UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo a la obtención del título de:
INGENIERA ELECTRÓNICA E INGENIERO ELECTRÓNICO

TEMA:
DISEÑO E IMPLEMENTACIÓN DEL PLAN PILOTO DE CONTROL DE ACCESO APLICANDO TECNOLOGÍA RFID PARA LABORATORIO 3 BLOQUE D UPS-SUR

AUTORES:
ESTEFANÍA ESMERALDA OLIVO BARCIA
CRISTIAN ADRIÁN PAZMIÑO CORTEZ

DIRECTOR:
JUAN CARLOS DOMÍNGUEZ AYALA

Quito, abril del 2017

CESIÓN DE DERECHOS DE AUTOR

Nosotros Estefanía Esmeralda Olivo Barcia y Cristian Adrián Pazmiño Cortez, con documento de identificación N° 1714195383 y N° 1716165533 respectivamente, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación intitulado: Diseño e implementación del plan piloto de control de acceso aplicando tecnología RFID para Laboratorio 3 Bloque D UPS-SUR, mismo que ha sido desarrollado para optar por el título de: Ingenieros Electrónicos, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

Estefanía Esmeralda Olivo Barcia

Cédula: 1714195383

Cristian Adrián Pazmiño Cortez

Cédula: 1716165533

Fecha: abril 2017

DECLARATORIA DE COAUTORÍA DEL DOCENTE TUTOR

Yo, declaro que bajo mi dirección y asesoría fue desarrollado el proyecto técnico, Diseño e implementación del plan piloto de control de acceso aplicando tecnología RFID para Laboratorio 3 Bloque D UPS-SUR, realizado por Estefanía Esmeralda Olivo Barcia y Cristian Adrián Pazmiño Cortez, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana para ser considerado como trabajo final de titulación.

Quito, abril 2017

Ing. Juan Carlos Domínguez Ayala

Cédula de identidad: 1713195590

DEDICATORIA

Dedicamos el presente proyecto a nuestros familiares por todo el ánimo y motivación que nos contagiaron en este proceso y en especial a nuestros padres por el apoyo y guía incondicional que nos brindaron en esta etapa.

De igual manera dedicamos este proyecto a nuestros maestros y nuestro tutor, quienes nos facilitaron su asesoría y préstamo de espacios físicos para el desarrollo del presente proyecto de titulación.

AGRADECIMIENTO

En primer lugar, expresamos nuestro sincero agradecimiento a la prestigiosa Universidad Politécnica Salesiana por la oportunidad de desarrollar nuestro proyecto técnico en el espacio físico del Laboratorio 3 del Bloque D del Campus Sur y el presupuesto económico brindado.

Por consiguiente, agradecemos a nuestro tutor de proyecto de titulación, el Ing. Juan Carlos Domínguez Ayala en especial por la confianza depositada en nosotros y su apoyo a la propuesta del proyecto técnico planteada desde el inicio; también por toda la asesoría y guía prestada en la corrección de detalles, resolución de dudas y sugerencias para el apropiado desarrollo del proceso de titulación.

Finalmente, a todos aquellos que participaron directa o indirectamente en la elaboración del presente proyecto técnico.

¡Gracias a ustedes!

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente proyecto técnico se desarrolló en la Universidad Politécnica Salesiana, específicamente en el Laboratorio 3 del Bloque D Campus sur, con la finalidad de realizar la instalación de un prototipo para el Control de Acceso por medio de tecnología RFID, mismo que permite controlar y registrar de forma automatizada los eventos de ingreso a dicho espacio físico. Para conseguir dicho propósito se optó por la transmisión de los datos de forma inalámbrica por medio de tarjetas inteligentes basadas en tecnología de Identificación por Radiofrecuencia (RFID) que permite almacenar y recuperar información de varios elementos de proximidad como etiquetas, tarjetas o llaveros, lo cual representa una ventaja en comparación a otras tecnologías como los de códigos de barras, ya que elimina la necesidad de trabajar con línea de vista y permite reprogramar las tarjetas incluso si están emitiendo datos; el sensor RFID se encarga de leer las etiquetas, se envían datos con comunicación RFID y SPI para posteriormente encapsularla en paquetes Ethernet TCP/IP. Adicionalmente, se incluyó un sistema de alerta en tiempo real mediante la programación de un API que puede configurarse para interactuar con redes sociales y mediante el envío de correos electrónicos al administrador; también ofrece la posibilidad de descargar el reporte de registros, con el fin de gestionar la seguridad en cuanto a los controles de acceso.

ABSTRACT

This technical project was developed at the Universidad Politécnica Salesiana, specifically in Lab 3 of the Campus South Block D, with the purpose of installing a prototype for Access Control using RFID technology, which allows control and registration in an automated way the events of entrance to said physical space. To achieve this purpose, the wireless transmission of data through smart cards based on Radio Frequency Identification (RFID) technology was used to store and retrieve information from several proximity elements such as labels, cards or keyboards, this represents an advantage compared to other technologies such as bar codes, since it eliminates the need to work with line of sight and allows to reprogram the cards even if they are emitting data, the RFID sensor is responsible for reading the labels, sending data with RFID and SPI communication for later encapsulation in Ethernet TCP / IP packets. In addition, a real-time alert system was included by programming an API that can be configured to interact with social networks and by sending emails to the administrator, also offers the possibility of downloading the logs report, to manage security regarding access controls.

INTRODUCCIÓN

El diseño e implementación del plan piloto de control de acceso propuesto con tecnología RFID, permite gestionar el registro de usuarios habilitados para acceder al espacio físico del laboratorio 3 y hacer uso de sus recursos en base a un horario, mediante la programación de una base de datos en PostgreSQL, el diseño de un servidor web y la configuración de dispositivos como la tarjeta Arduino y la placa Raspberry Pi 3.

El primer capítulo contiene información de la situación inicial del espacio físico, antecedentes del planteamiento del problema y alcance del proyecto, entre otros aspectos.

En el segundo capítulo se desarrolla el marco conceptual, colocando definiciones básicas acerca de los dispositivos utilizados en la elaboración del proyecto y los lenguajes de programación empleados para estructurar la base de datos, generar el proceso de control, la aplicación del servidor y la información de alerta que se envía a redes sociales.

El tercer capítulo contiene el diseño planteado para el prototipo, aquí se presentan los requerimientos tecnológicos, los diagramas de bloque y de flujo que exponen el funcionamiento del sistema, la programación y lógica de la base de datos, la página web, los dispositivos y el proceso a ejecutar para establecer la conexión y actualización de datos; a esto se añade un sistema de prevención frente a posibles fallas.

El cuarto capítulo presenta la implementación y puesta en marcha del prototipo, en este caso se describen los materiales utilizados, las pruebas y resultados de las etapas del proceso; también se añade información relevante acerca del sistema como presupuesto, conclusiones y recomendaciones acerca del modo de operación de control de acceso. En los anexos se presenta la configuración, programación y conexión de los equipos.

CAPÍTULO 1

ANTECEDENTES

1.1 Planteamiento del problema

Los espacios de uso estudiantil como laboratorios, contienen un gran número de equipos y flujo de usuarios lo que genera en muchas ocasiones accesos no autorizados, robos e incluso daño de la infraestructura y dispositivos que se manejan; éste es el caso del laboratorio 3 del bloque D de la Universidad Politécnica Salesiana - Campus Sur, que al ser un área de uso educativo requiere un mayor nivel de gestión sobre el acceso de personal y el grado de autorización asignado con la finalidad de proteger los activos de la organización.

1.2 Justificación

La implementación de sistemas de control de accesos automatizado para los laboratorios de la Universidad, constituyen hoy por hoy un requerimiento de vital importancia debido al notable aumento en la demanda de usuarios que ingresan a estos espacios, lo que ha ocasionado el incremento de riesgos y amenazas a la seguridad tales como robos, daños y/o mal uso de los equipos.

Actualmente el registro de usuarios que acceden al laboratorio se realiza de forma manual en hojas de registro que se almacenan en una bitácora que es empleada por los docentes encargados de laboratorio con el fin de tener información acerca del nombre de cada usuario con la fecha y hora de acceso al área del laboratorio, este proceso manual presenta fallas ya que en varias ocasiones no se registra la información de los datos de los usuarios que ingresan o cuando se realiza, muchas veces los datos son erróneos o de fácil adulteración.

Por lo tanto, la ejecución del presente proyecto técnico representa un plan piloto, que permitirá establecer medidas de control y gestión para fortalecer la seguridad del espacio

físico del laboratorio, usando tecnología de radio frecuencia que actúa con tarjetas inteligentes que poseen memoria de datos que pueden ser almacenados en cuatro bloques de datos, tres para identificación del usuario, cada uno protegido por contraseña para evitar la duplicación de la misma.

La adquisición de datos de las tarjetas se codifica en los dispositivos de control para posteriormente compararlas en la base de datos y ejecutar la acción programada en el proceso, siendo gestionable ya que la información puede ser actualizada por el administrador que posee una contraseña para autenticarse, brindando un mayor nivel de seguridad siendo una alternativa al uso tradicional de bitácoras de registro físicas.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar e implementar el plan piloto de un control de acceso utilizando tecnología de identificación por radio frecuencia para establecer la gestión del ingreso de usuarios al laboratorio 3 del bloque D de la Universidad Politécnica Salesiana - Campus Sur.

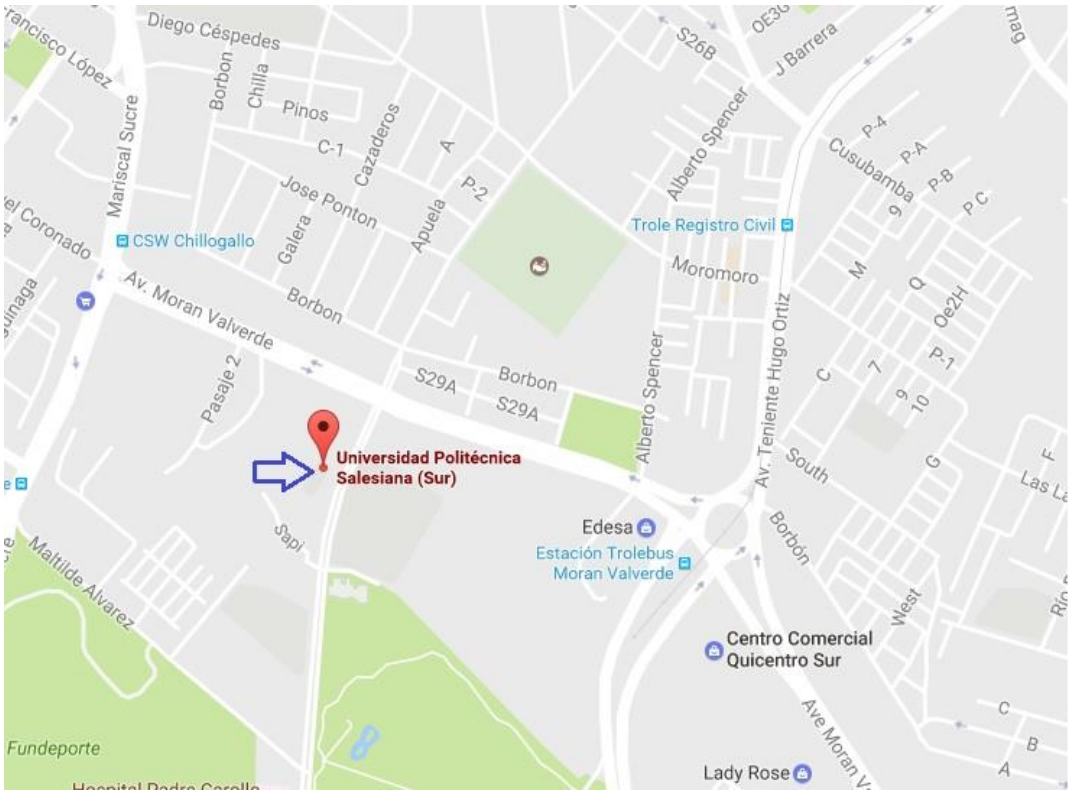
1.3.2 Objetivos Específicos

- Diseñar módulos para el registro de datos utilizando dispositivos programables como ARDUINO-MEGA y RASPBERRY PI que permitan controlar el acceso utilizando tarjetas MIFARE y activen actuadores magnéticos.
- Transmitir los registros adquiridos por los módulos de control de acceso mediante la encapsulación de los datos en paquetes ethernet y visualizarlos en el servidor que se encuentra en el departamento de Soporte Técnico para su gestión.
- Implementar el sistema de control de acceso en el laboratorio requerido, instalando los dispositivos de lectura, captura y almacenamiento de datos.
- Realizar la puesta en marcha del sistema de control de acceso, realizando pruebas del correcto funcionamiento del sistema al ejecutar las órdenes establecidas.

1.4 Delimitación Espacial

El presente proyecto se implementará en las instalaciones del Laboratorio 3 del Bloque D de la Universidad Politécnica Salesiana Campus Sur Sede Quito (Ver Figura 1), es un espacio que contiene dispositivos que utilizan estudiantes y docentes para realizar prácticas de Redes de Computadoras y otras asignaturas electivas que corresponden a la malla curricular de la Carrera de Ingeniería Electrónica.

