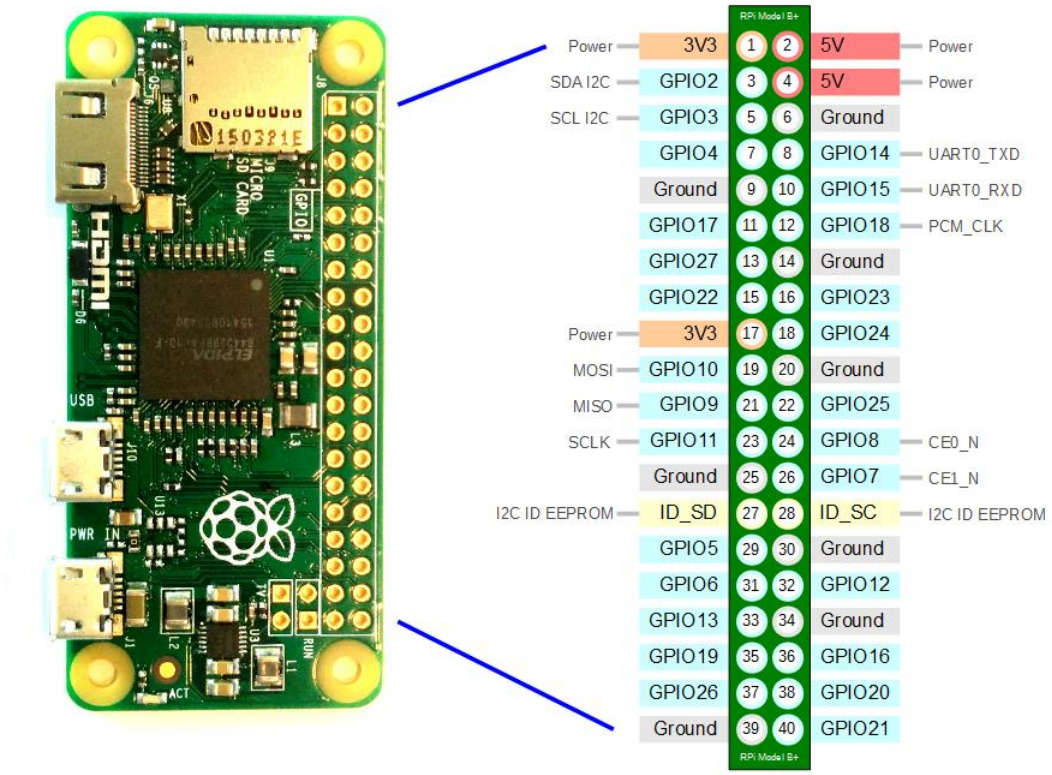


Fig. 2.11. Tarjeta PGIO de una Raspberry pi.

Fuente: http://www.softwaresamurai.org/2017/10/28/rpi-button-and-gpio-ports/

2.12 Arduino

La placa Arduino es una placa de circuito impreso (PCB) que está diseñada específicamente para usar un chip de microcontrolador, así como otras entradas y salidas. También tiene muchos otros componentes electrónicos que son necesarios para que el microcontrolador funcione o amplíe sus capacidades.

Los microcontroladores son computadoras pequeñas que se encuentran dentro de un solo circuito integrado o chip de computadora, y son una excelente manera de programar y controlar dispositivos electrónicos. Muchos dispositivos, denominados tableros de microcontroladores, tienen un chip de microcontrolador y otros conectores y componentes útiles que permiten al usuario conectar entradas y salidas.

2.12.1 Arduino Uno

Arduino Uno es una placa orientados al desarrollo de proyectos electrónicos basado en el microcontrolador ATmega328P. Posee 14 pines de entrada / salida digital, las cuales 6 estan diseñadas para salida PWM, entre sus componentes también constan

de 6 pines que son usadas como entradas analógicas, un cristal de cuarzo de 16 MHz, un puerto USB y un botón para el reinicio de la placa.

Características más relevantes:

- Microcontrolador utilizado: Atmega328
- Fuente de alimentación: 5 voltios.
- Voltios usados de entrada: 7-12v (usado), 6-20v (voltaje limite).
- Pines digitales de In/Out: 14 (6 salidas PWM).
- Intensidad máxima por In/Out: 40mA.

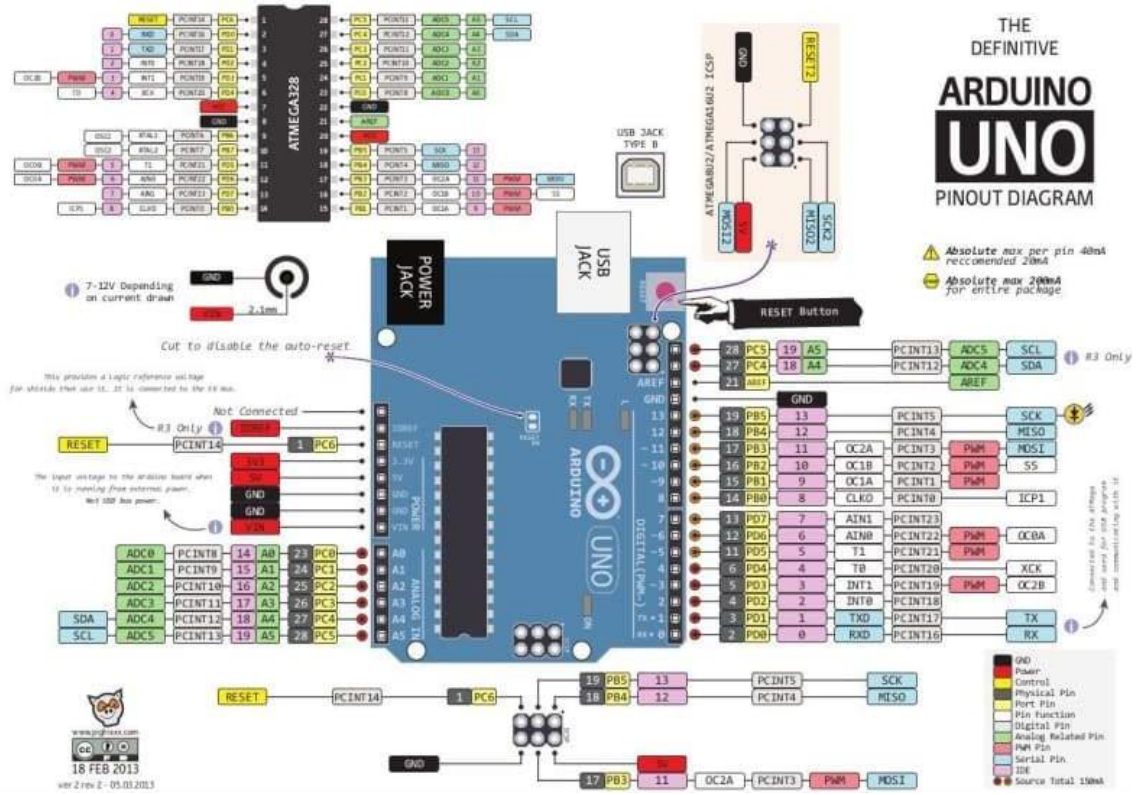


Fig. 2.12.1. Pinout Diagram Arduino Uno

Fuente: patagoniatec

2.12.2 Arduino Pro mini

A pesar de lo que sugieren los nombres, el Arduino Mini es más pequeño que el Nano. Esta placa también utiliza el mismo chip de microcontrolador ATmega328, pero se condensa aún más, eliminando todos los pines del cabezal y el conector mini-USB del Nano. Esta placa es excelente si el espacio es escaso, pero requiere mucha atención al conectarse porque una conexión incorrecta puede destruir fácilmente la placa.

El Pro mini es otro producto de SparkFun que lleva el minimalismo del Arduino Pro a nuevos límites. En la escala de Arduino, esto se encuentra perfectamente entre el Nano y el Mini. No tiene ninguno de los pines de cabecera o el puerto Mini-USB del Nano, y está un poco más extendido que el Arduino Mini. Esto tampoco tiene ninguna de las características de seguridad del Uno R3, así que tenga mucho cuidado al realizar el cableado, ya que una conexión incorrecta puede destruir fácilmente la placa.

Características más relevantes son:

- Microcontrolador utilizado: Atmega328
- Fuente de alimentación utilizable: 3.3voltios o 5voltios (dependiendo de la versión usada).
- Tensión utilizada de entrada: 3.35 a 12 voltios (versión Pro Mini 3.3volt.), 5 a 2 voltios (versión Pro Mini 5volt.).
- Puertos digitales In/Out: 14 (6 salidas PWM).
- Intensidad máxima por In/Out: 40mA.

Funcionamiento:

El módulo Arduino Pro mini tiene la capacidad de recibir fuente de alimentación a través de un cable al módulo FTDI o por un adaptador TTL a USB. Adicional posee la propiedad de funcionar a través de una alimentación regulado de 3.3v o 5v por el pin Vcc dependiendo del modelo, o por el pin RAW con una fuente sin regular.

Pines de alimentación:

- RAW: Para una fuente de alimentación no regulada.
- VCC: Fuente de alimentación regulada de 3.3 o 5 v.
- GND: Pin de conexión tierra o negativo.

Puertos In (Entrada) y Out (Salida):

El módulo Arduino de todas las versiones tienen la capacidad de programar el modo de los pines digitales que posee a través de la plataforma de desarrollo Arduino IDE, dando así la ventaja de ser programada de salida o de entrada a través del uso de funciones especiales. Los pines del Arduino trabajan en diferentes voltajes dependiendo del modelo que se usa, pudiendo ser de 3.3v o 5v, además de suministrar o recibir una intensidad máxima de 40 mA.

El módulo Arduino está compuesto por ciertos pines que pueden realizar funciones especiales:

- Comunicación UART: Son los pines utilizados especialmente para realizar la comunicación serial a través de un pin receptor y un pin transmisor. En el módulo Arduino pro mini los pines son 0 (RX) y 1 (TX) y se conectan con los pines TX-0 y RX-1 del adaptador TTL.

- Pines PWM: Están compuestos por 6 pines con la capacidad de generar una señal modulada de ancho de pulso de salida analógica. En el módulo Arduino Pro mini son los pines 3, 5, 6, 9, 10 y 11.

- Comunicación SPI: Son los pines encargados para la comunicación de interfaz periférica serial. En el módulo Arduino Pro mini los pines SPI son 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).

- Comunicación I2C: Son los pines que son utilizados para la comunicación con otros dispositivos utilizando la librería Wire.

- Led integrado: Dentro del Módulo Arduino Pro mini está equipado por un led para prueba de funcionamiento conectado al pin 13.

El módulo Arduino Pro mini posee 6 pines de entrada analógica, cada pin posee 10 bits de resolución (1024 valores) distribuidos en el conector lateral (4 pines) y 2 en los agujeros del interior (pin 4 y 5) del módulo.

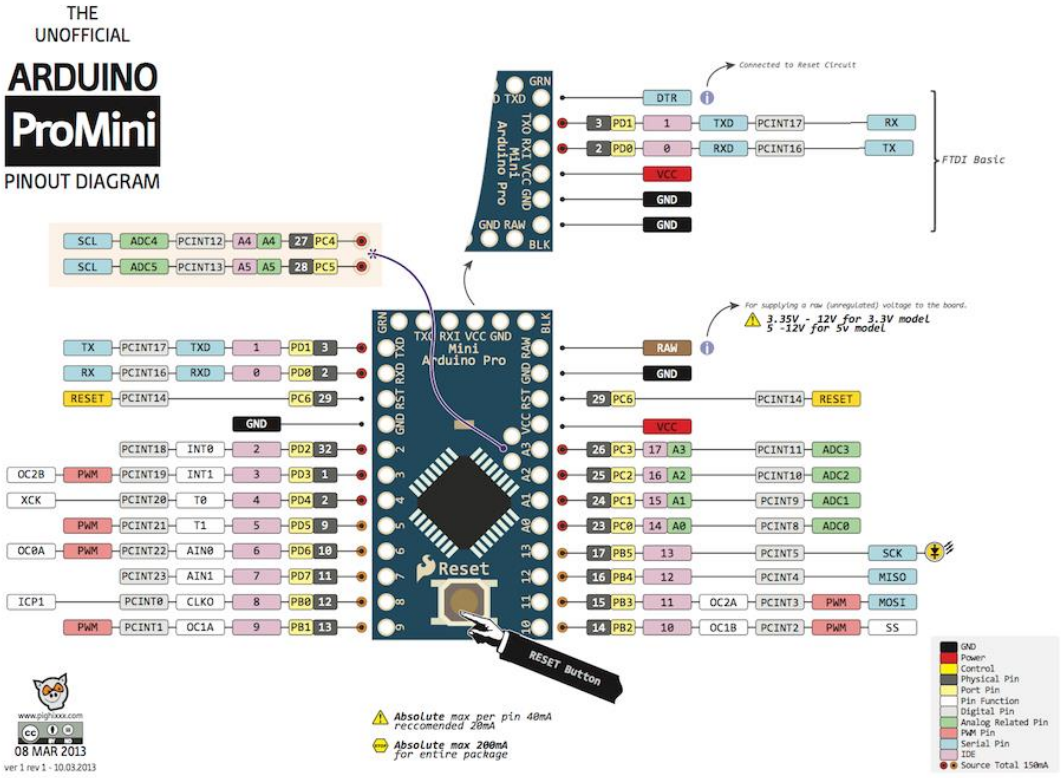


Fig. 2.12.2. Pinout Diagram Arduino Pro Mini

Fuente: patagoniatec

2.13 Modulo Quemador Arduino Pro Mini (USB to TLL)

Es una placa adaptadora de calidad alta que permite convertir de datos de USB a TTL, diseñados especialmente para proyecto que necesiten de una comunicación con puerto serial USB entre un ordenador, y microcontroladores o Arduino, ya que este adapta convirtiendo el puerto USB en un puerto serial UART. Ideal para la comunicación, conexión y compilación de código abierto para Arduino pro mini a través de un puerto USB.

Figura 2.13. Modulo Quemador USB to TTL,

Fuente: milyunpartes.com

2.14 GSM/GPRS SIM900 Shield

El módulo shield GSM GPRS SIM900 o también llamada ICOMSAT está desarrollado por Itead Studio, este modulo permite una conexión entre una red de telefonía celular con los módulos Arduino. El Arduino puede realizar funciones y hacer uso de servicios que ofrece voz y datos. El módulo de SIM900 tiene la capacidad de poder funcionar desde cualquier red GSM que existen en el mundo debido a que se trata de un módulo de trabajo en 4 bandas (850/900/1800/1900MHz).

Características más relevantes son:

- Operacionalidad global con cualquier proveedor, multibanda.
- Es controlada mediante comandos AT y AT extendidos.
- Incluye stack TCP/IP y soporta TCP, HTTP, FTP a través de comandos AT.
- Perfecto para transmisión de datos sobre GPRS.
- SMS en modo Texto y PDU.

Figure 2.14. Shield GSM SIM900

Fuente: electronics

2.15 Modem Arris TG862

El modem Arris TG862 del miembro de Touchstone tiene integrado puertos de enlace de telefonía residencial que admite comunicación de enlace de canales 8 x 4 para datos de hasta 320 Mbps de banda ancha. Se encuentra compuesto por la combinación de dos puertos de VoIP de clase portadora, un enrutador gigabit de 4 puertos y punto de acceso Wireless 802.11n de 2.4GHz con un respaldo de batería dentro de un solo dispositivo integrado.

TG862 es un modem que nuclea todos los servicios del usuario y distribuye servicios de datos, voz y video por IP de velocidad alta.

- Tiene la característica de poder ser configurable permitiendo personalizarla para gestión con una eficiencia eficaz, con diferentes métodos de aprovisionamiento.
- Tiene la propiedad de ser adaptable y proporciona una flexibilidad al momento de configurarlo y para el control con diferentes niveles de acceso local y remoto.
- La batería de respaldo cumple con los requisitos de la FCC garantizando 8 horas de duración. Batería disponible con mayor duración.

Especificaciones técnicas relevantes:

- Rango de frecuencia: 2400 a 2483.5 MHz.
- Interfaz de alimentacion USB 2.0.
- Voltaje de entrada: 100 a 240 Vca 50/60 Hz.
- Niveles de recepción: >-90dBM 802.11b 11mbps; > -76dBm 802.11g 54mbps,> -76dBM 802.11n HT20.
- Potencia de salida de transmisión: 18.5dBm + 1.5 / -1.5dB 802.11g (6mbps); 13.5dBm + 1.5 / -1.5dB 802.11n (MCS 7).

Fig.2.15. Modem TG862A

Fuente: HardReset99

2.16 High Power Wireless N Router TL-WR841HP

- Posee la característica de ser un amplificador de potencia alta y compuesto por dos antenas externas que se pueden desmontar la cual proporciona un alcance inalámbrico más grande por ser de alta ganancia de 9dBi.
- Posee una propagación de señal Wi-Fi potente capaz de atravesar paredes, obstáculos y elminar zonas muertas de señal.
- Tiene la característica de poseer una alta velocidad en la conexión inalámbrica a distancias amplias brindando una mejor experiencia de usuario.
- Velocidad Wireless de 300Mbps enfocados idealmente a la transmisión de video, juegos en línea y VoIP.

Especificaciones técnicas relevantes:

- Interfaz de 4 puertos LAN de 10 / 100Mbps y 1 puerto WAN de 10 / 100Mbps
- Fuente de alimentación de 12VDC / 1A
- Antena Omni Direccional Desmontable 2x9dBi (RP-SMA)
- Frecuencia de 2.4-2.4835GHz
- Inalámbrico: 64/128/152 bits WEP / WPA / WPA2, WPA-PSK / WPA2-PSK

Fig.2.16. Router TL-WR841HP

Fuente: tp-link.com

2.17 Nodemcu Esp8266

El módulo wifi NodeMCU es una placa de desarrollo basado en el microcontrolador ESP8266, además es un nombre de un firmware de fuente abierta.

El chip ESP8266 es altamente integrado desarrollado especialmente para las necesidades de conexión inalámbrica entre dispositivos. El chip ofrece una solución autónoma y completa de redes Wi-Fi, lo que le permite el alojamiento de una aplicación o funcionar como puente entre la red de Internet y un microcontrolador.

El ESP8266 posee grandes capacidades de procesamiento y almacenamiento, las cuales permiten el desarrollo de proyectos con sensores y dispositivos específicos integrados a través de sus pines de entrada y salida con un desarrollo de código

mínimo y carga mínima durante el tiempo de ejecución. Gracias a su alto grado de integración y adaptabilidad produce una reducción en la circuitería externa, y la totalidad de la solución, además de tener la capacidad de ocupar poco espacio en el diseño y elaboración de un circuito impreso.

Características:

- Microcontrolador ESP12E.
- Voltaje de alimentacion: 5v.
- Modulo Wi-Fi: 2.4 GHz.
- Comunicación UART Serial: GPIO3, GPIO1 (Rx, Tx), Puede ser configurado a GPIO15, GPIO13; Serial1: GPIO2 (solo Tx).
- Comunicación SPI: Pin Mosi: GPIO13 o SD1, Miso: GPIO12 o SD0, Sck GPIO14 o CLK, SS(Slave): GP, Level: 5V.
- Comunicación I2C: SDA: GPIO4, SCL: GPIO5.

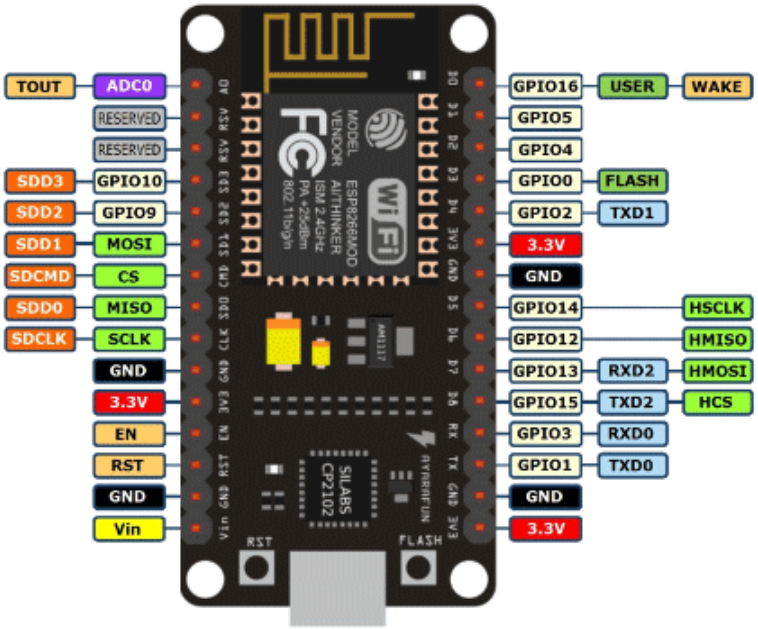


Figura 2.17. Diagrama Nodemcu Esp8266

Fuente: electronilab.co

2.18 Cámara

Las Raspberry Pi tiene integrado un puerto especial para la conexión de una cámara diseñada especialmente para el miniordenador, pese a la capacidad de soportar las tradicionales cámaras USB.

Este puerto especialmente diseñado permite una comunicación directa con el SOC

por lo que no existe la necesidad de instalar drivers especiales como con las tradicionales cámaras por USB, lo hace más sencilla la instalación además de que nos permite interactuar con la cámara directamente a través del lenguaje de programación Python.

Fig.2.18. Cámara

Fuente: BricoGeek

2.19 Lector Biométrico FPM10A

El sensor biométrico es un sensor el cuál reconoce la huella dactilar del usuario, este código biométrico se almacenará en la base de datos interna del dispositivo biométrico, pero podrá ser gestionada remotamente desde el aplicativo web.

Este sensor biométrico cuenta con una memoria interna para el almacenamiento de las huellas dactilares escaneadas y ofrece en API para poder realizar la gestión de estas.

Fig. 2.19.1. Sensor biométrico

Fuente: MaxElectrónica

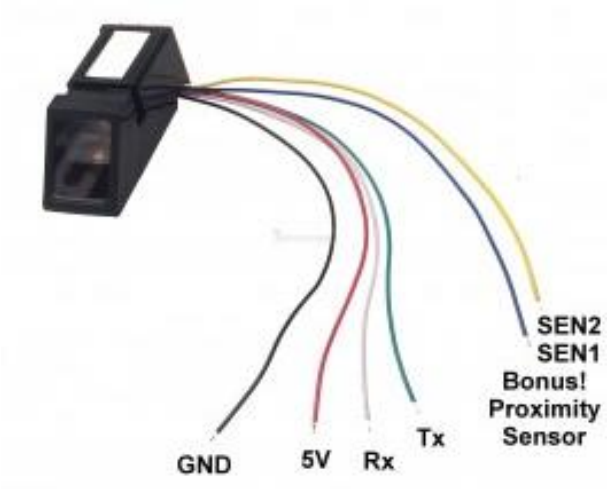


Fig. 2.19.2. Código de Colores

Fuente: Tinkersphere

2.20 RFID-RC 522

Las tarjetas de RFID son un mecanismo utilizado para almacenar información mediante campos magnéticos.

Estas tarjetas son ampliamente usadas ya que no necesitan una fuente de alimentación, sino que se polarizan con el campo magnético generado por el sensor RFID, la cual es necesario que las tarjetas magnéticas deban estar a una distancia cercana.

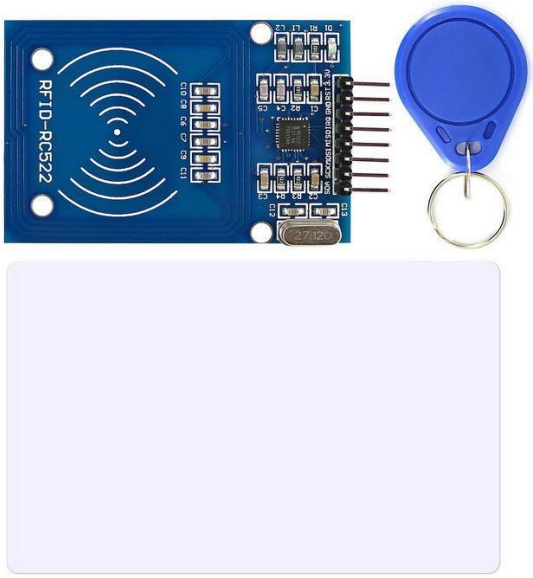


Fig.2.20. Lector y tarjetas RFID

Fuente: DealeXtreme

2.21 Módulo Relay 5V 1 Canal

El relé o también conocido como relevador es un dispositivo electromagnético. Realiza la función similar a un interruptor controlado por un circuito eléctrico en el interior en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes a través del paso o denegación del voltaje.

Especificaciones

- Voltaje entrada: 5 V.
- Voltaje de control:3 ~ 9 V.
- Voltaje de salida: 250 VCA o 30 VDC.
- Corriente a la salida: 10 A.
- Dimensiones: 43 x 17 mm.

Terminales

- Puerto VCC: Voltaje de entrada de 5 V.
- Puerto IN: Voltaje de entrada de la señal, comúnmente de 3.3 V.
- Puerto GND: Tierra o negativo.

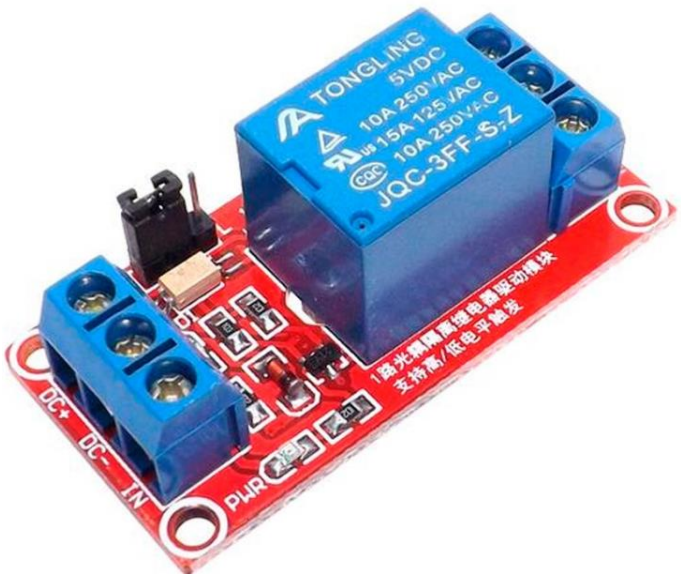


Fig.2.21. Modulo Relay 5V 1 canal

Fluente: planetaelectronico

2.22 Cerradura Electromagnética

La Cerradura Electromagnética son utilizados como dispositivos especialmente diseñado para la apertura en lugares de acceso limitado (control de acceso).

Las cerraduras electromagnéticas están compuestas por dos piezas principales, el electroimán que se alimenta por una fuente de energía, y de una lámina metálica conocida como pieza móvil o pieza polar. El electroimán se coloca comúnmente son instalados los marcos de las puertas gestionando el ingreso, trabaja como imán en la medida que circule corriente por su bobina y cierra la puerta.

Características:

- Fuerza que aplica: 140 Kg (300libras)
- Consumo de alimentación: 350 mA / 12VCC
- Funcionamiento de seguridad: Se desmagnetiza y abre la puerta cuando se quita la fuente de alimentación (Fail-safe).
- Campo de uso: Puertas de Vidrio, puertas de madera, puertas de metal.

Dimensiones y Peso

- Cerradura: 170 x 32 x 23 mm.
- Placa de ensamble: 152 x 32 x 10 mm.
- Peso: 1.3 kgs.

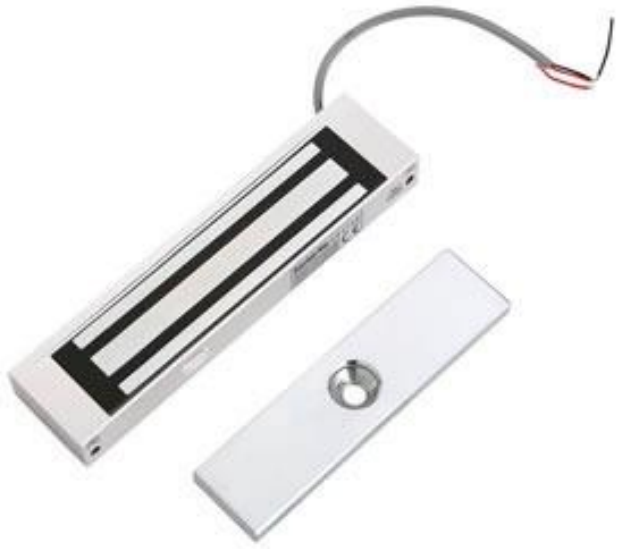


Fig.2.22. Cerradura Electromagnética

Fuente: gallerysecurity

2.23 Contacto Magnético MC270-S45

El dispositivo MC 270-S45 es un conjunto de contacto magnético robusto de alta seguridad en aluminio para instalaciones en puertas de acceso y otras instalaciones que requieran Grado de seguridad 3. Está diseñado con la capacidad de poseer larga durabilidad en ambientes ásperos. El kit consiste en MC 270-C que es la parte del conector con la función de contacto NC instalado en una carcasa de aluminio MC 200-4 y un imán potente en la carcasa de aluminio MC 200-5. El magnético potente permite una gran capacidad de apertura. El cable de cobrese encuentra protegido por una manguera protectora de acero MC T2.

Características más relevantes:

- Conexión eléctrica: 200 VDC / 500 mA / 10 VA.
- Iman compuesto por Alnico 5.
- Material de cobertura: Aluminio.
- Rango de Temperatura: -40-+70°C
- Conexión a través de cable.

Fig.2.23. Contacto Magnético MC270-S45

Fuente: hommaxsistemas

2.24 UPS NT-511 Forza

El dispositivo NT-511 de Forza se ha diseñado con la característica de estar estructurada con seis puertos de conexión eléctrica, como objetivo principal de mantener una protección total sobre sus equipos electrónicos personales y

periféricos. El dispositivo tiene la característica de ser de tamaño compacto que ocupa poco espacio, lo que facilita el transporte y la ubicación del equipo en espacios reducidos. Esta serie se ha desarrollado garantizando una mayor protección frente a problemas eléctricos e interrupciones de corto eléctrico. Los periodos de inactividad son eliminados virtualmente incrementando la disponibilidad de dispositivos críticos al mantener un suministro de energía eléctrica limpia y estable para todos los puertos de conexión.

Características más relevantes:

- Capacidad: 500VA/250W
- Topologíadiseñada: Interactiva
- Voltaje de alimentacion: 120V
- Tipo de puerto de entrada: NEMA 5-15P
- Tipos de puerto de salida: 6 x NEMA 5-15R

Fig.2.24. NT-511

Fuente: forzaups.com

CAPITULO 3

3 DISEÑO E IMPLEMENTACION DE LA PLATAFORMA

3.1 Especificaciones de la Plataforma

La plataforma de control de acceso y supervisión remota consta de las siguientes funciones establecidas por etapas: la obtención de datos del dispositivo magnético y biométrico, la comunicación entre los dispositivos, la obtención de imágenes, envío de mensajes de aviso y la supervisión remota.

Todas estas funciones corresponden al desarrollo de la plataforma, ya que cada una de ellas es de suma importancia porque todas conforman y dependen de la otra.

3.2 Características de las etapas de la Plataforma

3.2.1 Análisis de la Obtención de Datos

La obtención de datos se lo realiza a través del Arduino con una herramienta de software abierto para la escritura de código llamada "Arduino IDE", en la programación se encuentran códigos para la adquisición de datos, estos datos son códigos únicos obtenidos por el dispositivo RFID o del Sensor Biométrico.

3.2.1.1 Obtención de Datos del Dispositivo Magnético y Biométrico

La obtención de datos se realiza a través de dos maneras: la adquisición del código de una tarjeta magnética a través del dispositivo RFID o la adquisición del código dactilar a través del sensor biométrico, estos códigos son enviados al Arduino directamente, después según la función que se desea realizar, se puede ingresar la clave como código nuevo de usuario a la base de datos o compararlos con los datos ya existentes en la Base de datos.

Estos datos que son únicos para cada usuario son enviados como datos desde el Arduino Pro Mini hacia la Raspberry Pi 3 para ser ingresados o comparados en MySQL y además pueden ser editados y supervisados a través de la gestión remota con la aplicación web.

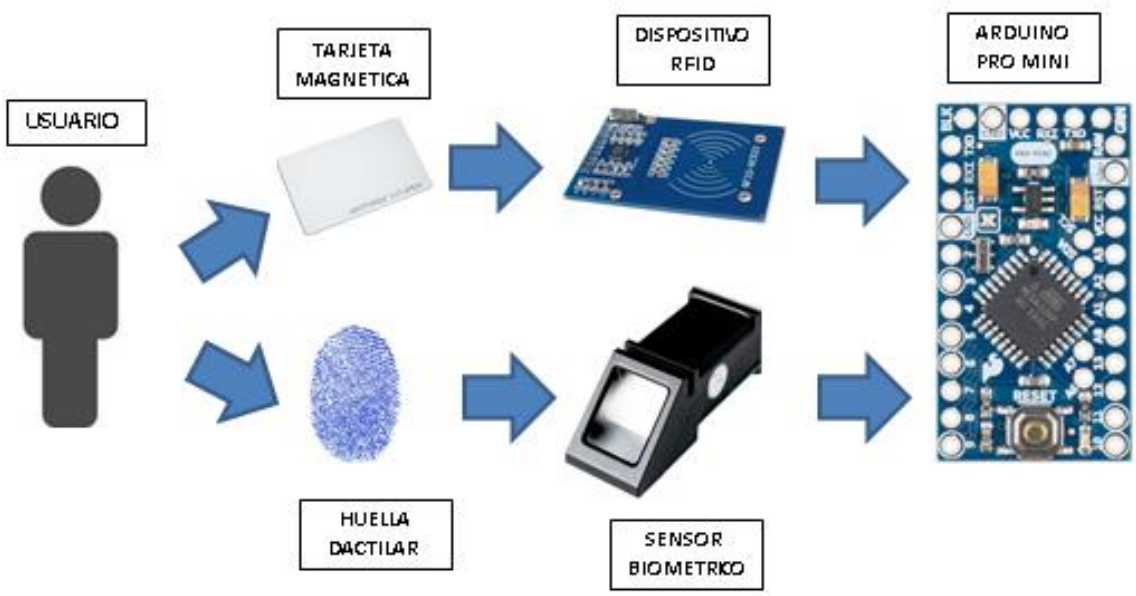


Fig. 3.2.1.1. Diagrama de Obtención de Datos.

Fuente: Los Autores

3.2.1.2 Bibliotecas para Arduino

La programación en Arduino comprende de varias bibliotecas que sirven para la obtención de datos a través de los sensores biométricos y magnéticos. Para la comunicación inalámbrica a través de un interfaz Wifi, las bibliotecas son llamadas en el código principal denominado "rfidAndBiometric.ino", en la programación también están definidos los pines que se usan para la comunicación de los dispositivos de adquisición de datos y la conexión inalámbrica; todos estos datos son analizados y comparados a través de varias programaciones realizados en los Raspberry Pi.

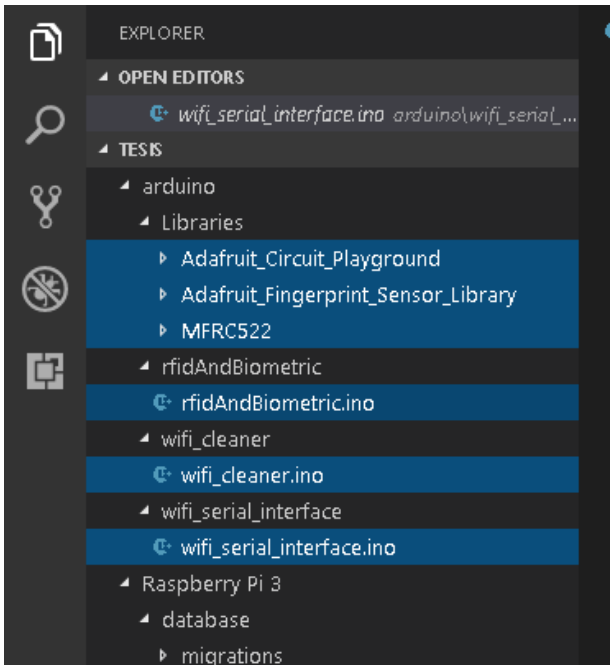


Fig. 3.2.1.2.1. Bibliotecas en Arduino Pro Mini.

Fuente: Los Autores

Estas bibliotecas son llamadas a través de la función "#include" seguido por el nombre de la biblioteca que se desea incluir dentro del proyecto; y los pines de uso son definidos a través de la función "#define" tanto para el dispositivo de RFID y del sensor magnético, estas funciones se las pueden observar en las dos figuras que se muestran a continuación.

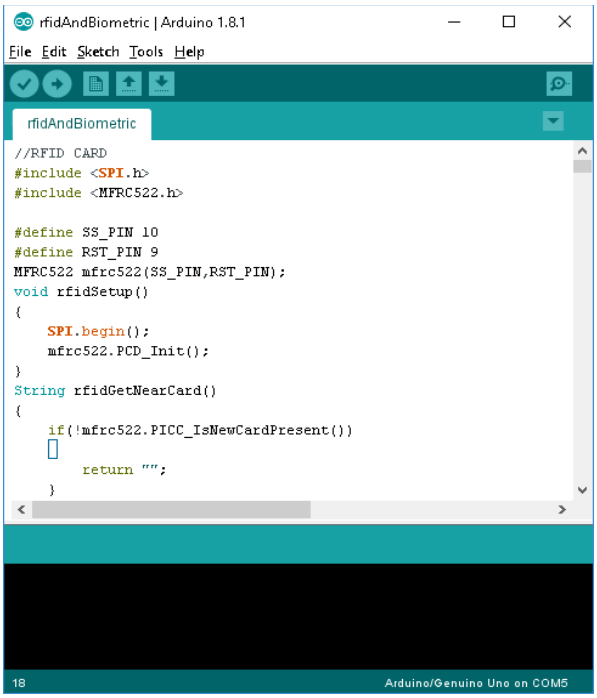


Fig. 3.2.1.2.2. Llamada de las Bibliotecas de RFID.

Fuente: Los Autores

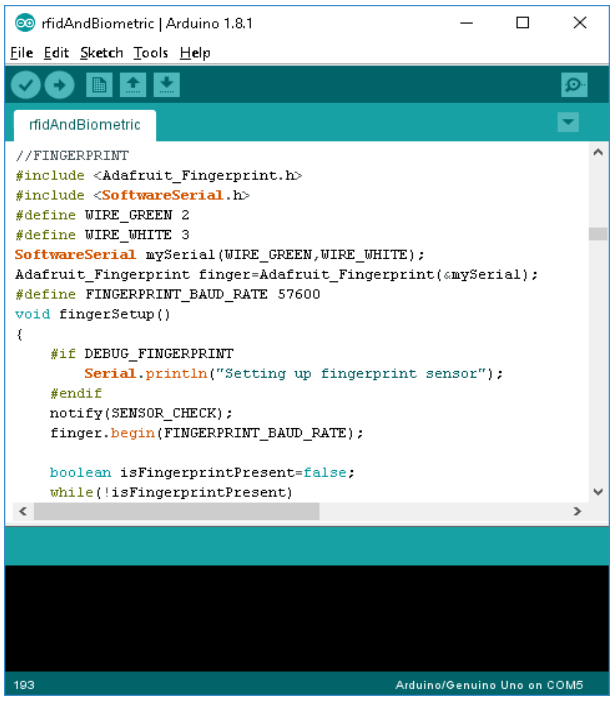


Fig. 3.2.1.2.3. Llamada de las Bibliotecas de FingerPrint.

Fuente: Los Autores

3.2.2 Análisis de la comunicación entre dispositivos

Todos los dispositivos de la plataforma de control de acceso y supervisión remota están conectados entre sí, para dar paso a la recepción y transmisión de información, los datos son trasmitidos en tiempo real; todos los dispositivos están enlazados por lo que entre ellos tienen una interdependencia para la tarea de acceso y gestión del almacén; cada comunicación tiene su propio proceso dependiendo de la función que se desee realizar.

3.2.2.1 Comunicación entre Raspberry Pi 3 y Arduino Pro Mini

La comunicación entre la Raspberry Pi 3 con el Arduino Pro Mini se puede desarrollar de dos maneras diferentes: la comunicación alámbrica y la comunicación inalámbrica para el desarrollo de la plataforma

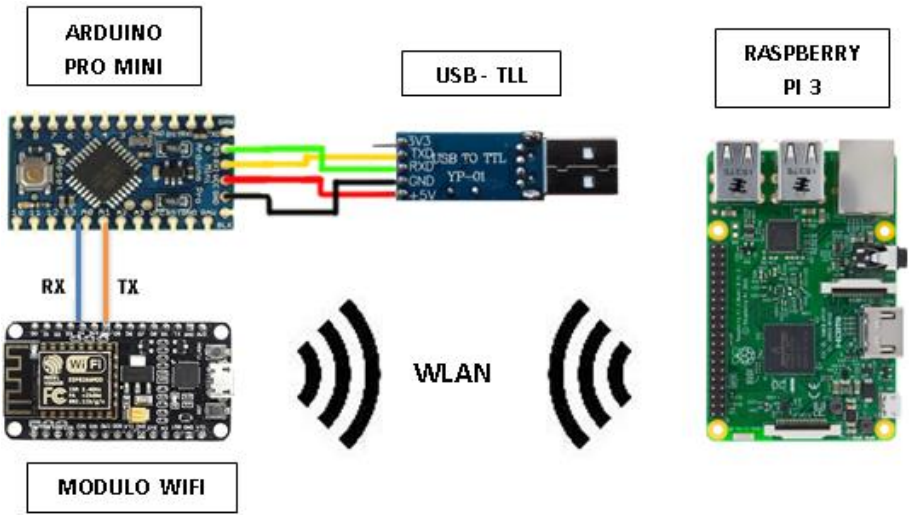


Fig. 3.2.2.1. Comunicación Raspberry Pi 3 a Arduino Pro Mini.

Fuente: Los Autores

3.2.2.1.1 Comunicación Alámbrica

El Raspberry Pi 3 se puede conectar con una conexión alámbrica a través del módulo quemador USB-TLL al Arduino Pro mini para realizar las funciones y a la vez se encuentran conectados inalámbricamente a través de un módulo Wifi para el chequeo del estado de comunicación.

3.2.2.1.2 Comunicación inalámbrica

El Raspberry Pi 3 se conecta inalámbricamente con el Arduino por medio del interfaz Wifi para el aviso apropiado del funcionamiento, el Arduino envía un script para la notificación del estado, el módulo Esp8266 envía datos de manera secuencial a la Raspberry utilizando wifi para informar el estado del Arduino, la Raspberry actualiza una variable con la fecha de la última vez que fue avisado; la Raspberry cada 10 segundos verifica cuanto tiempo ha pasado del último aviso que está activo, al pasar más de 10 segundos la Raspberry pi 3 da por caído al Arduino y muestra un mensaje en la interfaz web.

3.2.2.1.3 Control de acceso por Cerradura Electromagnética y Raspberry Pi 3

Se conoce muy bien que los dispositivos Raspberry Pi maneja niveles de voltaje menores a los 5VDC por lo que se dificulta activar la cerradura magnética con este

voltaje ya que la misma se acciona en 12 Vdc. Para lograr su funcionamiento se conecta con un relay que al recibir el pulso desde la Raspberry Pi 3 ocasionan la activación de la cerradura magnética.

Este módulo Relé será controlado directamente con los pines GPIO de la Raspberry Pi 3, para poder dar acceso de entrada al local comercial; el pulso se produce guacias a la programación del script que envía un mensaje de código binario el cual indicara la función de apertura o de cierre; el script está desarrollado a través de la programación de Node el cual implica que si el usuario corresponde con los datos respectivos de la base de datos Mysql se podrá abrir la cerradura, en caso de que no esté dentro de la base de datos, esta enviará un código binario negativo el cual impedirá la apertura al local y activará el sistema de envío de mensajes de aviso de alarma.

3.2.2.2 Comunicación Inalámbrica entre los dispositivos Raspberry Pi

La comunicación que se realiza entre los dispositivos Raspberry Pi 3 y la Pi Zero, se produce a través de una conexión por medio del Red de Área Local Inalámbrica WLAN para la transacción y recepción de datos y funciones, al momento de conectarse y recibir los datos, estos datos son enviados a la aplicación Web que se lo puede Gestionar desde cualquier Red.

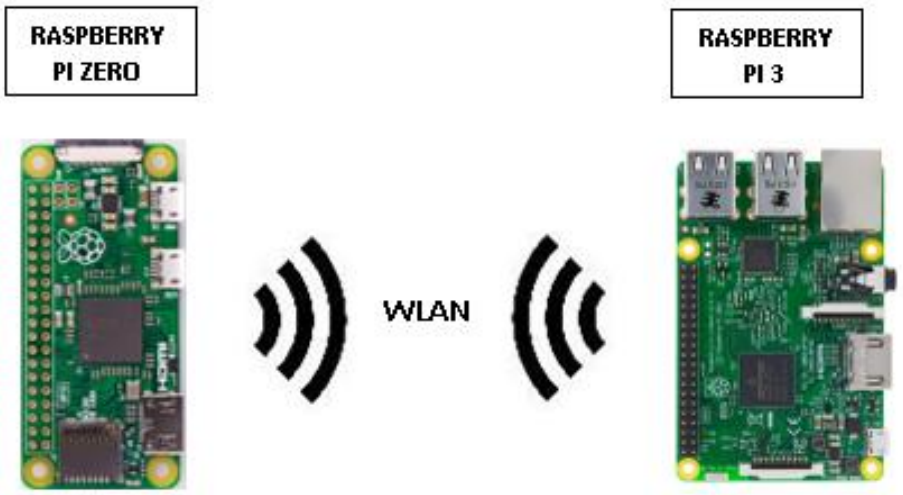


Fig. 3.2.2.2. Comunicación Raspberry Pi 3 a Raspberry Pi Zero.

Fuente: Los Autores

3.2.3 Análisis de la Obtención de Imágenes

La obtención de imágenes consta de una programación desarrollada en Python dentro de la Raspberry Pi Zero que se encuentra conectado con una cámara especial

desarrollada por la empresa Raspberry Pi, estas imágenes son mostradas en la aplicación web para uso de seguridad y gestión de entrada del almacenamiento

3.2.3.1 Tiempo de captura en tiempo real

El tiempo de capture es en tiempo real, se acciona cuando el contacto magnético se acciona y envía un código binario "1" y se para cuándo se envía un código binario diferente "0", está programado directamente desde los dispositivos Raspberry

3.2.3.2 Capacidad de almacenamiento

La capacidad de almacenamiento de fotos desarrollada en la aplicación web fue programada para albergar un total de 300 imágenes que son tomadas directamente desde la Cámara Raspberry Pi Zero.

La aplicación web tiene capacidad de albergar un número limitado de fotos ya que a razón de que mientras más datos son subidos a la aplicación web más información tiene que guardar, esta información puede exceder a la capacidad limitada y por causa de este efecto puede hacer que la aplicación se ralentice.

3.2.3.3 Contacto Magnético

El contacto magnético es un dispositivo que se usa como sensor para que la toma de imágenes sea automática, está conectado directamente en una comunicación con la Raspberry Pi Zero y la cámara Raspberry Pi, el cual al momento en que la cerradura y la puerta se abra, el sensor o contacto magnético al estar en contacto con su otra parte magnética enviará un script de código binario el cual producirá que la cámara Raspberry tome fotografías de manera automática.

3.2.3.4 Transmisión al aplicativo web

La transmisión de imágenes o fotos son en tiempo real, cuando el contacto magnético es activado al estar uno a lado del otro se envía un script, un código binario directamente a la Raspberry el cual lo lee, al enviar el código "1" se envía un mensaje el cual empieza a tomar fotos automáticamente hasta que el sensor deja de estar en contacto con su lado el cual envía el script "0" el cual detiene la toma de

fotos, estas fotos son enviadas directamente a la Raspberry Pi Zero nuevamente para ser subidas a la red.

3.2.4 Sistema de Aviso de intruso

El sistema de aviso consta de dos tipos de mensajes, el mensaje GSM vía telefonía y el mensaje email vía correo electrónico.

Estos mensajes son activados cuando una persona intenta ingresar al local y no está dentro de la base de datos, al momento que los sensores escanean y la huella o tarjeta no coinciden dentro de la base de datos o existe una ausencia de ellas, entonces se envían comandos por medio de la Raspberry pi 3.

3.2.4.1 Mensaje GSM

El mensaje vía GSM consta de un script que envía un mensaje de texto dirigido al número puesto en la programación la cual pertenece al propietario, avisándole sobre la detección de un intruso y de una posible amenaza, la Raspberry Pi 3 envía un script para la activación del módulo GSM/GRPS conectado directamente por un Arduino uno.

3.2.4.2 Mensaje email

El mensaje vía correo electrónico consta del mismo script de texto indicando el intento de entrada de un intruso adicional a una fotografía tomada directamente de la cámara de la Raspberry Pi 3, la programación está desarrollada en Python bajo la misma condición.

3.2.5 Análisis de la Supervisión Remota

La supervisión remota se encuentra conformado como una función que se puede realizar a través de la aplicación web, nuestro aplicativo web está desarrollada por medio de varios componentes que se encuentran codificados dentro de él, la gestión remota comprende de la programación en MySQL que corresponde a la gestión de la base de datos, el aplicativo está formado por VueJS que es un marco de trabajo orientado hacia el entorno de la gestión con el usuario del aplicación web y NodeJS que está diseñado para desarrollar aplicaciones de red escalables.

3.2.5.1 Capacidad de uso en tiempo real

La aplicación web está desarrollada en nodeJS la cual es un entorno de programación basado en Javascript, nodeJS trabaja a velocidades increíbles porque aprovecha el motor V8 desarrollado por Google para ejecutar las funciones de manera asíncrona, ya que está diseñado sin hilos lo que significa que no se necesita que una función haya terminado para comenzar otra nueva, la cual permite trabajar en tiempo real.

3.2.5.2 Gestión inalámbrica

La gestión inalámbrica está dirigida a la programación de MySQL la cual es un sistema de administración de base de datos basada en SQL.

La gestión inalámbrica se puede desarrollar a través de la programación de NodeJS la cual ejecuta funciones basadas en JavaScript en tiempo real para el uso del entorno MySQL para la gestión de la base de datos.

3.2.5.3 Funciones de la Plataforma

Al momento de entrar en el aplicativo web después de validar el usuario y la contraseña correspondientes para el encargado de la supervisión del local comercial, se puede dar uso de las funciones de agregar, comparar y eliminar información de la base de datos.

La supervisión del acceso del lugar se realiza por medio del chequeo de la información que es subida automáticamente desde los dispositivos de Raspberry Pi al aplicativo web, se puede observar la numeración o código de cada usuario que se ingresó por medio del código de la tarjeta magnética o por el código dactilar adquirida por el sensor biométrico, dicho código estará representado en el aplicativo web con el respectivo nombre del usuario correspondiente y adjuntado a la hora exacta en la que ingresaron, también se podrá visualizar imágenes tomadas en tiempo real al momento en que la puerta haya sido abierta.

La plataforma está compuesta por la ejecución de tres tipos de funciones la cual se basa en la comprobación de datos, el registro de usuario y la eliminación de usuario.

3.2.5.3.1 Comprobación de datos

Al momento de que el usuario intenta acceder al almacén comercial tiene que ingresar un código de acceso que puede ser por medio de su tarjeta magnética o por medio de su huella dactilar y al aplastar un pulsador realiza la ejecución del programa y genera el ingreso de información, al obtener el código que es único para cada usuario, la plataforma adquiere la información y es revisada por la base de datos para comprobar si el usuario existe, en caso de que el usuario sea reconocido como miembro de la empresa la plataforma envía un script para la apertura del local comercial, pero si se comprueba que no existe autorización de ingreso para el usuario entonces se envía un script negando el acceso y se activa el sistema de mensajes de alarma.

3.2.5.3.2 Registro de usuarios

Al momento que se desea ingresar algún usuario nuevo dentro de la base de datos, se debe de registrar a través del aplicativo web, se ingresa el nombre correspondiente al usuario y el código hexadecimal correspondiente a la tarjeta magnética, si se desea ingresar la información de la huella dactilar se envía un script que pasa por la Raspberry Pi 3 hacia al Arduino Pro Mini para hacer uso de la función de registro, el Arduino ejecutará la función que se dirigirá al sensor biométrico en el cual se hará el ingreso de la nueva huella dactilar, esta información del nuevo usuario será ingresada y formará parte de la base de datos MySQL.

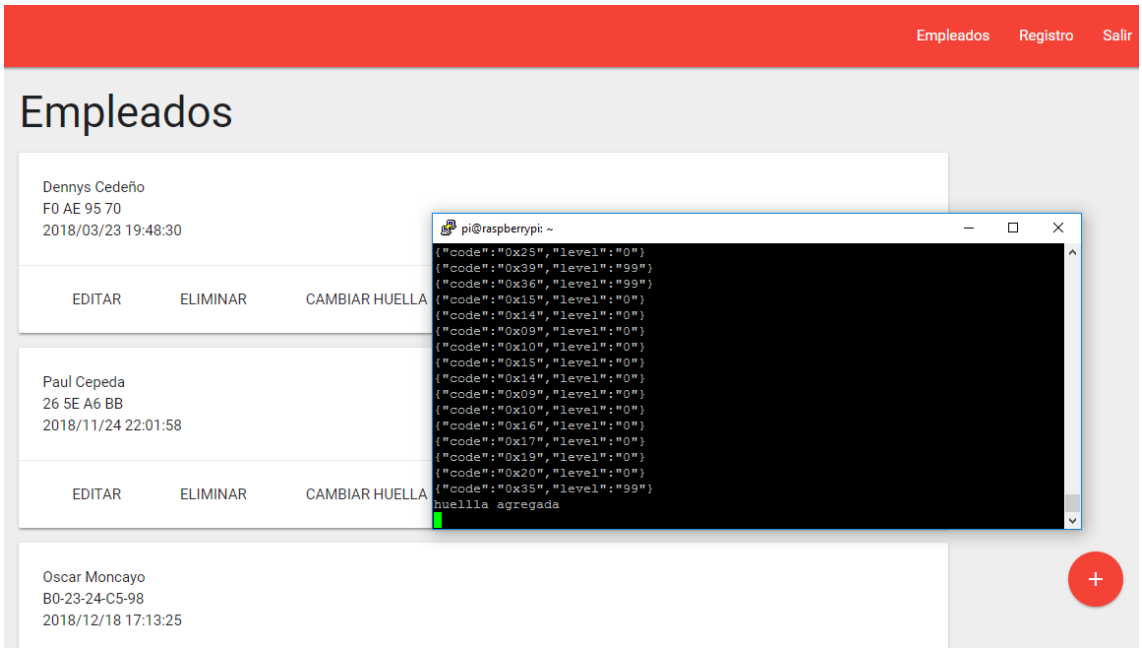


Fig. 3.2.5.3.2. Ingreso de Huella a la base de datos.

Fuente: Los Autores

3.2.5.3.3 Eliminación de usuario

Al momento de que los encargados de la empresa comercial ya no desee el servicio de algún usuario y deje de trabajar en el local comercial, se deberá realizar la respectiva eliminación de usuario a través del aplicativo web, ingresamos por medio del protocolo de seguridad para hacer uso de sus funciones, el cual una de ella es la eliminación de usuarios, el encargado debe hacer uso de la suspensión respectiva de la tarjeta RFID que es propiedad de la empresa y en la base de datos se elimina información de código magnético y del código dactilar del usuario que se desee.

3.3 Programaciones de la Plataforma

Dentro de la plataforma se encuentra desarrollada varias programaciones desarrolladas en entornos como NodeJs, VueJs, Mysql y Python.

En la Raspberry Pi 3 se encuentra realizada una programación en NodeJs basada en JavaScript para la comparación de los datos de código magnético y biométrico con la base de datos desarrollado en MySQL, y además de una programación en VueJs para el desarrollo visual de la aplicación web.

En la Raspberry Pi Zero se encuentra desarrollada una programación para la obtención de imágenes basado en Python, el cual se encuentra en comunicación constante a través de la red LAN.

Estas programaciones se desarrollan entre sí para la perfecta funcionalidad de la Plataforma de control de acceso y supervisión remota.

3.3.1 Programación de la base de datos MYSQL

MySQL nos permite desarrollar nuestra propia base de datos en un servidor privado creado en nuestro propio dispositivo, esta base de datos está conectado con la aplicación web realizado a través de VueJS, para crear la conexión debemos realizar sus propias formas de programación para que se realice la comunicación

Fig. 3.3.1. Conexión con la Base de Datos MySQL.

Fuente: Los Autores

Esta programación siempre es necesaria al momento de desarrollar un aplicativo web, ya que con esta programación se conecta a la base de datos MySQL, sin esta conexión el aplicativo web no poseería información dentro de sí misma para realizar las diferentes funciones de la plataforma.

3.3.2 Programación por NodeJs

Para que exista una comunicación exitosa con diferentes equipos (host) capaces de conectarse a la red como son los teléfonos celulares, de diferentes marcas y sistemas operativos, es necesario realizar la programación en NodeJs con una versión avanzada, superior a la versión 4, la cual nos permite acceder a la aplicación web desarrollada por medio de Vuejs y NodeJs.

3.3.3 Visualización y programación del aplicativo web por VUEJS

La programación en el entorno VueJS es desarrollada para la creación del aplicativo web, ya que VueJS es un marco de trabajo basado en programación JavaScript la cual permite crear el entorno, por la cual el encargado de la empresa podrá ingresar libremente por medio de los protocolos de seguridad para realizar la supervisión y gestión de la base de datos del local comercial.

La programación VueJS está enfocada en la visualización del aplicativo usando bibliotecas ya existentes dentro de ella la cual también se puede agregar más bibliotecas adicionales elaborados por agentes exteriores ya que es un entorno adaptable porque puede ser implementado en el HTML de cualquier proyecto web

3.3.4 Comunicación con el programa principal Python

La programación para la toma automática de fotos está desarrollada en el entorno de programación llamada Python lo cual es el entorno oficial de la Plataforma Raspberry, la codificación está desarrollada en la Raspberry Pi Zero lo cual estará encargada de la adquisición de fotos y la subida de las mismas para el proceso de supervisión.

3.4 Configuraciones de Red

3.4.1 Lan con la Wan

El usuario puede conectarse a través de cualquier parte del mundo a la red local de la empresa comercial a través de una configuración que se realiza en el router de redireccionamiento de puertos, a través de esta configuración se logra conectar desde cualquier red de internet y realiza una comunicación virtual con la dirección IP de nuestra red local y con un puerto especifico el cual están conectados nuestros equipos de la plataforma, a través de las programación de VueJs y NodeJs se puede acceder a nuestro aplicativo web y con sus datos privados y contraseña privada podrá ser uso del aplicativo, accedemos a nuestra base de datos MySQL para de esta manera supervisar los datos o gestionar en vista del control de acceso de los usuarios que trabajan dentro de la empresa.

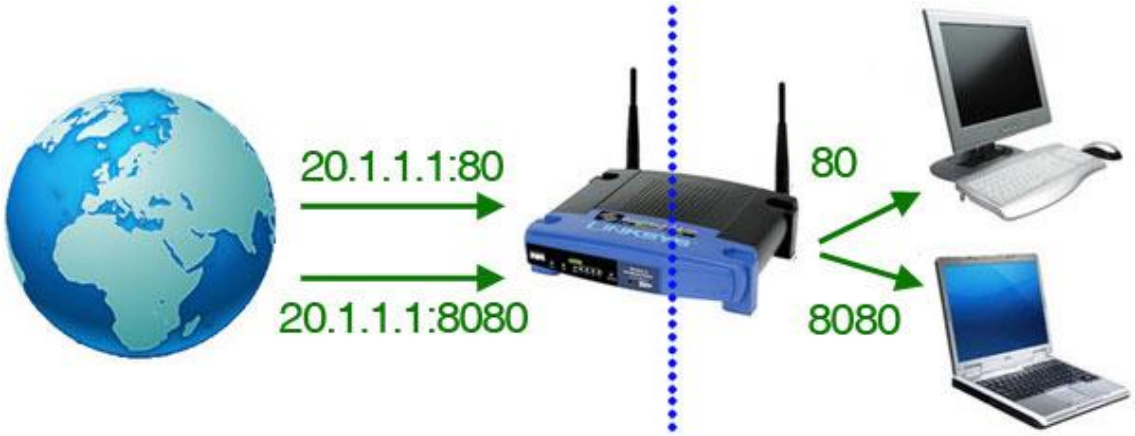


Fig.3.4.1.1. Redireccionamiento de puertos

Fuente: ElastixTech

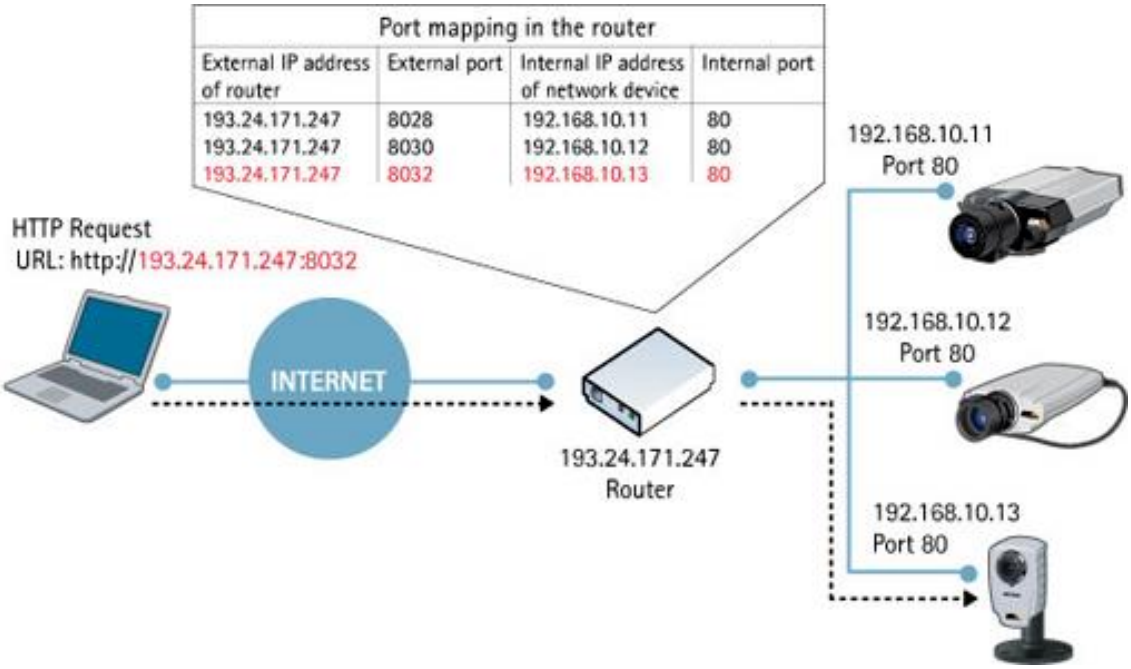


Fig.3.4.1.2. Tabla de la Estructura de Redireccionamiento de puertos

Fuente: Axis Communications

3.4.2 Configuración Router

Para realizar la conexión de red privada se debe realizar las configuración tanto en las Raspberry Pi como en el Router, en la Raspberry Pi 3 y la Raspberry Pi Zero se les configura la opción de conexión inalámbrica para asignarles una dirección ip estática para que al momento de encenderse la raspberry Pi no tenga que ser adquirida una dirección al azar por el Router, y así establecer una conexión estable

con una misma ip, esta dirección tiene que ser seleccionada dentro del rango de ip del router, preferiblemente se le asigna una de las últimas para que así no se produzca cualquier tipo de interferencia cada que vez que alguien se conecte a la red LAN.

El equipo arris brinda dos modos de servicio NAT y Bridge

- Modo NAT: Varios equipos comparten la conexión del equipo físico.
- Modo Bridge: Un equipo se conecta al modem de manera individual por cable LAN.

El modo bridge consiste en estar como puente (sin nateo) y brindarle el servicio a un solo equipo conectado por cable de red; solo proporcionara una IP pública directa al equipo receptor, este modo es más eficaz en la entrega del servicio, en este caso será el equipo TP LINK WR 841 HP quien administra la red del servicio, en este modo el equipo ARRIS de la operadora CLARO nos proporcionara el servicio mientras nosotros podremos realizar las configuraciones de habilitaciones de puertos específicos, y demás, por lo cual, nuestro equipo TP LINK WR841 será el AP quien brinda el servicio de internet.

IP ESTATICA

Luego de dejar nuestro equipo TP LINK WR 841 HP como administrador de la red, procedemos a realizar la respectiva configuración del router de manera ESTATICA

Tabla 3.4.2. *IP Estática de la Empresa*

IP	200.124.253.11
MASCARA DE SUBRED	255.255.255.192
PUERTA DE ENLACE	200.124.253.65
DNS 1	200.124.235.194
DNS 2	200.124.235.195

fuente: autor

Fig.3.4.2.1. Ip Publica Estática de la Empresa en el Modem Arris

Fuente: Autores

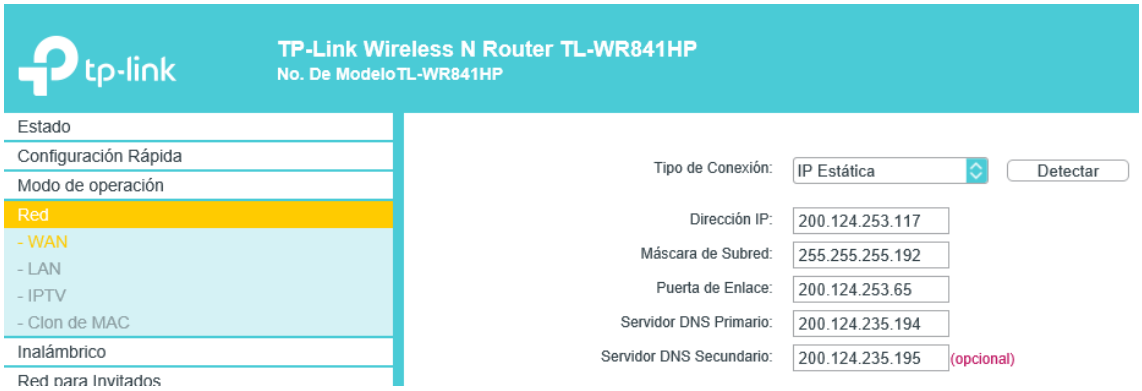


Fig.3.4.2.2. Router TL-WR841HP modo Bridge

Fuente: Autores

Se realiza la respectiva configuración de manera estática, en router realizamos la configuración de los puertos en Virtual Server, en este caso, los equipos se conectan a la red creada, verificando que exista conexión operativa por parte de estos, y se realiza la respectiva configuración de puertos.

Tabla 2.14. *Mapeo de rutas*

Puerto de servicio: 5000	DIRECCION IP: 192.168.0.200	PROTOCOLO: TCP/UDP
Puerto de servicio: 3000	DIRECCION IP: 192.168.0.100	PROTOCOLO: TCP/UDP

fuente: autor

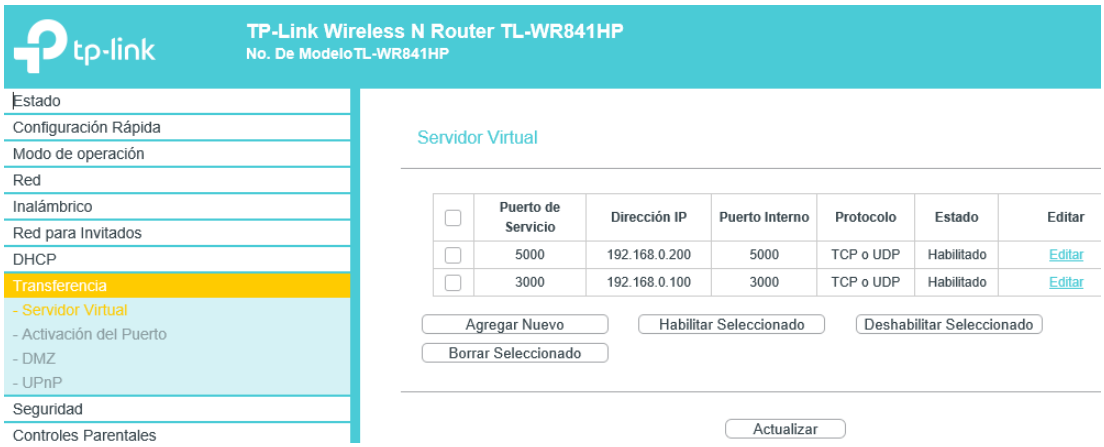


Fig.3.4.2.3. Mapeo de puertos y direcciones

Fuente: Autores

Esta configuración de puertos brinda un acceso específico a los equipos que están conectados a la red de la empresa, también se procede con la configuración DMZ la cual conlleva a aplicar la IP de un equipo que se desea el acceso desde cualquier red (interna – externa) y más la habilitación de puertos y la configuración de la IP Pública de manera estática se logra la entrada y comunicación desde cualquier parte remota del mundo que haya servicio de internet.

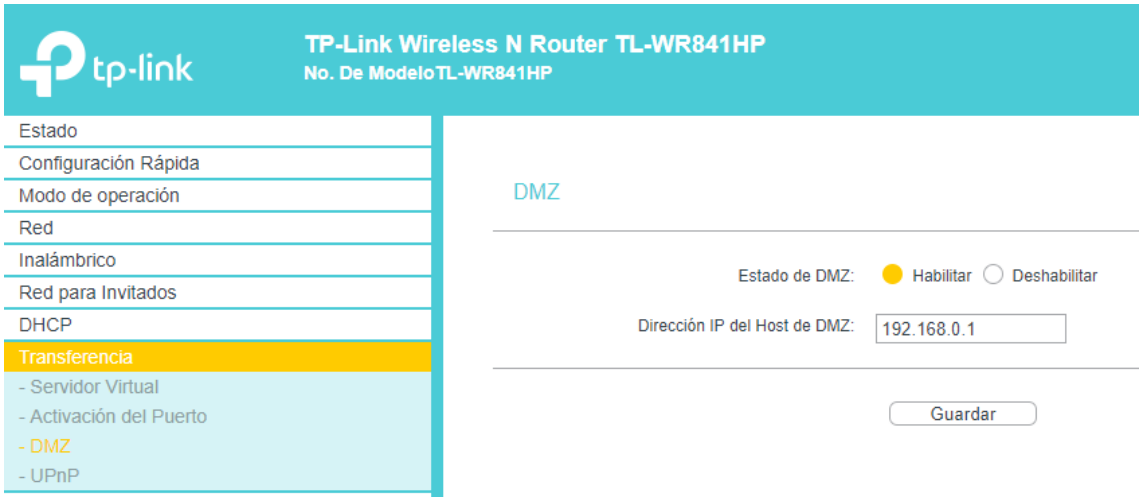


Fig.3.4.2.4. Configuración DMZ

Fuente: Autores

3.5 Diseño del circuito

El diseño del circuito impreso de la plataforma de control de acceso y supervisión remota se encuentra desarrollado en una aplicación virtual que la podemos encontrar en la Red conocida como EasyEDA, es una página web pública en el cual cualquier usuario puede crear PBC creando su propio usuario.

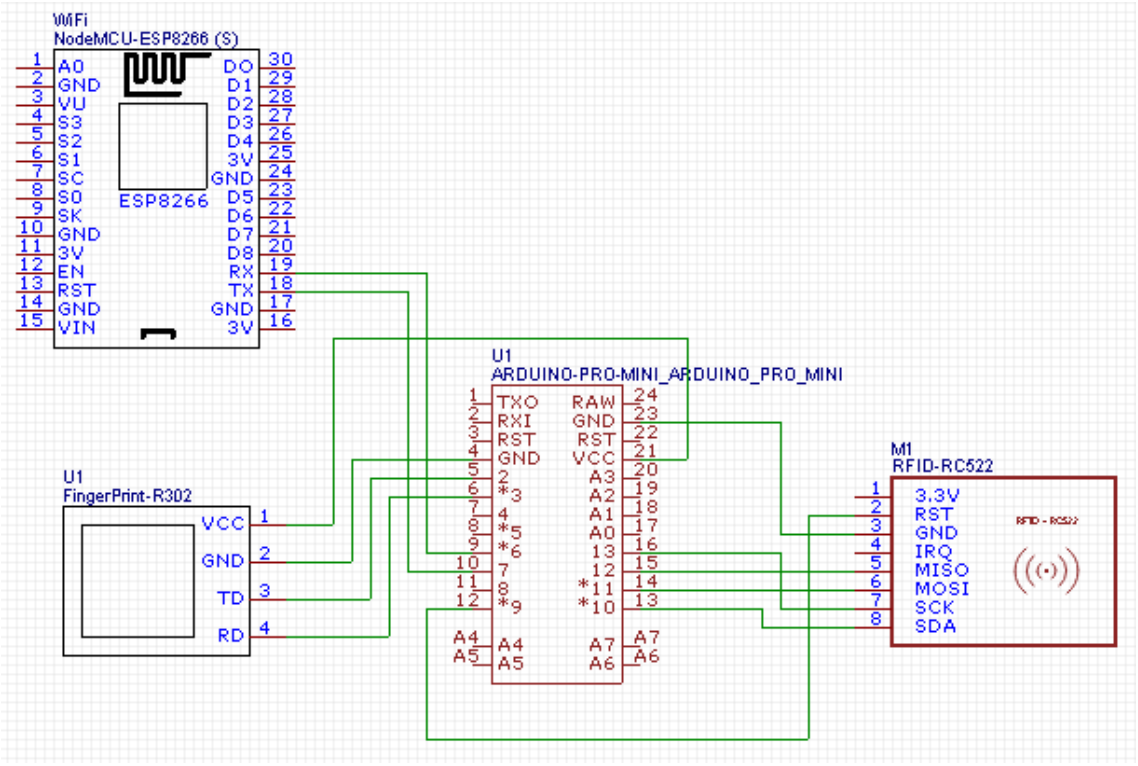


Fig. 3.5.1. PCB de las conexiones Arduino Pro Mini.

Fuente: Los Autores

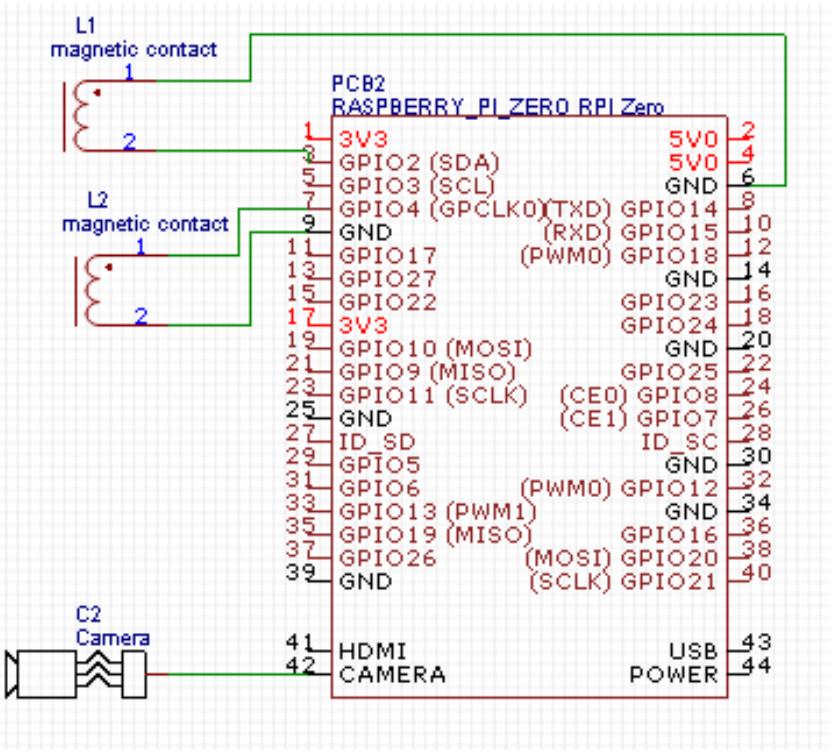


Fig. 3.5.2. PCB de las conexiones Raspberry Pi Zero.