Figura 1.1 Universidad Politécnica Salesiana Campus Sur



Ubicación de las instalaciones donde se implementó el Sistema de control de acceso, Fuente: (Google Maps, 2016)

1.5 Beneficiarios del proyecto

Los beneficiarios de este proyecto técnico son los usuarios del Laboratorio 3, entre los que se encuentran docentes, estudiantes y responsables de los laboratorios de la Carrera de Ingeniería Electrónica de la Universidad Politécnica Salesiana.

1.6 Estado actual de las instalaciones

Como estado inicial de las instalaciones, el Laboratorio 3 cuenta con:

- Una cámara IP
- Una alarma
- Una chapa empotrada de pestillo para llaves
- Una manija

1.7 Método de registro de usuarios

El registro de acceso se realiza por bitácoras con formularios físicos de los datos del usuario que utiliza las del Laboratorio, dicha información es manejada por los docentes encargados de las instalaciones.

CAPÍTULO 2

MARCO CONCEPTUAL

En este capítulo se presentan los fundamentos teóricos y técnicos necesarios para realizar la ingeniería del presente proyecto técnico, se describe la forma de operación de los recursos utilizados como instrumentos y lenguajes de programación en los que se basa el proceso de control y gestión de usuarios, de igual manera se presenta la descripción de términos nuevos cuya definición se requiere para tener una idea clara de su función.

2.1 Tecnología de Radiofrecuencia

La tecnología de identificación por radiofrecuencia (RFID) en la actualidad se utiliza ampliamente en diseño de proyectos, ya que presenta la ventaja de permitir realizar el proceso de identificación a distancia comparado con la tecnología de código de barras que requiere línea de vista para realizar la identificación de códigos, otra ventaja es el hecho de que RFID soporta un mayor número de identificaciones (IDs) y no es necesario incorporar datos adicionales de forma manual como comúnmente se realiza al utilizar código de barras. (Mandeep Kaur, 2011)

De forma general RFID describe un sistema que transmite información mediante un único dato serial que identifica un objeto o persona de forma inalámbrica usando ondas de radio, capturando datos automáticamente sin contacto, existen dos tipos de estructuras que se detallan a continuación:

- RFID de campo cercano: Las etiquetas que utilizan este tipo de estructura envían los datos de retorno al lector usando modulación de carga, es decir la bobina del lector detecta un pequeño incremento en la corriente que fluye a través de la misma lo que permite que se recupere la señal monitoreando el cambio de corriente.
- RFID de campo lejano: Las etiquetas con este tipo de estructura capturan ondas electromagnéticas que se propagan desde una antena dipolo acoplada al lector usando la técnica de back scattering o retro dispersión de ondas que absorbe la

mayor parte de energía que alcanza a cierta frecuencia, que por lo general al ser campo lejano opera en la banda UHF con un valor típico de 2,45 GHz, mientras que la de campo cercano opera por debajo de este valor; el alcance de operación del lector en modo de campo lejano es de 3 m.

Es importante conocer que ambas estructuras pueden transmitir potencia de forma remota a una etiqueta para mantener su operación con valores entre 10 µW y 1 mW, el rango de frecuencias con las que trabaja RFID en función de la distancia de lectura, se presentan a continuación en la tabla 2.1:

Tabla 2.1 Relación entre el rango de frecuencia y distancia de operación de RFID

Frecuencia	Distancia
9 KHz	1 cm
135 KHz	2 cm
13,56 MHz	1,5 m
30 MHz	4 m
120 MHz	4 m
245 MHz	6 m
850 MHz	6 m
2,45 GHz	90 m
5,8 GHz	100 m

Frecuencias y distancias de operación de tecnología RFID, Fuente: (Mandeep Kaur, 2011)

2.1.1 Funcionamiento de Identificación por Radiofrecuencia

En el proceso de identificación de etiquetas en un sistema RFID se requiere un dispositivo lector que detecte la información de las etiquetas que posteriormente se transmiten a través de ondas de radio, estos datos pueden ser procesados de forma digital a través de un dispositivo utilizando estándares para interfaces, como se aprecia en la figura 2.1.

La emisión de ondas del lector alcanza 30 m y el chip incorporado en la mayoría de tarjetas con tecnología RFID puede almacenar un máximo de 2 Kb de datos. (Mandeep Kaur, 2011)

Figura 2.1 Modo de operación entre etiquetas y lectoras con tecnología RFID

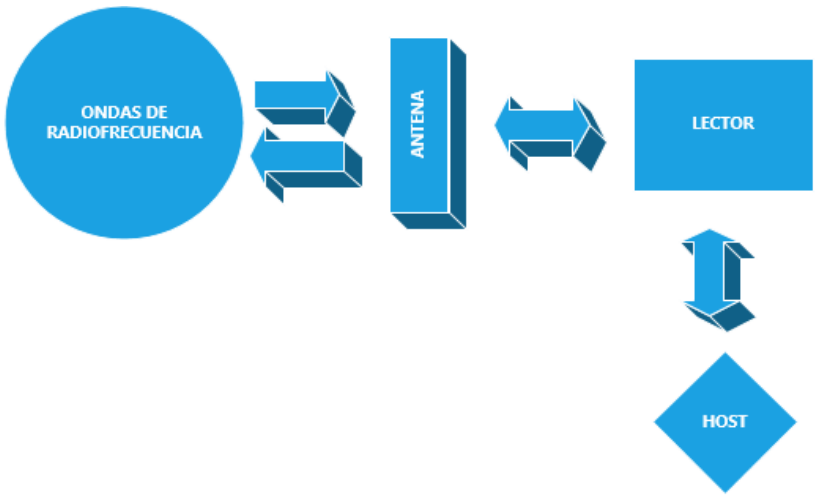


Diagrama de operación de radiofrecuencia y etiquetas, Elaborado por: Estefanía Olivo, Cristian Pazmiño.

2.1.2 Beneficios y limitaciones de los sistemas de Radiofrecuencia

Los beneficios más relevantes que brinda este tipo de tecnología es la detección de etiquetas sin requerir línea de vista, el almacenamiento, lectura y escritura de etiquetas de forma simultánea y la eliminación de intervención humana.

Las principales limitaciones de este tipo de tecnología son el costo que depende del tipo de etiquetas que se desea utilizar, la elección del rango de frecuencia de operación para que se pueda realizar el proceso de retro dispersión sin afectar el rango de lectura de datos y la estandarización de esta tecnología que tiene conflicto con la frecuencia asignada al Sistema de Posicionamiento Global (GPS) que también utiliza la frecuencia de 2,4 [GHz], aunque en la actualidad ya se han realizado conferencias, análisis y estudios para poder operar en dicha frecuencia sin afectar el modo de operación y transmisión de datos que realiza GPS.

2.2 Sistemas de Gestión de Bases de Datos (SGBD)

Como su nombre lo dice los sistemas de gestión de bases de datos se encargan de gestionar la información y distribuirla como datos individuales en arreglos de filas y columnas para poder realizar un proceso denominado indexación que consiste en el reconocimiento de usuarios, órdenes y adquisición de datos de forma simultánea.

2.2.1 Bases de datos (DB)

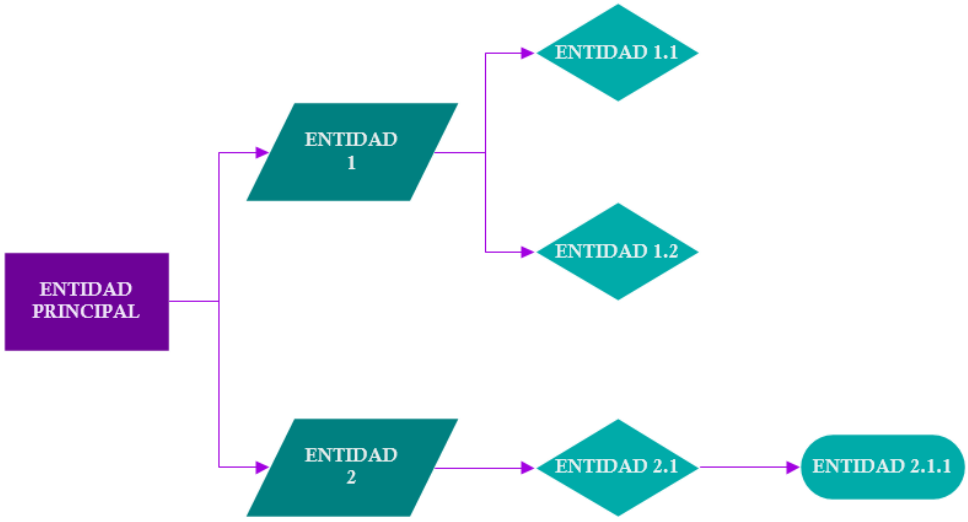
Una base de datos es considerada una reserva de información o acopio de datos que se relacionan entre sí y una lógica procedimental que representa un modelo de la realidad.

2.2.1.1 Modelos de Bases de datos

Un modelo de bases de datos establece la arquitectura en la que se manejan las tablas, variables y llaves relacionadas, a continuación, se describen los tipos de modelos que rigen la creación de una base de datos:

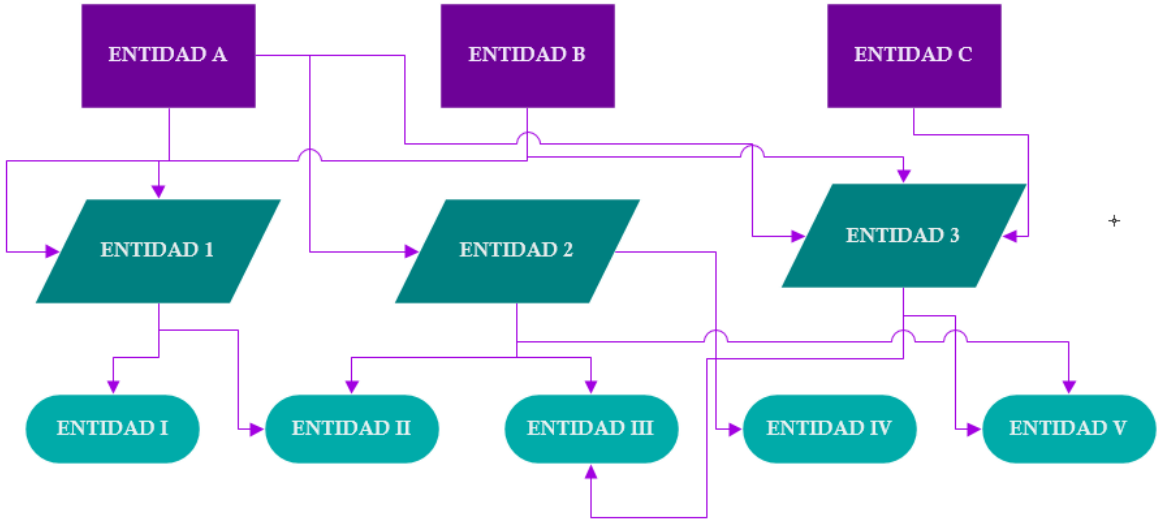
- Modelo jerárquico: Este modelo se basa en entidades o tablas que permiten acceso a los datos de forma derivada que puede representarse en forma de árbol ya que una entidad depende de otra y así sucesivamente por niveles (Ver Figura 2.2).
- Modelo en red: Este modelo también se conoce como las siglas de sus desarrolladores (CODASYL) Conferencia de Sistemas de Lenguajes de Datos, en este caso se considera como archivo a una tabla y establece una clave única como campo para poder acceder a la información contenida en otra tabla y así indexar un archivo y conocer los datos almacenados en el mismo, es importante gestionar correctamente los campos de clave única ya que se puede producir redundancia entre la codificación de claves como se observa en la figura 2.3. (González, 2011)
- Modelo relacional: Este modelo es una alternativa al gran número de relaciones entre entidades, se basa en tablas cuyas columnas corresponden a los campos y las filas a los registros, para generar una base de datos que aparte de realizar consultas pueda relacionar datos sin necesidad de una clave compleja (Ver figura 2.4).

Figura 3.2 Modelo de base de datos Jerárquico



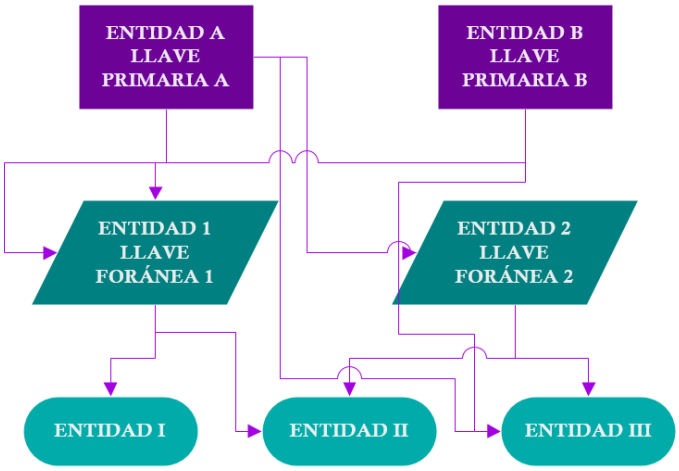
Esquema de modelo Jerárquico de base de datos, Elaborado por: Estefanía Olivo, Cristian Pazmiño.