Fuente: Los Autores

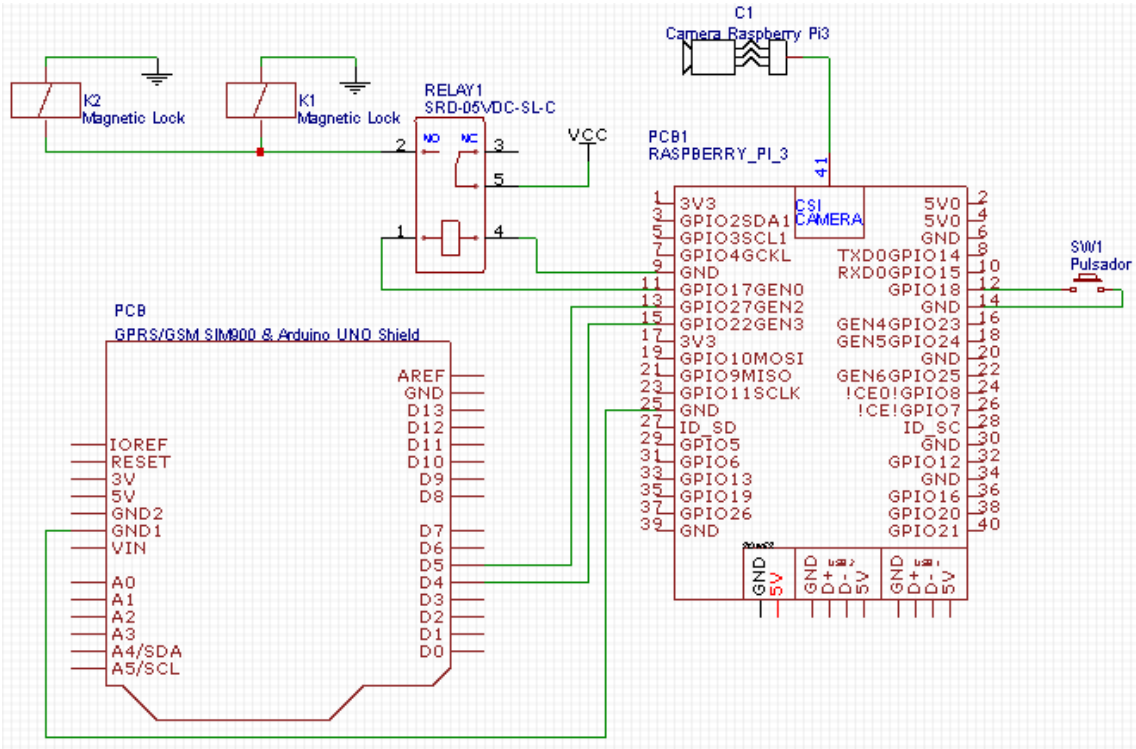


Fig. 3.5.3. PCB de las conexiones Raspberry Pi 3.

Fuente: Los Autores

CAPITULO 4

4 ANALISIS EXPERIMENTAL DE LA PLATAFORMA

4.1 Análisis Experimental de Capturas de las Cámaras

En la etapa de prueba de eficiencia y funcionamiento de la parte de capture de imágenes en tiempo real en la entrada y salida del local comercial, se comprobó la funcionalidad a través del método experimental.

Para la captura de imágenes se realizó la programación de manera específica a través de Python para tomar fotos solo en el momento de la apertura y cierre de la puerta principal y/o de la puerta secundaria.

4.1.1 Capture de fotos en momento de apertura de puertas

4.1.1.1 Acceso habilitado por comprobación correcta de personal

Cuando el acceso al local comercial sea válido por medio del código magnético o biométrico, se activará la apertura de la cerradura electromagnética permitiendo así que al momento de abrir la puerta principal y/o puerta secundaria se produzca una interrupción con los contactos magnéticos activando la captura de imágenes de manera automática.

Fig.4.1.1.1. Cerradura electromagnética y contacto magnético

Fuente: Autores

4.1.1.2 Obtención de fotografías en la puerta principal

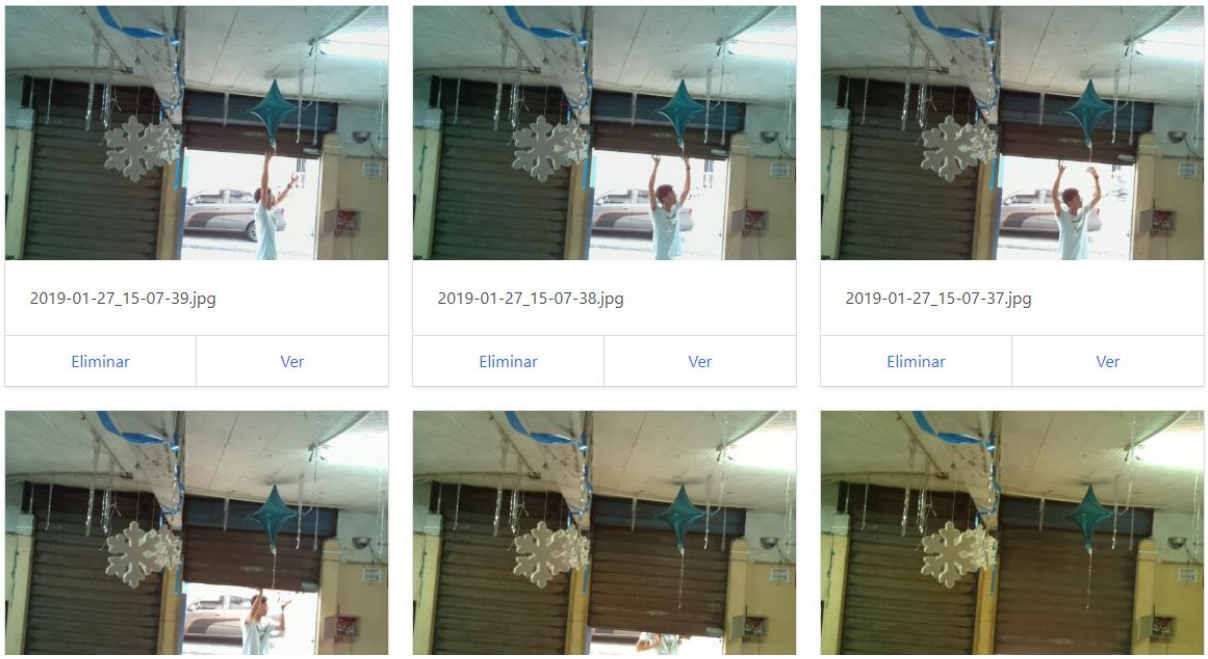


Fig. 4.1.1.2. Captura de imágenes por la apertura de la puerta principal

Fuente: Autores

4.1.1.3 Obtención de fotografías en la puerta secundaria

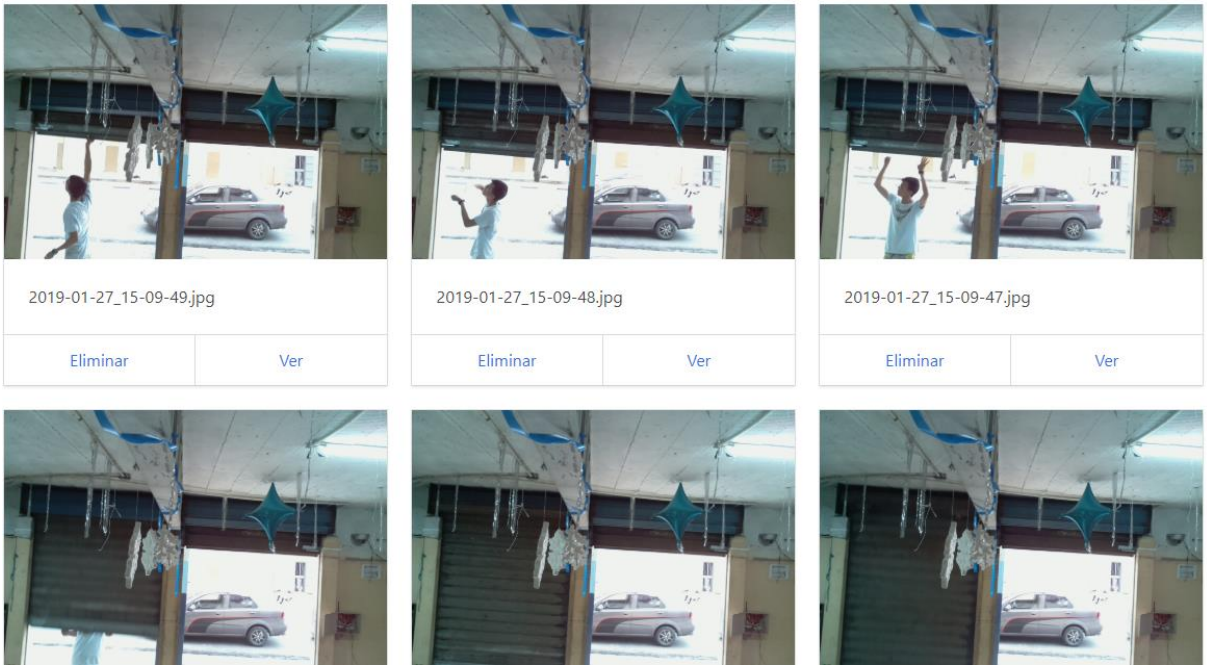


Fig. 4.1.1.3. Captura de imágenes por la apertura de la puerta secundaria

Fuente: Autores

4.1.2 Resultados y comparaciones

En la fase de experimentación y análisis de diversos tipos de cámara de vigilancia concluimos que al momento de instalación no poseían una buena estructura de conexión lo cual obtuvimos diferentes puntos de desventajas:

- Mucho cableado
- Poca ampliación y difícil movimiento
- Instalación de drivers

Por lo cual se decidió en implementar la plataforma con una cámara especialmente diseñada, la cámara Raspberry Pi posee un solo cable de bus de datos diseñados para cualquier versión del dispositivo Raspberry que presenta una gran ventaja por lo cual se procedió desde un principio a la obtención del dispositivo, el raspberry Pi Zero está conectado inalámbricamente con el dispositivo principal Raspberry Pi 3 por lo que permite una manejabilidad del dispositivo permitiéndonos instalarlo en cualquier lugar; al haber realizado el debido análisis investigativo se llegó a la conclusión de usar una cámara diseñada por la compañía Raspberry Pi ya que no sería necesario de instalar drivers adicionales y su conexión es directa.

4.2 Análisis Experimental del Dispositivos de Acceso Biométrico

En la realización de pruebas de funcionamiento de la parte de obtención del código dactilar por medio del sensor biométrico, se logra comprobar un eficiente rendimiento.

La plataforma realiza dos funciones, cada función se ejecuta condicionalmente por el resultado de la comprobación de la huella dactilar; al comprobar que el código dactilar se encuentra almacenados en la base de datos se procede con la habilitación del acceso y se registra el usuario, al comprobar que el código dactilar no se encuentra almacenada en la base de datos se procede con la activación del sistema de seguridad de aviso de intruso.

4.2.1 Acceso Habilitado por el Sensor Biométrico

4.2.1.1 Intento de ingreso al local comercial

Fig.4.2.1.1. Intento de acceso por el sensor biométrico

Fuente: Autores

4.2.1.2 Código de interpretación

```
'0x09':'Tomando imagen',
'0x10':'Imagen convertida',
'0x11':'Imagen en malas condiciones',
'0x12':'Error de función',
'0x13':'Imagen inválida',
'0x14':'Por favor ubique el dedo',
'0x15':'Por favor quite el dedo',
'0x16':'Creando modelo',
'0x17':'Huella encontrada',
'0x18':'Huellas no coinciden',
'0x19':'Modelo agregado',
'0x20':'Huella agregada',
'0x21':'Ubicación inválida',
'0x22':'Error escribiendo en el sensor',
'0x23':'Error buscando',
'0x24':'Huella encontrada',
```

```
#searcg finger
'0x39':'Huella desconocida',
'0x40': 'Coincidencia encontrada',
```

Fig. 4.2.1.2.1. Librería de código

Fuente: Autores

```
huella puesta
{"code":"0x14","level":"99"}
{"code":"0x14","level":"0"}
{"code":"0x09","level":"0"}
{"code":"0x10","level":"0"}
{"code":"0x23","level":"0"}
{"code":"0x24","level":"0"}
{"code":"0x40","level":"99","id":"1","confidence":"340"}
huella id  1
huella correcta
        ** puerta abierta **
```

Fig.3.4.1.2.2. Escaneo de la huella dactilar por el sensor biométrico

Fuente: Autores

4.2.1.3 Mensaje de comprobación de usuario correcto

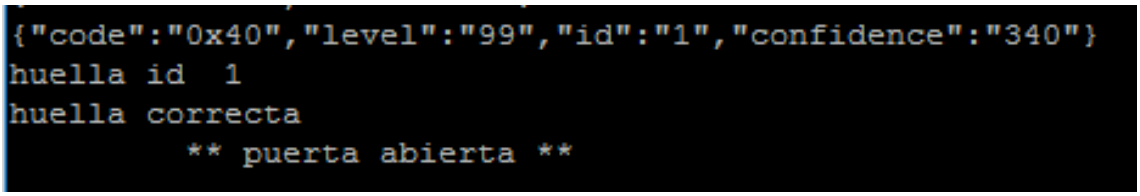


Fig.4.2.1.3. mensaje de detección de huella correcta

Fuente: Autores

4.2.1.4 Registro de entrada del usuario correcto.

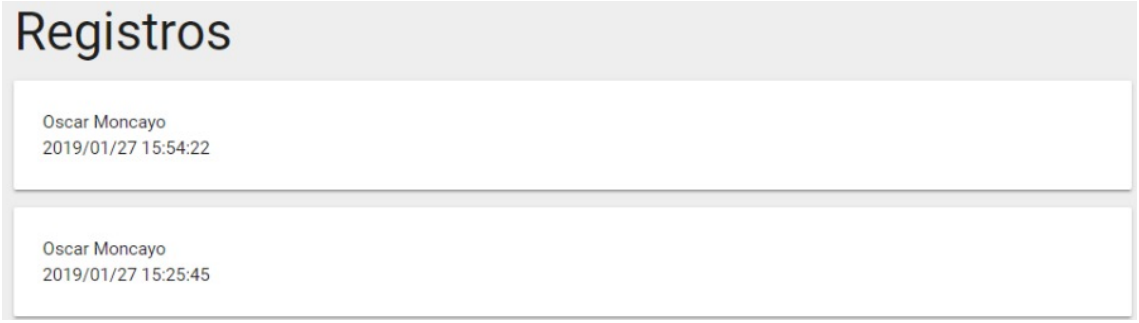


Fig. 4.2.1.4. Registro de ingreso

Fuente: Autores

4.2.1.5 Habilitación de la apertura de la chapa electromagnética.

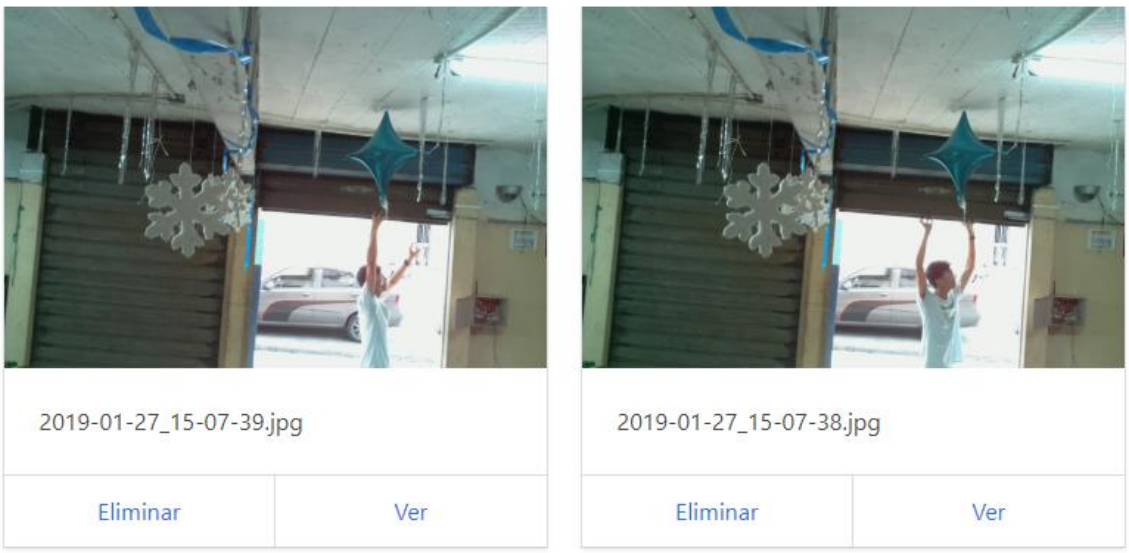


Fig. 4.2.1.5. Apertura de la puerta principal

Fuente: Autores

4.2.2 Acceso denegado por el Sensor Biométrico

4.2.2.1 Código de interpretación

```
huella puesta
{"code":"0x14","level":"99"}
{"code":"0x14","level":"0"}
{"code":"0x09","level":"0"}
{"code":"0x10","level":"0"}
{"code":"0x23","level":"0"}
{"code":"0x25","level":"0"}
{"code":"0x39","level":"99"}
huella id  -1
huella incorrecta
connecting to email
sending email
        done
         ** puerta no abierta **
```

Fig.4.2.2.1. Escaneo de la huella dactilar no reconocida

Fuente: Autores

4.2.2.2 Mensajes de comprobación de usuario incorrecto

```
huella id  -1
huella incorrecta
connecting to email
sending email
        done
         ** puerta no abierta **
```

Fig. 4.2.2.2. Mensaje de detección de huella incorrecta

Fuente: Autores

4.2.2.3 Envío de mensaje SMS de alerta de intrusión.

Fig. 4.2.2.3. Mensaje SMS de aviso de intrusión

Fuente: Autores

4.2.2.4 envió de mensaje de email adjuntado una foto de alerta de intrusión.

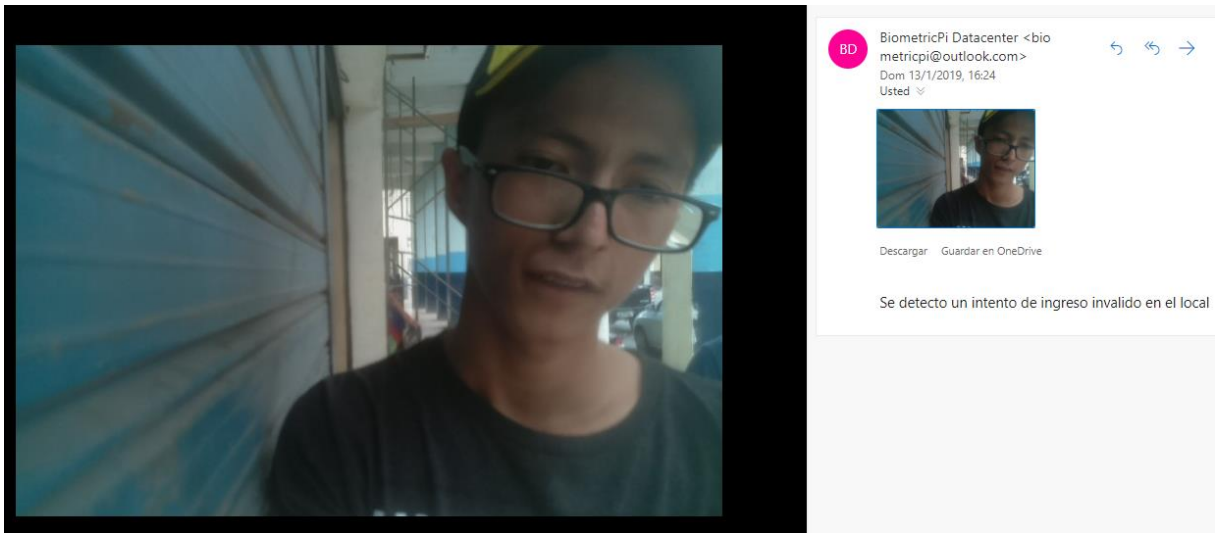


Fig. 4.2.2.4. Mensaje de correo electrónico con fotografía

Fuente: Autores

4.2.3 Resultados y comparaciones

En el tiempo de pruebas de la plataforma obtuvimos algunos inconvenientes con el dispositivo Biométrico ya que al principio realizábamos escaneo de la huella, pero el dispositivo no lo registraba y a veces se apagaba o se pausaba al momento de la obtención no mandaba la información de la huella; llegamos al análisis de diferentes posibles causas:

- Programación errónea
- Bibliotecas del Arduino erróneo
- Modulo Arduino dañado
- Modulo biométrico dañado

Por la cual procedimos a realizar el chequeo de cada opción, en la primera procedimos verificando que la programación al cual está conectado con el Arduino la cual se realiza la sincronización; en la segunda opción revisamos que la biblioteca del sensor biométrico este en llamadas en la programación del Arduino revisando las líneas de escritura el cual se lo descarto; para la tercera opción realizamos la verificación del dispositivo Arduino realizando el chequeo del funcionamiento con el sensor magnético el cual las pruebas se realizaron con éxito verificando que el dispositivo Arduino pro mini está en buen estado.

Al realizar el chequeo de todas las alternativas más sencillas llegamos a la conclusión que el módulo biométrico se encuentre dañado, realizamos varias pruebas revisando con otra fuente de voltaje para alimentarlo, pero seguía con el problema, entonces verificamos la plataforma con otro dispositivo biométrico la cual trabajo exitosamente y se rectificó el inconveniente.

4.3 Análisis Experimental de los Dispositivos de Acceso Magnético

En la realización de pruebas de funcionamiento de la parte de obtención del código magnético hexadecimal por medio del sensor RFID, se logra comprobar un rendimiento eficiente.

La plataforma realiza dos funciones, cada función se ejecuta condicionalmente por el resultado de la comprobación de la huella dactilar; al comprobar que el código magnético se encuentra almacenados en la base de datos se procede con la habilitación del acceso y se registra el usuario, al comprobar que el código magnético no se encuentra almacenada en la base de datos se procede con la activación del sistema de seguridad de aviso de intruso.

4.3.1 Acceso habilitado por el sensor RFID

4.3.1.1 Intento de ingreso al local comercial.

Fig.4.3.1.1. Intento de acceso por el sensor RFID

Fuente: Autores

4.3.1.2 Comprobación con la base de datos.

```
boton pulsado

esperando por tarjeta
{"card":"26 5E A6 BB"}
codigo leido 26 5E A6 BB , codigo esperado  F0 AE 95 70
tarjeta incorrecta
codigo leido 26 5E A6 BB , codigo esperado  26 5E A6 BB
tarjeta correcta
codigo leido 26 5E A6 BB , codigo esperado  B0-23-24-C5-98
tarjeta incorrecta
codigo leido 26 5E A6 BB , codigo esperado  F0 D4 F5 B3
tarjeta incorrecta
{"code":"0x42","level":"99"}
        ** puerta abierta **
```

Fig. 4.3.1.2. Comprobación de la tarjeta magnética con la base de datos

Fuente: Autores

4.3.1.3 Mensaje de comprobación de usuario correcto.

```
codigo leido 26 5E A6 BB , codigo esperado  26 5E A6 BB
tarjeta correcta
        ** puerta abierta **
```

Fig. 4.3.1.3. Mensaje de detección de tarjeta RFID correcta

Fuente: Autores

4.3.1.4 Registro de entrada del usuario correcto.

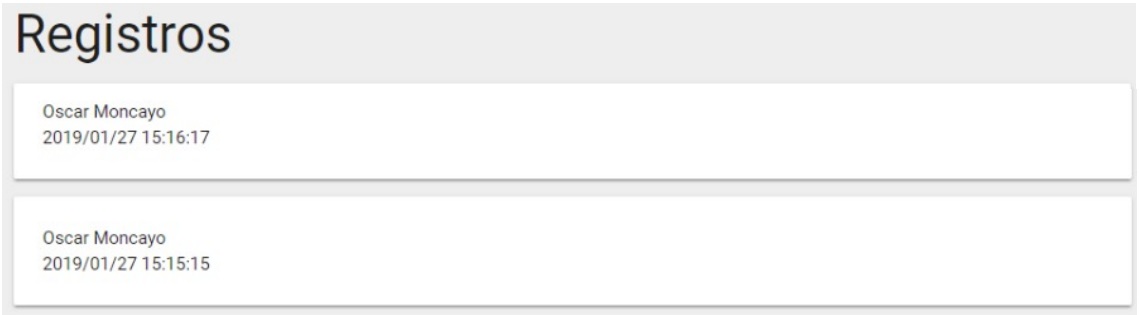


Fig. 4.3.1.4. Registro de ingreso

Fuente: Autores

4.3.1.5 Habilitación de la apertura de la chapa electromagnética.

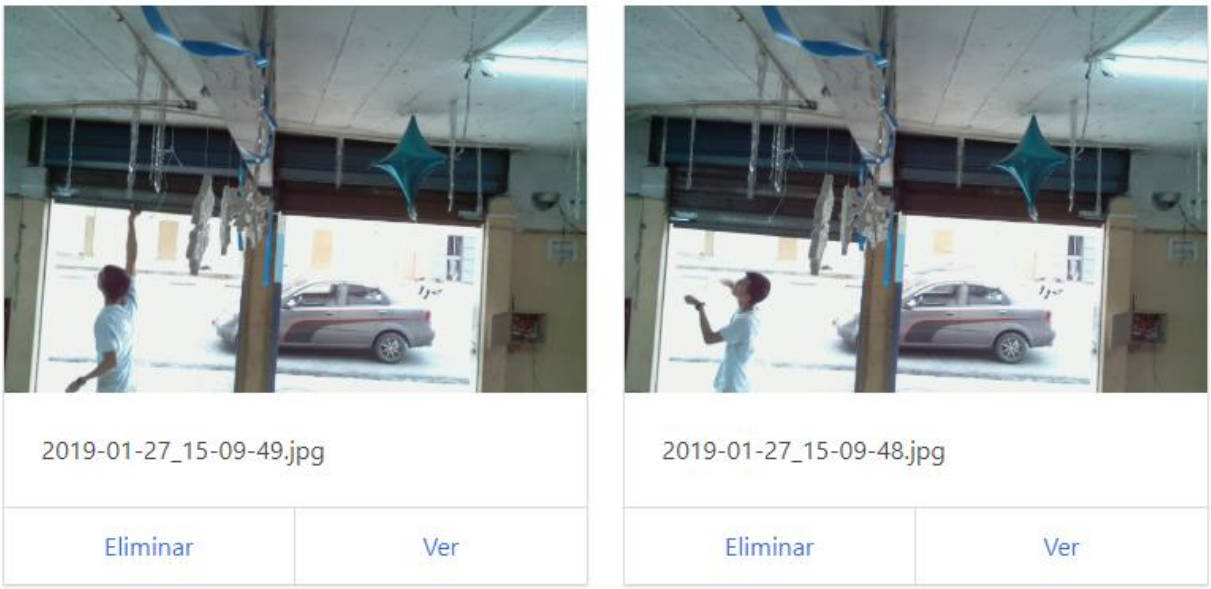


Fig. 4.3.1.5. Apertura de la puerta secundaria

Fuente: Autores

4.3.2 Acceso denegado por el Sensor RFID

4.3.2.1 Comprobación con la base de datos.

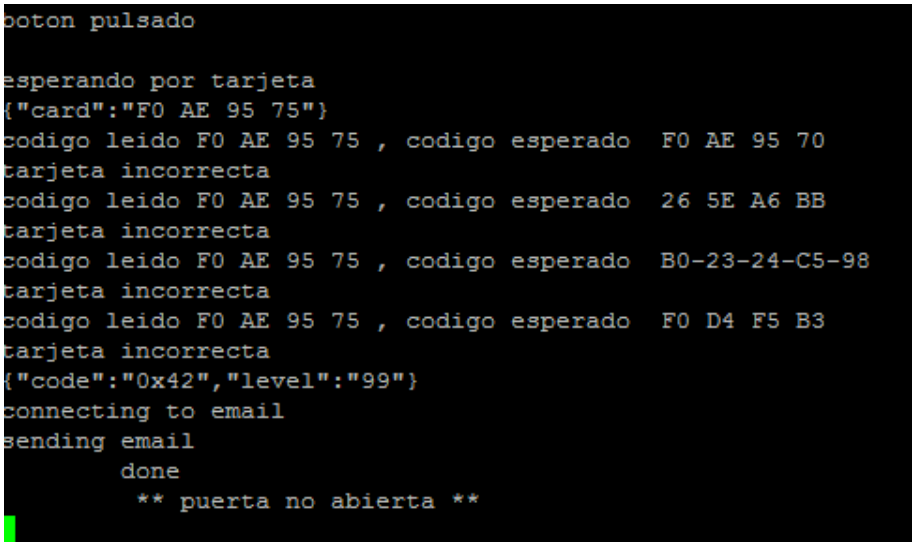


Fig. 4.3.2.1. Comprobación de la tarjeta magnética desconocida con la base de datos

Fuente: Autores

4.3.2.2 Mensaje de comprobación de usuario incorrecto.

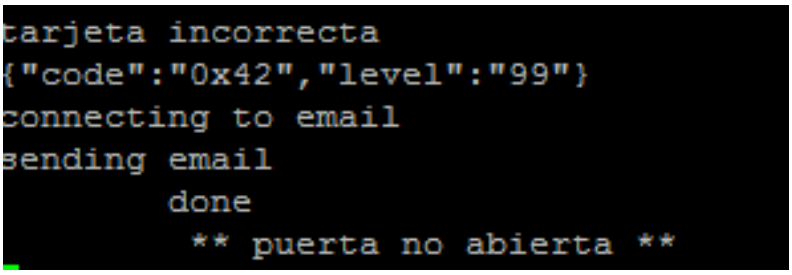


Fig. 4.3.2.2. Mensaje de detección de tarjeta RFID incorrecta

Fuente: Autores

4.3.2.3 Envío de mensaje SMS de alerta de intrusión.

Fig. 4.3.2.3. Mensaje SMS de aviso de intrusión

Fuente: Autores

4.3.2.4 Envío de mensaje de email adjuntado una foto de alerta de intrusión

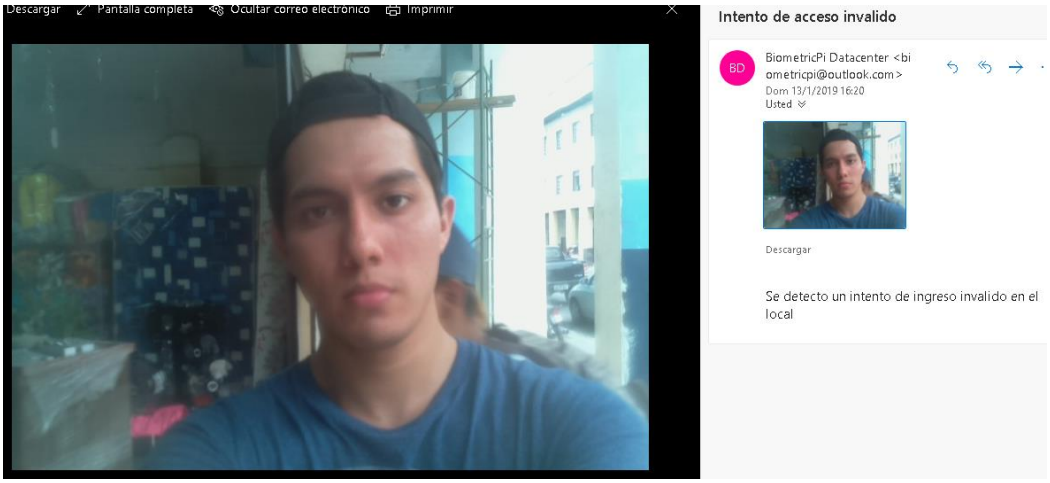


Fig. 4.3.2.4. Mensaje de correo electrónico con fotografía

Fuente: Autores

4.3.3 Resultados y comparaciones

En el procedimiento de chequeo del módulo de acceso magnético se desarrollaron varios tipos de revisiones exhaustivas en el cual comprenden de varios elementos como:

- Revisión del dispositivo Arduino
- Biblioteca en el Arduino
- Programación en el Arduino

Lo cual antes de proceder con la instalación final de la plataforma se revisa la primera opción la que da por resultado de que el dispositivo está en perfecto estado el cuándo dio por consiguiente realizar el chequeo del llamado de las bibliotecas del módulo magnético y la revisión de la programación la cual dio como resultado un trabajo exitoso y eficiente

4.4 Resultados finales

4.4.1 Modulo principal

Como resultado del proyecto de titulación se obtuvo el módulo principal en el cual se presenta las funciones principales de la plataforma donde se encuentra conformado por el control de acceso y la subida los datos a la aplicación web.

4.4.2 Control y visualización remota

La parte del control de usuario y gestión remota se lo puede realizar por medio de nuestro aplicativo web creada por nosotros y se encuentra desarrollada por un marco de trabajo vuejs, el entorno de javascript nodejs, la base de datos mysql y la programación para la visualización en Python. Esta aplicación se podrá acceder por medio de cualquier dispositivo que posea conexión a internet, esta puede estar conectada dentro de la misma red local como conectada a una diferente red en cualquier parte del mundo.

Al momento de querer entrar a nuestro aplicativo web se debe de digitar la dirección web estable ya configurada dentro de nuestro dispositivo con su respectivo número de puerto que se configura dentro del dispositivo, ubicado en la programación, y en la configuración del router del local comercial.

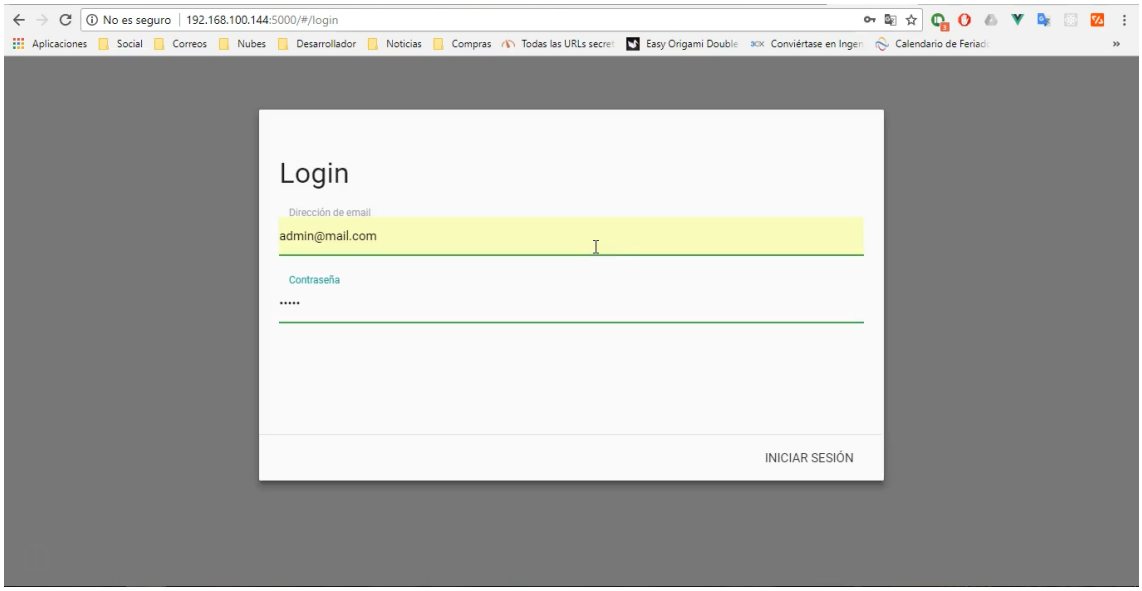
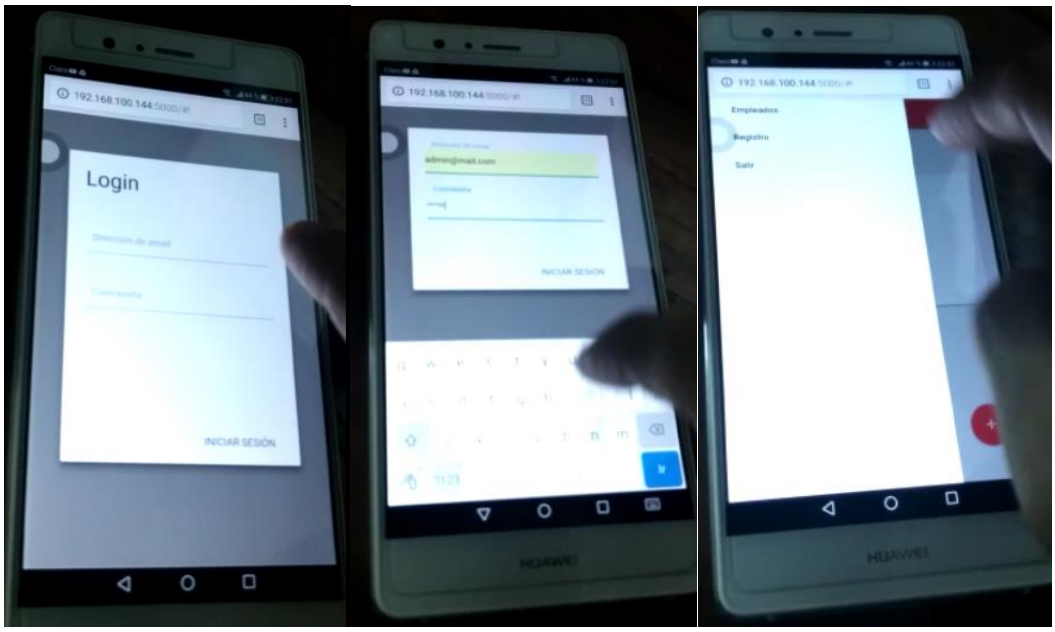


Fig. 4.4.2.1. Visualización del aplicativo web.

Fuente: Los Autores

Se puede visualizar mediante dispositivo fijos como móviles, ya que el aplicativo web constituye con las características de ser usable por la razón de ser de fácil uso, por intuición y accesible al poder visualizarse en diferentes dispositivos móviles o fijos sin tomar importancia al tamaño de la pantalla.



/Fig. 4.4.2.2. Visualización del aplicativo web en dispositivos móviles.

Fuente: Los Autores

Al acceder pedirá un email de usuario con su respectiva contraseña, la cual será única y solo la poseerá la persona a cargo de la vigilancia del interés del local.

4.4.3 Control de acceso del local comercial

El control de acceso se obtuvo un eficiente resultado ya que por medio de una cerradura electromagnética se permitió el control de entrada de la apertura del local como en el cierre del local, el cual está relacionado con la conexión directa con el dispositivo principal de la plataforma.

Fig.4.4.3.1. Sistema de apertura.

Fuente: Autores

El control de acceso está relacionado con la comparación de usuario dentro de la base de datos.

Las dos puertas corredoras son los objetos principales con el cual los empleados podrán acceder al local, la cual posee el contacto magnético para la toma de fotos de ingreso y de salida del local.

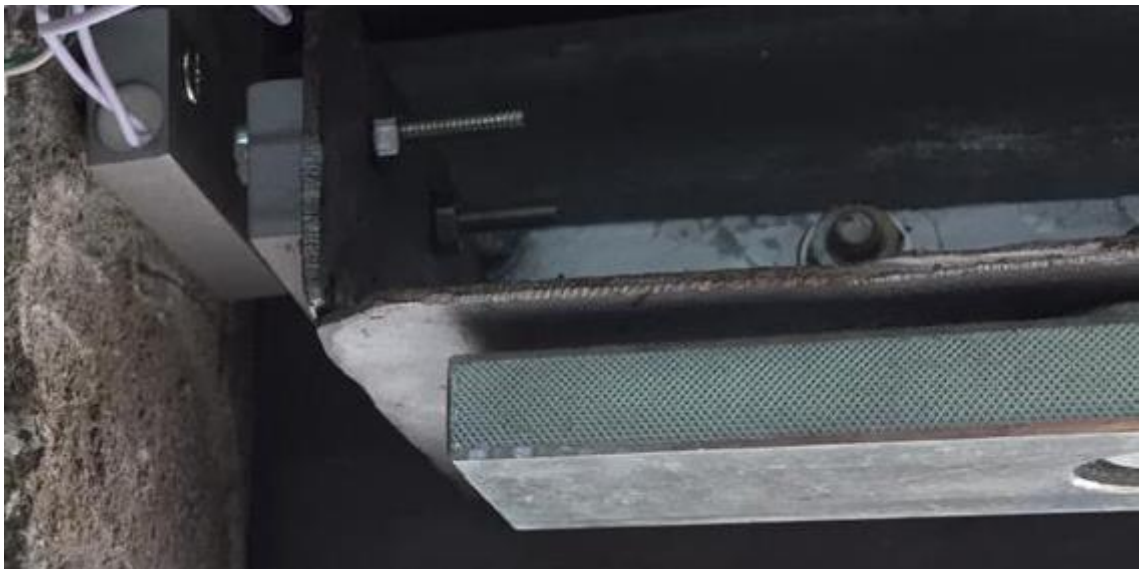


Fig.4.4.3.2. Contacto magnético de la parte superior.

Fuente: Autores

4.4.4 Vigilancia

La parte de la supervisión remota se obtuvo un resultado exitoso la cual consta de la visualización de las imágenes tomadas por segundo en tiempo real realizada por la cámara especial desarrollada por la empresa Raspberry Pi la cual esta sincronizada por un contacto magnético.

La cámara principal se encuentra ubicada en la parte principal del local comercial la cual toma fotos al instante para ser subidas al aplicativo web y la cámara secundaria de aviso de intruso se encuentra ubicada en el interfaz de entrada.

4.4.5 Acceso Biométrico / Magnético

En la entrada principal del local comercial se encuentra instalado el dispositivo de acceso biométrico y magnético para el ingreso de los empleados el cual se encuentra diseñado dentro de un dispositivo realizado por acrílico con los cables dentro y elaborado en PBC circuito impreso para que no haya interferencia o algún tipo de incidencia relacionada con el cableado.

CONCLUSIONES

Al haber implementado el proyecto de titulación se pudo percibir un gran éxito en la parte del desarrollo de la plataforma y en el incremento de la seguridad del local comercial, lo cual se llegó a la conclusión de que la plataforma fue instalada exitosamente sin problemas de funcionamiento.

Se pudo observar una gran eficiencia en cuanto a conocimientos adquiridos durante el periodo de aprendizaje en la carrera de Ingeniería Electrónica con mención en Telecomunicaciones de la Universidad Politécnica Salesiana, ya que se ha usado como el marco fundamental para la elaboración de la plataforma de Acceso Biométrico y Magnético para el diseño del sistema de seguridad con electrónica como los dispositivos de adquisición de datos, contactos magnéticos, dispositivos de control electromecánicos y el dispositivo Arduino y con telecomunicaciones en la configuración de routers, mapeo de puertos, configuración DMZ y asignación de dirección ip específicos para la plataforma.

También se pudo notar un gran apoyo en la parte de programación, conocimientos que fueron adquiridos fuera del entorno estudiantil universitario tales como la plataforma de desarrollo de páginas web, creación de base de datos, creación de un servidor web con apache, programación en marco de trabajo Framework Vuejs, configuración con el lenguaje de programación de comunicación cliente servidor como Nodejs, la creación de una base de datos dentro de un servidor web como MySQL y el entorno de programación de Python para la interacción con los demás dispositivos, todos los conocimientos adquiridos fueron de fundamentación para la realización de base de datos y el desarrollo del aplicativo web, así como la adquisición, transmisión y manejo de información de forma exitosa con los dispositivos Raspberry Pi.

En este proyecto de titulación existe un establecimiento seguro en la parte de la comunicación entre el sistema de seguridad implementado en el local comercial y el dueño del mismo, ya que se pueden enviar mensajes de texto y un mensaje de email con una foto tomada cuando se detecte una intromisión al local, además se suben las fotos tomadas cuando el acceso es autorizado y se registra automáticamente en el aplicativo web. Esto demuestra que el trabajo organizado es creativo y dinámico.

Se concluyó como resultado que las pruebas realizadas en el desarrollo de la plataforma fueron un principal aporte para la mejora de la misma, para el desarrollo de la plataforma en el software y la culminación total de la plataforma con resultados exitosos y eficiente.

RECOMENDACIONES

Las recomendaciones más importantes que se debe tener presente es la forma de manejo de los dispositivos Raspberry Pi ya que son miniordenadores con suplementos sensibles los cuales deben de estar protegidos en todo momento y ser usados con herramientas especiales; tener presente los datos de seguridad que son proporcionados por los desarrolladores como el porcentaje de humedad que tienen la capacidad de soportar para evitar el deterioro del equipo.

También se debe de tomar en cuenta acerca de la corriente para la alimentación de entrada, siempre tener presente cuales son las fuentes de alimentación correctas de cada equipo para así no generar un cortocircuito que pueda producir la quema de los elementos internos.

Además, tener precaución por la fragilidad de los módulos biométricos y magnéticos ya que cada uno posee un escáner para la lectura del código sea dactilar como magnético, los cuales son sensibles a los golpes, siempre tener en cuenta que estos dispositivos deben ser tratados con responsabilidad para el mantenimiento de estos equipos.

Se recomienda que al momento de elegir módulos con microcontroladores como el Arduino verificar que tipo y versión es el que se obtiene, ya que para la versión de Arduino Pro Mini existen dos versiones que son distinguidas por el voltaje de alimentación.

Cuando se procede con la instalación tanto eléctrica como electrónica siempre tener precaución con ciertas conexiones por lo que se aconseja tenerlos empatas de una manera segura y ser probados en varias ocasiones para seguridad de comunicación entre dispositivos.

En el caso de uso de cámaras de seguridad, realizar un análisis a profundidad del tipo de cámaras que se puede utilizar debido a que ciertas cámaras tienen como necesidad de funcionamiento la instalación de drivers. Existen cámaras diseñadas específicamente para módulos electrónicos las cuales son compatibles y se pueden utilizar sin necesidad de herramientas adicionales.

CRONOGRAMA

	AÑO 1												AÑO 2			
	MES															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1.COMPRA DE EQUIPOS																
1.1 Cotizacion de los elementos a usar																
1.2 Compra de equipos e implementos																
1.3 Pruebas de funcionamiento																
2.DISEÑO DE LA PLACA ELECTRONICA																
2.1 Construcción de la parte electrónica																
2.2Implementacion del escáner RFID																
2.3Implementacion del escáner biométrico																
3.DESARROLLO DEL SOFTWARE																
3.1 Programación en Raspberry																
3.2 Programación en Arduino																
3.3 Diseño del servidor web y base de datos																
4.DISEÑO DE LA IMPRESIÓN 3D																
4.1 Diseño e impresión de la estructura física de la parte electrónica del equipo																
4.2 Diseño e impresión del case de los sensores																
5. DESARROLLO DEL PROTOTIPO FINAL																
5.1 Verificación de funcionamiento																
5.2 Realización de correcciones y pruebas																
6.DESARROLLO DE MEJORAS																
6.1 Implementacion de sistema de mensaje SMS																
6.2 Implementacion de sistema de mensaje email																
6.3 Implementacion de sistema de capture de fotos de intrusión																

PRESUPUESTO

Descripción	Cantidad	Precio	Subtotal
Fuente de 12 voltios 2 Ampereos	2	$ 20,00	$ 40,00
Tarjeta microSD 8Gb clase 4	2	$ 18,00	$ 36,00
Case para Raspberry Pi	1	$ 35,00	$ 35,00
Camara Raspbery Pi Zero	1	$ 30,00	$ 30,00
Raspbery Pi Zero	1	$ 60,00	$ 60,00
Raspbery Pi 3	1	$ 70,00	$ 70,00
Cable de red	2	$ 15,00	$ 30,00
Switch de 8 puertos TPLink	1	$ 40,00	$ 40,00
Adaptador mini ESB-Ethernet	1	$ 20,00	$ 20,00
Modulo Wi-Fi Esp8266 V3	1	$ 20,00	$ 20,00
Arduino pro mini	1	$ 15,00	$ 15,00
Modulo quemador Arduino Pro Mini (USB toTLL)	1	$ 7,00	$ 7,00
Impresión 3D de la caja del prototipo	1	$ 90,00	$ 90,00
Capacitación de cableado estructurado	2	$ 120,00	$ 240,00
Lector de tarjetas	1	$ 40,00	$ 40,00
Servidor web en internet	1	$ 90,00	$ 90.00
Sensor biométrico	2	$ 50,00	$ 100.00
Modulo Relay 5V1 Canal	1	$ 7,00	$ 7,00
Cerradura Magnetica 300lb	1	$ 60,00	$ 60,00
Contacto Magnetico MC270-S45	1	$ 10,00	$ 10,00
UPS	1	$ 60,00	$ 60,00
Gastos varios	1	$200,00	$ 200,00
		Total	**$ 1300,00**

REFERENCIAS

Calderon, A. (2016, April 5). Disminuyen los robos a locales comerciales. *Expreso*, p. 5. Retrieved from http://www.expreso.ec/actualidad/disminuyen-los-robos-a-locales-comerciales-DY230151

Wallace, M. R. S. (2012). *Getting Started with Raspberry Pi*. Maker Media, Inc. Retrieved from https://books.google.es/books?hl=es&lr=&id=xYhMlilTwC4C&oi=fnd&pg=PR2&dq=raspberry+pi&ots=W46fiDnau4&sig=doTOmuD2Pbjm6DPhf87ceSTKmmk#v=onepage&q=raspberry pi&f=false

https://disenowebakus.net/. (2013). Estándares Web W3C - Qué son, cómo funcionan y para qué sirven. *Diseño de Páginas Web, Sitios de Internet y Posicionamiento SEO Akus.Net*. Retrieved from https://disenowebakus.net/estandares-web.php

Alvarez, M. A. (2003). Qué es Python. Retrieved July 24, 2018, from https://desarrolloweb.com/articulos/1325.php

Grados, J. G. (n.d.). ¿Qué es JavaScript? Retrieved July 24, 2018, from https://devcode.la/blog/que-es-javascript/

Eguíluz Pérez, J. (n.d.). Introducción a JavaScript. Retrieved from www.librosweb.es

Rosa, J. M. (2017). ¿Qué es Vue.js? | OpenWebinars.net. Retrieved July 24, 2018, from https://openwebinars.net/blog/que-es-vuejs/

Macrae, C. (n.d.). *Vue.js: Up and Running: Building Accessible and Performant Web Apps.*

NetConsulting. (2015). Node.js: ¿Qué es y para que sirve NodeJS? Retrieved July 24, 2018, from https://www.netconsulting.es/blog/nodejs/

Fossati, M. (2014). Todo sobre MySQL: Libro ideal para ingresar en el mundo de la base de datos ... - Natsys - Google Libros. Retrieved July 24, 2018, from https://books.google.com.ec/books?id=GS3kAgAAQBAJ&printsec=frontcover&dq=mysql&hl=es&sa=X&ved=0ahUKEwjlxMPZnrncAhVRq1kKHStSB6EQ6AEIMzAC#v=onepage&q=mysql&f=false

tinchorton. (2014). Arduino Pro Mini | Blog de PatagoniaTec Electronica. Retrieved September 17, 2018, from https://saber.patagoniatec.com/2014/07/arduino-pro-micro-5v-atmega32u4-micro-leonardo-pro-mini-arduino-argentina-ptec/

Arduino cc. (n.d.). Arduino Pro Mini. Retrieved September 17, 2018, from https://store.arduino.cc/usa/arduino-pro-mini

ANEXO

FOTOS DEL LOCAL COMERCIAL “CREDILIT”

Anexo 1. Estado del local comercial antes de iniciar (Vista exterior)

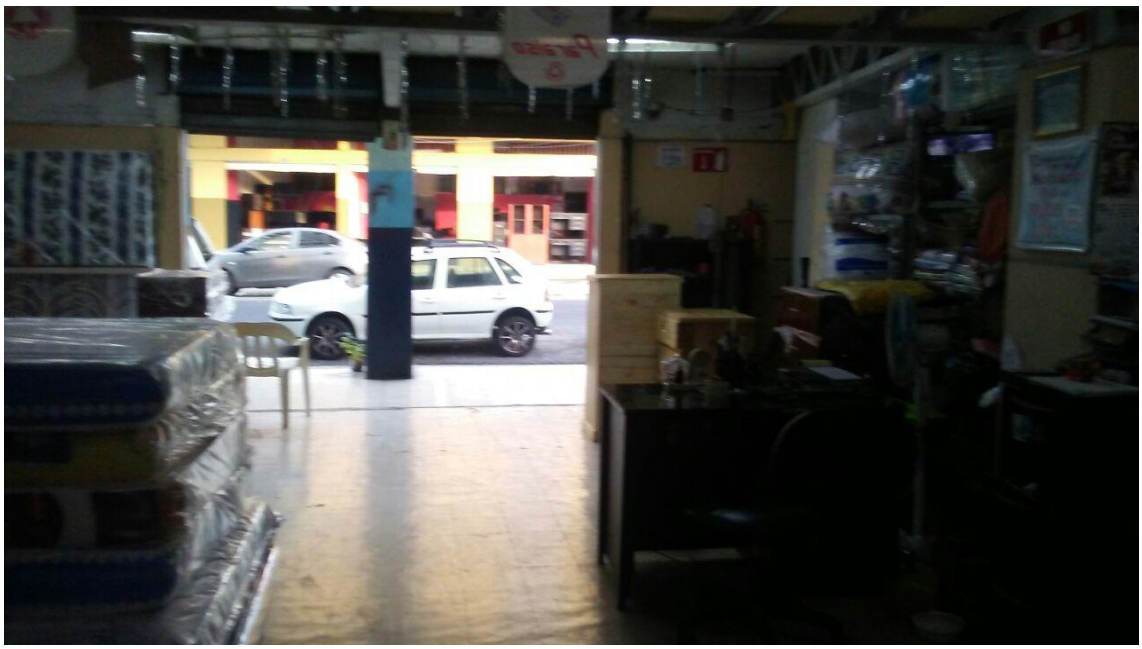
Fuentes: Los Autores

Anexo 2. Estado del local comercial antes de iniciar (Vista Interior 2)

Fuentes: Los Autores

Anexo 3. Estado del local comercial antes de iniciar (Vista interior 3)

Fuentes: Los Autores

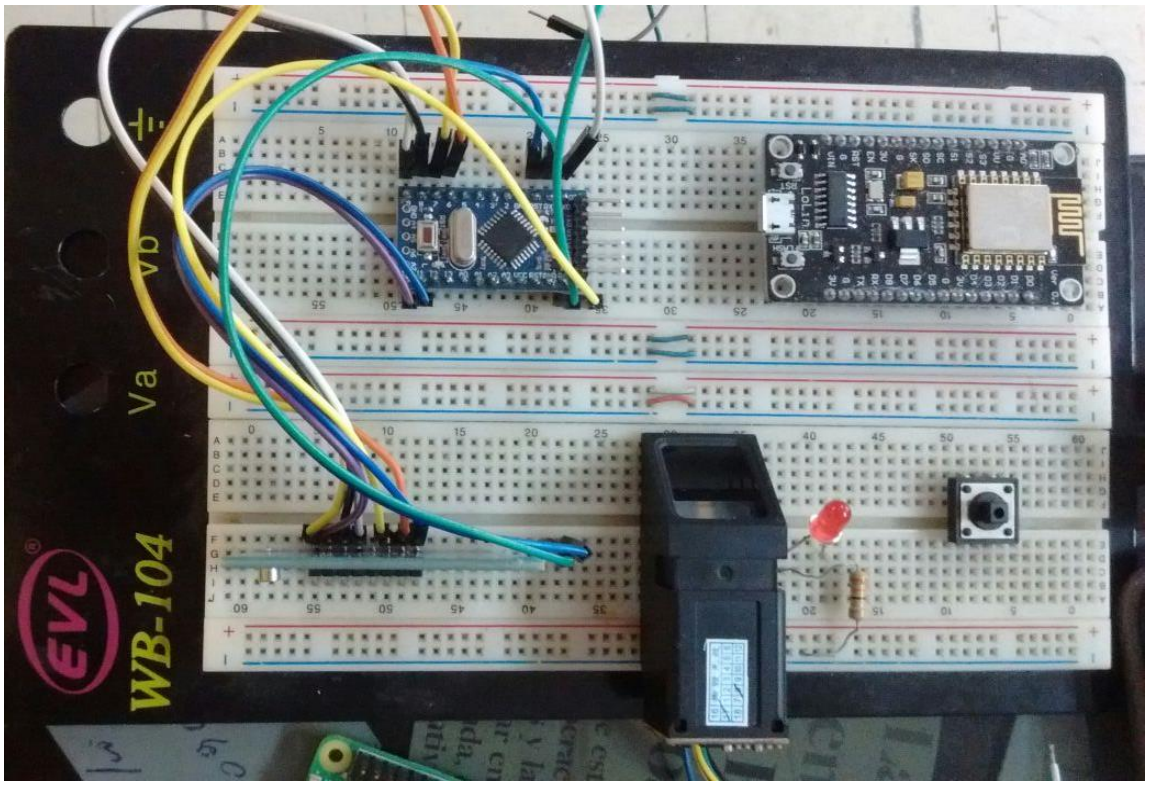


Anexo 4. Estado del local comercial antes de iniciar (Vista hacia el Exterior)

Fuentes: Los Autores

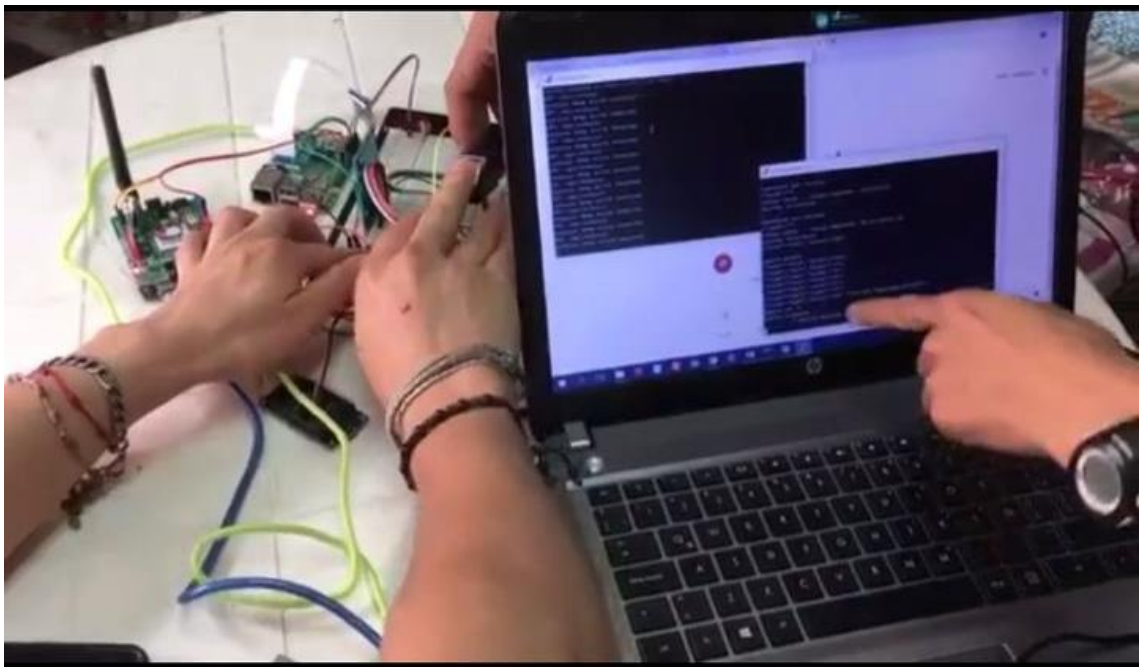
ANEXO

DESARROLLO DEL CODIGO DEL SENSOR BIOMÉTRICO



Anexo 1. Circuito de prueba del sensor Biométrico

Fuentes: Los Autores



Anexo 2. Prueba de funcionamiento del sensor biométrico a través de python

Fuentes: Los Autores

Anexo 3. Configuración de prueba del circuito principal

Fuentes: Los Autores

Anexo 4. Revisión de conexión del Arduino con la red

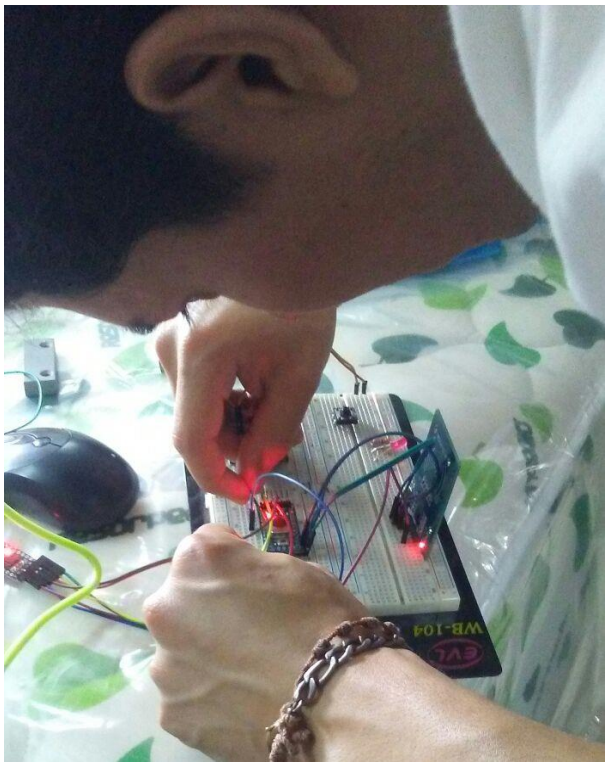
Fuentes: Los Autores

ANEXO

DESARROLLO DEL CODIGO DEL SENSOR MAGNÉTICO

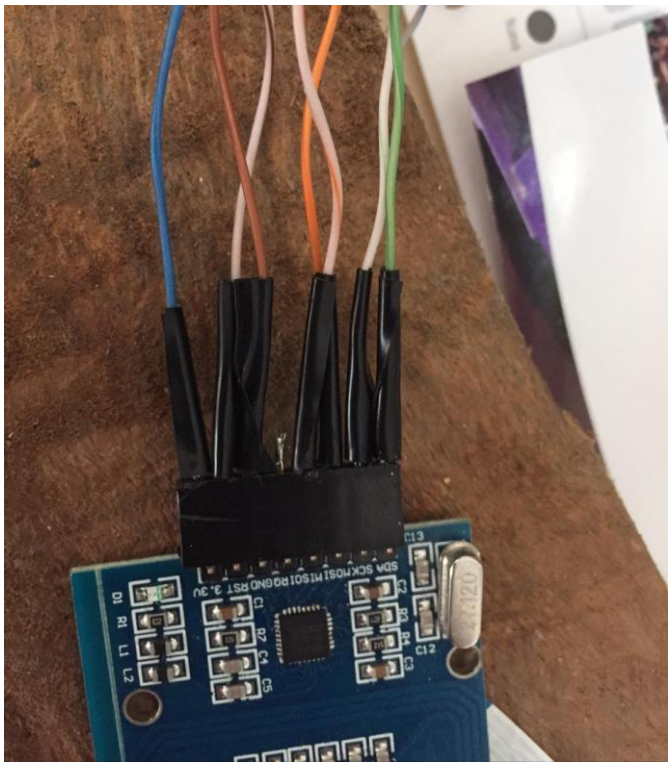
Anexo 1. Prueba de error del dispositivo Biométrico

Fuentes: Los Autores



Anexo 2. Prueba de funcionamiento de prototipo realizado en protoboard

Fuentes: Los Autores



Anexo 3. Conexión y prueba de funcionamiento del sensor RFID

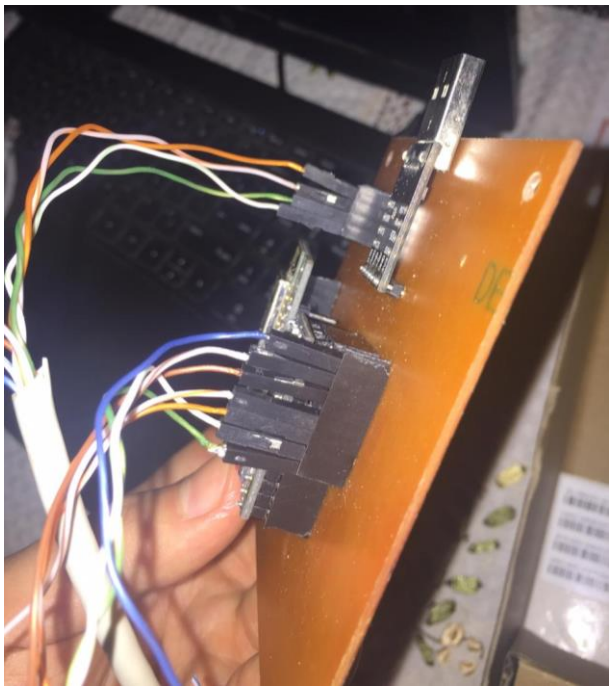
Fuentes: Los Autores

Anexo 4. Circuito de prueba del sensor RFID y el Biométrico

Fuentes: Los Autores

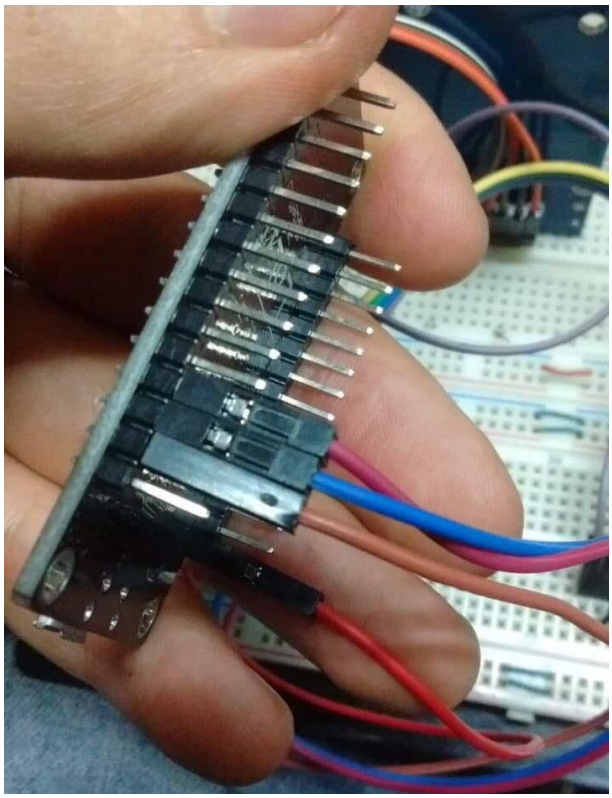
ANEXO

DESARROLLO DEL CIRCUITO IMPRESO Y MODULO WIFI



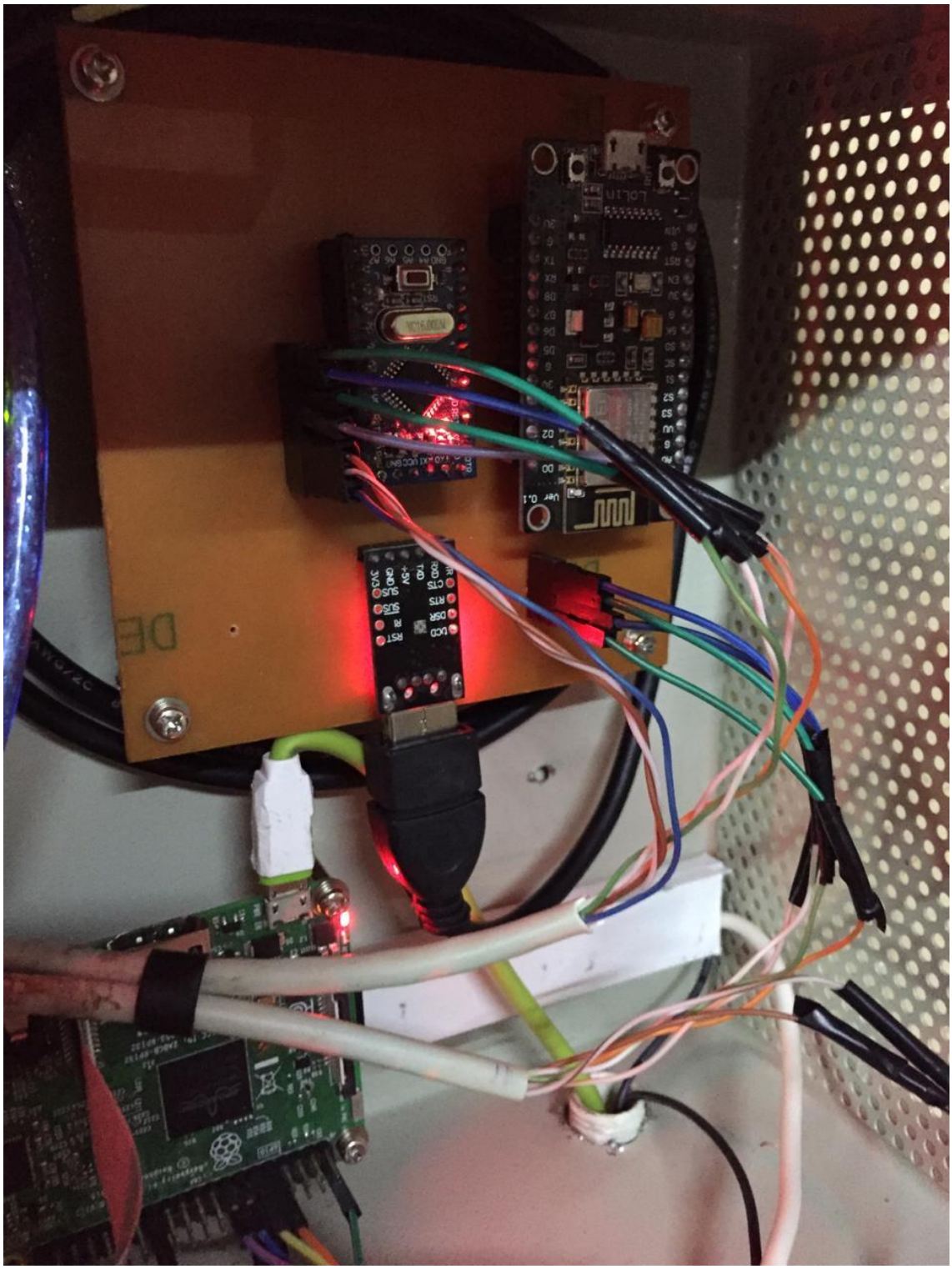
Anexo 1. Diseño del circuito impreso y conexiones directas hacia los sensores

Fuentes: Los Autores



Anexo 2. Conexión y Prueba de funcionalidad del módulo wifi

Fuentes: Los Autores

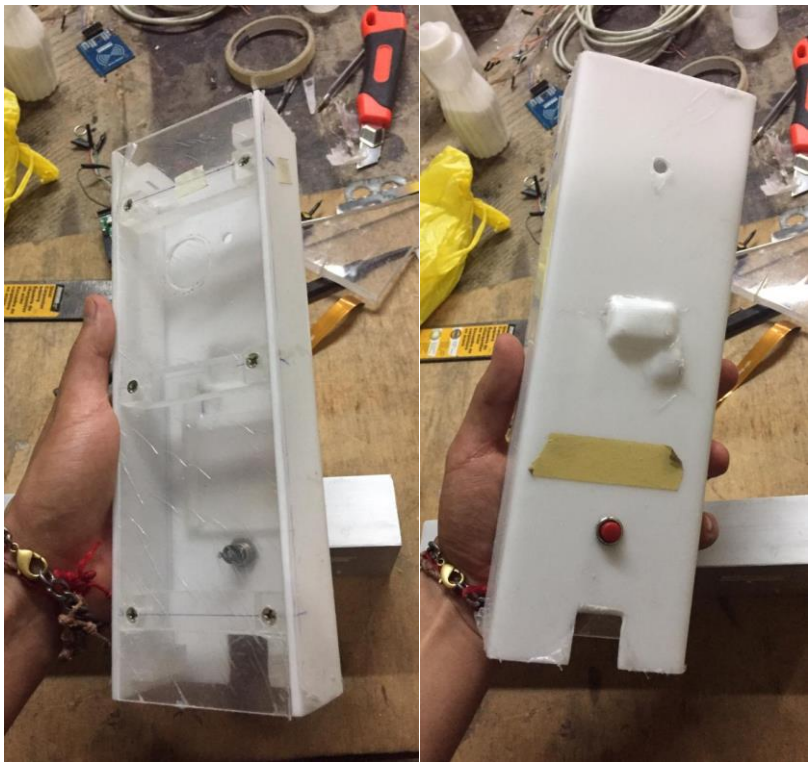


Anexo 3. Instalación y Prueba de los módulos usados

Fuentes: Los Autores

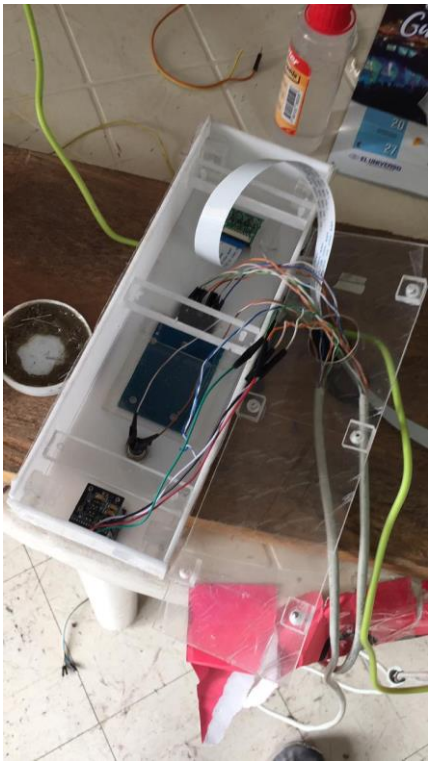
ANEXO

DISEÑO DE LA CARCAZA DEL CONTROL DE ACCESO



Anexo 1. Diseño y elaboración de carcaza para uso final

Fuentes: Los Autores

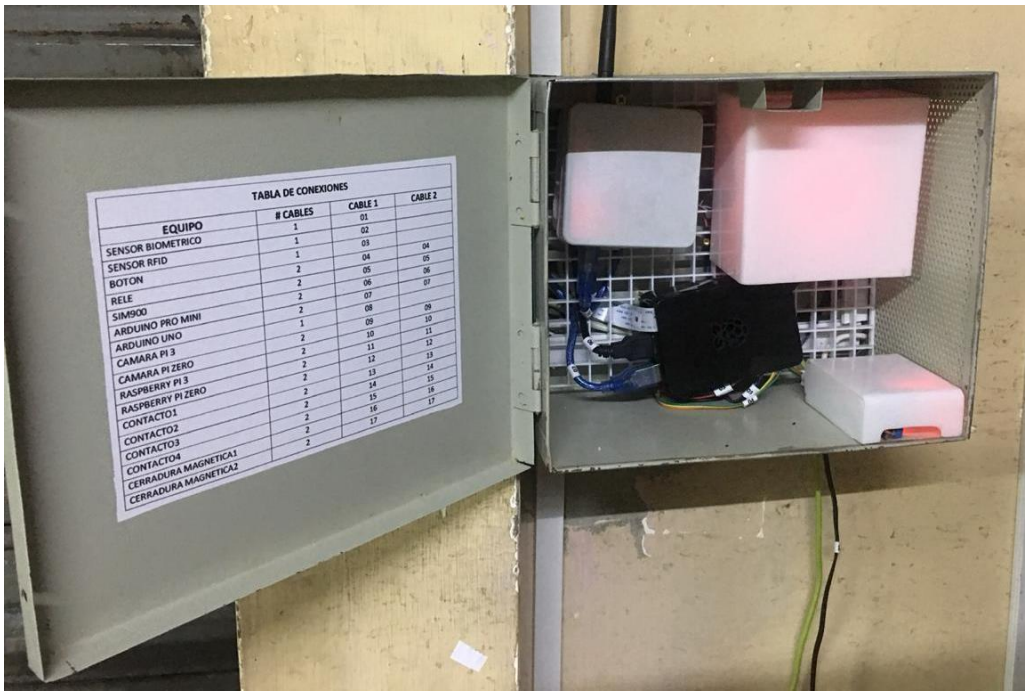


Anexo 2. Carcaza con los sensores ya integrados para la instalación.