Figura 4.3 Modelo de base de datos en red



Esquema de modelo en red de base de datos, Elaborado por: Estefanía Olivo, Cristian Pazmiño.

Figura 5.4 Modelo Relacional de base de datos



Esquema del modelo relacional de base de datos, Elaborado por: Estefanía Olivo, Cristian Pazmiño.

Es importante mencionar dentro de esta distribución a las bases de datos transaccionales que corren en base a bloques que establecen condiciones de ejecución utilizando el modelo relacional; presentan alto rendimiento por su forma de almacenamiento redundante que genera copias de datos constantemente y permite realizar las siguientes acciones:

- Suspender o eliminar entidades.
- Distribuir datos lógicos entre tablas sin modificar la información almacenada.
- Adquirir datos y controlar dispositivos conectados. (González, 2011)

2.2.3 Diagrama Entidad / Relación

Un diagrama de entidad / relación es una representación visual que permite plasmar la lógica que maneja la base de datos, definir campos o variables, determinar el tipo de datos con su longitud o extensión y escoger llaves, que se explicaran a continuación:

- Llave primaria (PRIMARY KEY): Una llave primaria se aplica a las columnas de una tabla donde se encuentran los campos y se utiliza para establecer una restricción de valor no nulo, esto quiere decir que todas las filas de la tabla deben tener un valor y se visualizan los elementos a través de la clave primaria que se seleccionó. (Celma Guiménez, Casamayor Ródenas, & Mota Herranz, 2003)

- Llave foránea o referencial (FOREIGN KEY): Una llave foránea es una clave ajena, que se encarga de hacer referencia desde una tabla hacia el registro de otra tabla para acceder a los datos y asociarlos.

2.2.4 Elementos

- Objetos: Se define como objeto a cualquier persona o evento de la cual se requiere conocer información, también se conoce como entidad.
- Operación: Una operación es una acción que se ejecuta en determinado objeto para añadir, modificar, consultar o eliminar datos.
- Transacción: Es un conjunto de operaciones que se realizan de forma secuencial.
- Tabla: Consta de filas en donde se ingresan registros o datos y columnas que definen campos de un formulario.

2.2.5 Seguridad de un Sistema de Gestión de Bases de Datos

Un sistema de gestión de bases de datos debe garantizar el acceso a la información de usuarios autorizados, para lo cual se toma en cuenta los siguientes elementos:

- Usuario: Para garantizar seguridad se debe conocer la identidad de la persona que requiere acceder, mediante una contraseña.
- Acceso: En este parámetro se asocia una lista de autorizaciones que determinan el nivel de acceso que se le otorga al usuario.
- Autorización: Corresponde al nivel o grado de privilegio del usuario.

2.3 PostgreSQL

PostgreSQL es una herramienta de gestión de bases de datos de tipo objeto-relacional (ORDBMS) que utiliza el lenguaje de programación SQL, posee una plataforma orientada a objetos que permite crear tablas, seleccionar tipo de datos y colocar restricciones basadas en funciones de consulta estructurada denominadas reglas query. (Lockhart, 1996)

2.3.1 Características

- Posee un entorno gráfico que facilita el diseño de bases de datos.
- Es un software de código abierto.
- Configuración de triggers o disparadores que se activan cuando se cumple una condición.
- Soporta diferentes modelos de base, añade la función de insertar estructuras en forma de array.
- Incorpora gestión para diferentes usuarios y asignación de permisos.

2.3.2 Beneficios y limitaciones de PostgreSQL

Entre los principales beneficios que brinda PostgreSQL se encuentran su capacidad de ejecutar un cuantioso número de transacciones que generan tráfico y soportarlo, posibilita almacenar procedimientos y ejecutarlos sobre la misma base como es el caso de Oracle; por otra parte al ser comparado con otros sistemas de gestión de bases de datos como MySQL presenta varias ventajas como la opción de rollback que actúa como un backup en caso de emergencia también se acopla al sistema sobre el cual se ejecuta y por último el mérito de PostgreSQL radica en la capacidad de ofrecer escalabilidad ya que opera con niveles de peticiones simultáneas. PostgreSQL posee un almacenamiento de 8 K que puede ser incrementado hasta 32 K por fila, aunque esto genera una limitación en el rendimiento. (Lockhart, 1996)

2.4 Creación de portales web y conexión al servidor

Para desarrollar el diseño de una aplicación web se requieren ciertas herramientas, como las que se explican a continuación:

- HTML: Esta herramienta se conoce como lenguaje marcado de hipertexto establece el diseño de la página web por medio de etiquetas que representan un elemento de la página ya sea una imagen, tabla o texto. Su estructura se basa en la cabecera donde se coloca información acerca de la página y el cuerpo que posee el contenido de la página web. (López, Fernández, & García, 2010)

- JavaScript: Es un tipo de lenguaje mejorado que ofrece calidad y rendimiento para el usuario o cliente, ya que permite la óptima visualización de la página web.
- PHP: Es una herramienta que se usa como pre procesador de hipertexto, es decir permite dar estructura a la página web de forma dinámica con el servidor y potenciar las funciones que se establecen en las bases de datos.
- SQL: Es un lenguaje de programación de consultas estándar para el manejo de datos y acciones almacenadas en bases de datos.

El diseño de un portal web requiere de etiquetas y atributos que son un conjunto de funciones de HTML que permiten la edición del texto, fondo, imágenes, entre otros elementos que contiene la página web, los principales atributos se detallan en el anexo 1.

2.4.1 Enlaces Web

Para realizar un enlace desde nuestro portal web es necesario definir una URL ò localizador uniforme de recursos especificando un protocolo, un dominio, un puerto opcional y la ruta al recurso, su función es permitir hallar un elemento web; los enlaces que se requieren para la visualización del portal web dependen de la programación realizada en HTML y se dividen en los siguientes destinos:

- Enlace a sitio web externo: El destino es una página web diferente a la del sistema gestionado internamente.
- Enlace a subpáginas: El destino son páginas web que se encuentran dentro del mismo sistema de gestión.
- Enlace interno: El destino son marcas realizadas dentro del mismo documento.
- Enlace a ficheros: El destino es la descarga de un archivo.
- Enlace a correo electrónico: El destino es una dirección e-mail. (López, Fernández, & García, 2010)

2.4.2 Servidores de aplicaciones

La función principal de un servidor de aplicaciones es establecer la brecha o conexión que puede ser unidireccional o bidireccional entre el cliente y el recurso web solicitado, es decir se ejecutan sin necesidad de ser programados y permiten gestionar servicios, entre las herramientas más conocidos están GNU/Linux, Apache, MySQL y PHP y entre todas conforman XAMP, un servidor de código abierto que gestiona dichas herramientas. (López, Fernández, & García, 2010)

- Apache: Es un componente imprescindible en el diseño de un servidor web, ya que permite establecer conexión con el puerto HTTP que se basa en peticiones y respuestas para establecer un intercambio de información entre el servidor y el cliente web.
- MySQL: Es un software de gestión de bases de datos de tipo relacional que trabaja de manera conjunta al lenguaje PHP, para llevar a cabo la comunicación entre el usuario y la base de datos que se ejecuta en el servidor MySQL es necesario instalar la interfaz phpMyAdmin o pgadmin3 en el caso de usar PostgreSQL con programación basada en SQL.
- PHP: Es un lenguaje de programación, cuya definición se encuentra en el apartado 2.4 del presente documento, su funcionalidad principal es la generación de páginas web desde el servidor realizando un vínculo con las paginas mediante el código realizado en HTML; entre otros aspectos relevantes se encuentran el ser orientado a objetos, el modo de ejecución se realiza en un servidor web que por lo general es MySQL y cuenta con una variedad de librerías que permiten la conexión con la mayoría de bases de datos.

2.5 Ordenador de placa única Raspberry Pi

El ordenador de placa única Raspberry Pi es un dispositivo de tipo SBC, es decir un Controlador de Sesión de Borde que permite realizar funciones entre la red Local y la red inalámbrica, al referirse a la placa Raspberry Pi como ordenador queda claro que se puede realizar una conexión similar a un ordenador con periféricos y comunicación a internet, su desarrollo inició en la Fundación Raspberry Pi como proyecto educativo en Reino

Unido en la Universidad de Cambridge y se lanzaron al mercado sus primeros modelos en el año 2012 con el objetivo de fomentar la formación técnica en colegios.

La estructura de la placa posee varios componentes que varían en función del modelo, entre los principales se encuentran un procesador, un chip, un PGU, puertos USB, puerto Ethernet y memoria RAM, en la tabla 2.2 se presenta una descripción y comparación entre las diferentes versiones de Raspberry Pi.

Tabla 2.2 Características de las diferentes versiones de Raspberry Pi

Características	Raspberry Pi	Raspberry Pi Zero	Raspberry Pi 2	Raspberry Pi 3
Lanzamiento	2014	11/2015	02/2015	2016
SoC	BCM 2835	BCM 2835	BCM 2836	BCM 2837
CPU	ARM 700MHz	ARM 1GHz	Quad Core 900MHz	Quad Core 1.2GHz
Procesador	ARMv6	ARMv6	ARMv7-A	ARMv8-A
GPU	250MHz	250MHz	250MHz	400MHz
Memoria RAM	512 MB	512 MB	1 GB	1 GB
GPIO	40	40	40	40
Ethernet	-	-	10/100	10/100
Wireless	-	-	-	802.11n/BT4.0

Descripción de componentes de los diferentes modelos de placa Raspberry, Fuente: (Benchoff, 2016), Elaborado por: Estefanía Olivo, Cristian Pazmiño.

2.5.1 Funcionalidades de Raspberry Pi

Para emplear el dispositivo Raspberry Pi en cualquiera de sus versiones se requiere una tarjeta de memoria SD, una corriente de alimentación de al menos 1 A para versiones antiguas y al menos 2,5 A para versiones recientes, un puerto HDMI para interfaz visual, puertos USB dependiendo de la versión para conectar dispositivos periféricos y un puerto para conexión Ethernet que está disponible en versiones actuales. (Raspberry Pi, 2012)

En la parte de Hardware los puertos USB tienen una limitación de corriente por lo cual se recomienda conectar dispositivos de almacenamiento externo mediante un hub y para el almacenamiento interno se utiliza una memoria SD de hasta 512 GB, este es un aspecto importante a considerar ya que el Sistema Operativo que se ejecuta en la placa Raspberry se almacena aquí, por lo que es recomendable usar una SD de menor capacidad para no exceder el rango de memoria y gozar de un óptimo rendimiento.

En cuanto al Sistema Operativo de la placa Raspberry, este no viene incorporado en el dispositivo, por lo cual es necesario descargarlo e instalarlo para poner en marcha su funcionamiento, los principales sistemas operativos que soporta están basado en Linux:

- Raspbian: Actualmente es un sistema operativo ampliamente utilizado para la placa Raspberry Pi, al ser una distribución de Linux es gratuito y presenta un alto grado de optimización de los requerimientos de la placa en cuestión de paquetes y el aporte de la comunidad de usuarios para el desarrollo de proyectos.
- Ubuntu Mate: En este caso Ubuntu Mate es un sistema operativo desarrollado principalmente para ejecutarse en la placa Raspberry versión 2, como característica principal permite la ejecución del internet de las cosas (IoT).
- Pidora: Es un sistema operativo basado en aplicaciones de Fedora, como ventaja presenta alto nivel de procesamiento, estabilidad y disponibilidad de paquetes.
- Microsoft Windows 10: Raspberry Pi posee la posibilidad de ejecutarse en Windows 10 diseñado para ofrecer el manejo de aplicaciones que permiten la conexión de varios dispositivos y es orientada a IoT. (Raspberry Pi, 2012)
- OpenELEC: Este sistema operativo se emplea en la placa Raspberry Pi con la finalidad de ofrecer al usuario prestaciones de Centro de Entretenimiento, ya que ejecuta aplicaciones multimedia de manera óptima disminuyendo el requerimiento de recursos tecnológicos y funciona correctamente al conectarlo con un TV.

La elección del sistema depende de las necesidades del usuario y herramientas como BerryBoot y Noobs realizan la instalación desde cero sin conexión a internet.

2.5.2 Lenguaje de programación Python

Python es un lenguaje de programación desarrollado en 1990 por Guido van Rossum, es conocido como uno de los mejores lenguajes de programación sucesor del lenguaje ABC debido a las funcionalidades que aporta y su simple manejo al ser considerado de alto nivel, en el año 2001 se formó la Fundación Software de Python (SPF) por un conjunto de desarrolladores de software libre que aportan a la comunidad con avances tecnológicos y nuevos algoritmos para el desarrollo de Python; es importante recalcar el rápido avance de su desempeño al ser utilizado en aplicaciones informáticas recurrentes en la actualidad entre las cuales se encuentran YouTube y el cliente de Dropbox. (Foundation, 2001-2017)

Las funcionalidades más relevantes que el lenguaje de programación Python aporta son la posibilidad de realizar programación orientada a objetos, el soporte multiplataforma que posee al contar con licencia de compatibilidad GPL para todo tipo de Unidad de Procesamiento Gráfico (GPU), también se otorga un mayor nivel de ejecución de algoritmos al utilizar un lenguaje de programación más corto a comparación del gestor Java o el lenguaje C++ y la opción de seleccionar el tipo de codificación de caracteres que se desea emplear utilizando el estándar industrial Unicode para uso informático.

Gracias a la gran cantidad de librerías de Python, es el lenguaje de programación más utilizado en proyectos de ingeniería de control que incluyen dispositivos como Arduino y Raspberry Pi. (Ángeles, José Luis, José Carlos, & Manuel, 2015)

2.5.3 Diseño de APIs

Una Interfaz de Programación de Aplicaciones, mejor conocida como API se considera una tendencia en la actualidad ya que permite establecer comunicación entre diferentes aplicaciones, conectarse a bases de datos e incluso comunicarse directamente con redes sociales, la principal ventaja de la implementación de APIs es la facilidad que ofrecen a los desarrolladores de software en cuanto a evitar la necesidad de volver a generar un código que ya se encuentra elaborado.

Por lo tanto al generar una API se establecen un conjunto de parámetros que pueden ser codificados en varios sistemas operativos como una capa de abstracción de código, que permite ejecutar un software sobre otro; para realizar todo el proceso se encuentra una biblioteca basada en el estándar HTTP que se conoce como API REST, en la cual se pueden encontrar funcionalidades previamente establecidas por otras aplicaciones como Twitter y el uso está restringido para la edición por parte de terceros, con el fin de evitar plagio. (Andrearrs, 2014)

2.6 Protocolos de comunicación y envió de datos entre dispositivos

Los protocolos de comunicación permiten la conexión entre diversos dispositivos y se encargan de definir un conjunto de normas para el intercambio de información, transmisión y adquisición de datos, a continuación, se van a describir los protocolos más utilizados.

2.6.1 Protocolos de transmisión serial

La comunicación serial es el estándar más utilizado en el campo de la transmisión de datos, se encarga del intercambio de bytes y el alcance en cuestión de distancia, depende de su versión al igual que la velocidad de transmisión, entre las principales características que se analizan en la comunicación serial se encuentran:

- Líneas de transmisión: Por lo general utiliza tres líneas de transmisión la línea de transmisión, la línea de recepción y la línea de referencia.
- Tipo de comunicación: El protocolo serial maneja comunicación asíncrona.
- Bits de datos: Determina la cantidad de bits que se envían en la transmisión, por lo general dependen del tipo de caracteres y su tamaño vario de 6, 7 u 8 bits.
- Bits de parada: Son los encargados de marcar el fin de una transmisión, permiten un aumento de sincronismo y una transmisión más lenta.
- Bits de paridad: Se utilizan para reducir los errores en la transmisión de datos.
- Velocidad de transmisión: La velocidad de transmisión detalla el número de bits que se envían por segundo y dicho valor es inversamente proporcional al valor de

la distancia, pues al encontrarse más cerca los dispositivos la distancia se acorta y se envían a mayor velocidad los datos.

- Distancia de transmisión: Es uno de los aspectos fundamentales en la transmisión de datos, al ser más extensa provoca pérdida de velocidad de transmisión.
- Codificación: La comunicación serial utiliza codificación ASCII normal y extendida.
- Dispositivos: Existe un término en general para equipos terminales (DTE) y otro para equipos de control (DCE).

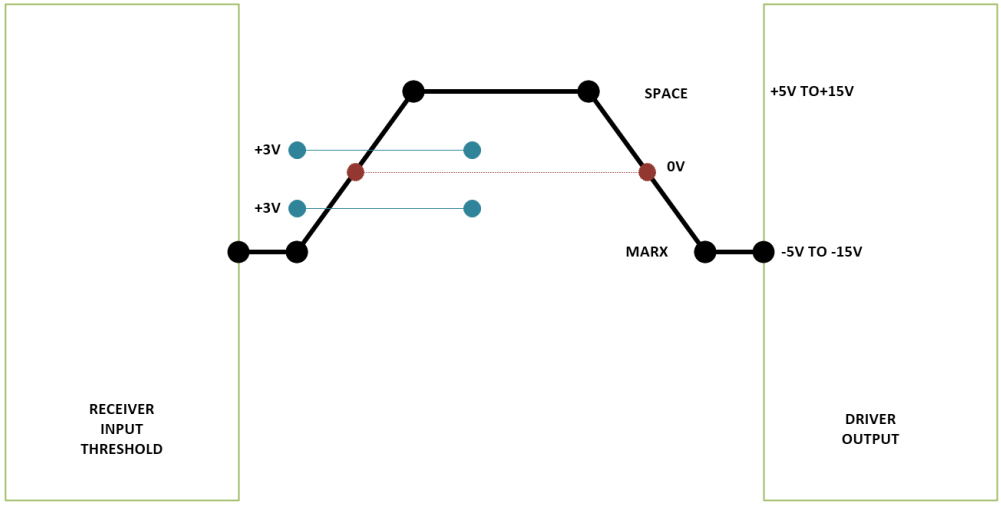
2.6.1.1 Protocolo RS-232

En primer lugar, es necesario conocer que las siglas RS significan Estándar Recomendado para todas las versiones de comunicación serial, el protocolo RS-232 fue desarrollado por el departamento de ingenieros de la Alianza de Industrias Electrónicas (EIA) en 1969, para este protocolo las señales poseen dirección de entrada o salida que dependen del DTE.

Por otra parte, el protocolo RS-232 fue mejorado con los estándares RS-422 y RS-485, ya que entre sus limitaciones se encuentra el utilizar una única línea para la transmisión de valores lógicos y su capacidad de alcance no excede los 12 m; para lo cual en los estándares mejorados se implementó un tipo de comunicación balanceada para el envío de valores lógicos 0 y 1.

En la figura 6 se puede apreciar que el nivel de voltaje más alto para el controlador es de +5 V a +15 V y el más bajo -5 V a -15 V, con un nivel de ruido equivalente a 2 V y para el receptor el nivel más alto va de +3 V a +15 V y el más bajo -3 V a -15 V; esto permite identificar a los niveles más bajos como 1L y a los más bajos como 0L; esto indica que para disminuir el efecto de diafonía entre señales adyacentes el valor que se establece para velocidad de transmisión de datos equivale a 20 kbps, para reducir la probabilidad de cruce de señales. (B.R. & Jaganmohan , 2015)

Figura 6.5 Especificación de niveles de voltaje



Relación de niveles de voltaje entre el controlador y el quipo terminal receptor, Elaborado por: Estefanía Olivo, Cristian Pazmiño.

2.6.1.2 Protocolo RS-422

El protocolo RS-422 es un tipo de interfaz que posee un único controlador y soporta más de 10 dispositivos, su método de transmisión de datos es full dúplex y a diferencia del protocolo RS-232 utiliza un par de cables al igual que el protocolo RS-485 para balancear la señal de voltaje en la transmisión y recepción de valores lógicos.

2.6.1.3 Protocolo RS-485

El protocolo RS-485 es un tipo de interfaz que mejora las limitaciones del protocolo RS-232 y RS-422 ya que soporta 32 dispositivos transmisores y 32 dispositivos receptores, su alcance es aproximadamente 1200 m a 200 kbps en condiciones ideales y a menor distancia de hasta 50 m aumenta a 10 Mbps; con respecto al nivel de voltaje de alimentación se maneja entre -7 V y 12 V.

2.6.1.4 Características principales de los Protocolos de Comunicación Serial

En la siguiente tabla 2.3 se presenta una comparación entre los diferentes protocolos de comunicación serial:

Tabla 2.3 Características de versiones de Protocolos de Comunicación Serial

Características	RS-232	RS-422	RS-485
Número de Dispositivos	1 TX, 1 RX	5 TX, 10 RX	32 TX, 32 RX
Modo de comunicación	Full Duplex	Full Duplex & Half Duplex	Half Duplex
Distancia Máxima con velocidad de referencia	15 m 19.2 kbps	1200 m 100 kbps	1200 m 100 kbps
Velocidad datos a 15[m]	19.2 kbps	10 Mbps	10 Mbps
Señalización	No Balanceada	Balanceada	Balanceada
1 Lógico	-5 V a -15 V	2 V a 6 V	1.5 V a 5 V
0 Lógico	5 V a 15 V	2 V a 6 V	1.5 V a 5 V

Descripción de diferencias entre comunicaciones seriales, Fuente: (B.R. & Jaganmohan , 2015), Elaborado por: Estefanía Olivo, Cristian Pazmiño.

2.6.2 Protocolo SPI

El protocolo SPI como sus siglas lo indican es una Interfaz Serial Periférica, esto quiere decir que permite transmitir datos en una comunicación maestro-esclavos entre dispositivos controladores y periféricos, brindando flexibilidad ya que permite evitar la necesidad de configurar un único bus de comunicación al separar la sincronización del reloj y envío de datos.

La forma de comunicación que efectúa SPI está basada en un dispositivo Maestro que controla y selecciona el envío de datos a una frecuencia de señal de reloj apropiada hacia los dispositivos esclavos por lo cual se modifica al comunicarse con un esclavo diferente; para el envío de datos desde el Maestro hacia los esclavos se utiliza la línea de

comunicación MOSI y por el contrario cuando los datos se envían desde un dispositivo esclavo hacia un maestro se utiliza la línea MISO. (Gonzales, 2006)

Al realizar una comparación entre el protocolo SPI y el protocolo Serial se aprecia que el bus SPI no establece un valor exacto de velocidad de datos, elimina la comunicación entre dispositivos esclavos y permite una comunicación independiente al estándar que utiliza cada dispositivo en particular; otro aspecto importante es la clasificación SPI:

- SPI Nivel 1: En un SPI de nivel 1, el registro de datos se almacena dos veces para permitir que un byte de datos sea leído mientras se recibe un nuevo byte de datos en la comunicación.
- SPI Nivel 2: Un SPI de nivel 2 permite entrelazar los circuitos periféricos en una sola comunicación SPI incrementando la posibilidad de configurar una conexión a nivel de software y hardware.

En la figura 2.6, se presenta el esquemático del bus de comunicación SPI en un ejemplo de un maestro con dos esclavos y el proceso de transmisión de bits de datos, en donde se observan las tres líneas de interfaz que son la señal de reloj controlada por el dispositivo maestro, las líneas de selección del dispositivo esclavo y las líneas de envío de datos hacia y desde el dispositivo maestro MISO y MOSI.

Figura 7.6 Diagrama de transmisión SPI entre un dispositivo Maestro y dos Esclavos

Esquemático de la comunicación SPI y el proceso de transmisión de datos, Fuente: (Kugelstadt, 2011)

2.7 Tarjetas de Radiofrecuencia MIFARE

MIFARE es la tecnología de tarjetas sin contacto más empleada alrededor del mundo desarrollada por Philips, estas tarjetas permiten realizar aplicaciones de automatización de procesos y operan a una distancia determinada por el alcance del módulo lector, poseen un rango de almacenamiento de 1 Kbyte hasta 8 Kbyte, la memoria puede ser extendida, pero provoca latencia en la lectura de datos; el promedio de tiempo de lectura corresponde a 848 KB/s. (Antonio J., Alberto F. , Miguel A., & Antonio F. G. , 2010)

Por seguridad necesario utilizar una técnica de integridad de datos que se conoce como campo de Comprobación de Redundancia Cíclica (CRC), este campo permite detectar errores de repetición de registros o instrucciones que provocan la necesidad de aumento de capacidad de almacenamiento; en el capítulo 3 correspondiente al diseño del presente proyecto presentara la descripción de los parámetros a los que las tarjetas y llaveros MIFARE operan.

CAPÍTULO 3

DISEÑO DEL PROTOTIPO

En este capítulo se describe la configuración, programación, conexión y parámetros del modo de operación entre los diferentes elementos que componen el control de acceso.

3.1 Descripción del modo de operación del Sistema de Control de Acceso

El modo de operación del sistema de control de acceso se divide en dos fases, la primera corresponde a la parte de hardware que se encarga de adquirir las señales y procesarlas en datos que serán enviados por la red codificados en forma binaria y hexadecimal, la segunda fase se encarga de almacenar los datos receptados en una base de datos que contiene información de los usuarios y los permisos que se asignan, para posteriormente compararlos mediante variables tipo texto en la programación almacenada en el ordenador de placa única Raspberry Pi 3 y así enviar órdenes de activación del actuador magnético tomando en cuenta el horario de cada docente, por otra parte en el servidor web se permite gestionar la información de usuarios, credenciales, acceder a consultas y registro de eventos. Otra ventaja que posee el prototipo de control de acceso planteado es el envío de mensajes de alerta a redes sociales mediante la programación de un API que es una interfaz de programación que permite realizar funciones establecidas dentro de otro software como es el caso de la placa Raspberry Pi 3.

Por ultimo al prototipo se le añade un sistema de prevención frente a posibles fallas conformado por un pulsador puertas adentro que se activará en caso de emergencia, un teclado puertas afuera que posee una clave única que será digitada en el caso de falla en la energía eléctrica o perdida de conexión con el servidor y un Sistema de Alimentación Ininterrumpida (UPS) que permita al sistema seguir funcionando y alimentando el actuador magnético en ausencia de energía eléctrica.

En la figura 3.1 de la página 26, se presenta el diagrama de bloques funcional que detalla de forma gráfica el modo de ejecución del Sistema de control de acceso.