Fuentes: Los Autores

Anexo 3. Análisis de instalaciones eléctricas y electrónicas

Fuentes: Los Autores



Anexo 4. Equipos instalados en el rack del local comercial.

Fuentes: Los Autores

TABLA DE CONEXIONES			
EQUIPO	**# CABLES**	**CABLE 1**	**CABLE 2**
SENSOR BIOMETRICO	1	01	
SENSOR RFID	1	02	
BOTON	1	03	
RELE	2	04	04
SIM900	2	05	05
ARDUINO PRO MINI	2	06	06
ARDUINO UNO	2	07	07
CAMARA PI 3	1	08	
CAMARA PI ZERO	2	09	09
RASPBERRY PI 3	2	10	10
RASPBERRY PI ZERO	2	11	11
CONTACTO1	2	12	12
CONTACTO2	2	13	13
CONTACTO3	2	14	14
CONTACTO4	2	15	15
CERRADURA MAGNETICA1	2	16	16
CERRADURA MAGNETICA2	2	17	17

Anexo 5. Tabla de conexiones de los equipos.

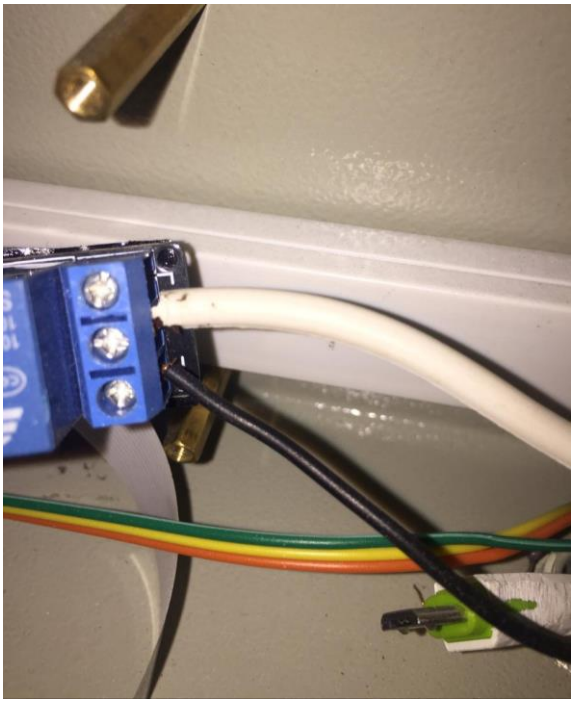
Fuentes: Los Autores

ANEXO

INSTALACION ELECTRONICO DE LA CERRADURA ELECTROMAGNETICA

Anexo 1. Conexión de alimentación y señal de la cerradura electromagnética

Fuentes: Los Autores

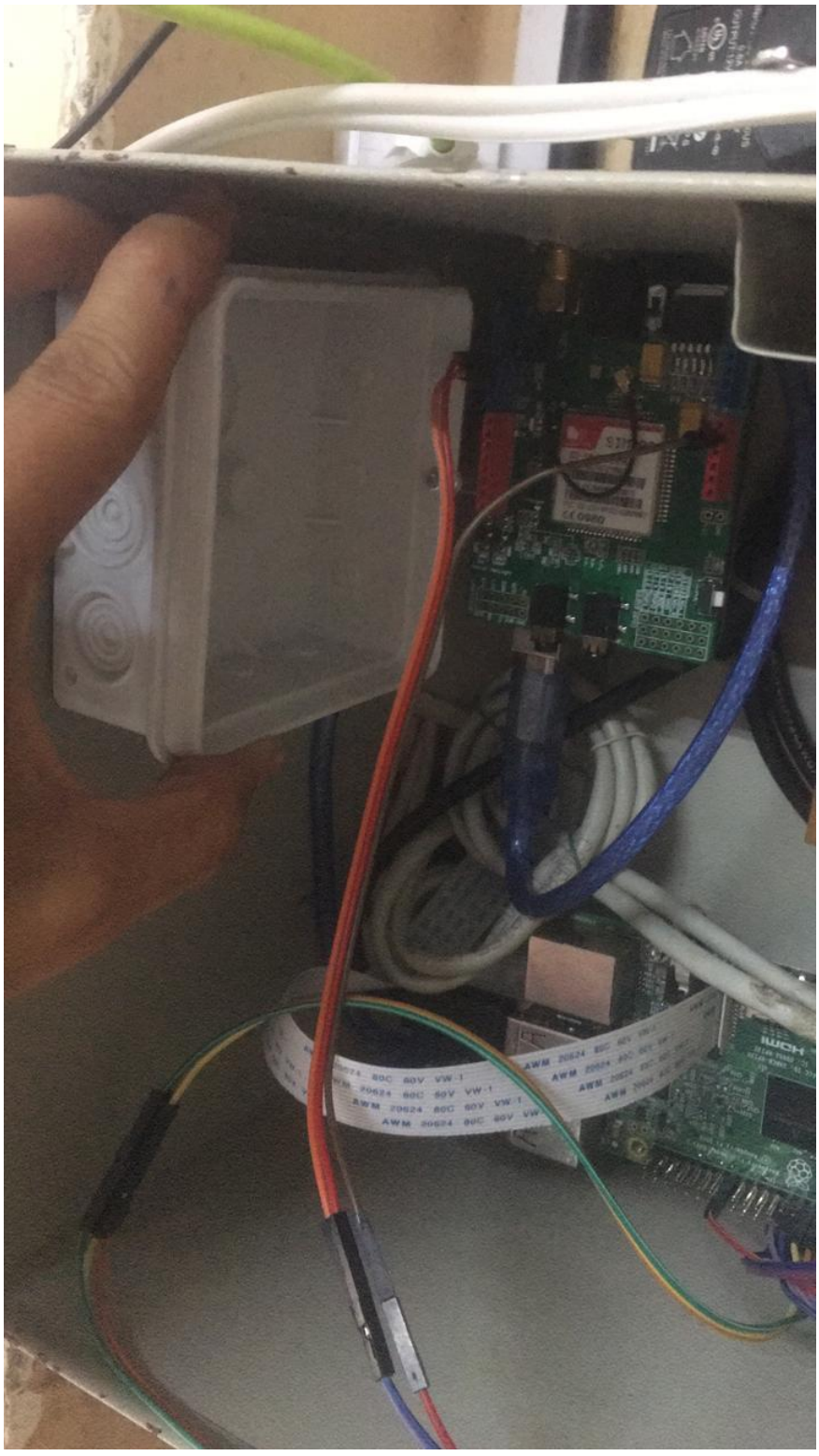


Anexo 2. Prueba de funcionamiento y conexión de alimentación

Fuentes: Los Autores

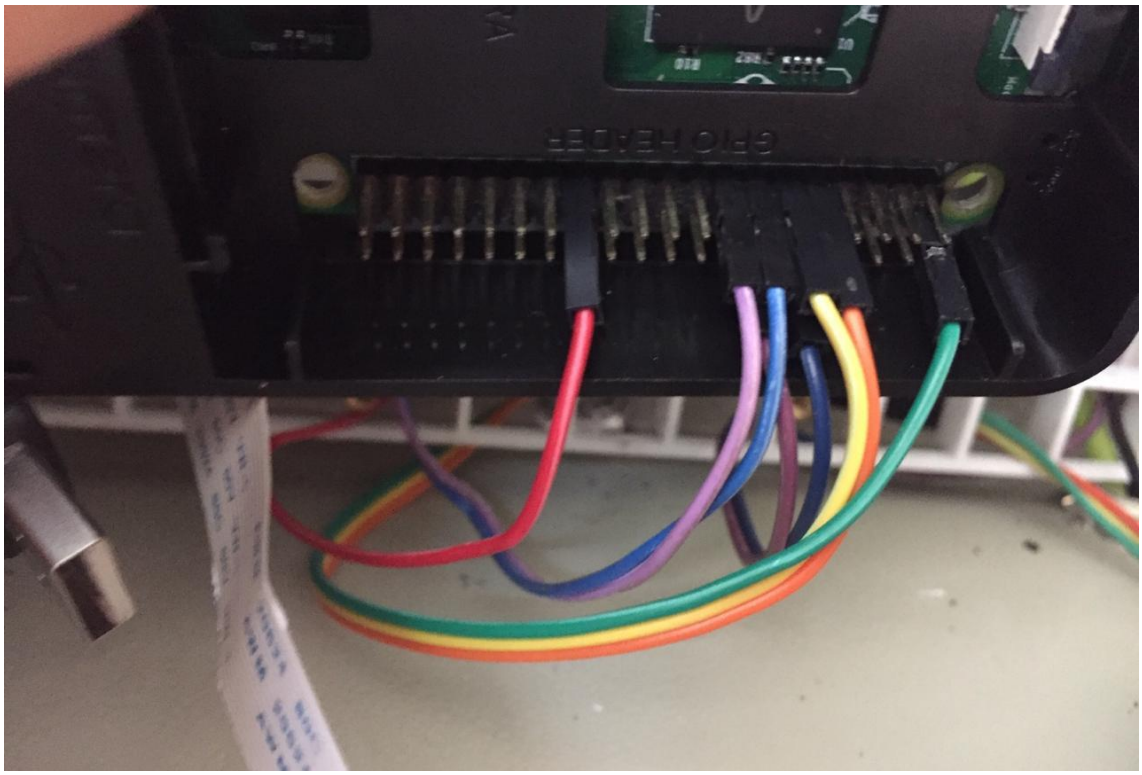
ANEXO

DESARROLLO DE LOS APLICATIVOS WEB Y CODIGOS DE PROGRAMACION EN LA RASPBERRY PI 3 Y PI ZERO



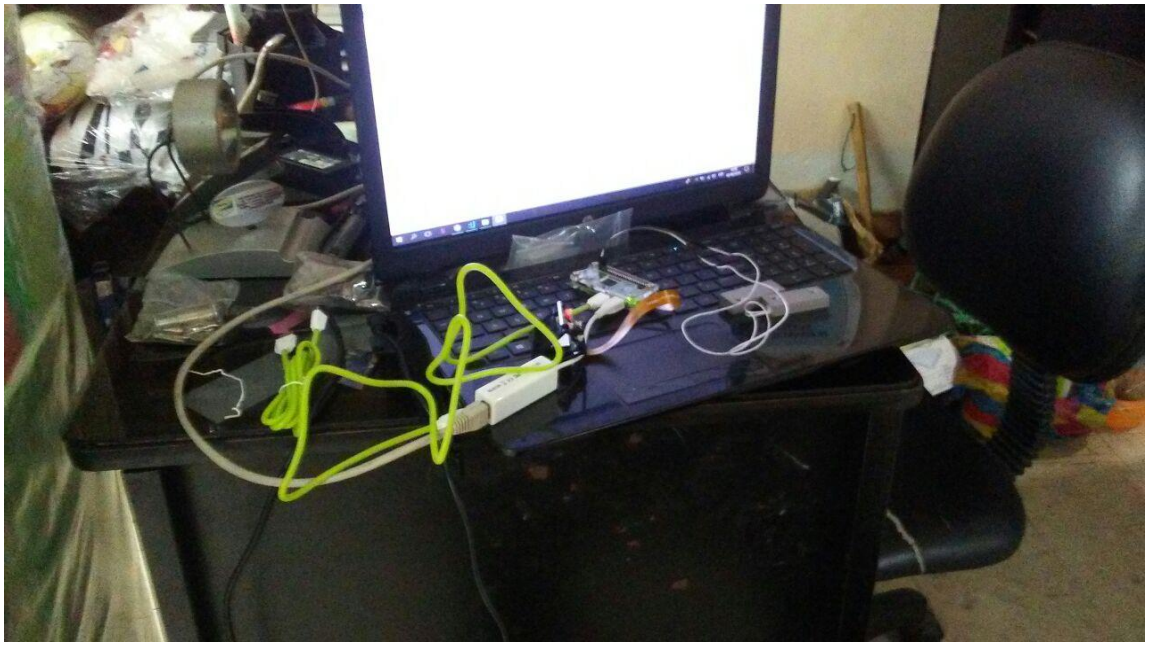
Anexo 1. Prueba de envío de mensaje de texto.

Fuentes: Los Autores



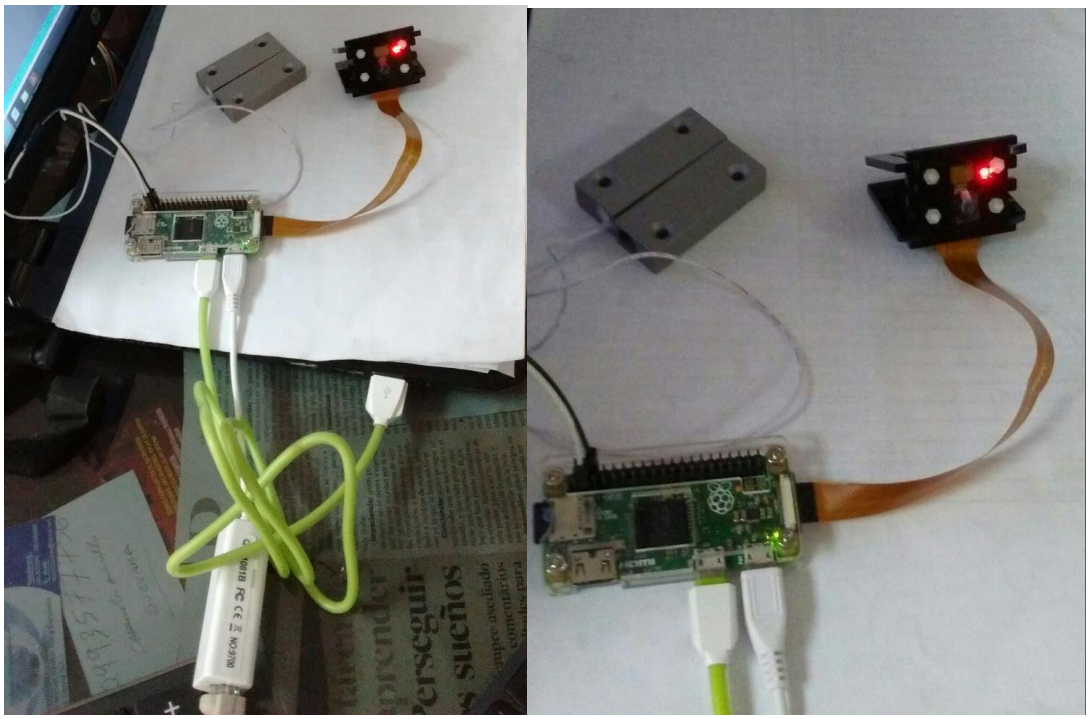
Anexo 2. Conexión de la Raspberry Pi3

Fuentes: Los Autores



Anexo 3. Diseño del circuito de la Raspberry Pi Zero

Fuentes: Los Autores



Anexo 4. Prueba de conexión y funcionamiento de la Cámara y la Raspberry Pi Zero

Fuentes: Los Autores

Anexo 5. Creación de la programación en Python con pines especifico.

Fuentes: Los Autores

Anexo 6. Revisión de la programación en Python para desarrollarlo dinámicamente

Fuentes: Los Autores

Anexo 7. Diseño y compilación del programa de comunicación de la raspberry pi 3 con el Arduino.

Fuentes: Los Autores

Anexo 8. Diseño del aplicativo web de la plataforma

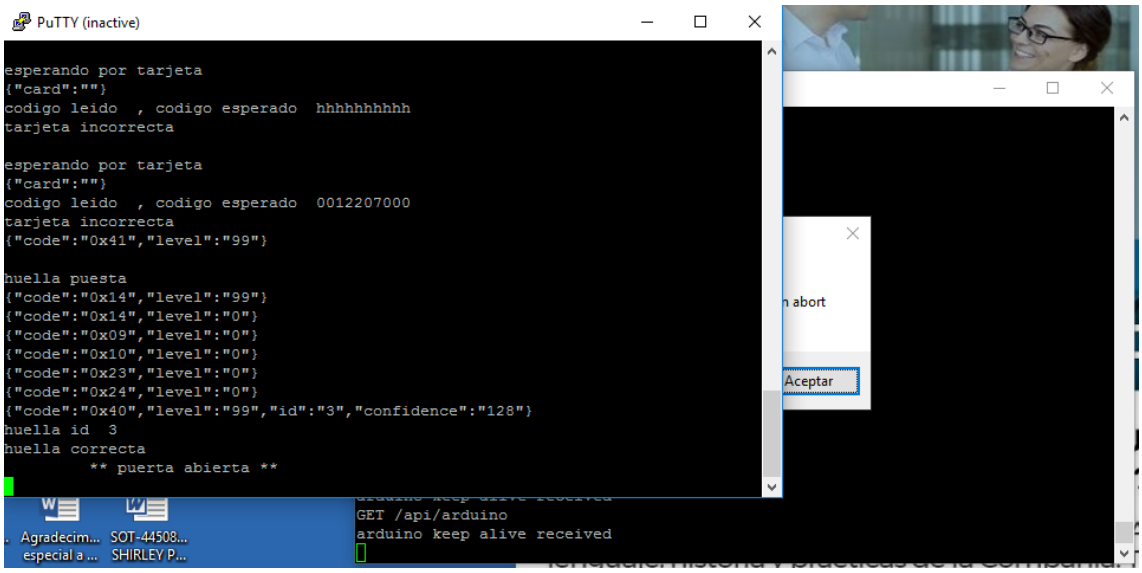
Fuentes: Los Autores

Anexo 9. Revisión y prueba del circuito con el aplicativo web

Fuentes: Los Autores

Anexo 10. Prueba de funcionamiento de los aplicativos web.

Fuentes: Los Autores

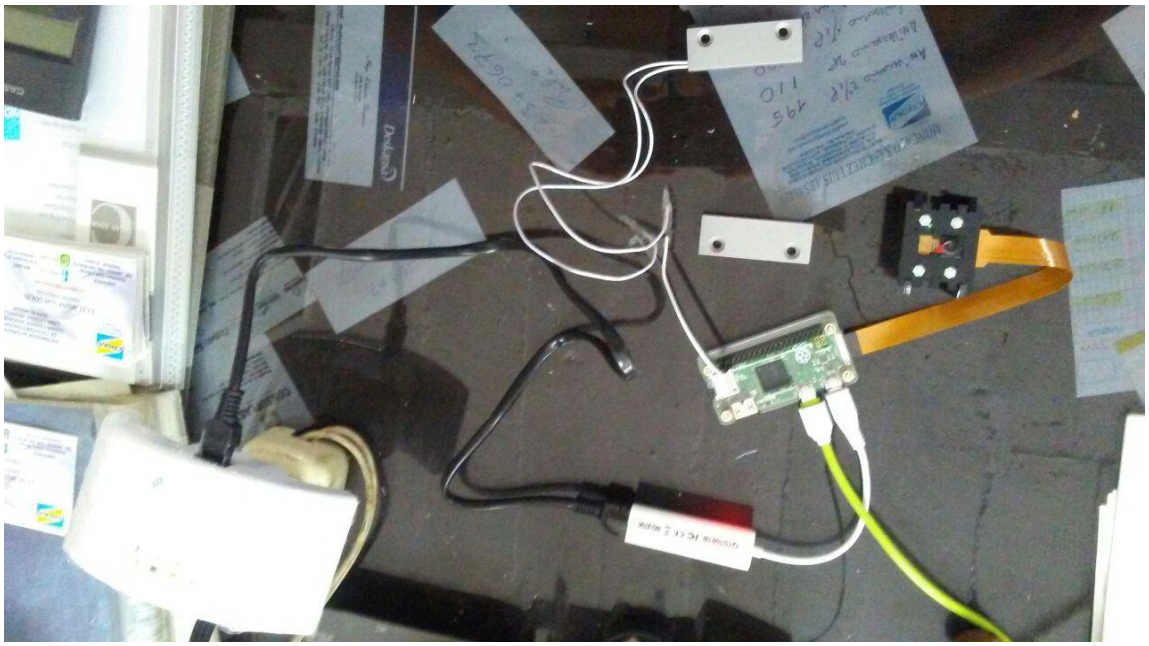


Anexo 11. Comandos de ejecución de Prueba en Python

Fuentes: Los Autores

Anexo 12. Configuración de la red del local comercial

Fuentes: Los Autores

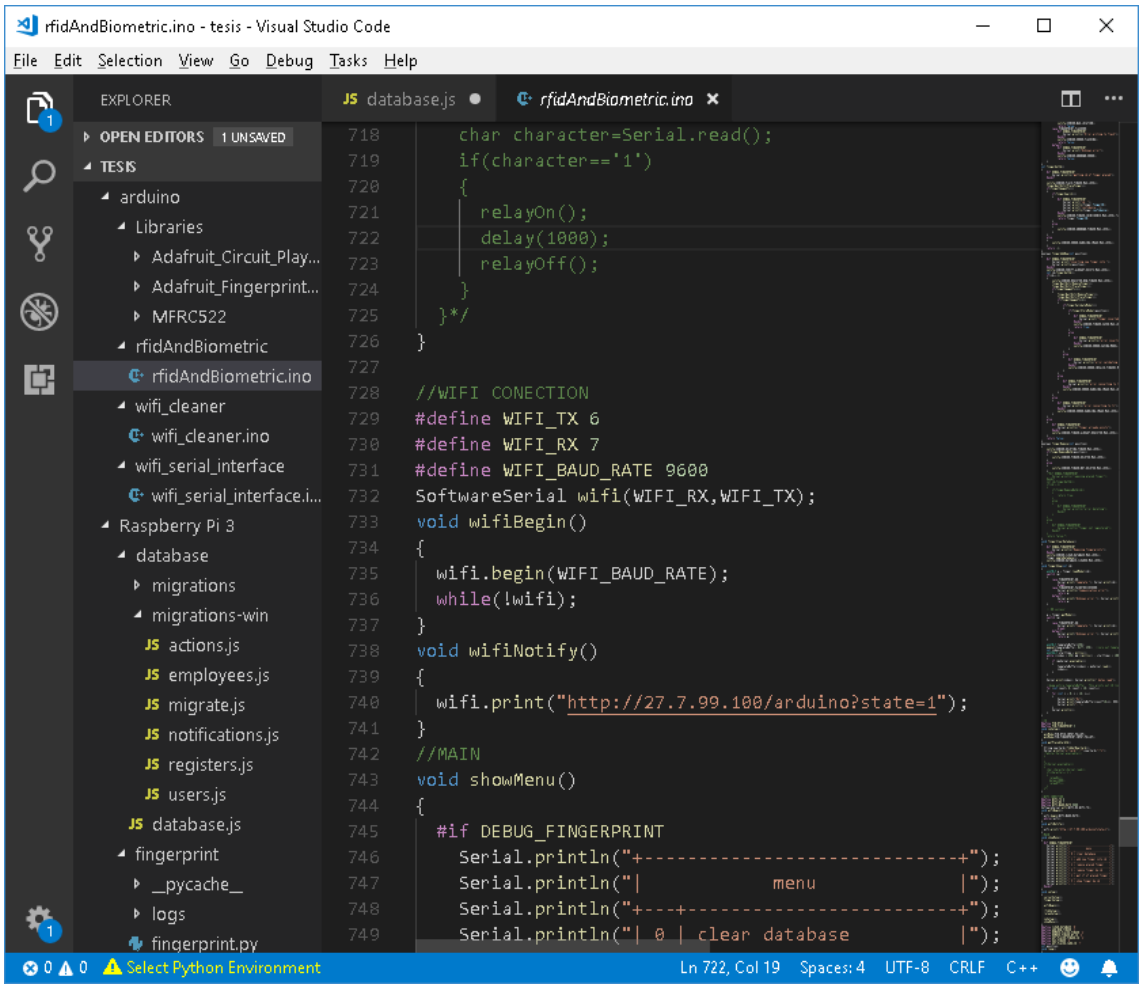


Anexo 13. Prueba de conexión de un duplicador WIFI

Fuentes: Los Autores

ANEXO

DESARROLLO DEL CODIGO DEL MODULO ARDUINO PRO MINI



Anexo 1. Desarrollo de la programación de la Plataforma

Fuentes: Los Autores

```
//RFID CARD
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);
void rfidSetup()
{
    SPI.begin();
    mfrc522.PCD_Init();
}
String rfidGetNearCard()
{
    if(!mfrc522.PICC_IsNewCardPresent())
    {
        return "";
```

```
    }
    if(!mfrc522.PICC_ReadCardSerial())
    {
        return "";
    }
    String content="";
    byte letter;
    for (byte i=0;i<mfrc522.uid.size;i++)
    {
        if(i==0)
        {
            content.concat(String(mfrc522.uid.uidByte[i]<0x10?"
0":""));
        }
        else
        {
            content.concat(String(mfrc522.uid.uidByte[i]<0x10?" 0":"
"));
        }

        content.concat(String(mfrc522.uid.uidByte[i],HEX));
    }
    content.toUpperCase();
    return content;
}
String rfidWaitForCard()
{
    String nearCard="";
    while(nearCard.equals(""))
    {
        nearCard=rfidGetNearCard();
    }
    return nearCard;
}

//RELAY
#define PIN_RELAY 8
void relaySetup()
{
    pinMode(PIN_RELAY,OUTPUT);
}
void relayOn()
{
    digitalWrite(PIN_RELAY,HIGH);
}
void relayOff()
{
    digitalWrite(PIN_RELAY,LOW);
}
```

```
//MAIN

#define DEBUG_FINGERPRINT false

//SERIAL
#define SERIAL_BAUD_RATE 9600
void serialSetup()
{
    Serial.begin(SERIAL_BAUD_RATE);
    while(!Serial);
}
//LEVELS
#define MAX_LEVEL 99

//GENERAL MESSAGES
#define SENSOR_CHECK "0x01"
#define SENSOR_READY "0x02"
#define SENSOR_NOT_FOUND "0x03"
#define SENSOR_GETTING_IMAGE "0x04"
#define SENSOR_IMAGE_TAKEN "0x05"
#define SENSOR_COMMUNICATION_ERROR "0x06"
#define SENSOR_IMAGE_ERROR "0x07"
#define SENSOR_UNKNOWN_ERROR "0x08"
#define SENSOR_TAKING_IMAGE "0x09"
#define SENSOR_IMAGE_CONVERTED "0x10"
#define SENSOR_IMAGE_TOO_MESSY "0x11"
#define SENSOR_FEATURE_FAIL "0x12"
#define SENSOR_IMAGE_INVALID "0x13"
#define SENSOR_PLACE_FINGER "0x14"
#define SENSOR_REMOVE_FINGER "0x15"
#define SENSOR_CREATING_MODEL "0x16"
#define SENSOR_FINGER_MATCHED "0x17"
#define SENSOR_FINGER_MISMATCH "0x18"
#define SENSOR_STORE_MODEL "0x19"
#define SENSOR_FINGER_STORED "0x20"
#define SENSOR_BAD_LOCATION "0x21"
#define SENSOR_ERROR_FLASHING "0x22"
#define SENSOR_FINGER_SEARCH "0x23"
#define SENSOR_FINGER_FOUND "0x24"
#define SENSOR_FINGER_NOT_FOUND "0x25"
#define SENSOR_DELETE_MODEL "0x26"
#define SENSOR_FINGER_DELETED "0x27"

//CLEAR DATABASE
#define SENSOR_CLEAR_DATABASE "0x28"
#define SENSOR_DATABASE_CLEARED "0x29"

//ADD NEW FINGER
```

```
#define SENSOR_VERIFY_ALREADY_EXISTS "0x30"
#define SENSOR_FINGER_ALREADY_REGISTER "0x31"
#define SENSOR_ERROR_HANDLING_IMAGE "0x32"
#define SENSOR_ERROR_INVALID_FINGERS "0x33"
#define SENSOR_ERROR_SAVING_MODEL "0x34"
#define SENSOR_FINGER_SAVED "0x35"
#define SENSOR_REGISTER_NEW_FINGER "0x36"

//REMOVE PLACED FINGER
#define SENSOR_DELETING_FINGER "0x37"
#define SENSOR_FINGER_NOT_DELETED "0x38"

//SEARCH FINGER
#define SENSOR_UNKNOWN_FINGER "0x39"
#define SENSOR_FINGER_COINCIDENCE "0x40"

//CHECK FINGER
#define SENSOR_FINGER_PLACED "0x41"
#define SENSOR_NO_FINGER_PLACED "0x42"

#define NOTIFICATION_INTERVAL 100
void notify(String code)
{
    notify(code,0);
}
void notify(String code,byte level)
{
    Serial.println("{\"code\":\""+code+"\",\"level\":\""+level+"\"}");
    //delay(NOTIFICATION_INTERVAL);
}

void notify(String code,byte level,byte id,int confidence)
{

Serial.println("{\"code\":\""+code+"\",\"level\":\""+level+"\",\"id\":\
""+id+"\",\"confidence\":\""+confidence+"\"}");
    //delay(NOTIFICATION_INTERVAL);
}
int serialGetNumer()
{
    #if DEBUG_FINGERPRINT
        Serial.println("get the database position");
    #endif
    while(!Serial.available());

    char hundredsChar=Serial.read();
    int hundreds=hundredsChar-48;
    delay(100);
```

```
    char tensChar=Serial.read();
    int tens=tensChar-48;
    delay(100);

    char unitsChar=Serial.read();
    int units=unitsChar-48;

    int position=hundreds*100+tens*10+units;

    #if DEBUG_FINGERPRINT
        Serial.println(position);
    #endif
    return position;
}

//FINGERPRINT
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
#define WIRE_GREEN 2
#define WIRE_WHITE 3
SoftwareSerial mySerial(WIRE_GREEN,WIRE_WHITE);
Adafruit_Fingerprint finger=Adafruit_Fingerprint(&mySerial);
#define FINGERPRINT_BAUD_RATE 57600
void fingerSetup()
{
    #if DEBUG_FINGERPRINT
        Serial.println("Setting up fingerprint sensor");
    #endif
    *notify(SENSOR_CHECK);
    finger.begin(FINGERPRINT_BAUD_RATE);

    boolean isFingerprintPresent=false;
    while(!isFingerprintPresent)
    {
        if(finger.verifyPassword())
        {
            #if DEBUG_FINGERPRINT
                Serial.println("Fingerprint done");
            #endif
            isFingerprintPresent=true;
            notify(SENSOR_READY);
        }
        else
        {
            #if DEBUG_FINGERPRINT
                Serial.println("Did not find fingerprint sensor,
retrying");
            #endif
            notify(SENSOR_NOT_FOUND);
```

```
        }
    }
}
boolean fingerGetImage()
{
    #if DEBUG_FINGERPRINT
        Serial.println("getting image");
    #endif
    notify(SENSOR_GETTING_IMAGE);
    uint8_t image=finger.getImage();
    switch(image)
    {
        case FINGERPRINT_OK:
            #if DEBUG_FINGERPRINT
                Serial.println("Image taken");
            #endif
            notify(SENSOR_IMAGE_TAKEN);
            return true;
        case FINGERPRINT_NOFINGER:
            #if DEBUG_FINGERPRINT
                Serial.println("No finger detected");
            #endif
            notify(SENSOR_NOT_FOUND);
            return false;
        case FINGERPRINT_PACKETRECIEVEERR:
            #if DEBUG_FINGERPRINT
                Serial.println("Communication error");
            #endif
            notify(SENSOR_COMMUNICATION_ERROR);
            return false;
        case FINGERPRINT_IMAGEFAIL:
            #if DEBUG_FINGERPRINT
                Serial.println("Imaging error");
            #endif
            notify(SENSOR_IMAGE_ERROR);
            return false;
        default:
            #if DEBUG_FINGERPRINT
                Serial.println("Unknown error");
            #endif
            notify(SENSOR_UNKNOWN_ERROR);
            return false;
    }
}
boolean fingetImage2Tz(int slot)
{
    #if DEBUG_FINGERPRINT
        Serial.print("converting image to tz in slot");
        Serial.println(slot);
```

```
    #endif
    notify(SENSOR_TAKING_IMAGE);
    uint8_t tz=finger.image2Tz(slot);
    return fingerCheckTz(tz);
}
boolean fingetImage2Tz()
{
    #if DEBUG_FINGERPRINT
        Serial.println("converting image to tz");
    #endif
    notify(SENSOR_TAKING_IMAGE);
    uint8_t tz=finger.image2Tz();
    return fingerCheckTz(tz);
}
boolean fingerCheckTz(uint8_t tz)
{
    switch(tz)
    {
        case FINGERPRINT_OK:
            #if DEBUG_FINGERPRINT
                Serial.println("Image converted");
            #endif
            notify(SENSOR_IMAGE_CONVERTED);
            return true;
        case FINGERPRINT_IMAGEMESS:
            #if DEBUG_FINGERPRINT
                Serial.println("Image too messy");
            #endif
            notify(SENSOR_IMAGE_TOO_MESSY);
            return false;
        case FINGERPRINT_PACKETRECIEVEERR:
            #if DEBUG_FINGERPRINT
                Serial.println("Communication error");
            #endif
            notify(SENSOR_COMMUNICATION_ERROR);
            return false;
        case FINGERPRINT_FEATUREFAIL:
            #if DEBUG_FINGERPRINT
                Serial.println("Could not find fingerprint features");
            #endif
            notify(SENSOR_FEATURE_FAIL);
            return false;
        case FINGERPRINT_INVALIDIMAGE:
            #if DEBUG_FINGERPRINT
                Serial.println("Invalid finger image");
            #endif
            notify(SENSOR_IMAGE_INVALID);
            return false;
        default:
```

```
            #if DEBUG_FINGERPRINT
                Serial.println("Unknown error");
            #endif
            notify(SENSOR_UNKNOWN_ERROR);
            return false;
    }
}
void fingerHasFingerPlaced()
{
    if(finger.getImage()==FINGERPRINT_OK)
    {
        notify(SENSOR_FINGER_PLACED,MAX_LEVEL);
    }
    else
    {
        notify(SENSOR_NO_FINGER_PLACED,MAX_LEVEL);
    }
}
void fingerWaitUntilPlaceFinger()
{
    #if DEBUG_FINGERPRINT
        Serial.println("please, place finger");
    #endif
    notify(SENSOR_PLACE_FINGER);
    uint8_t fingerImageState=finger.getImage();
    while(fingerImageState!=FINGERPRINT_OK)
    {
      Serial.println(fingerImageState);
      fingerImageState=finger.getImage();
    }
    while(finger.getImage()!=FINGERPRINT_OK);

}
void fingerWaitUntilRemoveFinger()
{
    #if DEBUG_FINGERPRINT
        Serial.println("please, remove finger");
    #endif
    notify(SENSOR_REMOVE_FINGER);
    while(finger.getImage()!=FINGERPRINT_NOFINGER);
}
boolean fingerValidateModel()
{
    #if DEBUG_FINGERPRINT
        Serial.println("creating model");
    #endif
    notify(SENSOR_CREATING_MODEL);
    uint8_t model=finger.createModel();
    switch(model)
```

```
    {
        case FINGERPRINT_OK:
            #if DEBUG_FINGERPRINT
                Serial.println("Finger matched!");
            #endif
            notify(SENSOR_FINGER_MATCHED);
            return true;
        case FINGERPRINT_PACKETRECIEVEERR:
            #if DEBUG_FINGERPRINT
                Serial.println("Communication error");
            #endif
            notify(SENSOR_COMMUNICATION_ERROR);
            return false;
        case FINGERPRINT_ENROLLMISMATCH:
            #if DEBUG_FINGERPRINT
                Serial.println("Fingerprints did not match");
            #endif
            notify(SENSOR_FINGER_MISMATCH);
            return false;
        default:
            #if DEBUG_FINGERPRINT
                Serial.println("Unknown error");
            #endif
            notify(SENSOR_UNKNOWN_ERROR);
            return false;
    }
}
boolean fingerStoreModel(int position)
{
    #if DEBUG_FINGERPRINT
        Serial.print("storing model into ");
        Serial.println(position);
    #endif
    notify(SENSOR_STORE_MODEL);
    uint8_t model=finger.storeModel(position);
    switch(model)
    {
        case FINGERPRINT_OK:
            #if DEBUG_FINGERPRINT
                Serial.println("Finger stored");
            #endif
            notify(SENSOR_FINGER_STORED);
            return true;
        case FINGERPRINT_PACKETRECIEVEERR:
            #if DEBUG_FINGERPRINT
                Serial.println("Communication error");
            #endif
            notify(SENSOR_COMMUNICATION_ERROR);
            return false;
```

```
        case FINGERPRINT_BADLOCATION:
            #if DEBUG_FINGERPRINT
                Serial.println("Could not store in that location");
            #endif
            notify(SENSOR_BAD_LOCATION);
            return false;
        case FINGERPRINT_FLASHERR:
            #if DEBUG_FINGERPRINT
                Serial.println("Error writing to flash");
            #endif
            notify(SENSOR_ERROR_FLASHING);
            return false;
        default:
            #if DEBUG_FINGERPRINT
                Serial.println("Unknown error");
            #endif
            notify(SENSOR_UNKNOWN_ERROR);
            return false;
    }
}
boolean fingerSearch()
{
    #if DEBUG_FINGERPRINT
        Serial.println("searching for finger");
    #endif
    notify(SENSOR_FINGER_SEARCH);
    uint8_t result=finger.fingerFastSearch();
    switch(result)
    {
        case FINGERPRINT_OK:
            #if DEBUG_FINGERPRINTG
                Serial.println("Found a print match!");
            #endif
            notify(SENSOR_FINGER_FOUND);
            return true;
        case FINGERPRINT_PACKETRECIEVEERR:
            #if DEBUG_FINGERPRINT
                Serial.println("Communication error");
            #endif
            notify(SENSOR_COMMUNICATION_ERROR);
            return false;
        case FINGERPRINT_NOTFOUND:
            #if DEBUG_FINGERPRINT
                Serial.println("Did not find a match");
            #endif
            notify(SENSOR_FINGER_NOT_FOUND);
            return false;
        default:
            #if DEBUG_FINGERPRINT
```

```
                Serial.println("Unknown error");
            #endif
            notify(SENSOR_UNKNOWN_ERROR);
            return false;
    }
}
boolean fingerRemoveById(int id)
{
    #if DEBUG_FINGERPRINT
        Serial.print("removing finger from ");
        Serial.println(id);
    #endif
    notify(SENSOR_DELETE_MODEL);
    uint8_t removed=finger.deleteModel(id);
    switch(removed)
    {
        case FINGERPRINT_OK:
            #if DEBUG_FINGERPRINT
                Serial.println("finger deleted");
            #endif
            notify(SENSOR_FINGER_DELETED);
            return true;
        case FINGERPRINT_PACKETRECIEVEERR:
            #if DEBUG_FINGERPRINT
                Serial.println("Communication error");
            #endif
            notify(SENSOR_COMMUNICATION_ERROR);
            return false;
        case FINGERPRINT_BADLOCATION:
            #if DEBUG_FINGERPRINT
                Serial.println("Could not delete in that location");
            #endif
            notify(SENSOR_BAD_LOCATION);
            return false;
        case FINGERPRINT_FLASHERR:
            #if DEBUG_FINGERPRINT
                Serial.println("Error writing to flash");
            #endif
            notify(SENSOR_ERROR_FLASHING);
            return false;
        default:
            #if DEBUG_FINGERPRINT
                Serial.print("Unknown error");
            #endif
            notify(SENSOR_UNKNOWN_ERROR);
            return false;
    }
}
int fingerGetId()
```

```
{
    #if DEBUG_FINGERPRINT
        Serial.println("getting id of finger placed");
    #endif

    notify(SENSOR_PLACE_FINGER,MAX_LEVEL);
    fingerWaitUntilPlaceFinger();
    if(fingetImage2Tz())
    {
        if(fingerSearch())
        {
            #if DEBUG_FINGERPRINT
                Serial.print("id: ");
                Serial.println(finger.fingerID);
                Serial.print("confidence: ");
                Serial.println(finger.confidence);
            #endif

notify(SENSOR_FINGER_COINCIDENCE,MAX_LEVEL,finger.fingerID,finger.confi
dence);
            return finger.fingerID;
        }
        else
        {
            notify(SENSOR_UNKNOWN_FINGER,MAX_LEVEL);
        }
    }
    else
    {
        notify(SENSOR_ERROR_HANDLING_IMAGE,MAX_LEVEL);
    }
    return -1;
}
boolean fingerAddNew(int position)
{
    #if DEBUG_FINGERPRINT
        Serial.print("inserting new finger into ");
        Serial.println(position);
    #endif
    notify(SENSOR_VERIFY_ALREADY_EXISTS,MAX_LEVEL);
    int id=fingerGetId();
    if(id==-1)
    {
        notify(SENSOR_REGISTER_NEW_FINGER,MAX_LEVEL);
        fingerWaitUntilRemoveFinger();
        fingerWaitUntilPlaceFinger();
        if(fingetImage2Tz(1))
        {
            fingerWaitUntilRemoveFinger();
```

```
            fingerWaitUntilPlaceFinger();
            if(fingetImage2Tz(2))
            {
                if(fingerValidateModel())
                {
                    if(fingerStoreModel(position))
                    {
                        #if DEBUG_FINGERPRINT
                            Serial.print("finger inserted");
                        #endif
                        notify(SENSOR_FINGER_SAVED,MAX_LEVEL);
                        return true;
                    }
                    else
                    {
                        #if DEBUG_FINGERPRINT
                            Serial.println("error inserting finger");
                        #endif
                        notify(SENSOR_ERROR_SAVING_MODEL,MAX_LEVEL);
                    }
                }
                else
                {
                    #if DEBUG_FINGERPRINT
                        Serial.println("error validating model");
                    #endif
                    notify(SENSOR_ERROR_INVALID_FINGERS,MAX_LEVEL);
                }
            }
            else
            {
                #if DEBUG_FINGERPRINT
                    Serial.println("error converting to tz");
                #endif
                notify(SENSOR_ERROR_HANDLING_IMAGE,MAX_LEVEL);
            }
        }
        else
        {
            #if DEBUG_FINGERPRINT
                Serial.println("error converting to tz");
            #endif
            notify(SENSOR_ERROR_HANDLING_IMAGE,MAX_LEVEL);
        }
    }
    else
    {
        #if DEBUG_FINGERPRINT
            Serial.println("finger already exists");
```

```
        #endif
        notify(SENSOR_FINGER_ALREADY_REGISTER,MAX_LEVEL);
    }
    return false;
}
boolean fingerRemove(int position)
{
    notify(SENSOR_DELETING_FINGER,MAX_LEVEL);
    if(fingerRemoveById(position))
    {
        notify(SENSOR_FINGER_DELETED,MAX_LEVEL);
    }
    else
    {
        notify(SENSOR_FINGER_NOT_DELETED,MAX_LEVEL);
    }
}
void fingerClearDatabase()
{
    #if DEBUG_FINGERPRINT
        Serial.println("Removing fingerprints");
    #endif
    notify(SENSOR_CLEAR_DATABASE,MAX_LEVEL);
    finger.emptyDatabase();
    notify(SENSOR_DATABASE_CLEARED,MAX_LEVEL);
}
void fingerShow(int id)
{
    uint8_t p = finger.loadModel(id);
    switch (p)
    {
        case FINGERPRINT_OK:
            Serial.print("template "); Serial.print(id);
Serial.println(" loaded");
            break;
        case FINGERPRINT_PACKETRECIEVEERR:
            Serial.println("Communication error");
            return p;
        default:
            Serial.print("Unknown error "); Serial.println(p);
            return p;
    }
    // OK success!

    p = finger.getModel();
    switch (p)
    {
        case FINGERPRINT_OK:
```

```
            Serial.print("template "); Serial.print(id);
Serial.println(" transferring");
            break;
        default:
            Serial.print("Unknown error "); Serial.println(p);
            return p;
    }

    uint8_t templateBuffer[256];
    memset(templateBuffer, 0xff, 256);  //zero out template buffer
    int index=0;
    uint32_t starttime = millis();
    while ((index < 256) && ((millis() - starttime) < 1000))
    {
        if (mySerial.available())
        {
            templateBuffer[index] = mySerial.read();
            index++;
        }
    }

    Serial.print(index); Serial.println(" bytes read");

    //dump entire templateBuffer.  This prints out 16 lines of 16 bytes
    for (int count= 0; count < 16; count++)
    {
        for (int i = 0; i < 16; i++)
        {
            Serial.print("0x");
            Serial.print(templateBuffer[count*16+i], HEX);
            Serial.print(", ");
        }
        Serial.println();
    }
}
//IO
#define PIN_RFID 4
#define PIN_FINGERPRINT 5
void ioSetup()
{
  pinMode(PIN_RFID,INPUT_PULLUP);
  pinMode(PIN_FINGERPRINT,INPUT_PULLUP);
}
void getPlacedCardId()
{
  String nearCard=rfidGetNearCard();
  Serial.println("{\"card\":\""+nearCard+"\"}");
}
```

```
//WIFI CONECTION
#define WIFI_TX 6
#define WIFI_RX 7
#define WIFI_BAUD_RATE 9600
SoftwareSerial wifi(WIFI_RX,WIFI_TX);
void wifiBegin()
{
  wifi.begin(WIFI_BAUD_RATE);
  while(!wifi);
}
void wifiNotify()
{
  wifi.print("http://192.168.200.100/arduino?state=1");
}
//MAIN
void showMenu()
{
  #if DEBUG_FINGERPRINT
    Serial.println("+----------------------------+");
    Serial.println("|            menu            |");
    Serial.println("+---+------------------------+");
    Serial.println("| 0 | clear database         |");
    Serial.println("+---+------------------------+");
    Serial.println("| 1 | add new finger into id |");
    Serial.println("+---+------------------------+");
    Serial.println("| 2 | remove placed finger   |");
    Serial.println("+---+------------------------+");
    Serial.println("| 3 | remove finger by id    |");
    Serial.println("+---+------------------------+");
    Serial.println("| 4 | get if of placed finger |");
    Serial.println("+---+------------------------+");
    Serial.println("| 5 | show finger by id      |");
    Serial.println("+---+------------------------+");
  #endif
}
void setup()
{
  serialSetup();
  fingerSetup();
  wifiBegin();
  rfidSetup();
  relaySetup();
  ioSetup();
  showMenu();
}
#define CLEAR_DATABASE '0'
#define ADD_NEW_FINGER '1'
#define REMOVE_PLACED_FINGER '2'
#define REMOVE_FINGER_BY_ID '3'
```

```
#define GET_PLACED_FINGER_ID '4'
#define HAS_FINGER '6'
#define GET_PLACED_CARD_ID '7'
int position;
void loop()
{  wifiNotify();
  if(Serial.available())
  {
    int id=-1;
    char cmd=Serial.read();
    switch(cmd)
    {     case CLEAR_DATABASE:
        fingerClearDatabase();
        break;
      case ADD_NEW_FINGER:
        position=serialGetNumer();
        fingerAddNew(position);
        break;
      case REMOVE_FINGER_BY_ID:
        position=serialGetNumer();
        fingerRemove(position);
        break;
      case GET_PLACED_FINGER_ID:
        fingerGetId();
        break;
      case HAS_FINGER:
        fingerHasFingerPlaced();
        break;
      case GET_PLACED_CARD_ID:
        getPlacedCardId();
        break;
      default:
        #if DEBUG_FINGERPRINT
          Serial.println("invalid cmd");
        #endif
        break;
    }
    showMenu();
  }  }
```