3.2 Diseño de módulos del Sistema de Control de Acceso

3.2.1 Módulo de Control y Potencia

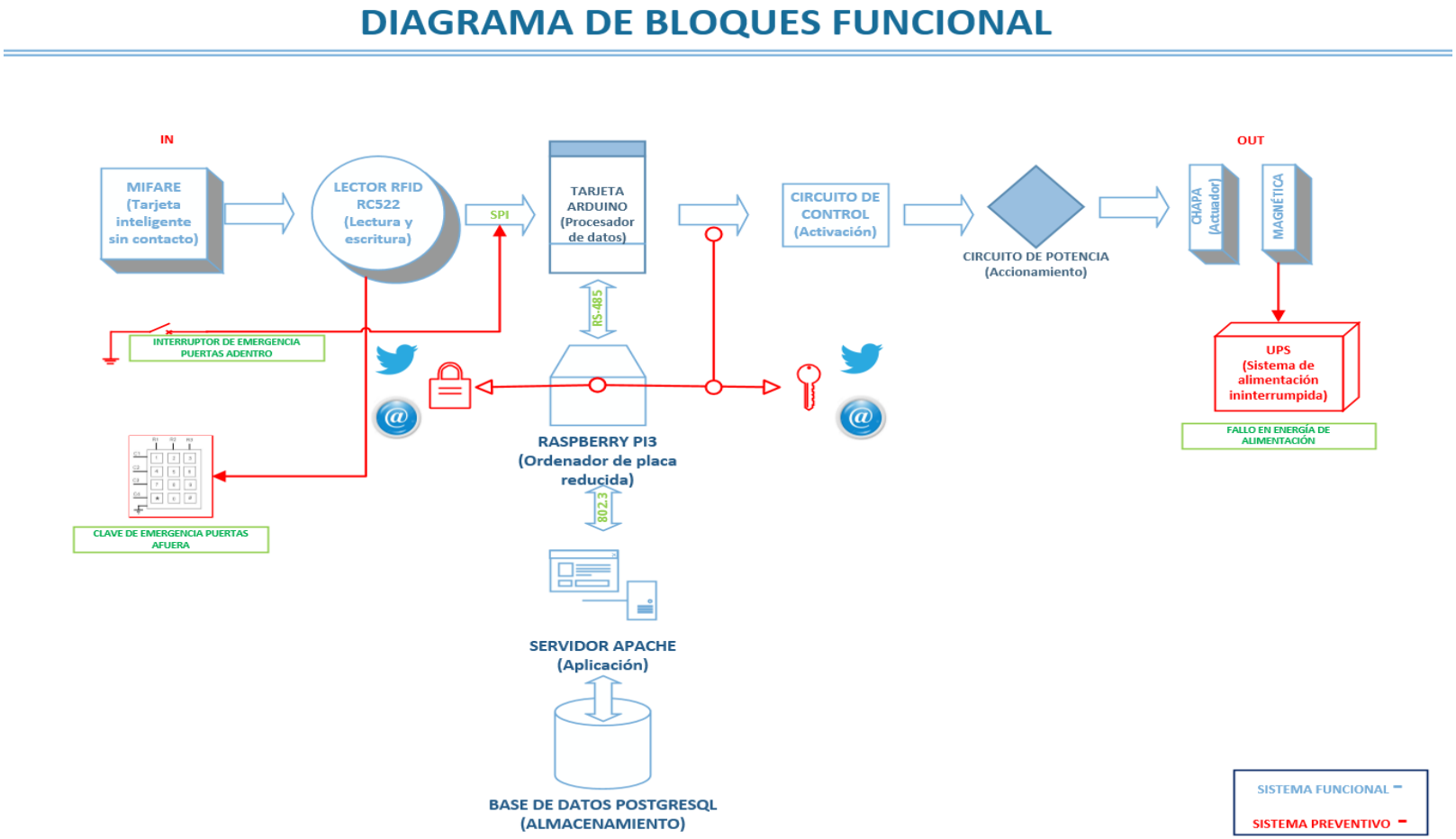
El circuito de control es el encargado de ordenar y dirigir el proceso de activación de entradas y salidas de datos que se obtienen a partir de los recursos conectados al sistema, la placa Arduino Mega actúa como dispositivo Maestro en el cual se seleccionan entradas digitales y voltajes de alimentación necesarios para el funcionamiento de los dispositivos esclavos; la tabla 3.1 detalla la distribución de pines, en el Anexo 2 se encuentra la conexión realizada y el Anexo 3 presenta la programación de control.

Tabla 3.1 Conexión de dispositivos esclavos hacia la placa Arduino Mega.

Dispositivo	**Entradas Digitales requeridas**	**Pines de conexión**
Tarjeta lectora de ingreso	8	SS: PIN 49 Ejecución: PIN50 - PIN52
Tarjeta lectora de salida	8	SS: PIN 42 Ejecución: PIN50 - PIN52
Teclado matricial 4*3	8	Filas: PIN 22-24-26-28 Columnas: PIN 30-32-34
Buzzer	1	PIN 10
Relé de activación	1	PIN 11
Pulsador de emergencia	1	PIN 12

Descripción de pines requeridos para la conexión con dispositivos esclavos, Elaborado por: Estefanía Olivo, Cristian Pazmiño.

Figura 8.1 Diagrama Funcional del Sistema de Control de Acceso



Modo de operación del Diseño del Control de Acceso, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

El control que se ejecuta en la placa Arduino captura el ID de las tarjetas MIFARE como dato hexadecimal mediante el protocolo de comunicación SPI, por consiguiente, según la validación de dicha ID se manda la señal de activación o desactivación al relé para la apertura de la puerta, esto se consigue mediante la generación de un lazo de programación FOR que compara los pines de selección de dispositivos esclavos (ssPIN) para identificar el lector y establecer una condición de asignación de valor de 0 para las lecturas de salida y un valor de 1 para las lecturas de ingreso. En el caso de la configuración del teclado, se descarga la librería correspondiente y se programa una clave única de 4 dígitos que puede ser modificada y activa la apertura de la puerta; para el caso del pulsador de emergencia se le asigna el permiso de apertura sin condicionamiento.

En el caso del circuito de potencia para la activación de la chapa magnética se utilizaron los elementos que se describen en la tabla 3.2 y la conexión que se presenta en la figura 3.2, el circuito mantiene energizado el actuador magnético al utilizar la conexión en modo normalmente cerrado del relé, al activarse una salida del Arduino Mega llega un pulso de 5 V a la base del transistor que energiza la bobina del relé provocando la conmutación del mismo y cortando la alimentación de 12 V de la cerradura para su apertura.

Tabla 3.2 Elementos del circuito de potencia.

Dispositivo	**Función**
Transistor 2N3904 NPN	Conmutador
Diodo 4007	Protección
Relé de activación 5 V	Conmutador
Chapa magnética 12 V	Actuador

Descripción de elementos del circuito de potencia, Elaborado por: Estefanía Olivo, Cristian Pazmiño.

Figura 9.2 Circuito de Potencia

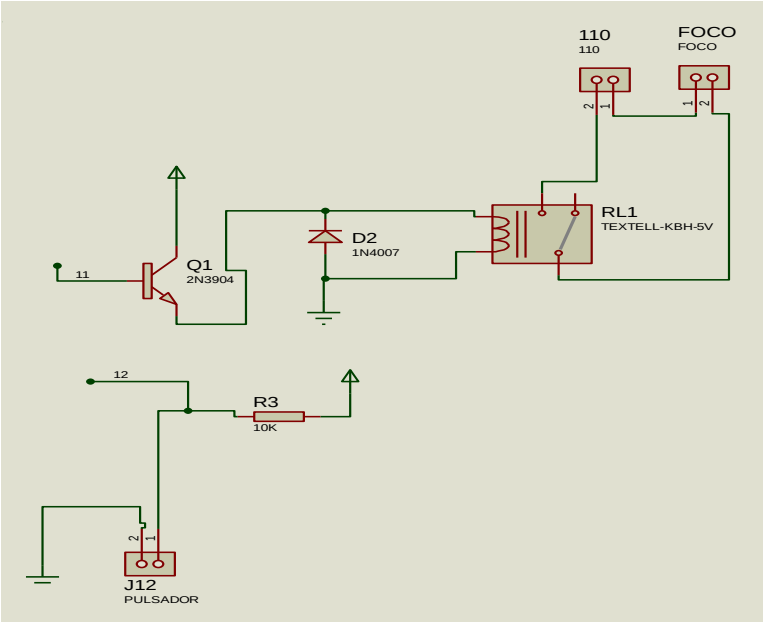
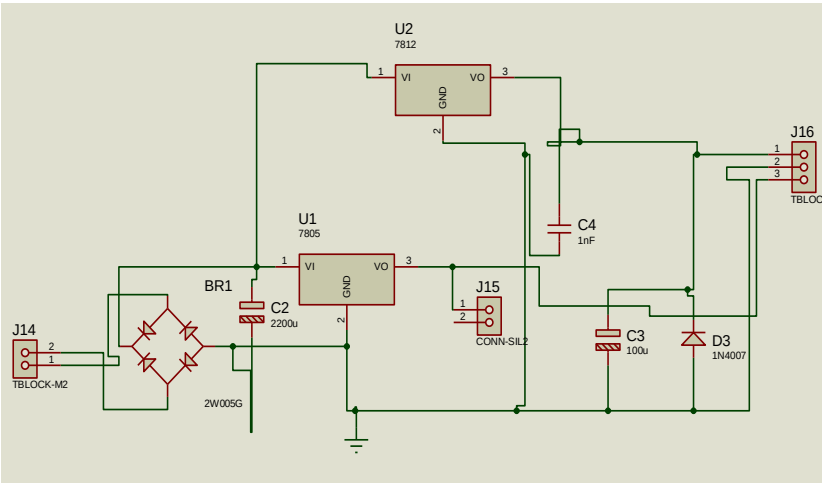


Diagrama del circuito de Potencia del Sistema de Control de Acceso, Elaborado por: Estefanía Olivo y Cristian Pazmiño

Por último, en este módulo se añade un circuito regulador de voltaje para la alimentación de 5 V de la placa de control y 12 V para la alimentación del circuito de potencia, su diseño se presenta a continuación en la figura 3.3.

Figura 10.3 Diagrama del circuito regulador



Circuito Regulador de 5 V y 12 V, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

3.2.2 Módulo de Adquisición y Procesamiento de Datos

Este módulo es el encargado de capturar datos a través de las tarjetas lectoras de radiofrecuencia y procesarlos en la placa Arduino Mega para ser enviados utilizando comunicación RS-485 al Ordenador de placa Única Raspberry Pi3.

Para obtener información de etiquetas de llaveros y tarjetas MIFARE es necesario acoplar al circuito de control dos lectoras RFID RC522 que actúan como sensores de señales de ondas de radio frecuencia, operando bajo los parámetros detallados en la tabla 3.3.

Para conocer la distancia de lectura de ondas de radiofrecuencia que el lector RFID RC522 alcanza, se analiza la frecuencia estándar del dispositivo lector, que en este caso equivale a 13.56 MHz y se compara con los datos de la tabla 2.1 del presente documento, donde se detalla el alcance que corresponde a 1,5 m para este rango de frecuencia. La ecuación 1 presenta la relación directamente proporcional entre distancia y frecuencia.

$$f \ \alpha \ Dmáx \qquad \text{Ec. (1.1)}$$

Inicialmente como dato se conoce la frecuencia de operación estándar de las tarjetas lectoras y se toma como velocidad de onda el valor estándar de la velocidad de la luz, refiriéndose a un sistema ideal sin interferencia; para determinar el valor de longitud de onda que se maneja en la propagación de ondas electromagnéticas en el sistema de radiofrecuencia, mediante la siguiente ecuación:

$$\lambda = \frac{c}{f} \qquad \text{Ec. (2.2)}$$

$$\lambda = \frac{3x10^{8}\ m/s}{13{,}56\ MHz}$$

$$\lambda = 22{,}12\ m$$

Al conocer la longitud de onda, se puede determinar la pérdida de potencia en la propagación de ondas, en función de la distancia de alcance; entre la etiqueta y el lector utilizando la ecuación 3.3.

$$Ploss = 20 * log[(4 * \pi * d)/\lambda] \quad \text{Ec. (3.3)}$$

$$Ploss = 20 * \log\left[\frac{(4 * \pi * 1{,}5\ m)}{22{,}12\ m}\right]$$

$$Ploss = -1{,}38\ dB \approx 0{,}72\ mW$$

El valor hallado representa la pérdida de potencia en la propagación de ondas del sistema de control de acceso al trabajar con un alcance de 1,5 m.

Tabla 3.3 Parámetros de lectura de etiquetas.

Parámetros De transmisión	**Lectores MFRC522**	**Etiquetas MIFARE**
Distancia de lectura	6 cm	5 cm – 10 cm
Frecuencia de operación	13.56 MHz	13.56 MHz
Rango de longitud de onda	22,12 m	22,12 m
Velocidad de transmisión de datos	10 Mbps	848 KB/s

Descripción de rangos de distancia y velocidad a la que operan las tarjetas lectoras y las etiquetas,

Elaborado por: Estefanía Olivo, Cristian Pazmiño.

En el diseño del control de acceso se utiliza una tarjeta lectora para la salida de usuarios y otra tarjeta para la lectura del ingreso de usuarios al espacio físico del laboratorio, para llevar a cabo dicha ejecución se conectan ambas tarjetas en paralelo con comunicación SPI como se detalla en la tabla 3.4.

Otro punto importante a tomar en cuenta es la distancia de posición entre lectoras, en este caso la distancia entre lectoras del diseño corresponde a 15 cm y al ocupar cada una un bus individual para la transmisión de datos se evita la interferencia entre ondas electromagnéticas.

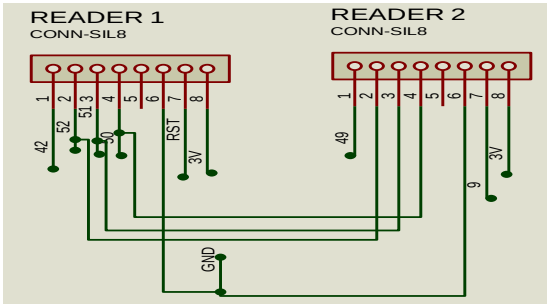
Tabla 3.4 Modo de conexión de tarjetas lectoras.

Líneas de Control de Comunicación SPI	Lector 1 MFRC522	Lector 2 MFRC522
SPI SCK	PIN 52	Paralelo
SPI SDA (SS)	PIN 42	PIN 49
SPI MISO	PIN 50	Paralelo
SPI MOSI	PIN 51	Paralelo

Descripción de la conexión en paralelo de las tarjetas lectoras, Elaborado por: Estefanía Olivo, Cristian Pazmiño.

La figura 3.4 presenta la conexión en paralelo de las tarjetas lectoras de ingreso y salida, dicha conexión permite enviar datos independientes de cada lectora ya que el responsable de habilitar la tarjeta correspondiente es el pin SDA.

Figura 11.4 Conexión de Adquisición de datos por Radiofrecuencia



Circuito de adquisición de datos, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

La segunda parte corresponde al procesamiento de datos con el Ordenador de Placa Única Raspberry Pi 3, en este caso el dispositivo Maestro Arduino Mega posee la lectura del ID capturado por el dispositivo lector en forma hexadecimal, para que el dato sea procesado por el Lenguaje Python que maneja Raspberry Pi 3 se utiliza el estándar industrial Unicode que admite cualquier codificación de caracteres como se explicó en el apartado 2.5.2 del presente documento, dicho ID se almacena en una variable tipo texto para hacer una consulta y realizar la comparación con las IDs registradas en la base de datos y verificar

si posee o no permiso de acceso y realizar el proceso de envío de mensajes de alerta que se detalla en la parte 3.4 del presente capítulo.

3.3 Gestión y Administración

En esta parte se realiza el software y administración de la información recolectada por los dispositivos del Sistema de Control de Acceso.

3.3.1 Creación de la Base de datos

La base de datos seleccionada para la realización del Sistema de Control de acceso es PostgreSQL puesto que a comparación de otras bases de datos presenta mayor escalabilidad dado que el número de peticiones que soporta es alto, maneja código abierto por lo cual se puede utilizar en diferentes aplicaciones de forma económica, genera respaldos de información y su memoria es extensible.

3.3.1.1 Diagrama Entidad/Relación

En el diagrama entidad/relación se toma en cuenta las diferentes tablas que se generan para definir los campos y almacenar los registros de los usuarios y elementos sobre los que se va realizar el control; la figura 3.5 presenta el diagrama entidad/relación basado en el modelo relacional de base de datos estableciendo llaves primarias y foráneas para el Control de Acceso al Laboratorio 3 del Bloque D de CISCO tomando en cuenta el docente, la materia, el horario; se crea relación entre la cedula de cada usuario y el ID de tarjeta que se le asigna, también se toma en cuenta el código de la materia que dicta el docente para asignarle un horario que se almacena en variables de timestamp.

Figura 12.5 Diagrama Entidad/Relación de la Base de Datos PostgreSQL

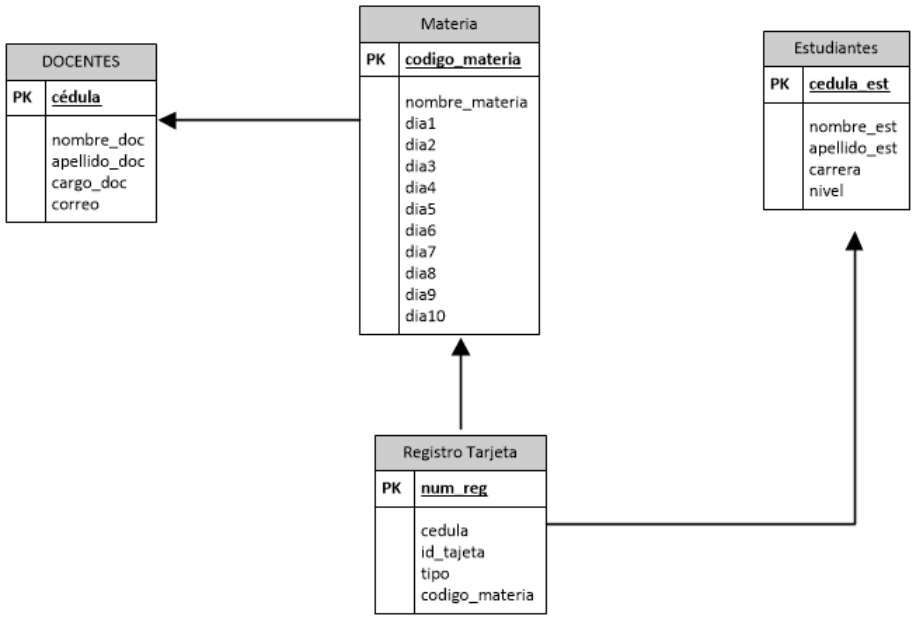


Diagrama entidad/relación de la lógica empleada en la base de datos, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

A continuación se explica el proceso planteado en la base de datos, lo primero que se analiza después de obtener el ID almacenado es la tabla Materia que determina el acceso según el día y la hora, realizando una consulta para establecer la coincidencia con los registros almacenados del horario de cada docente, se extrae dicha información comparando el número de día del horario, por ejemplo si el docente tiene clases el martes el número de día sería 2 una vez extraído el número de día, se hace una consulta del día actual y se compara si las dos variables son iguales, en el caso de que sea falso se envía un registro a la base indicando que el usuario no tiene acceso; por otro lado si la comparación es verdadera entra a la siguiente sentencia de comparación y se envía mediante comunicación serial el carácter ASCII “a” para indicar al dispositivo Maestro Arduino Mega que tiene que activar el actuador y se registra el evento en la base. La lógica empleada en el flujo de información entre la base, el servidor y actuadores se detalla esquemáticamente en el Anexo 4.

3.3.2 Conexión al servidor

La gestión del control de acceso se almacena en la plataforma Windows 8 Server, sobre el paquete XAMP que posee la funcionalidad de integrar los servicios que se requieren como es el caso de MySQL, Apache y PHP; en esta fase se realiza el diseño de la aplicación o página web que actúa como interfaz visual para el usuario, la programación se realizó sobre un archivo index cargado en el dominio de la aplicación con el fin de estructurar una página principal a la cual se van a indexar archivos PHP que establecen ordenes de consulta y formularios, el proceso de ejecución de los archivos es secuencial y se presenta en el diagrama correspondiente al Anexo 5 y el Anexo 6 contiene una lista de atributos para crear la interfaz web. Por último, la tabla 3.5 presenta los principios básicos que se siguen para establecer la conexión de un gestor de bases de datos hacia el servidor:

Tabla 3.5 Actividades a desarrollar para establecer la conexión con el servidor.

Actividad	Función	Codificación
Conexión con el servidor	Asignar una dirección IP, nombre de usuario y contraseña.	**$conexion = pg connect ("localhost", "user", "password")**
Seleccionar base de datos	Se selecciona a través de codificación en PHP.	**pg select db ("name", $conexion)**
Realizar consultas SQL	Se asocia consulta a la base de datos por medio de un enlace y una instrucción.	**$consulta = "SELECT * FROM table name;"** **$res =pg query ($consulta, $conexion) or die ("incorrecto")**
Procesar resultados	Se basa en programación de comandos para devolver datos e insertar o modificar registros, se presenta en orden la codificación de numero de filas, modificación de un registro y resultado.	**$nfilas = pg num rows ($res);** **$modi = pg affected rows ($res);** **$resultado = pg fetch rows ($res);**

Continuación Tabla 3.6 Actividades a desarrollar para establecer la conexión con el servidor.

Actividad	Función	Codificación
Terminar la conexión	Su función es evitar que la conexión permanezca activada de forma innecesaria.	**pg close ($conexion);**
Habilitar extensiones	Las extensiones en el archivo index.php permite selecciona la base de datos PostgreSQL en XAMPP	**extensión=php_pdo_pgsql.dll** **extensión=php_pgsql.dll**

Actividades, funciones y codificación de parámetros para la conexión de la base de datos y el servidor, Elaborado por: Estefanía Olivo, Cristian Pazmiño.

Es importante revisar que el puerto 5432 se encuentre activo para escuchar las peticiones de la base de datos PostgreSQL.

3.4 Mensajes de alerta

Esta sección es la encargada de aumentar el nivel de seguridad del Sistema de control de Acceso, ya que bajo la generación de un API se establecen los parámetros necesarios para el envío de mensajes de alerta a la red social Twitter y la aplicación de correo.

3.4.1 Mensajes de alerta a Twitter

Twitter es una de las redes sociales más utilizadas hoy en día, gracias a sus herramientas de desarrollador se puede crear aplicaciones o API, con el fin de mandar mensajes desde ordenadores como Raspberry Pi utilizando lenguaje de programación Python, el cual se seleccionó para llevar a cabo el diseño del presente proyecto.

Para poder mandar tweets desde la script de Python es necesario crear el API correspondiente enlazando a una cuenta de Twitter, una vez creado el API se procede a

instalar la librería en el Raspberry Pi 3 desde el terminal ejecutando el comando “sudo apt-get install tweepy”.

Tweepy es la librería que se requiere para poder enviar tweets desde Python, su uso depende del desarrollador y brinda la capacidad de ver tweets, actualizar contenido como: el estado, foto de perfil, entre otros.

Para que el control de acceso determine en qué momento enviar una alerta se requiere configurar una serie de parámetros que se detallan a continuación:

- Incluir la librería en el script
- Realizar un query o consulta a la base de datos con la petición de los datos del docente al que se enviará en el mensaje.
- Definir las claves necesarias para que la aplicación se enlace con la API de Twitter.
- Crear una variable que contenga las claves para la autenticación del usuario.
- Enviar el mensaje de alerta a twitter.

La variable “y” que se observa en el anexo 7 correspondiente al código empleado para la generación de APIs contiene la autentificación necesaria del usuario de la cuenta para permitir realizar cualquier acción posible en twitter, en este caso se hace una actualización del estado, para esto, se guarda el mensaje en la variable tweet donde se agrega todo el contenido necesario como el nombre, apellido, fecha y hora de ingreso del docente y un mensaje en caso de que el botón de emergencia se accione por algún tipo de evento inesperado.

3.4.2 Mensajes de correo

El correo electrónico es otra herramienta muy útil a la hora de enviar o recibir mensajes de alerta de accesos, con el propósito de tener respaldo de dichos mensajes en un servidor; por esta razón se generan alertas del control de acceso para enviar automáticamente un correo a cualquier servidor SMTP ya sea Hotmail, Gmail e incluso correo institucional

cuando el usuario ingrese, salga o se active el pulsador de emergencia. Es importante abrir el puerto de mensajería SMTP 587.

Para poder enviar un correo electrónico se utilizó la librería smtplib la cual se encuentra incluida en Python y se configuran los siguientes parámetros:

- Crear el objeto smtplib.SMTP, el cuál recibirá como parámetro el localhost y el servidor SMTP correspondiente, por ejemplo, el de Gmail smtp.gmail.com:587.
- Preparar el mensaje donde se define el asunto, encabezado y mensaje.
- Para el envío se realiza una llamada al método sendmail del objeto SMTP.

3.5 Criterios de dimensionamiento del Proyecto

El presente diseño se realizó considerando parámetros de interfaces de comunicación entre dispositivos y características del ambiente al trabajar con rangos de frecuencia.

3.5.1 Cálculo de la resistencia de PULL-UP

El protocolo SPI permite establecer parámetros de velocidad de transmisión y frecuencias de reloj en el orden de los MHz (Ultra Fast mode).

Para el diseño se utilizó un Arduino Mega como dispositivo maestro cuya frecuencia de funcionamiento viene dada por el cristal que posee y corresponde a 16 [MHz], por otra parte se conoce que el bus SPI posee 2 bits SPR1 Y SPR2 que controlan la velocidad SCK del dispositivo maestro y un tercer bit SPI2X para proporcionar el doble de velocidad de transmisión como se presenta en el Anexo 8 dependiendo del valor lógico que se envía; se obtiene una relación de frecuencia SPI en función de la frecuencia del oscilador que obedece a la ecuación 3.4 de la librería SPI que maneja Arduino.

$$fspi = \frac{fosc}{2} \qquad \text{Ec. (4.4)}$$

$$fspi = \frac{16MHz}{2} = 8MHz$$

$$fspi = 8MHz$$

Por lo tanto, la frecuencia SPI opera en el rango de alta frecuencia, que posee mayor facilidad de interferencia provocada por ruido, esto se puede evitar utilizando resistencias de pull-up que permiten un adecuado manejo de las señales que se envían, sin perder la velocidad de transmisión aunque aumenta el nivel de consumo; para el presente diseño se determina el valor de resistencia más pequeño que se limita por la corriente de operación máxima de los dispositivos conectados al bus que corresponde a 3 mA, dicho valor depende de la tensión de alimentación de las lectoras RFID que equivale a 3.3 V.