Anexo 2. Programación en el Arduino Pro-Mini – Programa de adquisición de datos de los dispositivos RFID y Biométrico para la transmisión y gestión de la información de los empleados.

Fuentes: Los Autores

ANEXO

DESARROLLO DEL CODIGO DEL MODULO WIFI NODECMU ESP 8266

```
#include <ESP8266WiFi.h>

const char* ssid     = "biometricpi";
const char* password = "biometricpi";

const char* host = "http://192.168.10.2";
const char* streamId   = ".....................";
const char* privateKey = ".....................";

void setup() {
  Serial.begin(9600);
  Serial.setDebugOutput(true);
  delay(10);

  // We start by connecting to a WiFi network

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

int value = 0;

void loop() {
  delay(5000);
  ++value;

  Serial.print("connecting to ");
  Serial.println(host);

  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
```

```
  }

  // We now create a URI for the request
  String url = "/input/";
  url += streamId;
  url += "?private_key=";
  url += privateKey;
  url += "&value=";
  url += value;

  Serial.print("Requesting URL: ");
  Serial.println(url);

  // This will send the request to the server
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  unsigned long timeout = millis();
  while (client.available() == 0) {
    if (millis() - timeout > 5000) {
      Serial.println(">>> Client Timeout !");
      client.stop();
      return;
    }
  }

  // Read all the lines of the reply from server and print them to
Serial
  while(client.available()){
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }

  Serial.println();
  Serial.println("closing connection");
}
```

Anexo 1. Programación para formatear el Módulo WIFI – Programa que permite al Modulo WIFI empezar desde cero eliminando todos los datos guardados con anticipación para la compilación de datos nuevos.

Fuentes: Los Autores

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
```

```
#include <ESP8266HTTPClient.h>

ESP8266WiFiMulti WiFiMulti;

String sendRequest(String url)
{
  int wifiState=WiFiMulti.run();
  if(wifiState==WL_CONNECTED)
  {
    HTTPClient http;
    http.begin(url);
    int httpCode=http.GET();
    String payload=http.getString();
    http.end();
    if(httpCode>0)
    {
      if(httpCode==HTTP_CODE_OK)
      {
        return payload;
      }
      else
      {
        return "{\"state\":\""+String(httpCode)+"\"}";
      }
    }
    else
    {
      return http.errorToString(httpCode).c_str();
    }
  }
  else
  {
    return "NO CONNECTED";
  }
}

//DEBUG
#define DEBUG true
void printDebug(String message)
{
  #if DEBUG
    Serial.print(message);
  #endif
}
void printDebug(char character)
{
  #if DEBUG
    Serial.print(character);
  #endif
```

```
}
void printlnDebug(String message)
{
  #if DEBUG
    Serial.println(message);
  #endif
}

//MAIN PROGRAM
#define BAUD_RATE 9600
#define SERIAL_DELAY 25
void setup()
{
  Serial.begin(BAUD_RATE);
  while(!Serial);

  WiFi.mode(WIFI_STA);
  WiFiMulti.addAP("CREDILIT", "Creditos1994");
  printlnDebug("Connecting to Wifi");

  int wifiState = -1;
  do {
    wifiState = WiFiMulti.run();
    printDebug(".");
    delay(1000);
  } while(wifiState != WL_CONNECTED);
}

void loop()
{
  String URL = "http://192.168.0.200:5000/api/arduino?state=1";
  printDebug("GET: "); printlnDebug(URL);
  String response=sendRequest(URL);
  Serial.println(response);
  delay(8000);
  }
```

Anexo 2. Programación para conexión el Módulo WIFI – Programa que permite la comunicación del Módulo WIFI con las Raspberry Pi 3, el cual lleva configurada la Red inalámbrica a la que está conectada y la ip del Raspberry Pi 3.

Fuentes: Los Autores

ANEXO

DESARROLLO DEL CODIGO DEL SISTEMA DE SEGURIDAD DE ENVIO SMS

```
//SMS
char phone_number[]="0999638782";
char sms_text[]="Se detecto un intento de ingreso invalido en el
local";
char aux_string[30];
int8_t answer;

#define Rx 8
#define Tx 7
#define GSM_LED 13
#define GPRS_BAUD_RATE 9600
#include <SoftwareSerial.h>
SoftwareSerial gprs(Tx,Rx);
int8_t sendATcommand(char* ATcommand, char* expected_answer, unsigned
int timeout) {
    uint8_t x=0,  answer=0;
    char response[100];
    unsigned long previous;

    memset(response, '\0', 100);
    delay(100);

    while( gprs.available() > 0) {
        gprs.read();
    }
    gprs.println(ATcommand);
    Serial.print(">>");
    Serial.println(ATcommand);

    x = 0;
    previous = millis();
    Serial.print("<<");

    do {
        if(gprs.available() != 0) {
            response[x] = gprs.read();
            Serial.print(response[x]);
            x++;
            if (strstr(response, expected_answer) != NULL) {
                answer = 1;
            }
        }
    }
    while((answer == 0) && ((millis() - previous) < timeout));
    Serial.println("");

    return answer;
}
void gprsSetup() {
```

```
    gprs.begin(GPRS_BAUD_RATE);
    while(!gprs);
    Serial.println("Gprs serial done");
    pinMode(GSM_LED, OUTPUT);
}
void gprsWaitUntilConnect() {
  digitalWrite(GSM_LED, HIGH);
  Serial.println("Connecting to network");
  while( (sendATcommand("AT+CREG?", "+CREG: 0,1", 500) ||
          sendATcommand("AT+CREG?", "+CREG: 0,5", 500)) == 0 );
  digitalWrite(GSM_LED, LOW);
}
//#define CALL_SECONDS 12

void gprsSendSms() {
    Serial.println("Set to sms mode");
    sendATcommand("AT+CMGF=1", "OK", 1000);

    Serial.println("set receiver number");
    sprintf(aux_string,"AT+CMGS=\"%s\"", phone_number);
    answer = sendATcommand(aux_string, ">", 2000);

    if (answer == 1) {
        gprs.println(sms_text);
        gprs.write(0x1A);
        answer = sendATcommand("", "OK", 20000);
        if (answer == 1) {
            Serial.println("message sent");
        } else {
            Serial.println("error 1");
        }
    }
    else {
        Serial.print("error 2");
        Serial.println(answer, DEC);
    }
}
//IO
#define PIN_S1 4
#define PIN_S2 5
void ioSmsSetup() {
  pinMode(PIN_S1, INPUT_PULLUP);
  pinMode(PIN_S2, INPUT_PULLUP);
}
boolean ioSMSHaveToSendSMS() {
  boolean s1 = digitalRead(PIN_S1) == LOW;
  boolean s2 = digitalRead(PIN_S2) == LOW;
  return s1 && s2;
}
```

```
//SERIAL
#define BAUD_RATE 9600
void serialSetup() {
  Serial.begin(BAUD_RATE);
}
//MAIN
void setup() {
  ioSmsSetup();
  serialSetup();
  gprsSetup();
  gprsWaitUntilConnect();
}

void loop() {
  if(ioSMSHaveToSendSMS()) {
    delay(1000);
    Serial.println("Send sms");
    gprsSendSms();
  }
}
```

Anexo 1. Programación para GSM/GRP SIM900 Shield – Líneas de programación realizada en Arduino Uno para la comunicación del GSM/GPRS SIM900 Shield realizando comunicación por la conexión del Raspberry Pi 3.

Fuentes: Los Autores

ANEXO

DESARROLLO DEL CODIGO DE LA RASPBERRY PI 3

```
import os

os.popen("sudo node /home/pi/Downloads/index.js")
```

Anexo 1. Programación main.py de la Raspberry Pi 3 – Líneas de programación en el cual se produce el comando principal de NodeJs en la parte del Raspberry Pi 3.

Fuentes: Los Autores

```
const mysql=require('mysql')
const connection=mysql.createConnection({
    /*
    host:'localhost',
    user:'user',
    password:'123',
    database:'biometric'
    */
    host:'localhost',
    user:'root',
    password:'',
    database:'biometric-pi'
})
connection.connect(error=>{
    if(error){
        throw error
    }
})
exports.getConnection=()=>{
    return connection
}
```

Anexo 2. Programación de la Base de Datos – Programación realizada en la Raspberry Pi 3 para la conexión con la Base de datos MYSQL.

Fuentes: Los Autores

```
print('importing libraries')
from gpiozero import Button
from gpiozero import LED
from logger_tools import logger
from httpclient import HttpClient
from time import sleep
from sys import argv
```

```
from sys import exit
from mysmtp import Smtp
from time import time
from time import sleep
from picamera import PiCamera
import MySQLdb
from fingerprint import Fingerprint
from time import sleep
print('\t--done')

camera = PiCamera()

haveToSendSMS = False
haveToSendEmail = False
for argument in argv:
    if argument == '--sms':
        haveToSendSMS = True
    if argument == '--email':
        haveToSendEmail = True

server='localhost'
username='user'
password='123'
database='biometric'

print('connection to database')
database=MySQLdb.connect(server,username,password,database)
cursor=database.cursor()
print('\t--done')

ID=0
CODE=1
NAME=2

print('connection to fingerprint')
fingerprint=Fingerprint.instance()
print('\t--done')

button = Button(18)
led = LED(17)

if haveToSendSMS:
    sendSms1 = LED(27)
    sendSms2 = LED(22)

    sendSms1.on()
    sendSms2.on()

def sendSms():
```

```
    print('send SMS')
    sendSms1.off()
    sendSms2.off()
    sleep(1)
    sendSms1.on()
    sendSms2.on()

while True:
    try:
        while True:
            if button.is_pressed:
                while button.is_pressed:
                    pass
                print("\nboton pulsado")

                estadoTarjeta = False
                cursor.execute('select id, card from employees')
                cards = cursor.fetchall()
                    print("\nesperando por tarjeta")
                    card_string = fingerprint.get_rfid()
                for card in cards:
                    print("codigo leido", card_string, ", codigo
esperado ", card[CODE])
                    if card_string == card[CODE]:
                        id_employee = card[ID]
                        estadoTarjeta = True
                        print("tarjeta correcta")
                    else:
                        print("tarjeta incorrecta")

                estadoHuella = False
                if fingerprint.has_finger_placed() == True:
                    print("\nhuella puesta")
                    id_finger = fingerprint.get_id()
                    print("huella id ",id_finger)
                    if id_finger != -1:
                        cursor.execute('select id from employees where
code=' + id_finger + ' limit 1')
                        id_employee = cursor.fetchall()
                        id_employee = id_employee[ID][ID]
                        estadoHuella = True
                        print("huella correcta")
                    else:
                        print("huella incorrecta")
                if estadoHuella == True or estadoTarjeta == True:
                    print("\t ** puerta abierta **")
                    led.on()
                    sleep(3)
                    led.off()
```

```
                    cursor.execute('insert into registers (employee_id)
values (' + str(id_employee) + ')')
                else:
                    if haveToSendSMS:
                        sendSms()

                    if haveToSendEmail:
                        photoWasTaked = False
                        photosRetry = 0
                        while not photoWasTaked:
                            try:
                                photosRetry = photosRetry + 1
                                timestamp = str(time()).replace('.',
'')

camera.capture('{timestamp}.jpg'.format(timestamp = timestamp))
                                photoWasTaked = True
                            except Exception as error:
                                logger.error(error,exc_info=True)
                                if photosRetry > 3:
                                    raise Exception('Failed 3 times
taking photo')

                        emailWasSent = False
                        emailsRetry = 0
                        while not emailWasSent:
                            try:
                                emailsRetry = emailsRetry + 1
                                print('connecting to email')
                                smtp = Smtp.getOutlook()
                                smtp.login('biometricpi@outlook.com',
'Password2018*')
                                print('sending email')
                                smtp.send(
    content = 'Se detecto un intento de ingreso invalido en el local',
    receiver = 'credilit2009@hotmail.com',
    subject = 'Intento de acceso invalido',
    attachment_path = '{timestamp}.jpg'.format(timestamp = timestamp)
                                )
                                print('\tdone')
                                emailWasSent = True
                            except Exception as error:
                                logger.error(error,exc_info=True)
                                if emailsRetry > 3:
                       raise Exception('Failed 3 times sending email')
                                sleep(1)

                    print("\t ** puerta no abierta **")
```

```
            cursor.execute('select id,code,name from actions')
            actions=cursor.fetchall()

            for action in actions:
                id=action[ID]
                position=int(action[CODE])

                if action[NAME]=='remove':
                    print("entro remove")
                    print('removing from position
"{position}"'.format(position=position))
                    fingerprint.remove_finger(position)
                    cursor.execute('delete from actions where
id="{id}"'.format(id=id))
                    cursor.execute('insert into notifications (content)
values ("Huella borrada")')
                    print("salio remove")

                if action[NAME]=='add':
                    print("entro add")
                    print('adding into position
"{position}"'.format(position=position))
                    position=fingerprint.add_new_finger(position)
                    cursor.execute('delete from actions where
id="{id}"'.format(id=id))
                    if position>0:
                        cursor.execute('insert into notifications
(content) values ("Huella agregada")')
                        print("huellla agregada")
                    elif position==0:
                        cursor.execute('insert into notifications
(content) values ("Huella previamente registrada")')
                        print("huella repetida")
                    else:
                        cursor.execute('insert into notifications
(content) values ("Error registrando la huella")')
                        print("huella error")
            database.commit()
        fingerprint.close()
    except Exception as error:
        print(error)
        logger.error(error,exc_info=True)
```

Anexo 3. Programación principal de la plataforma realizada en Raspberry Pi 3 donde llaman las demás librerías

Fuentes: Los Autores

```
from serial_tools import Serial
from json import loads as json_decoder
from serial import SerialException

class FingerprintNotFoundException(Exception):
    pass
class Fingerprint:
    levels = {
        'MAX_LEVEL':'99'
    }

    actions = {
        'CLEAR_DATABASE': '0',
        'ADD_NEW_FINGER': '1',
        'REMOVE_PLACED_FINGER': '2',
        'REMOVE_FINGER_BY_ID': '3',
        'GET_PLACED_FINGER_ID': '4',
        'HAS_FINGER': '6',
        'GET_RFID': '7',
    }

    responses = {
        '0x01':'Verificando sensor',
        '0x02':'Sensor listo',
        '0x03':'Sensor no encontrado',
        '0x04':'Obteniendo imagen',
        '0x05':'Imagen capturada',
        '0x06':'Error de comunicación',
        '0x07':'Error de imagen',
        '0x08':'Error desconocido',
        '0x09':'Tomando imagen',
        '0x10':'Imagen convertida',
        '0x11':'Imagen en malas condiciones',
        '0x12':'Error de función',
        '0x13':'Imagen inválida',
        '0x14':'Por favor ubique el dedo',
        '0x15':'Por favor quite el dedo',
        '0x16':'Creando modelo',
        '0x17':'Huella encontrada',
        '0x18':'Huellas no coinciden',
        '0x19':'Modelo agregado',
        '0x20':'Huella agregada',
        '0x21':'Ubicación inválida',
        '0x22':'Error escribiendo en el sensor',
        '0x23':'Error buscando',
        '0x24':'Huella encontrada',
        '0x25':'Huella no encontrada',
        '0x26':'Eliminar modelo',
        '0x27':'Huella eliminada',
```

```
        #clear database
        '0x28':'Borrando la base de datos',
        '0x29':'Base de datos borrada',

        #add new finger
        '0x30':'Verificando si ya está registrada',
        '0x31': 'Huella ya registrada',
        '0x32': 'Error digitalizando la huella',
        '0x33': 'Huellas inválidas',
        '0x34': 'Error guandando los modelos',
        '0x35': 'Huella registrada',
        '0x36': 'Registrando nueva huella',

        #remove finger
        '0x37':'Eliminando huella',
        '0x38':'Huella no eliminada',

        #searcg finger
        '0x39':'Huella desconocida',
        '0x40': 'Coincidencia encontrada',

        #check finger
        '0x41':'Huella detectada',
        '0x42': 'Huella no detectada'
    }

    ports = (
        '/dev/ttyACM0',
        '/dev/ttyACM1',
        '/dev/ttyUSB0',
        '/dev/ttyUSB1',
        'com1',
        'com2',
        'com3',
        'com4',
        'com5'
    )
    def instance():
        for port in Fingerprint.ports:
            try:
                print(port)
                fingerprint=Fingerprint(port=port)
                fingerprint.wait_for_sensor()
                return fingerprint
            except SerialException as error:
                pass
            except OSError as error:
                pass
```

```
        raise SerialException('Serial device not found')

    def remove_finger_at(position):
        try:
            fingerprint=Fingerprint.instance()
            fingerprint.remove_finger(position)
            fingerprint.close()
            return None
        except SerialException as error:
            return ('Sensor no encontrado','No se pudo establecer
conexión con el sensor')
        except JSONDecodeError as error:
            return ('Respuesta inválida','El sensor envió una respuesta
que no pudo ser procesada')
        except KeyError as error:
            return ('Respuesta inválida','La respuesta recibida por el
sensor no tiene un formato correcto')

    def print_by_level(self,json):
        if json['level'] in self.levels['MAX_LEVEL']:
            pass#print(self.responses[json['code']])
        else:
            pass#print('\t', self.responses[json['code']])

    def send_position(self, position):
        if position < 10:
            self.arduino.write('00'+str(position))
        elif 10 <= position < 100:
            self.arduino.write('0'+str(position))
        elif 100 <= position < 1000:
            self.arduino.write(str(position))

    def __init__(self,port='COM3', baud_rate=9600):
        self.arduino = Serial(port=port, baud_rate=baud_rate)

    def wait_for_sensor(self):
        while True:
            data = self.arduino.read(initializer='{', finisher='}')
            if data not in '':
                json = json_decoder(data)
                self.print_by_level(json)
                if json['code'] in '0x01':
                    while True:
                        data = self.arduino.read(initializer='{',
finisher='}')
                        if data not in '':
                            json = json_decoder(data)
                            self.print_by_level(json)
                            if json['code'] in '0x02':
```

```
                            return

    def clear_database(self):
        self.arduino.write(self.actions['CLEAR_DATABASE'])
        while True:
            data = self.arduino.read(initializer='{', finisher='}')
            if data not in '':
                json = json_decoder(data)
                self.print_by_level(json)
                if json['code'] in '0x29':
                    return

    def add_new_finger(self, position):
        self.arduino.write(self.actions['ADD_NEW_FINGER'])
        self.send_position(position)
        while True:
            data = self.arduino.read(initializer='{', finisher='}')
            if data not in '':
                json = json_decoder(data)
                self.print_by_level(json)
                if json['code'] in '0x31':
                    return 0
                elif json['code'] in '0x35':
                    return position
                if json['code'] in ['0x32', '0x33', '0x34']:
                    return -1

    def remove_finger(self, position):
        self.arduino.write(self.actions['REMOVE_FINGER_BY_ID'])
        self.send_position(position)
        while True:
            data = self.arduino.read(initializer='{', finisher='}')
            if data not in '':
                json = json_decoder(data)
                self.print_by_level(json)
                if json['code'] in ['0x38','0x27']:
                    return

    def get_id(self):
        self.arduino.write(self.actions['GET_PLACED_FINGER_ID'])
        while True:
            data = self.arduino.read(initializer='{', finisher='}')
            if data not in '':
                json = json_decoder(data)
                self.print_by_level(json)
                if json['code'] in ['0x39','0x32']:
                    return -1
                if json['code'] in '0x40':
                    return json['id']
```

```
    def has_finger_placed(self):
        self.arduino.write(self.actions['HAS_FINGER'])
        while True:
            data = self.arduino.read(initializer='{', finisher='}')
            if data not in '':
                json = json_decoder(data)
                self.print_by_level(json)
                if json['code'] in '0x41':
                    return True
                if json['code'] in '0x42':
                    return False

    def get_rfid(self):
        self.arduino.write(self.actions['GET_RFID'])
        while True:
            data = self.arduino.read(initializer='{', finisher='}')
            if data not in '':
                json = json_decoder(data)
                if 'card' in json:
                    #self.print_by_level(json)
                    return json['card']

    def close(self):
        self.arduino.close()
    def get_arduino(self):
        return self.arduino
```

Anexo 4. Programación para comparación de obtención de datos por el Arduino.

Fuentes: Los Autores

```
import serial
class Serial:
    def __init__(self, port, baud_rate=9600):
        self.port = port
        self.baud_rate = baud_rate
        self.port = serial.Serial(port=port, baudrate=baud_rate)

    def read_decoded(self):
        character = self.port.read()
        if character == '':
            return ''
        else:
            return character.decode('utf-8')
```

```
    def read(self, initializer='', finisher=''):
        if initializer in '':
            return self.read_decoded()

        character = self.read_decoded()
        while character not in initializer:
            character = self.read_decoded()
        data = initializer

        if finisher in '':
            return self.read_decoded()

        while character not in finisher:
            character = self.read_decoded()
            data = data + character
        return data

    def write(self,data):
        self.port.write(str(data).encode('utf-8'))

    def close(self):
        self.port.close()

if __name__=='__main__':
    arduino = Serial(port='com3')
    data = arduino.read(initializer='{', finisher='}')
    print(data)
    data = arduino.read(initializer='{', finisher='}')
    print(data)
```

Anexo 5. Programación realizada para la lectura del Arduino

Fuentes: Los Autores

```
<html lang="es">
    <head>
        <!-- metas -->
        <meta name="viewport" content="width=device-width, initial-
scale=1">
        <meta charset="utf-8">
        <title>
            Biometric Pi
        </title>
        <!-- metas -->
        <!-- styles -->
        <link rel="stylesheet"
href="/plugins/materialize/css/materialize.css">
```

```
        <!--<link rel="stylesheet" href="/plugins/font-
awesome/css/font-awesome.css">-->
        <link rel="stylesheet" href="/plugins/google-
fonts/css/icon.css">
        <link rel="stylesheet" href="/plugins/app/css/app.css">
        <!-- styles -->

        <!-- scripts -->
        <script type="text/javascript"
src="/plugins/jquery/js/jquery.js"></script>
        <script type="text/javascript"
src="/plugins/materialize/js/materialize.js"></script>
        <script type="text/javascript"
src="/plugins/vue/js/vue.js"></script>
        <script type="text/javascript"
src="/plugins/axios/js/axios.js"></script>
        <!-- scripts -->
    </head>
    <body class="grey lighten-3">
        <!-- app -->
        <div id="app">

            <!-- login modal -->
            <div v-if="!app.isAuthenticated">
                <div class="modal modal-fixed-footer open" style="z-
index: 1005; display: block; opacity: 1; transform: scaleX(1); top:
10%;">
                    <div class="modal-content">
                        <h4>
                            Login
                        </h4>
                        <br>
                        <form v-on:submit.prevent="loginUser">
                            <div class="input-field col s6">
                                <input v-model="login.email" id="email"
type="email" class="validate">
                                <label for="email">
                                    Dirección de email
                                </label>
                            </div>
                            <div class="input-field col s6">
                                <input v-model="login.password"
id="password" type="password" class="validate">
                                <label for="password">
                                    Contraseña
                                </label>
                            </div>
                        </form>
                    </div>
```

```
                    <div class="modal-footer">
                        <a v-on:click="loginUser" href="#!"
class="modal-action waves-effect waves-green btn-flat">
                            Iniciar sesión
                        </a>
                    </div>
                </div>
                <div class="modal-overlay" style="z-index: 1004;
display: block; opacity: 0.5;"></div>
            </div>
            <!-- login modal -->

            <template v-else="app.isAuthenticated">
                <!-- navbar -->
                <nav class="red">
                    <div class="nav-wrapper">
                        <a v-bind:href="app.href" class="brand-logo">
                             {{app.name}}
                        </a>
                        <a href="#" data-activates="mobile-demo"
class="button-collapse">
                            <i class="material-icons">
                                menu
                            </i>
                        </a>
                        <ul class="right hide-on-med-and-down">
                            <li v-for="item in navbar.items">
                                <a v-bind:href="item.href">
                                    {{item.label}}
                                </a>
                            </li>
                        </ul>
                        <ul class="side-nav" id="mobile-demo">
                            <li v-for="item in navbar.items">
                                <a v-bind:href="item.href">
                                    {{item.label}}
                                </a>
                            </li>
                        </ul>
                    </div>
                </nav>
                <!-- navbar -->

                <!-- loader -->
                <div v-show="isLoading" class="progress">
                    <div class="indeterminate"></div>
                </div>
                <!-- loader -->
```

```
                <!-- employees -->
                <div class="container">
                    <h3 class="header">
                        {{list.title}}
                    </h3>

                    <!-- no employees -->
                    <div v-if="list.employees.length==0 && !isLoading">
                        <div class="card horizontal">
                            <div class="card-stacked">
                                <div class="card-content">
                                    {{list.empty}}
                                </div>
                            </div>
                        </div>
                    </div>
                    <!-- no employees -->

                    <!-- employee -->
                    <div v-for="employee in list.employees" class="card
horizontal">
                        <!--<div class="card-image">
                            <img
src="https://lorempixel.com/100/190/nature/6">
                        </div>-->
                        <div class="card-stacked">
                            <div class="card-content">
                                <div>
                                    {{employee.name}}
                                </div>
                                <div>
                                    {{employee.card}}
                                </div>
                                <div>
                                    {{formatDate(employee.created_at)}}
                                </div>
                            </div>
                            <div class="card-action">
                                <a v-
on:click="eval(action.onClick+'(employee)')" v-for="action in
list.actions" class="modal-action waves-effect btn-flat">
                                    {{action.label}}
                                </a>
                            </div>
                        </div>
                    </div>
                    <!-- employee -->
                </div>
                <!-- employees -->
```

```
                <!-- fab button -->
                <div class="fixed-action-btn" style="bottom: 45px;
right: 24px;">
                    <a v-on:click="showCreateModal" class="btn-floating
btn-large waves-effect waves-light red">
                        <i class="material-icons">
                            add
                        </i>
                    </a>
                </div>
                <!-- fab button -->

                <!-- delete modal -->
                <div v-if="modals.delete.employee">
                    <div class="modal modal-fixed-footer open"
style="z-index: 1005; display: block; opacity: 1; transform: scaleX(1);
top: 10%;">
                        <div class="modal-content">
                            <h4>
                                <span v-if="isDeleting">
                                    <div class="modal-overlay"
style="z-index: 1004; display: block; opacity: 0.5;"></div>
                                    <div class="preloader-wrapper small
active">
                                        <div class="spinner-layer
spinner-blue-only">
                                            <div class="circle-clipper
left">
                                                <div
class="circle"></div>
                                            </div>
                                            <div class="gap-patch">
                                                <div
class="circle"></div>
                                            </div>
                                            <div class="circle-clipper
right">
                                                <div
class="circle"></div>
                                            </div>
                                        </div>
                                    </div>
                                </span>
                                {{modals.delete.title}}
                            </h4>
                            <p>
                                {{modals.delete.message}}
"{{modals.delete.employee.name}}"?
```

```
                            </P>
                        </div>
                        <div class="modal-footer">
                            <a v-on:click="eval(action.onClick+'()')"
v-for="action in modals.delete.actions" href="#!" class="modal-action
waves-effect waves-green btn-flat">
                                {{action.label}}
                            </a>
                        </div>
                    </div>
                    <div class="modal-overlay" style="z-index: 1004;
display: block; opacity: 0.5;"></div>
                </div>
                <!-- delete modal -->

                <!-- create modal -->
                <div v-if="modals.create.employee">
                    <div class="modal modal-fixed-footer open"
style="z-index: 1005; display: block; opacity: 1; transform: scaleX(1);
top: 10%;">
                        <div class="modal-content">
                            <h4>
                                <span v-if="isCreating">
                                    <div class="modal-overlay"
style="z-index: 1004; display: block; opacity: 0.5;"></div>
                                    <div class="preloader-wrapper small
active">
                                        <div class="spinner-layer
spinner-blue-only">
                                            <div class="circle-clipper
left">
                                                <div
class="circle"></div>
                                            </div>
                                            <div class="gap-patch">
                                                <div
class="circle"></div>
                                            </div>
                                            <div class="circle-clipper
right">
                                                <div
class="circle"></div>
                                            </div>
                                        </div>
                                    </div>
                                </span>
                                {{modals.create.title}}
                            </h4>
                            <form v-on:submit.prevent="createEmployee">
```

```
                                <div class="input-field col s6">
                                    <input v-
model="modals.create.employee.name" id="name" type="text"
class="validate">
                                    <label for="name">

{{modals.create.fields.name.label}}
                                    </label>
                                </div>
                                <div class="input-field col s6">
                                    <input v-
model="modals.create.employee.card" id="card" type="text"
class="validate">
                                    <label for="card">

{{modals.create.fields.card.label}}
                                    </label>
                                </div>
                            </form>
                        </div>
                        <div class="modal-footer">
                            <a v-on:click="eval(action.onClick+'()')"
v-for="action in modals.create.actions" href="#!" class="modal-action
waves-effect waves-green btn-flat">
                                {{action.label}}
                            </a>
                        </div>
                    </div>
                    <div class="modal-overlay" style="z-index: 1004;
display: block; opacity: 0.5;"></div>
                </div>
                <!-- create modal -->

                <!-- update modal -->
                <div v-if="modals.update.employee">
                    <div class="modal modal-fixed-footer open"
style="z-index: 1005; display: block; opacity: 1; transform: scaleX(1);
top: 10%;">
                        <div class="modal-content">
                            <h4>
                                <span v-if="isUpdating">
                                    <div class="modal-overlay"
style="z-index: 1004; display: block; opacity: 0.5;"></div>
                                    <div class="preloader-wrapper small
active">
                                        <div class="spinner-layer
spinner-blue-only">
                                            <div class="circle-clipper
left">
```

```
                                                <div
class="circle"></div>
                                            </div>
                                            <div class="gap-patch">
                                                <div
class="circle"></div>
                                            </div>
                                            <div class="circle-clipper
right">
                                                <div
class="circle"></div>
                                            </div>
                                        </div>
                                    </div>
                                </span>
                                {{modals.update.title}}
                            </h4>
                            <form v-on:submit.prevent="updateEmployee">
                                <div class="input-field col s6">
                                    <input v-
model="modals.update.employee.name" id="name" type="text"
class="validate">
                                    <label for="name">

{{modals.update.fields.name.label}}
                                    </label>
                                </div>
                                <div class="input-field col s6">
                                    <input v-
model="modals.update.employee.card" id="card" type="text"
class="validate">
                                    <label for="card">

{{modals.update.fields.card.label}}
                                    </label>
                                </div>
                            </form>
                        </div>
                        <div class="modal-footer">
                            <a v-on:click="eval(action.onClick+'()')"
v-for="action in modals.update.actions" href="#!" class="modal-action
waves-effect waves-green btn-flat">
                                {{action.label}}
                            </a>
                        </div>
                    </div>
                    <div class="modal-overlay" style="z-index: 1004;
display: block; opacity: 0.5;"></div>
                </div>
```

```
                <!-- update modal -->
            </template>

            <script src="/plugins/app/js/app.js"></script>
        <!-- app -->

        <!-- vue -->
        <script src="/plugins/app/js/employees.js"></script>
        <!-- vue -->
    </body>
</html>
```

Anexo 6. Aplicativo web principal en donde se muestra los empleados del local "CREDILIT".

Fuentes: Los Autores

```
<html lang="es">
    <head>
        <!-- metas -->
        <meta name="viewport" content="width=device-width, initial-
scale=1">
        <meta charset="utf-8">
        <title>
            Biometric Pi
        </title>
        <!-- metas -->

        <!-- styles -->
        <link rel="stylesheet"
href="/plugins/materialize/css/materialize.css">
        <!--<link rel="stylesheet" href="/plugins/font-
awesome/css/font-awesome.css">-->
        <link rel="stylesheet" href="/plugins/google-
fonts/css/icon.css">
        <link rel="stylesheet" href="/plugins/app/css/app.css">
        <!-- styles -->
        <!-- scripts -->
        <script type="text/javascript"
src="/plugins/jquery/js/jquery.js"></script>
        <script type="text/javascript"
src="/plugins/materialize/js/materialize.js"></script>
        <script type="text/javascript"
src="/plugins/vue/js/vue.js"></script>
        <script type="text/javascript"
src="/plugins/axios/js/axios.js"></script>
```

```
        <!-- scripts -->
    </head>
    <body class="grey lighten-3">
        <!-- app -->
        <div id="app">

            <!-- navbar -->
            <nav class="red">
                <div class="nav-wrapper">
                    <a v-bind:href="app.href" class="brand-logo">
                         {{app.name}}
                    </a>
                    <a href="#" data-activates="mobile-demo"
class="button-collapse">
                        <i class="material-icons">
                            menu
                        </i>
                    </a>
                    <ul class="right hide-on-med-and-down">
                        <li v-for="item in navbar.items">
                            <a v-bind:href="item.href">
                                {{item.label}}
                            </a>
                        </li>
                    </ul>
                    <ul class="side-nav" id="mobile-demo">
                        <li v-for="item in navbar.items">
                            <a v-bind:href="item.href">
                                {{item.label}}
                            </a>
                        </li>
                    </ul>
                </div>
            </nav>
            <!-- navbar -->
            <!-- loader -->
            <div v-show="isLoading" class="progress">
                <div class="indeterminate"></div>
            </div>
            <!-- loader -->
            <!-- registers -->
            <div class="container">
                <h3 class="header">
                    {{list.title}}
                </h3>
                <!-- no registers -->
                <div v-if="list.registers.length==0 && !isLoading">
                    <div class="card horizontal">
                        <div class="card-stacked">
```

```
                            <div class="card-content">
                                {{list.empty}}
                            </div>
                        </div>
                    </div>
                </div>
                <!-- no registers -->
                <!-- register -->
                <div v-for="register in list.registers" class="card
horizontal">
                    <!--<div class="card-image">
                        <img
src="https://lorempixel.com/100/190/nature/6">
                    </div>-->
                    <div class="card-stacked">
                        <div class="card-content">
                            <div>
                                {{register.employee}}
                            </div>
                            <div>
                                {{formatDate(register.created_at)}}
                            </div>
                        </div>
                        <!--<div class="card-action">
                            <a v-
on:click="eval(action.onClick+'(employee)')" v-for="action in
list.actions">
                                {{action.label}}
                            </a>
                        </div>-->
                    </div>
                </div>
                <!-- register -->
            </div>
            <!-- registers -->

            <script src="/plugins/app/js/app.js"></script>
        <!-- app -->

        <!-- vue -->
        <script src="/plugins/app/js/registers.js"></script>
        <!-- vue -->
    </body></html>
```

Anexo 7. Página web secundaria que muestra el registro de entrada de los empleados del local

Fuentes: Los Autores

```
const database=require('/home/pi/Downloads/migrations/database.js')
const connection=database.getConnection()

console.log('creating users')
connection.query('drop table if exists users',(error,rows)=>{
    if(error){
        throw error
    }
    connection.query('create table users (id integer unsigned
auto_increment primary key,email varchar(50) not null,password
varchar(50) not null,created_at datetime default now())
engine=innodb',(error,rows)=>{
        if(error){
            throw error
        }
        console.log('users table created')
        connection.query('insert into users (email,password) values
("admin@mail.com","admin")',(error,rows)=>{
            if(error){
                throw error
            }
            console.log('default user created')
        })
    })
})
```

Anexo 8. Programación de la pagina de autenticación por protocolo de seguridad de clave de usuario.

Fuentes: Los Autores

```
const vue=new Vue({
    el:'#app',
    data:{
        app:{
            href:'/',
            name:'Biometric Pi',
            isAuthenticated:false,
        },
        login:{
            email:'',
            password:'',
        },
        urls:{
            employees:{
                get:'/api/employees',
```

```
            create:'/api/employees',
            delete:'/api/employees',
            update:'/api/employees',
            fingerprint:'/api/fingerprint',
        },
        notifications:{
            get:'/api/notifications',
        },
        login:'api/login'
    },
    navbar:{
        items:[
            {
                label:'Empleados',
                href:'/employees',
            },
            {
                label:'Registro',
                href:'/registers',
            },
            /*{
                label:'Mi cuenta',
                href:'/account',
            },*/
            {
                label:'Salir',
                href:'/logout',
            },
        ]
    },
    isLoading:true,
    isCreating:false,
    isDeleting:false,
    isUpdating:false,
    list:{
        title:'Empelados',
        empty:'No hay empleados registrados en el sistema',
        actions:[
            {
                label:'Editar',
                onClick:'showUpdateModal',
            },
            {
                label:'Eliminar',
                onClick:'showDeleteModal',
            },
            {
                label:'Cambiar huella',
                onClick:'changeFingerprint',
```

```
            },
        ],
        employees:[]
    },
    modals:{
        delete:{
            title:'Eliminar empleado',
            message:'¿Realmente desea eliminar al empleado',
            actions:[
                {
                    onClick:'deleteEmployee',
                    label:'Eliminar',
                },
                {
                    onClick:'closeDeleteModal',
                    label:'Cerrar',
                },
            ],
            employee:null,
        },
        update:{
            title:'Editar empleado',
            fields:{
                name:{
                    label:'Nombre',
                },
                card:{
                    label:'Tarjeta',
                },
            },
            actions:[
                {
                    onClick:'updateEmployee',
                    label:'Actualizar',
                },
                {
                    onClick:'closeUpdateModal',
                    label:'Cerrar',
                },
            ],
            employee:null,
        },
        create:{
            title:'Crear empleado',
            fields:{
                name:{
                    label:'Nombre',
                },
                card:{
```

```
                        label:'Tarjeta',
                    },
                },
                actions:[
                    {
                        onClick:'createEmployee',
                        label:'Crear',
                    },
                    {
                        onClick:'closeCreateModal',
                        label:'Cerrar',
                    },
                ],
                employee:null,
            }
        }
    },
    mounted:function(){
        this.getEmployees()
        this.fetchNotifications()
    },
    methods:{
        fetchNotifications:function(){
            axios.get(this.urls.notifications.get).then(response=>{
                response.data.notifications.map(notification=>{
                    Materialize.toast(notification,4000)
                })
            }).finally(()=>{
                setTimeout(this.fetchNotifications,1000)
            })
        },
        loginUser:function(){

axios.post(this.urls.login,{email:this.login.email,password:this.login.
password}).then(response=>{
                if(response.data=='Bienvenido'){
                    this.app.isAuthenticated=true
                }
                Materialize.toast(response.data,4000)
                setTimeout(()=>{
                    $(".button-collapse").sideNav()
                },1000)
            }).catch(error=>{
                Materialize.toast(error.response.data,4000)
            })
        },
        getEmployees:function(){
            this.isLoading=true
            axios.get(this.urls.employees.get).then(response=>{
```

```
                this.list.employees=response.data.employees
                this.isLoading=false
            }).catch(error=>{
                Materialize.toast(error.response.data,4000)
                this.isLoading=false
            })
        },
        showCreateModal:function(){
            this.modals.create.employee={
                name:'',
                card:'',
            }
        },
        closeCreateModal:function(){
            this.modals.create.employee=null
        },
        createEmployee:function(){
            this.isCreating=true

axios.post(this.urls.employees.create,this.modals.create.employee).then
(response=>{
                Materialize.toast(response.data,4000)
                this.isCreating=false
                this.closeCreateModal()
                this.getEmployees()
            }).catch(error=>{
                Materialize.toast(error.response.data,4000)
                this.isCreating=false
            })
        },
        showDeleteModal:function(employee){
            this.modals.delete.employee=employee
        },
        closeDeleteModal:function(employee){
            this.modals.delete.employee=null
        },
        deleteEmployee:function()
        {
            this.isDeleting=true

axios.delete(this.urls.employees.delete+'/'+this.modals.delete.employee
.id).then(response=>{
                Materialize.toast(response.data,4000)
                this.isDeleting=false
                this.closeDeleteModal()
                this.getEmployees()
            }).catch(error=>{
                Materialize.toast(error.response.data,4000)
                this.isDeleting=false
```

```
            })
        },
        showUpdateModal:function(employee){
            this.modals.update.employee={
                id:employee.id,
                name:employee.name,
                card:employee.card,
                created_at:employee.created_at,
            }
        },
        closeUpdateModal:function(employee){
            this.modals.update.employee=null
        },
        updateEmployee:function(){
            this.isUpdating=true

axios.patch(this.urls.employees.update+'/'+this.modals.update.employee.
id,this.modals.update.employee).then(response=>{
                Materialize.toast(response.data,4000)
                this.isUpdating=false
                this.closeUpdateModal()
                this.getEmployees()
            }).catch(error=>{
                Materialize.toast(error.response.data,4000)
                this.isUpdating=false
            })
        },
        changeFingerprint:function(employee){

axios.get(this.urls.employees.fingerprint+'/'+employee.code).then(respo
nse=>{
                Materialize.toast(response.data,4000)
            }).catch(error=>{
                Materialize.toast(error.response.data,4000)
            })
        },
        twoDigits:function(value){
            return value<10?'0'+value:value
        },
        formatDate:function(datetime){
            datetime=new Date(datetime)
            const year=datetime.getFullYear()
            const month=this.twoDigits(datetime.getMonth()+1)
            const day=this.twoDigits(datetime.getDate())

            const hours=this.twoDigits(datetime.getHours())
            const minutes=this.twoDigits(datetime.getMinutes())
            const seconds=this.twoDigits(datetime.getSeconds())
```

```
            return year+'/'+month+'/'+day+'
'+hours+':'+minutes+':'+seconds
        }
    },
})
```

Anexo 9. Programacion de la estructura del aplicativo web principal
Fuentes: Los Autores

```
<template lang="pug">
    div
        .container
            h3.header Empleados
            Empty(v-bind:list='employees')
            Card(v-bind:key='employee.id' v-for='employee in
employees')
                div(slot='content')
                    div {{employee.name}}
                    div {{employee.card}}
                    div {{formatDate(employee.created_at)}}
                div(slot='actions')
                    .modal-action.waves-effect.btn-flat(v-
on:click='showUpdateModal(employee)') Editar
                    .modal-action.waves-effect.btn-flat(v-
on:click='showDeleteModal(employee)') Eliminar
                    .modal-action.waves-effect.btn-flat(v-
on:click='changeFingerprint(employee)') Cambiar huella
        Fab(v-on:clicked='showCreateModal')

        Modal(v-if='employeeToDelete')
            div(slot='content')
                h4 Eliminar empleado
                p Realmente desea eliminar a
"{{employeeToDelete.name}}"
                Loading(v-if='isDeleting')
            div(slot='footer')
                a.modal-action.waves-effect.waves-green.btn-flat(v-
on:click='deleteEmployee') Eliminar
                a.modal-action.waves-effect.waves-green.btn-flat(v-
on:click='closeDeleteModal') Cerrar

        Modal(v-if='employeeToUpdate')
            div(slot='content')
                h4 Actulizar empleado
                Loading(v-if='isUpdating')
                form(v-on:submit.prevent="updateEmployee")
```

```
                    .input-field.col.s6
                        input.validate(v-model='employeeToUpdate.name'
id='name' type='text')
                        label(for='name') Nombre del empleado
                    .input-field.col.s6
                        input.validate(v-model='employeeToUpdate.card'
id='card' type='text')
                        label(for='card') Tarjeta del empleado
            div(slot='footer')
                a.modal-action.waves-effect.waves-green.btn-flat(v-
on:click='updateEmployee') Actualizar
                a.modal-action.waves-effect.waves-green.btn-flat(v-
on:click='closeUpdateModal') Cerrar

        Modal(v-if='employeeToCreate')
            div(slot='content')
                h4 Crear empleado
                Loading(v-if='isCreating')
                form(v-on:submit.prevent="createEmployee")
                    .input-field.col.s6
                        input.validate(v-model='employeeToCreate.name'
id='name' type='text')
                        label(for='name') Nombre del empleado
                    .input-field.col.s6
                        input.validate(v-model='employeeToCreate.card'
id='card' type='text')
                        label(for='card') Tarjeta del empleado
            div(slot='footer')
                a.modal-action.waves-effect.waves-green.btn-flat(v-
on:click='createEmployee') Crear
                a.modal-action.waves-effect.waves-green.btn-flat(v-
on:click='closeCreateModal') Cerrar

</template>
<script>
    import {mapState} from 'vuex'
    import Navbar from '../partials/navbar.vue'
    import Empty from '../partials/empty.vue'
    import Fab from '../partials/fab.vue'
    import Card from '../partials/card.vue'
    import Modal from '../partials/modal.vue'
    import Loading from '../partials/loading.vue'
    import {formatDate} from '../../tools/date'
    import axios from 'axios'

    export default {
        data(){
            return{
                imAlive:true,
```

```
            employeeToDelete:null,
            isDeleting:false,
            employeeToUpdate:null,
            isUpdating:false,
            employeeToCreate:null,
            isCreating:false,
            employees:[
                {
                    id:1,
                    code:10,
                    name:'Danny Vaca',
                    card:'27071993',
                    created_at:new Date()
                }
            ]
        }
    },
    created(){
        this.getEmployees()
        this.fetchNotifications()
    },
    beforeDestroy(){
        this.imAlive=false
    },
    methods:{
        fetchNotifications:function(){
            axios.get('api/notifications').then(response=>{
                response.data.notifications.map(notification=>{
                    Materialize.toast(notification,4000)
                })
                if(this.imAlive){
                    setTimeout(this.fetchNotifications,1000)
                }
            }).catch((error)=>{
                if(this.imAlive){
                    setTimeout(this.fetchNotifications,1000)
                }
            })
        },
        getEmployees:function(){
            this.$store.commit('loading',true)
            axios.get('/api/employees').then(response=>{
                this.employees=response.data.employees
                this.$store.commit('loading',false)
            }).catch(error=>{
                console.log(error)
                Materialize.toast(error.response.data,4000)
                this.$store.commit('loading',false)
            })
```

```
            },
            showDeleteModal(employee){
                this.employeeToDelete=employee
            },
            closeDeleteModal(){
                this.employeeToDelete=null
            },
            deleteEmployee(){
                this.isDeleting=true

axios.delete('/api/employees/'+this.employeeToDelete.id).then(response=
>{
                    Materialize.toast(response.data,4000)
                    this.closeDeleteModal()
                    this.getEmployees()
                    this.isDeleting=false
                }).catch(error=>{
                    console.log(error)
                    Materialize.toast(error.response.data,4000)
                    this.isDeleting=false
                })
            },
            showUpdateModal(employee){
                this.employeeToUpdate=employee
            },
            closeUpdateModal(){
                this.employeeToUpdate=null
            },
            updateEmployee(){
                this.isUpdating=true

axios.patch('/api/employees/'+this.employeeToUpdate.id,this.employeeToU
pdate).then(response=>{
                    Materialize.toast(response.data,4000)
                    this.closeUpdateModal()
                    this.getEmployees()
                    this.isUpdating=false
                }).catch(error=>{
                    console.log(error)
                    Materialize.toast(error.response.data,4000)
                    this.isUpdating=false
                })
            },
            changeFingerprint(employee){

axios.get('/api/fingerprint/'+employee.code).then(response=>{
                    Materialize.toast(response.data,4000)
                }).catch(error=>{
                    console.log(error)
```

```
                    Materialize.toast(error.response.data,4000)
                })
            },
            showCreateModal(){
                this.employeeToCreate={
                    name:'',
                    card:''
                }
            },
            closeCreateModal(){
                this.employeeToCreate=null
            },
            createEmployee(){
                this.isCreating=true

axios.post('/api/employees',this.employeeToCreate).then(response=>{
                    Materialize.toast(response.data,4000)
                    this.closeCreateModal()
                    this.getEmployees()
                    this.isCreating=false
                }).catch(error=>{
                    console.log(error)
                    Materialize.toast(error.response.data,4000)
                    this.isCreating=false
                })
            },
            formatDate(date){
                return formatDate(date)
            }
        },
        computed:{
            ...mapState(['isLoading'])
        },
        components:{
            Navbar,
            Empty,
            Fab,
            Card,
            Modal,
            Loading
        }
    }
</script>
```

Anexo 10. Framework Vuejs para la pagina de empleados de la base de datos.

Fuentes: Los Autores

```
const vue=new Vue({
    el:'#app',
    data:{
        app:{
            href:'/',
            name:'Biometric Pi',
        },
        urls:{
            registers:{
                get:'/api/registers',
            },
        },
        navbar:{
            items:[
                {
                    label:'Empleados',
                    href:'/employees',
                },
                {
                    label:'Registro',
                    href:'/registers',
                },
                {
                    label:'Salir',
                    href:'/employees',
                },
            ]
        },
        isLoading:true,
        list:{
            title:'Registros',
            empty:'No hay registros de accesos en el sistema',
            registers:[]
        },
    },
    mounted:function(){
        this.getRegisters()
        $(".button-collapse").sideNav()
    },
    methods:{
        getRegisters:function(){
            this.isLoading=true
            axios.get(this.urls.registers.get).then(response=>{
                this.list.registers=response.data.registers
                this.isLoading=false
            }).catch(error=>{
                Materialize.toast(error.response.data,4000)
                this.isLoading=false
            })
```

```
        },
        twoDigits:function(value){
            return value<10?'0'+value:value
        },
        formatDate:function(datetime){
            datetime=new Date(datetime)
            const year=datetime.getFullYear()
            const month=this.twoDigits(datetime.getMonth()+1)
            const day=this.twoDigits(datetime.getDate())
            const hours=this.twoDigits(datetime.getHours())
            const minutes=this.twoDigits(datetime.getMinutes())
            const seconds=this.twoDigits(datetime.getSeconds())
            return year+'/'+month+'/'+day+'
'+hours+':'+minutes+':'+seconds
        }
    },})
```

Anexo 11. Estructura del aplicativo web de la pagina de Registro

Fuentes: Los Autores

```
<template lang="pug">
    div
        .container
            h3.header Registros
            Empty(v-bind:list='registers')
            Card(v-bind:key='register.id' v-for='register in
registers')
                div(slot='content')
                    div {{register.employee}}
                    div {{formatDate(register.created_at)}}
</template>
<script>
    import {mapState} from 'vuex'
    import Navbar from '../partials/navbar.vue'
    import Empty from '../partials/empty.vue'
    import Card from '../partials/card.vue'
    import {formatDate} from '../../tools/date'
    import axios from 'axios'
    export default {
        data(){
            return{
                registers:[]
            }
        },
        created(){
            this.getRegisters()
```

```
        },
        methods:{
            getRegisters:function(){
                this.$store.commit('loading',true)
                axios.get('/api/registers').then(response=>{
                    this.registers=response.data.registers
                    this.$store.commit('loading',false)
                }).catch(error=>{
                    console.log(error)
                    Materialize.toast(error.response.data,4000)
                    this.$store.commit('loading',false)
                })
            },
            formatDate(date){
                return formatDate(date)
            }
        },
        components:{
            Navbar,
            Empty,
            Card
        }
    }
</script>
```

Anexo 12. Framework Vuejs para la pagina de empleados de la base de datos.

Fuentes: Los Autores

```
var exec=require('child_process').exec
const express=require('express')
var path = require('path');
const router=express.Router()
const logger=require(__dirname+'/../tools/logger')
const database=require(__dirname+'/../database/database.js')

var lastArduinoMessage=new Date()
const checkInterval=10000
setInterval(()=>{
    const now=new Date()
    const interval=now-lastArduinoMessage
    if(interval>checkInterval){
        const connection=database.getConnection()
        console.log('arduino down')
        connection.query('insert into notifications (content) values
("El arduino está caido")',(error,employees)=>{
```

```
            if(error){
                logger.error(error)
            }
        })
    }
},checkInterval)

router.get('/employees',(request,response)=>{
    const connection=database.getConnection()
    connection.query('select id,name,card,code,created_at from
employees',(error,employees)=>{
        if(error){
              logger.error(error)
            return response.status(500).send('Error obteniendo
empleados')
        }
        response.json({employees:employees})
    })
})

router.post('/employees',(request,response)=>{
    const connection=database.getConnection()
    connection.query('select code from employees order by
code',(error,rows)=>{
        existsCodeInArray=(array,code)=>{
            for(index in array) if(parseInt(array[index].code)==code)
return true
        }
        var code=1
        if(rows.length!=0){
            while(existsCodeInArray(rows,code)){
                code++
            }
        }
        connection.query('insert into employees (name,card,code) values
("'+request.body.name+'","'+request.body.card+'","'+code+'")',(error,ro
ws)=>{
            if(error){
                logger.error(error)
                return response.status(500).send('Error creando
empleado')
            }
            return response.send('Empleado creado')
        })
    })
})
router.delete('/employees/:id',(request,response)=>{
    const connection=database.getConnection()
```

```
    connection.query('select code from employees where
id="'+request.params.id+'"',(error,rows)=>{
        if(error){
            return response.send('Error conectando a la base de datos')
        }
        code=rows[0]['code']
        connection.query('select count(*) as quantity from actions
where name="remove" and code="'+request.params.code+'"',(error,rows)=>{
            if(rows[0]['quantity']>0){
                return response.send('Solicitud ya enviada')
            }
            connection.query('insert into actions (code,name) values
("'+code+'","remove")',(error,rows)=>{
                if(error){
                    return response.send('Error borrando huella')
                }
                connection.query('delete from employees where
id="'+request.params.id+'"',(error,rows)=>{
                    if(error){
                        return response.send('Error borrando empleado')
                    }
                    return response.send('Borrando empleado')
                })
            })
        })
    })
})
router.patch('/employees/:id',(request,response)=>{
    const connection=database.getConnection()
    connection.query('update employees set
name="'+request.body.name+'",card="'+request.body.card+'" where
id="'+request.params.id+'"',(error,rows)=>{
        if(error){
            return response.status(500).send('Error actualizando
empleado')
        }
        return response.send('Empleado actualizado')
    })
})

router.get('/registers',(request,response)=>{
    const connection=database.getConnection()
    connection.query('select registers.created_at,employees.name as
employee from registers inner join employees on
employees.id=registers.employee_id order by registers.id
desc',(error,registers)=>{
        if(error){
            return response.status(500).send('Error obteniendo
registros')
```

```
        }
        response.json({registers:registers})
    })
})

router.get('/cards/:card',(request,response)=>{
    const connection=database.getConnection()
    connection.query('select id,name from employees where
card="'+request.params.card+'"',(error,employees)=>{
        if(error){
            return response.status(500).send('Error buscando tarjeta')
        }
        if(employees.length==0){
            return response.status(404).send('No existe la tarjeta')
        }
        connection.query('insert into registers (employee_id) values
("'+employees[0].id+'")',(error)=>{
            if(error){
                return response.status(500).send('Error registrando el
ingreso')
            }
            return response.json({employee:employees[0]})
        })
    })
})
router.get('/fingerprint/:code',(request,response)=>{
    const connection=database.getConnection()
    connection.query('select count(*) as quantity from actions where
name="add" and code="'+request.params.code+'"',(error,rows)=>{
        if(error){
            return response.send('Error buscando huella')
        }
        if(rows[0]['quantity']>0){
            return response.send('Solicitud ya enviada')
        }
        connection.query('insert into actions (code,name) values
("'+request.params.code+'","add")',(error,rows)=>{
            if(error){
                return response.send('Error agregando huella')
            }
            return response.send('Agregando huella')
        })
    })
})
router.get('/notifications',(request,response)=>{
    const connection=database.getConnection()
    connection.query('select content from
notifications',(error,rows)=>{
        notifications=[]
```

```
      if(rows){
            rows.map(row=>{
                  notifications.push(row.content)
              })
            response.json({notifications:notifications})
              connection.query('truncate table notifications')
      }else{
            response.json({notifications:[]})
      }
    })

})

router.post('/login',(request,response)=>{
    const connection=database.getConnection()
    const email=request.body.email
    const password=request.body.password
    connection.query('select count(*) as counter from users where
email="'+email+'" and password="'+password+'"',(error,rows)=>{
        if(error){
            return response.send('Error en la base de datos')
        }
        if(rows[0]['counter']==0){
            return response.send('Credenciales incorrectas')
        }
        return response.send('Bienvenido')
    })
})
router.get('/arduino',(request,response)=>{
    const state=request.query.state
    if(state==='1'){
        lastArduinoMessage=new Date()
      console.log('arduino keep alive received')
    }
    response.send()
})
module.exports=router
```

Anexo 13. Programacion basado en Javascript en NodeJS para realizar una pagina web dinámica.
Fuentes: Los Autores

ANEXO

DESARROLLO DEL CODIGO DE LA RASPBERRY PI ZERO

```
<!DOCTYPE html>
<html lang=en>
      <head>
      <meta charset=utf-8>
      <meta http-equiv=X-UA-Compatible content="IE=edge">
      <meta name=viewport content="width=device-width,initial-scale=1">
      <title>photos</title>
      <link rel=icon type=image/png sizes=32x32
href=/static/img/icons/favicon-32x32.png><link rel=icon type=image/png
sizes=16x16 href=/static/img/icons/favicon-16x16.png><link rel=manifest
href=/static/manifest.json><meta name=theme-color content=#4DBA87><meta
name=apple-mobile-web-app-capable content=yes><meta name=apple-mobile-
web-app-status-bar-style content=black><meta name=apple-mobile-web-app-
title content=photos>
<link rel=apple-touch-icon href=/static/img/icons/apple-touch-icon-
152x152.png><link rel=mask-icon href=/static/img/icons/safari-pinned-
tab.svg color=#4DBA87><meta name=msapplication-TileImage
content=/static/img/icons/msapplication-icon-144x144.png><meta
name=msapplication-TileColor content=#000000><link rel=preload
href=/static/js/vendor.464a22de5cb9edd95f02.js as=script><link
rel=preload href=/static/js/app.15148c152612c109f796.js as=script>
<link rel=preload
href=/static/css/app.d84fa79e6719a032e4dbc027ce90180b.css as=style>
<link rel=preload href=/static/js/manifest.2ae2e69a05c33dfc65f8.js
as=script>
<link href=/static/css/app.d84fa79e6719a032e4dbc027ce90180b.css
rel=stylesheet>
</head>
      <body>
      <noscript>This is your fallback content in case JavaScript fails
to load.</noscript><div id=app></div>
<script>!function(){"use strict";var
o=Boolean("localhost"===window.location.hostname||"[::1]"===window.loca
tion.hostname||window.location.hostname.match(/^127(?:\.(?:25[0-5]|2[0-
4][0-9]|[01]?[0-9][0-
9]?)){3}$/));window.addEventListener("load",function(){"serviceWorker"i
n
navigator&&("https:"===window.location.protocol||o)&&navigator.serviceW
orker.register("service-
worker.js").then(function(o){o.onupdatefound=function(){if(navigator.se
rviceWorker.controller){var
n=o.installing;n.onstatechange=function(){switch(n.state){case"installe
d":break;case"redundant":throw new Error("The installing service worker
became redundant.")}}}}}).catch(function(o){console.error("Error during
service worker registration:",o)})})}();</script>
<script type=text/javascript
src=/static/js/manifest.2ae2e69a05c33dfc65f8.js></script>
<script type=text/javascript
src=/static/js/vendor.464a22de5cb9edd95f02.js></script>
```

```
<script type=text/javascript
src=/static/js/app.15148c152612c109f796.js></script>
      </body></html>
```

Anexo 1. Programación Aplicativo web de las imágenes fotograficas

Fuentes: Los Autores

```
<template lang="pug">
  .container
    transition(name='fade' mode='out-in')
      Modal(v-if='imageToShow' v-on:close='imageToShow = null')
        figure.image
          img(v-bind:src='hostname + "/" + imageToShow' width='100%')
    Notification(v-bind:notification='notification' v-
on:close='notification = null')
    .columns(v-for='chunk in photos')
      .column(v-for='photo in chunk')
        .card
          .card-image
            figure.image
              img(v-bind:src='hostname + "/" + photo')
          .card-content
            .media
              .media-content.hide-overflow {{photo}}
          .card-footer
            a.card-footer-item(v-on:click='deletePhoto(photo)')
Eliminar
            a.card-footer-item(v-on:click='showPhoto(photo)') Ver
    Paginator(
      v-bind:last='paginator.last'
      v-bind:current='paginator.current'
      v-on:next='next'
      v-on:last='last'
      v-on:to='to')
    br
</template>
<script>
import photos from '@/services/photos'
import array from '@/tools/array'
import Notification from '@/components/Notification'
import Modal from '@/components/Modal'
import Paginator from '@/components/Paginator'
export default {
  components: {
    Notification,
```

```
    Modal,
    Paginator
  },
  data () {
    return {
      hostname: null,
      photos: [],
      notification: null,
      imageToShow: null,
      paginator: {
        perPage: 9,
        last: 0,
        current: 0
      } }
  },
  created () {
    this.hostname = window.location.protocol + '//' +
window.location.hostname + ':80'
    this.downloadPhotos()
  },
  methods: {
    split: array.split,
    downloadPhotos () {
      this.$store.commit('app/setLoading', true)
      photos.list(this.paginator.perPage,
this.paginator.current).then(paginator => {
        this.paginator.last = paginator.last
        this.paginator.current = paginator.current
        this.photos = array.chunk(paginator.items, 3)
      }).catch(error => {
        this.notification = {
          title: 'Error del servidor ' + error.response.status,
          type: 'error'
        }
      }).finally(() => {
        this.$store.commit('app/setLoading', false)
      })    },
    deletePhoto (image) {
      this.$store.commit('app/setLoading', true)
      photos.delete(image, this.paginator.perPage,
this.paginator.current).then(paginator => {
        this.paginator.last = paginator.last
        this.paginator.current = paginator.current
        this.photos = array.chunk(paginator.items, 3)
      }).catch(error => {
        this.notification = {
          title: 'Error del servidor ' + error.response.status,
          type: 'error'
        }
```

```
      }).finally(() => {
        this.$store.commit('app/setLoading', false)
      })
    },
    showPhoto (image) {
      this.imageToShow = image
    },
    next () {
      if (this.paginator.current <= this.paginator.last) {
        this.paginator.current++
        this.downloadPhotos()
      }
    },
    last () {
      if (this.paginator.current > 0) {
        this.paginator.current--
        this.downloadPhotos()
      }
    },
    to (page) {
      if (this.paginator.last > page) {
        this.paginator.current = page
        this.downloadPhotos()
      }
    }
  }
}
</script>
<style>
  .hide-overflow {
    width: 10px;
    white-space: nowrap;
    overflow: hidden;
    text-overflow: ellipsis;  }
</style>
```

Anexo 2. Framework de la pagina web de las imagenes fotograficas.

Fuentes: Los Autores

```
print('importing libraries')
from gpiozero import Button
import picamera
import os
import glob
PHOTO_PATH = '/var/www/html'
```

```
button = Button(2)
button2 = Button(4)
print('setting up camera')
camera = picamera.PiCamera()
while True:
    try:
        print('main program')
        while True:
            if not button.is_pressed or not button2.is:pressed:
                print('button is pressed')
                Numero = (glob.glob(PHOTO_PATH + '*.jpg'))
                if len(Numero) > 200:
                    print('delete frirst photo')
                    Numero.sort()
                    print(Numero[0])
                    os.remove(Numero[0])
                f = os.popen('sudo TZ=America/Guayaquil date +%F_%H-%M-
%S')
                now = f.read()
                now = now[0:-1]
                print(PHOTO_PATH + now + '.jpg')
                camera.capture(PHOTO_PATH + now + '.jpg')
                print('done')
    except Exception as error:
        print(error)
```

Anexo 3. Programación Python para la toma de fotos dentro del local por medio de la Raspberry pi Zero.

Fuentes: Los Autores

```
<template lang="pug">
  .columns
    .column
      br
    .column
      br
      .card
        .card-content
          form(v-on:submit.prevent='login')
            .field
              label.label Usuario
              .control
                input.input(type='text' v-model='username'
placeholder='Usuario' autofocus)
              //p.help.is-success This username is available
```