La ecuación 3.5 toma en cuenta los valores de corriente máxima y tensión de alimentación máxima para hallar el valor mínimo de resistencia de pull-up para evitar la interferencia de ruido en el bus SPI.

$$Rmp > \frac{VDD}{ID} \qquad \text{Ec. (5.5)}$$

$$Rmp > \frac{3.3\ V}{3\ mA}$$

$$Rmp > 1{,}1k\Omega$$

Dónde:

Rmp = Resistencia mínima de pull-up

VDD = Tensión de alimentación

ID = Corriente máxima

3.5.2 Cálculo de la longitud máxima del bus SPI

El protocolo SPI soporta una capacidad máxima de 400 pF entre todos los dispositivos conectados, para el diseño del control de acceso esto puede limitar el número de dispositivos esclavos que se utilizan como es el caso de la lectora de ingreso y la lectora de salida, esto depende del tipo de bus a utilizar; en este caso para el diseño se toma en cuenta el cable F/UTP categoría 6A cuya capacitancia de desbalanceo equivale a 330 pF

por cada 100 metros lo que presenta una relación de 3.3 pF por metro como se observa en la tabla 3.6.

Tabla 3.7 Especificaciones eléctricas del cable UTP cat6A.

Características eléctricas	**Rango**
Resistencia DC	<8,5 Ω / 100 m
Resistencia de desbalanceo DC	5%
Capacitancia mutua	5,6 nF / 100 m
Capacitancia de desbalanceo	<330 pF / 100 m
Retardo de inclinación	≤ 45 ns

Características del cable UTP cat6A, Elaborado por: Estefanía Olivo, Cristian Pazmiño.

Sí para el diseño se van a utilizar dos lectoras RFID con comunicación SPI, sumando los 5 pF de capacitancia que cada tarjeta añade y conociendo la capacitancia de desbalanceo del cable se puede calcular la capacitancia total que maneja el bus SPI. La ecuación 3.6 presenta el cálculo para hallar la capacitancia total del sistema.

$$Cap.Total = 2 * Capacitancia(RFID) + Capacitancia\ de\ desbalaceo\ del\ cable$$ Ec. (6.6)

$$Capacitancia\ Total = 2 * \left(\frac{5\ pF}{1\ m}\right) + 3.3\frac{pF}{m}$$

$$Capacitancia\ Total = 13.3\ pF/m$$

Entonces se comprueba que las dos lectoras conectadas no superan el valor de capacitancia máxima del protocolo correspondiente a 400 pF y al conocer los valores de capacitancia máxima y la total del diseño del Sistema de control de acceso, se puede calcular la ecuación 3.7 correspondiente a la longitud máxima soportada por el diseño.

$$Longitud\ Máxima = \frac{Capacitancia\ Total}{Capacitancia\ Máxima/m}$$ Ec. (7.7)

$$Longitud\ Máxima = \frac{400pF}{13.3pF/m}$$

$$Longitud\ Máxima = 30{,}08\ m$$

3.5.3 Escalabilidad del proyecto en el área de los laboratorios del Bloque D

Según el análisis realizado para hallar parámetros de distancia de comunicación de los protocolos SPI, serial y conexión entre el dispositivo maestro Arduino Mega y los dispositivos esclavos; se determina que el Sistema de control de acceso aporta escalabilidad puesto que se puede implementar el prototipo hasta en 5 laboratorios, tomando en cuenta el valor de capacitancia máxima de la ecuación 8 añadiendo 4 puertas más como se presenta a continuación en la ecuación 3.8:

$$C.Total = \#lectores * Capacitancia(RFID) + Cap.de\ desbalaceo\ del\ cable \qquad \text{Ec. (8.8)}$$

$$Capacitancia\ Total = 10 * \left(\frac{4\ pF}{1\ m}\right) + 3.3\ pF/m$$

$$Capacitancia\ Total = 43.3\frac{pF}{m}$$

Por lo tanto, el radio de la longitud que soporta se plantea en la ecuación 9:

$$Longitud\ Máxima = \frac{Capacitancia\ Total}{Capacitancia\ Máxima/m} \qquad \text{Ec. (9.9)}$$

$$Longitud\ Máxima = \frac{400pF}{43.3pF/m} = 9.24\ m$$

$$Longitud\ Máxima = 9.24\ m$$

Por otra parte, respecto al dispositivo maestro se requiere un Arduino Mega para realizar el control de las 5 puertas ya que cuenta con 54 puertos de conexión digital y en el caso de los mensajes de alerta y el Sistema de alimentación ininterrumpida se requiere un ordenador Raspberry Pi 3 y un UPS por cada laboratorio.

El anexo 9 presenta un plano del espacio físico de los laboratorios del bloque D y el alcance que abarca el presente diseño para brindar escalabilidad.

CAPÍTULO 4

IMPLEMENTACIÓN Y PRUEBAS

En este capítulo se detalla la puesta en marcha del plan piloto de Control de Acceso para el laboratorio 3 y las pruebas realizadas al integrar la interfaz web con cada módulo de adquisición de datos.

4.1 Requerimientos tecnológicos y físicos para el montaje del plan piloto de Control de Acceso

Para llevar a cabo el montaje del plan piloto de control de Acceso se presenta la siguiente lista de requerimientos tecnológicos y físicos:

- Placa Arduino Mega
- Ordenador de placa única Raspberry Pi 3
- Lectoras RFID
- Teclado matricial 4x3
- Pulsador para paro de emergencia
- Cable UTP Cat6A
- Tubería EMT
- Cable gemelo
- Fuente de alimentación Ininterrumpida (UPS)
- Caja de aluminio 20x20
- Servidor
- Resistencias
- Capacitores electrolíticos y cerámicos
- Placa de fibra de vidrio
- Acrílico
- Diodo
- Transistores
- Fuente de alimentación 12 VDC
- Transformador 120/12 VAC

- Chapa magnética
- Brazo hidráulico
- Tarjetas y llaveros MIFARE
- Buzzer

El presupuesto correspondiente se encuentra en el Anexo 10 y la forma en la cual se llevó a cabo la instalación en el espacio físico del laboratorio se presenta en el Anexo 11 en donde se indica la posición en la cual se instaló cada punto.

4.2 Normas de calidad para el montaje de Sistemas de Radiofrecuencia

En ambientes de radiofrecuencia se utilizan por lo general circuitos integrados de tarjetas lectoras, para este caso se eligieron las tarjetas MFRC522 de tipo lectura / escritura altamente integrado para comunicaciones sin contacto.

Como se colocó en la tabla 3.3 del presente documento estas tarjetas operan a una frecuencia estándar de 13,56 MHz, para este tipo de lectores se utilizan normas de acuerdo a la aplicación, en este caso para el control de acceso se utilizan las normas de la tabla 4.1.

Tabla 4.1 Estándares para la instalación de Sistemas basados en Radiofrecuencia.

Denominación	Descripción
ISO/IEC 11784-11785	Norma de privacidad y seguridad de los datos
ISO 14443	Sistemas de identificación y documentación personal
ISO 24710	Técnicas para gestión de objetos

Características del cable UTP cat6A. Elaborado por: Estefanía Olivo, Cristian Pazmiño.

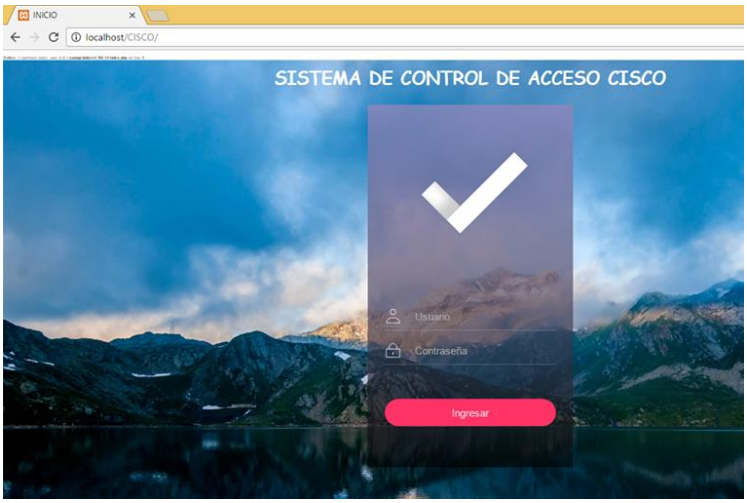
El presente proyecto se basa principalmente en la norma ISO 14443 que es el estándar indicado para Sistemas de Radiofrecuencia que manejan una frecuencia de operación equivalente a 13,56 MHz de la banda HF, aquí se plantean especificaciones para la instalación y puesta en marcha determinando las características físicas que se soportan,

los niveles de potencia en los que operan las lectoras y el protocolo de transmisión SPI; al basarse en dichos estándares se realizó la conexión en paralelo entre las dos tarjetas lectoras que se muestra en la figura 3.4 del presente documento y el cálculo de la capacidad máxima tolerada en cuestión de longitud.

4.3 Modo de funcionamiento de la aplicación web

La aplicación Web define dos tipos de usuarios que son el administrador que puede visualizar todas las opciones de la aplicación y el monitor que posee un nivel de privilegio inferior y solo puede visualizar ciertos parámetros; para empezar a cada uno se le asigna una contraseña diferente y acceden al Sistema de gestión al abrir un Buscador de internet y colocar la dirección http://172.17.34.25/CISCO; para posteriormente identificarse con el nombre de usuario y contraseña correspondiente, se puede observar un formulario como se presenta en la figura 4.1:

Figura 4.1 Dirección Web del servidor



Inicio de sesión, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

4.3.1 Usuario Monitor

Al ingresar el usuario y contraseña correspondiente se visualiza la página de inicio para este tipo de usuario, que presenta dos opciones, la primera permite consultar usuarios por categoría y la segunda brinda acceso a los registros almacenados (Ver figura 4.2).

Figura 13.2 Pantalla de inicio para el usuario monitor

Menú de opciones para el usuario monitor, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

La primera pestaña del menú posee las opciones que se describen en la figura 4.3.

Opción 1: Este enlace permite realizar una consulta de usuario ingresando los apellidos.

Opción 2: Permite realizar una consulta de usuario ingresando la ID de tarjeta asignada.

Opción 3: Aquí se ingresa el número de cedula para consultar si el usuario está registrado.

Figura 4.3 Menú de opción de consulta para el usuario monitor

Enlaces de consulta para el usuario monitor. Elaborado por: Estefanía Olivo y Cristian Pazmiño.

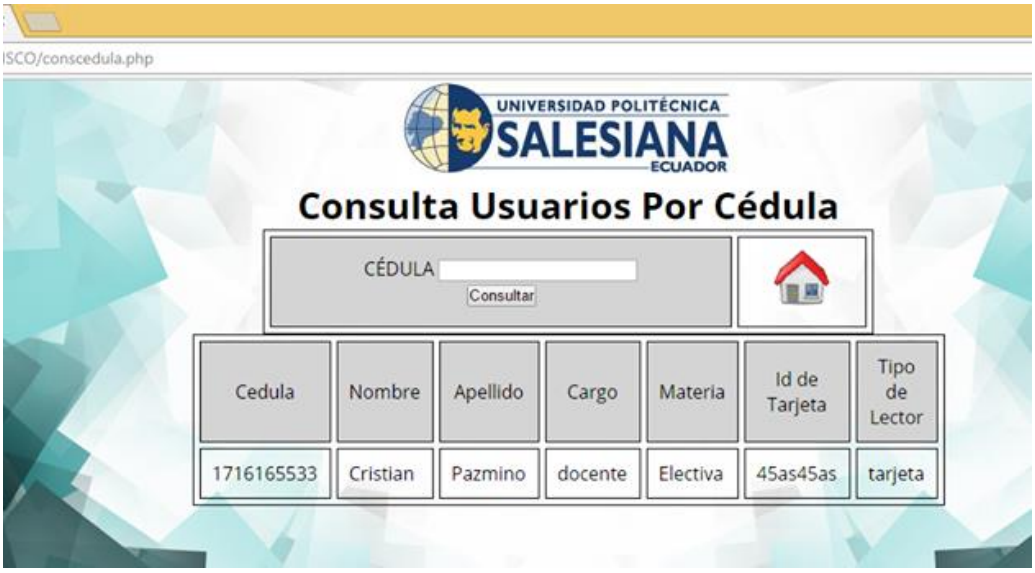
Como resultado se ejecuta de forma independiente el formulario de la figura 4.4.

Figura 4.4 Modo de consulta de usuarios por apellido

Formulario de consulta por apellido, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

La opción de consulta por número de cedula se presenta en la figura 4.5.