```
            .field
              label.label Contraseña
              .control
                input.input(type='password' v-model='password'
placeholder='Contraseña')
              //p.help.is-success This username is available
            .field.is-grouped
              .control
                button.button.is-link.is-fullwidth Login
    .column
      br
      Notification(v-bind:notification='notification' v-
on:close='notification = null')
</template>
<script>
  import base from '@/tools/faker/base'
  import Notification from '@/components/Notification'
  export default {
    components: {
      Notification
    },
    data () {
      return {
        username: '',
        password: '',
        notification: null
      }   },
    methods: {
      login () {
        this.$store.commit('app/setLoading', true)
        setTimeout(() => {
          if (this.username === 'admin' && this.password === 'admin') {
            this.$store.commit('auth/login', {token: base.token(20)})
          } else {
            this.notification = {
              title: 'No se pudo iniciar sesión',
              message: 'Las credenciales son incorrectas',
              type: 'error'
            }      }
          this.$store.commit('app/setLoading', false)
        }, base.numberBetween(500, 2000))
      }    }  }
</script>
```

Anexo 4. Pagina principal de ingreso con protocolos de seguridad mediante contraseña para el aplicativo de gestión de fotos

Fuentes: Los Autores

ANEXO

DATASHEET DE LOS EQUIPOS UTILIZADOS EN ELPROYECTO

TG862G DOCSIS® 3.0 Residential Gateway with 802.11n, 4 Port Router and 2 Voice Lines

Specifications

Physical	
	Continue to next page.
Operating Temperature ° F (° C)	41 to 122 (5 to 40)
Operating Relative Humidity (Min-Max)	5-85% (Non condensing)
Storage Temperature ° F (° C)	-40 to 158 (-40 to 70)
Color	Black
Dimensions (H x W x D) in.	9 x 8.2 x 2 - excluding F-connector
Weight lbs	No battery included: 1.8
Backup Capacity	Lithium-ion 2.6 Ah for 8 hours operation. Consult ARRIS for other battery holdup time options.
Battery Dimensions in.	1.2 x 1.85 x 3.1
Weight lbs	1.0
Battery Storage Temperature ° F (° C)	-4 to 140 (-20 to 60) Note: Storage above 77° F (25° C) will significantly reduce life of the battery and is not recommended.
Telemetry	AC Fail, Battery Low, Battery Missing, Replace Battery
Diagnostic LEDs	Power, DS, US, Online, Ethernet, WiFi, Secure, Tel 1, Tel 2, Battery
Interfaces	
RF Interface	External 'F' type connector
Data Interfaces (bridged)	4 x 10/100/1000 Base-T Ethernet (RJ-45 connector)
Telephony Interface	2 lines; RJ-14 ("Line 1/2"), RJ-11 ("Line 2")
USB Interface	USB 2.0 Powered Host Port
Input Voltage (nominal)	100-240 Vac 50/60 Hz
Telephony	
Supervisory Voltage	48 Vdc nominal
Maximum Loop Length to CPE	1000 ft (457M) of 26 AWG (0.4 mm) wire
Ringing Load Capacity	10 REN total; 5 per line
Provisionable High Loop Current Mode	Yes (40mA constant current source, NA templates ONLY)
Telcordia™ GR 1089 (Lightning and Power Surge) Tested	Yes
Programmable Interface for Worldwide Applications	Yes (supports multiple country templates)
RF Downstream	
Bonded Channels	Up to 8
Tuner Configuration	2 tuning ranges of 48MHz each supporting 4 bonded channels
Frequency Range (MHz)	108-1002 DOCSIS
Carrier Bandwidth (MHz)	6 (DOCSIS)
Modulation (QAM)	64 or 256
Data Rate (Mbps Max.)	Up to 320 DOCSIS
RF Input Sensitivity Level (dBmV)	-15 to +15 (DOCSIS))

Technical Specification

ARRIS Technical Specification

Specifications Continued

RF Upstream	
Bonded Channels	Up to 4
Frequency Range (MHz)	5 to 42 (DOCSIS)
Modulation	QPSK, 8 QAM, 16 QAM, 32 QAM, 64 QAM & 128 QAM (S-CDMA only)
Data Rate (Mbps Max.)	up to 160
RF Output Level (dBmV)	ATDMA: +8 to 54 dBmV (32 QAM, 64 QAM) +8 to 55 dBmV (8 QAM, 16 QAM) +8 to 58 dBmV (QPSK) S-CDMA: +8 to +53 dBmV (all modulations)
Automatic Level Adjust	Yes
Frequency Stability (kHz)	±5
Output Impedance (Ohms)	75

Wireless	
Transmit Power Output (EIRP)	19.5dBm +1.5/-1.5dB 802.11b; 17.5dBm +1.5/-1.5dB 802.11n (MCS 7)
Receive Levels	>-86dBM 802.11b 11mbps; >-76dBm 802.11g 54mbps, >-76dBM 802.11n HT20 MCS7
Frequency Range	2400-2483.5 MHz
Antennas	2 transmit and receive

Standards

DOCSIS 3.0	IEEE 802.3, IEEE 802.3ab, IEEE 802.3x
PacketCable 1.0 & 1.5	IEEE 802.1p, IEEE 802.1Q
Codec: G.711, 64 kbps, μ and A-law encoded speech	IEEE Std. 802.11b, 1999 Edition
Enhanced codec firmware support (G.726, G.828, G.729E)	IEEE Std. 802.11g, 2002 Edition
T.38 Fax Relay	IEEE Std. 802.11n
	WiFi Alliance Certified
	UL® 60950
	FCC Part 15 Class B

Battery back-up times are typical and can be influenced by the age of the battery, the charging state, storage conditions and operating temperature as well as factors such as data activity and length of active telephone calls. The capabilities, system requirements and/or compatibility with third-party products described herein are subject to change without notice . Actual operating range may vary according to environmental conditions at the time of use.

Ordering Information

Part Number	Description
DOCSIS 3.0 Models	PacketCable 1.0/1.5 Compliant Gateway, Integrated 100-240 VAC, 50/60 HZ Power supply with 6 foot power cord . Includes Quick Install Guide, Ethernet Cable, and CD-ROM with users guide
790682	Touchstone TG862G/CE-0 - CEE 7/16 Euro power plug - battery capable but no battery included
790683	Touchstone TG862G/CE-8 - CEE 7/16 Euro power plug - 2 cell, Li-Ion Battery Pack for up to 8 hours of battery back-up
790680	Touchstone TG862G/NA-0 - NEMA 1 power plug - battery capable but no battery included
790681	Touchstone TG862G/NA-8 - NEMA 1 power plug - 2 cell, Li-Ion Battery Pack for up to 8 hours of battery back-up
Accessories	
789699	2-Cell, 2.6 Ah, Li-Ion Battery Pack for TG862 (Battery models)
722805	North American NEMA 1-15 plug Power Cord, 6 foot
721197	European CEE 7/16 plug Power Cord, 6 foot

Specifications are subject to change without notice.

ARRIS

www.arrisi.com

Anexo 1. Datasheet Modem Arris

Fuentes: Arris

tp-link

300Mbps High Power Wireless N Router

Boosted Coverage,
Breaks Through Walls

300Mbps | 9dBi Antennas | Maximum Range

TL-WR841HP

Specifications

Hardware

- **Ethernet Ports:** 4 10/100Mbps LAN Ports, 1 10/100Mbps WAN Port
- **Buttons:** Power On/Off Button, Wi-Fi On/Off Button, Reset Button, WPS Button, RE Button
- **Antennas:** Two 9dBi Detachable Antennas
- **External Power Supply:** 12V/1A
- **Dimensions (W x D x H):** 9.0× 7.5×1.9 in. (228×190×48mm)

Wireless

- **Wireless Standards:** IEEE 802.11b/g/n
- **Frequency:** 2.4GHz
- **Signal Rate:** 300Mbps
- **Transmit Power:** CE: < 20dBm, FCC: <30dBm
- **Reception Sensitivity:**
  2.4GHz:
  270M: -71dBm@10% PER
  130M: -74dBm@10% PER
  108M: -74dBm@10% PER
  54M: -77dBm@10% PER
  11M: -94dBm@10% PER
  6M: -93dBm@10% PER
  1M: -97dBm@10% PER
- **Wireless Function:** Enable/Disable Wireless Radio, WDS Bridge, WMM, Wireless Statistics
- **Wireless Security:** 64/128-bit WEP, WPA/WPA2, WPA-PSK/WPA2-PSK encryptions

Software

Quality of Service: WMM, Bandwidth Control

WAN Type: Dynamic IP/Static IP/PPPoE/PPTP(Dual Access)/L2TP(Dual Access)/BigPond

Management: Access Control, Local Management, Remote Management

DHCP: Server, DHCP Client List, Address Reservation

Port Forwarding: Virtual Server, Port Triggering, UPnP, DMZ

Dynamic DNS: DynDns, NO-IP

Access Control: Parental Controls, Local Management Control, Host List, White List, Black List

Firewall Security: DoS, SPI Firewall, IP and MAC Address Binding

Protocols: IPv4, IPv6

Guest Network: 2.4GHz guest network

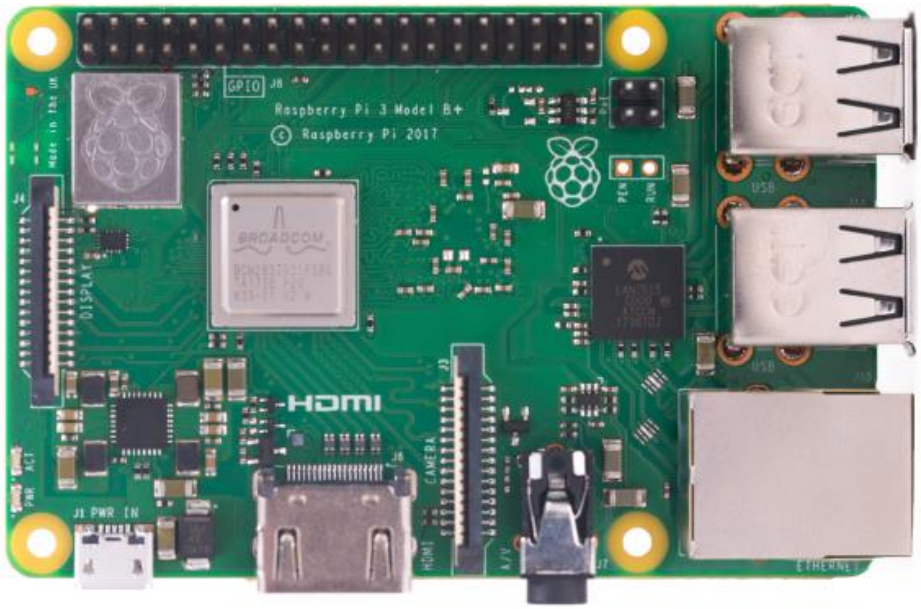
Others

- **Certification:**
  CE, FCC, RoHS
- **System Requirements:**
  Microsoft Windows 98SE/NT/2000/XP/Vista™/7/8/8.1/10, MAC OS, NetWare, UNIX or Linux
  Internet Explorer 11, Firefox 12.0, Chrome 20.0, Safari 4.0, or other Java-enabled browser
- **Environment:**
  Operating Temperature: 0℃~40℃ (32℉~104℉)
  Storage Temperature: -40℃~70℃ (-40℉~158℉)
  Operating Humidity: 10%~90% non-condensing
  Storage Humidity: 5%~90% non-condensing
- **Package Contents**
  High Power Wireless Router TL-WR841HP
  Power Supply Unit
  Ethernet Cable
  Quick Installation Guide

Anexo 2. Datasheet Router TP LINK WR-841 HP

Fuentes: tp-link

Overview



The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT

The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market.

The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

raspberrypi.org

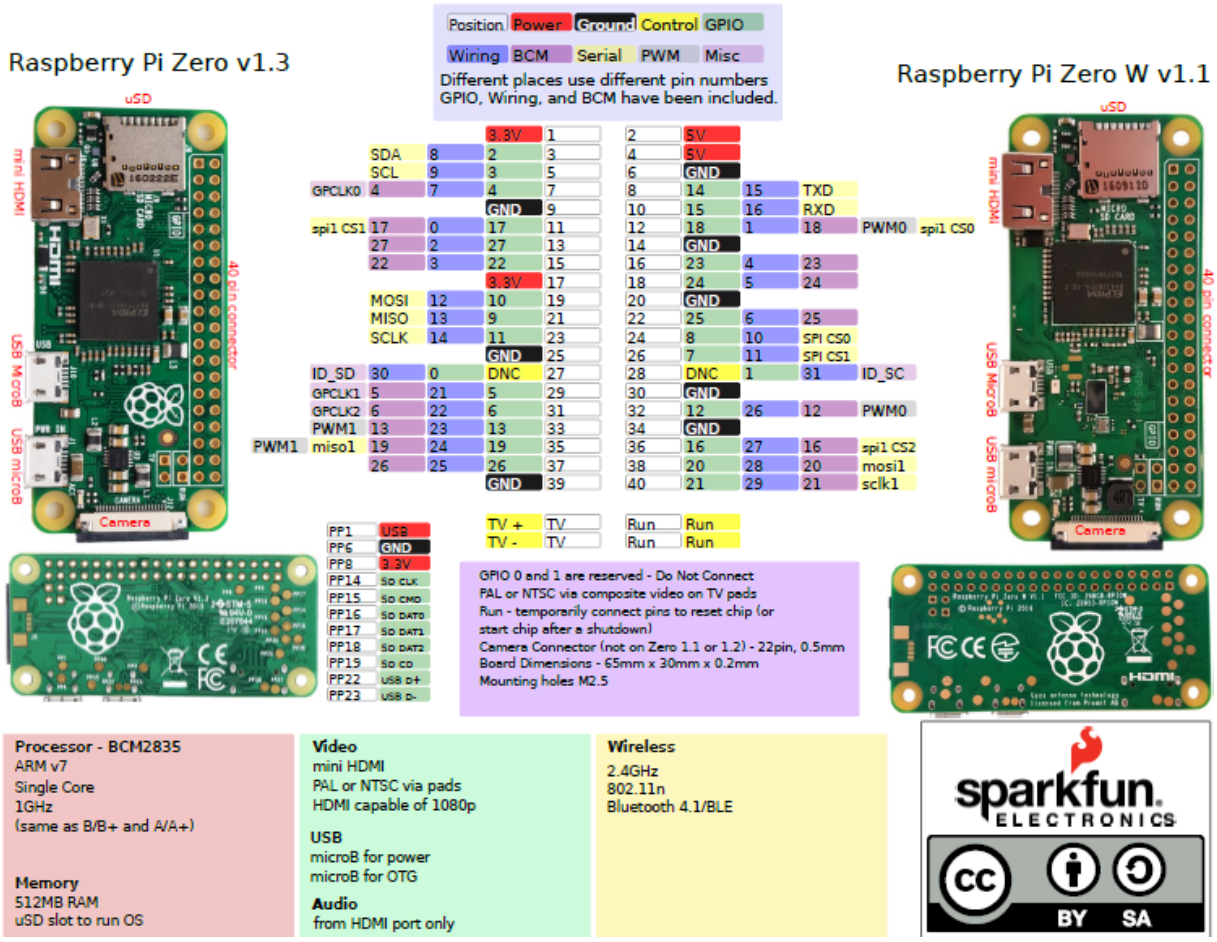
Specifications

Processor:	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz
Memory:	1GB LPDDR2 SDRAM
Connectivity:	■ 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE ■ Gigabit Ethernet over USB 2.0 (maximum throughput 300Mbps) ■ 4 × USB 2.0 ports
Access:	Extended 40-pin GPIO header
Video & sound:	■ 1 × full size HDMI ■ MIPI DSI display port ■ MIPI CSI camera port ■ 4 pole stereo output and composite video port
Multimedia:	H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
SD card support:	Micro SD format for loading operating system and data storage
Input power:	■ 5V/2.5A DC via micro USB connector ■ 5V DC via GPIO header ■ Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment:	Operating temperature, 0–50°C
Compliance:	For a full list of local and regional product approvals, please visit www.raspberrypi.org/products/raspberry-pi-3-model-b+
Production lifetime:	The Raspberry Pi 3 Model B+ will remain in production until at least January 2023.

raspberrypi.org

Anexo 3. Datasheet Raspberry Pi 3 B+

Fuentes: Raspberry.org



Anexo 4. Datasheet Raspberry Pi Zero W V1.1

Fuentes: Sparkfun Electronics

Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-9V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) (0.5 KB used by bootloader)
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm centre-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

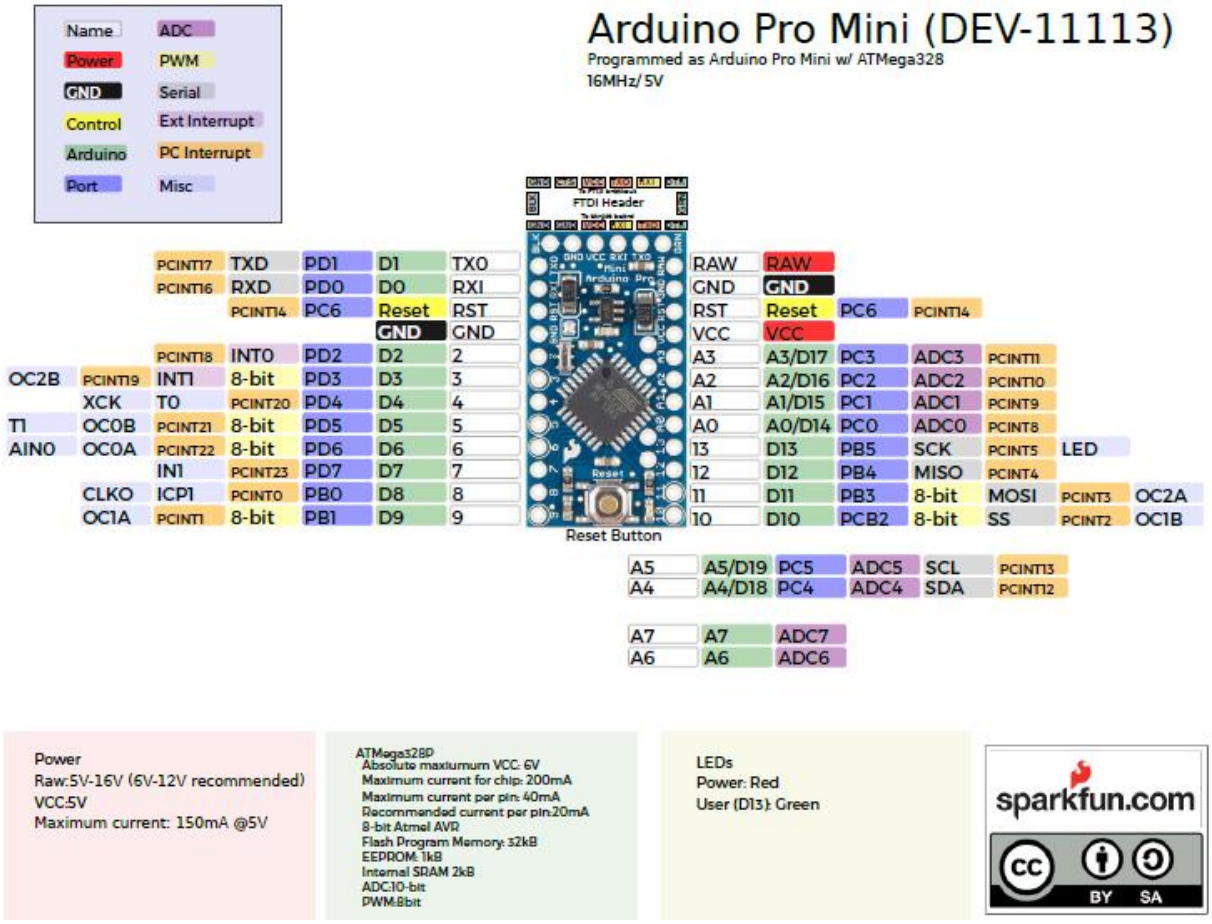
Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication using the SPI library.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

Anexo 5. Datasheet Arduino Uno

Fuentes: Arduino



Anexo 6. Datasheet Arduino Pro Mini

Fuentes: Sparkfun Electronics

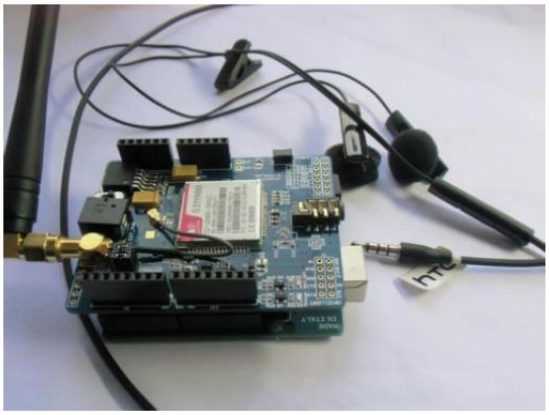
AZL GPRS/GSM SHIELD SIM900

Introduction

The GPRS/GSM Shield provides you a way to use the GSM cell phone network to receive data from a remote location. The shield allows you to achieve this via any of the three methods:

- Short Message Service
- Audio
- GPRS Service

The GPRS/GSM Shield is compatible with all boards which have the same form factor (and pin out) as a standard Arduino Board. The GPRS/GSM Shield is configured and controlled via its UART using simple AT commands. Based on the SIM900 module from SIMCOM, it is like a cell phone. Besides the communications features, the GPRS/GSM Shield has 6 GPIOs, 2 PWMs and an ADC.



Features

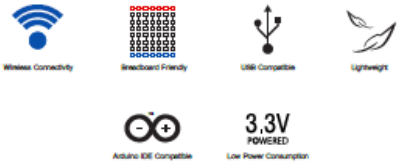
- **Quad-Band 850 / 900/ 1800 / 1900 MHz** - would work on GSM networks in all countries across the world.
- **GPRS multi-slot class 10/8**
- **GPRS mobile station class B**
- **Compliant to GSM phase 2/2+**
  - **Class 4 (2 W @ 850 / 900 MHz)**
  - **Class 1 (1 W @ 1800 / 1900MHz)**
- **Control via AT commands** - Standard Commands: GSM 07.07 & 07.05 | Enhanced Commands: SIMCOM AT Commands.
- **Short Message Service** - so that you can **send and receive** small amounts of data over the network (ASCII or raw hexadecimal).
- **Embedded TCP/UDP stack** - allows you to upload data to a web server.
- **RTC supported.**
- **Selectable serial port.**
- **2 in 1 head set jack**
- **Low power consumption** - 1.5mA(sleep mode)
- **Industrial Temperature Range** - -40°C to +85°C

Anexo 7. Datasheet GSM/GPRS SIM900 Shield

Fuentes: Tiny Sine

NodeMCU ESP8266 ESP-12E WiFi Development Board

NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the DevKit. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson, and spiffs.

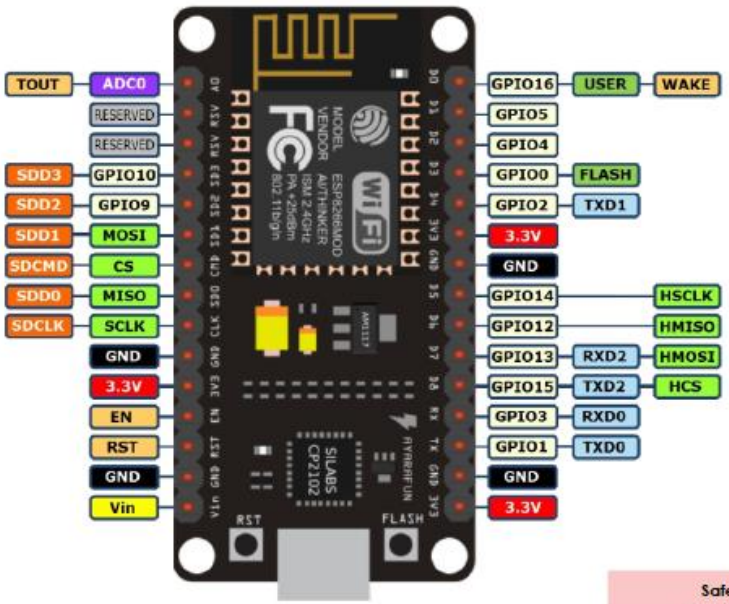


Features

- Version : DevKit v1.0
- Breadboard Friendly
- Light Weight and small size.
- 3.3V operated, can be USB powered.
- Uses wireless protocol 802.11b/g/n.
- Built-in wireless connectivity capabilities.
- Built-in PCB antenna on the ESP-12E chip.
- Capable of PWM, I2C, SPI, UART, 1-wire, 1 analog pin.
- Uses CP2102 USB Serial Communication interface module.
- Arduino IDE compatible (extension board manager required).
- Supports Lua (alike node.js) and Arduino C programming language.

PINOUT DIAGRAM

NodeMCU ESP8266 v1.0



Safety Precaution
All GPIO runs at 3.3V !!

Source
https://iotbytes.wordpress.com/nodemcu-pinout/

NodeMCU ESP8266

Front View

Front View

Specifications of ESP-12E WiFi Module

Wireless Standard	IEEE 802.11 b/g/n
Frequency Range	2.412 - 2.484 GHz
Power Transmission	802.11b : +16 ± 2 dBm (at 11 Mbps) 802.11g : +14 ± 2 dBm (at 54 Mbps) 802.11n : +13 ± 2 dBM (at HT20, MCS7)
Receiving Sensitivity	802.11b : -93 dBm (at 11 Mbps, CCK) 802.11g : -85 dBm (at 54 Mbps, OFDM) 802.11n : -82 dBm (at HT20, MCS7)
Wireless Form	On-board PCB Antenna
IO Capability	UART, I2C, PWM, GPIO, 1 ADC
Electrical Characteristic	3.3 V Operated 15 mA output current per GPIO pin 12 - 200 mA working current Less than 200 uA standby current
Operating Temperature	-40 to +125 °C
Serial Transmission	110 - 921600 bps, TCP Client 5
Wireless Network Type	STA / AP / STA + AP
Security Type	WEP / WPA-PSK / WPA2-PSK
Encryption Type	WEP64 / WEP128 / TKIP / AES
Firmware Upgrade	Local Serial Port, OTA Remote Upgrade
Network Protocol	IPv4, TCP / UDP / FTP / HTTP
User Configuration	AT + Order Set, Web Android / iOS, Smart Link APP

Anexo 8. Datasheet Modulo Node MCU esp8266

Fuentes: Einstronic

1. Introduction

The Finger Print Sensor is one optical fingerprint sensor which will make fingerprint detection and verification adding super simple. There's a high powered DSP chip AS601 that does the image rendering, calculation, feature-finding and searching. You can also enroll new fingers directly - up to 162 finger prints can be stored in the onboard FLASH memory. There's a red LED in the lens which will light up during taking photos so that you know its working condition. It is easy to use and by far the best fingerprint sensor you can get.

2. Specification

Supply voltage	3.6~6.0 V
Operating current(Max)	120 mA
Fingerprint imaging time Match Mode:	1.0 S Compare Mode 1:1
Search Mode	1:N
Storage capacity	162 templates
False Acceptance Rate	0.001% (Security level 3)
False Reject Rate	1.0% (Security level 3)
Baud rate	9600, 19200, 28800, 38400, 57600bps (default is 57600)
Interface	TTL Serial
Interface	TTL Serial

3. Interface

Pin number	Name	Type	Function Description
1	Vin	in	Positive Power Supply Input Terminal(Line color:Red)
2	TD	out	Serial data output, TTL logic levels(Line color: Yellow)
3	RD	in	Serial data input, TTL logic levels(Line color: White)
4	GND	-	Signal ground(Line color: Black)

Anexo 9. Datasheet FingerPrint Sensor

Fuentes: Seeed Studio Electronics

MFRC-522 RFID NFC Reader with card and tag

Technical Manual Rev 1r0

MFRC-522 RFID NFC Reader with Card and tag is based on RF module RC522 near field communication module. With operating frequency of 13.56Mhz where you can read and write a tag. Compatible in all gizDuino/Arduino Microcontroller boards.

General Specifications:

Input Supply Voltage: 3.3 VDC
Working Current: 13 to 26mA
Part Number: MF522-ED
Card reading distance: 0 to 60mm
Interface: SPI communication
Data Communication speed: 10Mbit/s Max.
Operarting Frequency: 13.56Mhz
Supported card types:
Mifare1 S50, Mifare1 S70, Mifare UltraLight, Mifare Pro, Mifare Desfire
Weight: 8g
Dimensions: 60mm x 40mm

Anexo 10. Datasheet MFRC-522 RFID Sensor

Fuentes: e-Gizmo Mechatronix Central

Introducción:

La serie de cerraduras electromagnéticas modelo E-941SA es la manera ideal de asegurar una puerta en contra de una entrada no autorizada. Cuando se aplica energía a la cerradura electromagnética, se crea un campo magnético extremadamente fuerte. El electroimán se atrae fuertemente a la armadura de la placa de acero la cual se monta en la puerta asegurada. Una vez que el electroimán se desactiva, la puerta asegurada funcionará de manera normal sin ningún magnetismo residual.

Características:

- Aluminio anodizado.
- Sin magnetismo residual.
- Protección con MOV contra sobretensiones.
- Soporte de montaje ajustable.
- Hardware completo de montaje para instalaciones típicas.
- Soporte tipo "L" y soportes tipo "Z" disponibles para un fácil montaje.
- 12/24 VCC seleccionable.*
- Placa de pared desmontable.*
- **E-941SA-300** es apto para uso en intemperie para utilizarse en exterior.
- **E-941SA-1K2PQ** y **E-941SA-600PQ** también tienen la característica de incluir un LED bicolor y un sensor de retención para indicación de estado para mostrar el estado de cierre:

Verde	La puerta está cerrada y bloqueada
Rojo	La puerta no está cerrada y/o bloqueada
Apadado	Puerta en uso / Sin alimentación

*E-941SA-300 no tiene esta característica.

Lista de partes:

1 x placa de montaje
1 x electroimán
1 x placa del armazón
1 x tornillo del armazón
2 x arandelas de acero
1 x arandela de hule
1 x espaciador de puerta
1 x perno de sujeción con tornillo
2 x pernos guía
4 x tornillos autorroscantes largos
2 x tornillos autorroscantes cortos
2 x tornillos de montaje con cabeza hexagonal
2 x tapas inviolables
1 x llave tipo Allen

Especificaciones:

		300 libras	600 libras	1,200 libras
Voltaje de operación		12VCC ±10%	12 ó 24 VCC ±10%	
Flujo de corriente	12VCC	315mA@	500mA	
	24VCC	N/A	250mA	
Resistencia de la bobina		48Ω ±10% por bobina (vea la página 8)		
Relevador del sensor de retención		3A@12VCC		
Dimensiones	Imán	6 3/4" x 1" x 1 1/4" (170 x 23 x 32 mm)	9 7/8" x 1 1/16" x 1 5/8" (250 x 26 x 40 mm)	10 1/2" x 1 5/8" x 2 5/8" (268 x 42 x 67 mm)
	Placa del armazón	6" x 3/8" x 1 1/4" (152 x 10 x 32 mm)	7 1/4" x 1/2" x 1 1/2" (185 x 12 x 38 mm)	7 1/4" x 5/8" x 2 3/8" (185 x 16 x 61 mm)
Temperatura de operación		14°~131° F (-10°~55° C)		
Peso		2 libras 13.5-oz (1.3kg)	4 libras 6-oz (2kg)	11 libras (5kg)
Listado en UL		No	Si	Si

Anexo 11. Datasheet Cerradura Electromagnética

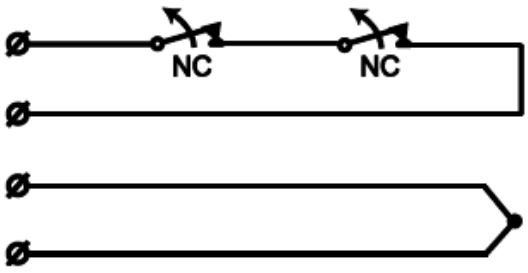
Fuentes: Ecolarm

ALARMTECH Detectors

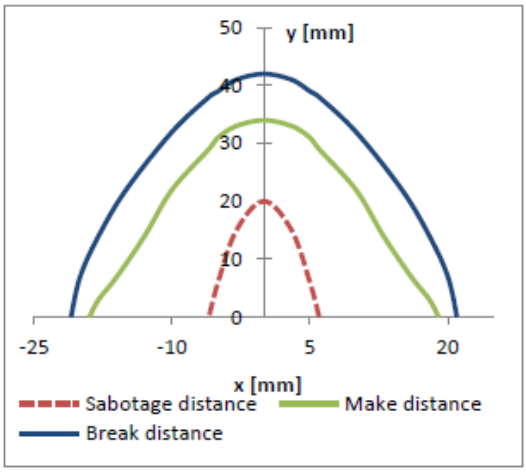
Magnetic Contact
Surface mounting

Instruction Manual **MC 270-S45**

Circuit diagram



DISTANCE DIAGRAM – WOOD



DESCRIPTION

MC 270-S45 is a heavy duty high security magnetic contact used in both alarm and security access control systems for protection of garage doors, industrial gates etc. against unauthorized opening and external magnetic field.

MC 270-S45 is certified according to EN 50131-2-6:2008.

MOUNTING INSTRUCTIONS

- Contact and magnet should be installed in parallel, corresponding to each other. Offset will reduce the working distances and may result in faulty operation or lower security.
- Magnetic contact should be installed in accordance with the installation drawings.

TECHNICAL DATA

Working environment	Wood	Steel
Sabotage distance	max. 20 mm	not recommended
Make distance	typ. 34mm +/– 40 %	not recommended
Break distance	typ. 42mm +/– 40 %	not recommended
Contact type	form A, SPST	
Switching voltage max.	48 V DC/AC	
Switching current max.	400 mA DC/peak AC	
Contact rating max.	10 W	
Estimated life expectancy	>20 million switching operations at 10 V/4 mA	
Cable	6 m, ϕ 3,2 mm, 4x0,14 mm^2	
Sleeving	1 m, ϕ 8,2 mm, stainless steel	
Environmental class (EN50130-5:2011)	IIIA	
Operating temperature range	-40°C to +55°C	
Operating humidity	max. 95% RH	
Housing material	Aluminum	
Dimensions:		
Contact part	73,5 x 15 x 25 mm	
Magnet part	73,5 x 15 x 25 mm	
Security grade (EN50131-2-6:2008)	3	
Approvals	ITR 14/2014	

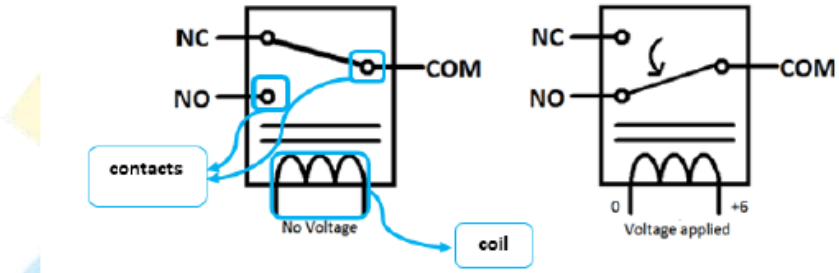
Anexo 12. Datasheet Contacto Magnético

Fuentes: Alarmtech

RELAY MODULES

RELAY WORKING IDEA

Relays consist of three pins normaly open pin , normaly closed pin, common pin and coil. When coil powerd on magntic field is generated the contacts connected to each other.



Relay modules 1-channel features

- Contact current 10A and 250V AC or 30V DC.
- Each channel has indication LED.
- Coil voltage 12V per channel.
- Kit operating voltage 5-12 V
- Input signal 3-5 V for each channel.
- Three pins for normally open and closed for each channel.

How to connect relay module with Arduino

As shown in relay working idea it depends on magnetic field generated from the coil so there is power isolation between the coil and the switching pins so coils can be easily powered from Arduino by connecting VCC and GND bins from Arduino kit to the relay module kit after that we choose Arduino output pins depending on the number of relays needed in project designed and set these pins to output and make it out high (5 V) to control the coil that allow controlling of switching process.

Anexo 13. Datasheet Relay 5V 1 canal

Fuentes: Future Electronic Corporation

forza POWER TECHNOLOGIES

forzaups.com

NT-511

La serie NT de Forza ofrece protección eléctrica a su computadora personal y periféricos.

De tamaño compacto, la unidad es ideal para espacios de trabajo limitados en la oficina y el hogar. Aunque pequeña en tamaño, proporciona la máxima protección ante la constante amenaza de fallas en el suministro eléctrico.

Características

- Sistema de alimentación ininterrumpible
- Protección eléctrica para equipos de uso doméstico y comercial
- Regulador de voltaje (AVR), corrige automáticamente las variaciones en el suministro
- 6 tomacorrientes con supresión de sobretensiones, respaldo de batería y regulación automática de voltaje AVR
- Protección para teléfono, fax y módem (RJ-11)
- Tres años de garantía

500VA

NIVEL DE PROTECCIÓN 5

- + Interruptor
- + Supresor de sobretensión
- + Protector de voltaje
- + Regulador de voltaje
- + Respaldo de batería

MPN	NT-511
Aspectos generales	
Capacidad	500VA/250W
Entrada	
Tensión nominal	120V
Margen de tensión	89-145VCA
Frecuencia	45-65Hz (detección automática)
Tipo de enchufe	NEMA 5-15P
Salida	
Tensión nominal	120V +/- 10%
Frecuencia	50/60Hz
Estabilidad de frecuencia	+/- 1Hz en modo de batería
Forma de onda	Onda sinusoidal modificada
Número total de salidas	6 (NEMA 5-15R)
Respaldo total con batería y protección contra sobrecargas	4
Protección exclusiva contra sobretensión	2
Protección para teléfono/módem/fax	RJ-11
Batería	
Tipo y número de baterías	12V 4.5Ah (1)
Tiempo de autonomía	18 min*
Tiempo de recarga	Hasta el 90 % de su capacidad en 6 horas
Regulación de tensión	
Regulación de tensión (120/220V)	120V
Característica de refuerzo (120/220V)	Vin x 1.18
Característica de compensación (120/220V)	Vin x 0.85
Regulación de frecuencia	
Selección de frecuencia automática	SI
Transferencia a línea/batería	
Tiempo de transferencia típico	2-4ms
Transferencia por baja tensión de línea a batería (120/220V)	89VCA
Transferencia por alta tensión de línea a batería (120/220V)	145VCA
Alarmas/Indicadores	
Indicadores visuales	Modo de CA: azul fijo Modo de batería: azul intermitente Modo de falla: luz azul apagada
Alarma audible	Modo de batería: se activa cada 10 segundos Bajo voltaje de la batería: se activa cada 1 segundo Sobrecarga: se activa cada 0,5 segundo Falla: sonido continuado
Protección	
Protección total	Regulación de tensión de línea: 110%+20%/-10%; después de 5 minutos interrumpe el paso de corriente y pasa al modo de falla. 120%+20%/-10%; interrumpe de inmediato el paso de corriente y pasa al modo de falla. Modo de batería:120%+20%/-0%; interrumpe de inmediato el paso de corriente
Joules	200J
Características especiales	
Opción de arranque en frío	SI
Recarga automática	SI
Ambiente	
Temperatura de funcionamiento	0-40°C
Temperatura de almacenamiento	-15-45°C
Humedad relativa	De 0 a 90% no condensada
Nivel de ruido	<40 dB a un metro de distancia de cualquier superficie
Características físicas	
Interruptor de encendido	Sí (iluminado)
Carcasa	Plástico retardador de llama
Color	Negro
Longitud del cable	1,2m
Dimensiones	279x101x142mm
Peso	4,8kg
Información adicional	
Garantía	3 años**

Anexo 14. Datasheet NT-511Forza

Fuentes: forzaups.com