Figura 4.5 Modo de consulta de usuarios por cédula de identidad



Formulario de consulta por cédula, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

De igual manera en el caso de consulta por ID de tarjeta (Ver figura 4.6).

Figura 4.6 Modo de consulta de usuarios por ID de tarjeta

Formulario de consulta por ID, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

La segunda pestaña del menú posee las opciones que se describen en la figura 4.7.

Opción 1: Permite visualizar el registro de los accesos permitidos y rechazados.

Opción 2: Permite la descarga de un archivo tipo PDF de la información de registros.

Opción 3: Esta opción genera un gráfico estadístico del número de eventos realizados.

Figura 4.7 Menú de opción de Registro de Eventos para el usuario Monitor

Opciones de registro de eventos. Elaborado por: Estefanía Olivo y Cristian Pazmiño.

4.3.2 Usuario Administrador

En este caso el administrador posee el máximo nivel de privilegio y puede visualizar todas las opciones en el menú de inicio, también puede ingresar a la opción de configuración, en la cual realiza la parametrización de envío de mensajes de alerta (Ver figura 4.8)

Figura 4.8 Pantalla de inicio para el usuario administrador

Menú de opciones para el usuario administrador, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

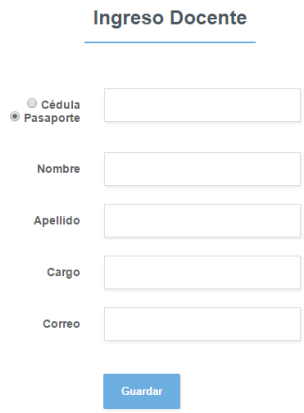
En el apartado 4.3.1 se detallaron las opciones de consulta de usuarios por categorías y registro de eventos que también se encuentran a disposición del usuario monitor; por dicho motivo se van a presentar pruebas del funcionamiento de las opciones de enlace para editar usuarios, materia y registrar tarjetas.

Donde solamente el usuario administrador posee permisos para configurar o editar campos e información almacenada en la base de datos y se presentan las siguientes opciones:

Opción 1: Corresponde al ingreso de usuarios y crea un enlace a un formulario, donde se llenan datos personales como nombre y cédula, también se permite actualizar o borrar un usuario como se presenta en las figuras 4.9, 4.10 y 4.11.

Opción 2: En el ingreso de materia se coloca el código de la asignatura que dicta el docente, el horario de clases y permite actualizar o borrar la materia, como se puede observar en las figuras 4.12 y 4.13.

Figura 4.9 Formulario de Ingreso Docente



Formulario de datos personales del docente. Elaborado por: Estefanía Olivo y Cristian Pazmiño.

Figura 4.10 Función de actualización del cargo de un docente

Editar cargo del docente, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

Figura 4.11 Función de borrar un usuario

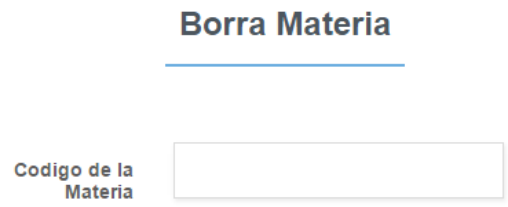
Borrar un docente, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

Las siguientes figuras presentan el resultado de la opción 2 correspondiente a la configuración de materias.

Figura 4.12 Función de Ingreso de materia

Ingresar el horario de una materia y su código, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

Figura 4.13 Función de borrar una materia



Borrar materia, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

Opción 3: Esta opción permite al usuario administrador asignar una tarjeta a un usuario como se observa en la figura 4.14; y la figura 4.15 presenta la opción de asignar una nueva tarjeta en caso de pérdida por parte del usuario.

Figura 4.14 Función de asignación de tarjeta a un usuario

Asignación de ID de tarjeta, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

Figura 4.15 Función de asignación de tarjeta nueva un usuario

Actualización de ID de tarjeta, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

4.3.3 Pruebas de registros y reportes

La figura 4.16 presenta el modo en el que se visualiza el registro de eventos.

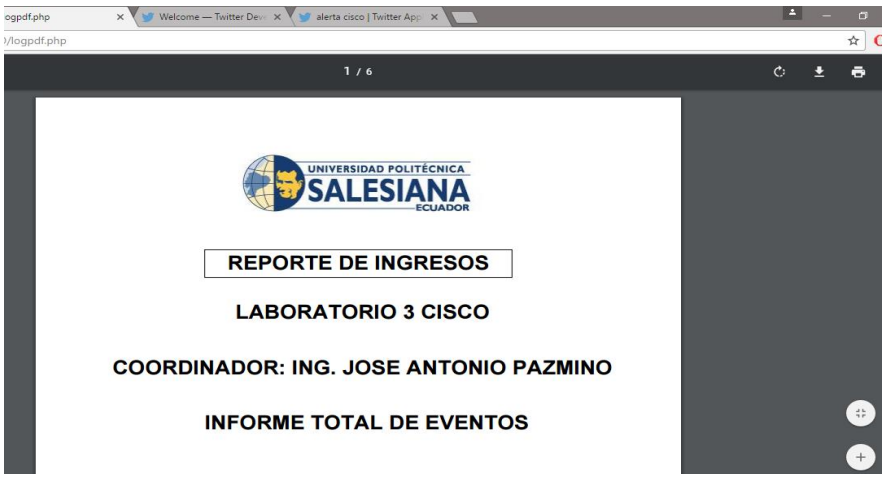
Figura 4.16 Registro de eventos de los usuarios

Número	ID de tarjeta	Nombre del docente	Apellido del docente	Fecha	Comentario
1	242D59EB	Alejandro	Moreno	2016-12-15 14:41:07.561395	no tiene acceso
2	242D59EB	Alejandro	Moreno	2016-12-15 14:41:07.561395	no tiene acceso
3	242D59EB	Alejandro	Moreno	2016-12-15 14:43:20.662666	no tiene acceso

Registro de accesos, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

En la figura 4.17 se puede observar el formato PDF en el cual se descarga el reporte de eventos almacenados.

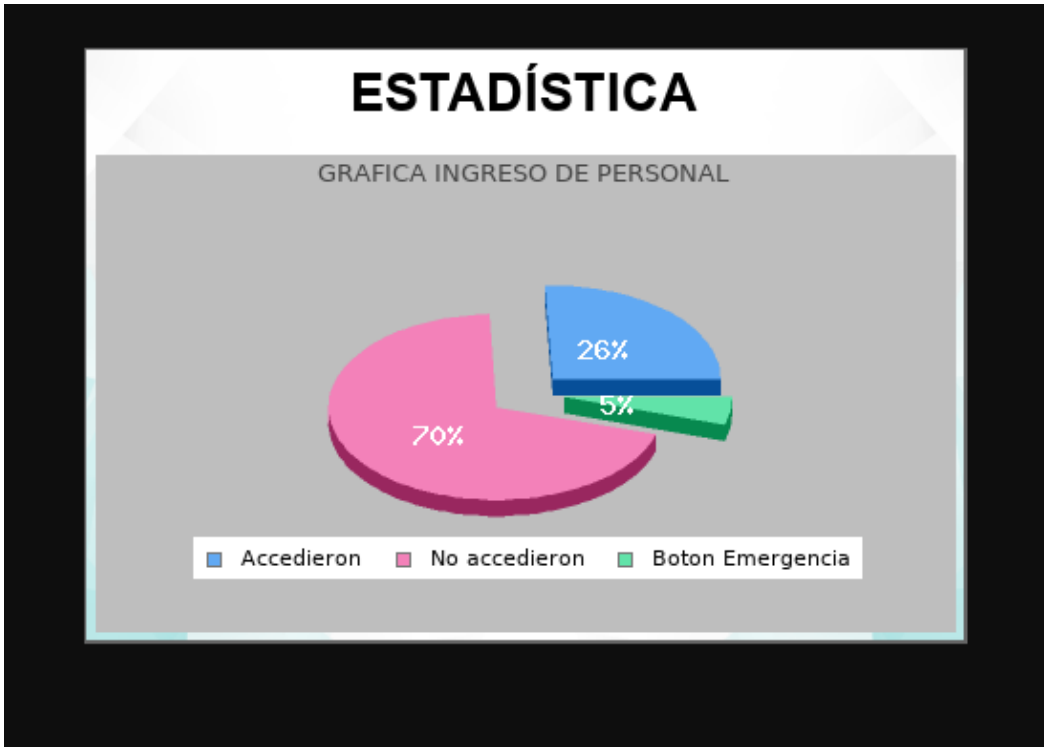
Figura 4.17 Descarga de archivo de reporte en formato PDF.



Reporte de eventos en archivo PDF, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

Por último, en la figura 4.18 se presenta el diagrama estadístico del registro de eventos, donde se toma en cuenta el número de veces que se intentó acceder con permiso, sin permiso y el número de ocasiones que se activó el botón de emergencia.

Figura 4.18 Estadística de eventos almacenados.

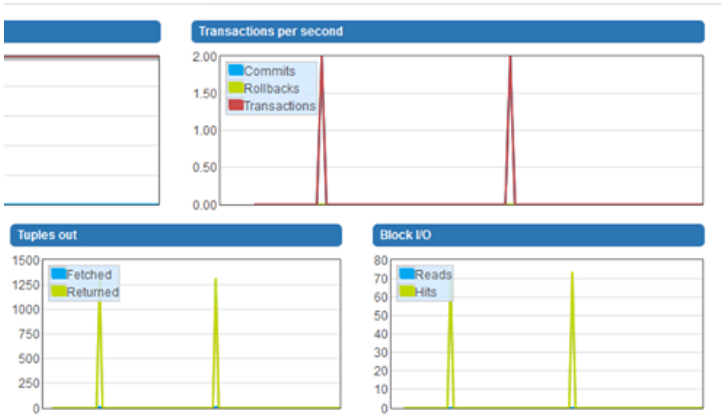


Gráfica de estadística de eventos, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

4.3.4 Diagrama de flujo de transacciones

Los diagramas de la figura 4.19 registran y muestran el número de transacciones por segundo, el número de tuplas que corresponde a la agrupación de valores como un único valor y los bloques de entrada y de salida que se procesan, se observan 2 transacciones por segundo, un rango de 1000 a 1250 tuplas y un rango de 60 a 70 bloques de entrada y salida al poner en marcha el sistema.

Figura 4.19 Estadística de transacciones por segundo, tuplas y bloques I/O.



Gráficas del comportamiento de la base de datos, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

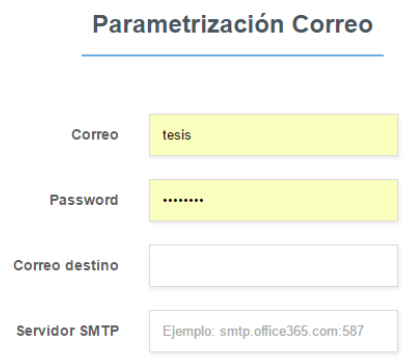
4.4 Pruebas de envío de mensajes de alerta

Como se mencionó en el punto 4.3.2 se otorgan permisos de configuración de los mensajes de alerta al usuario administrador, la pestaña de opciones se denomina configuración y permite realizar dos acciones que se detallan a continuación:

Opción 1: Permite parametrizar el correo al cual serán enviados los mensajes de alerta (Ver figura 4.20).

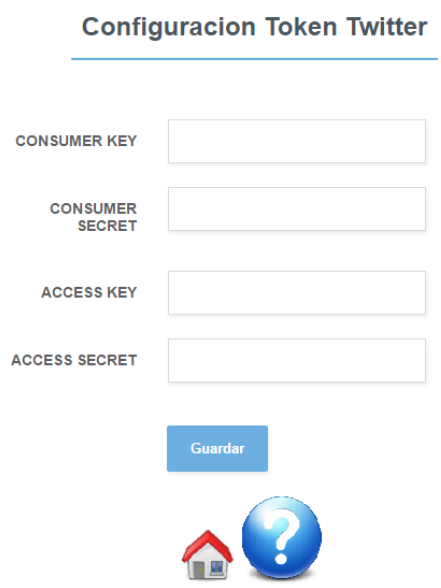
Opción 2: Permite parametrizar las claves que asigna Twitter para poder enviar mensajes de alerta (Ver figura 4.21).

Figura 4.20 Parámetros de configuración de correo.



Parametrización de correo de envío, destino y tipo de servidor SMTP, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

Figura 4.21 Asignación de tokens para Twitter.



Parametrización de Token de Twitter, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

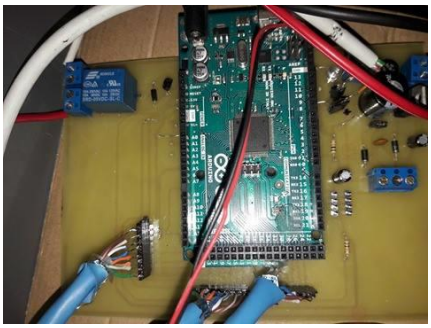
El símbolo de ayuda de la figura 4.21 contiene un manual de configuración de envío de alertas a Twitter que se presenta en el anexo 12.

4.6 Instalación final del Sistema de Control de Acceso

En este punto se presenta el resultado de la instalación realizada en el espacio físico del laboratorio 3, montando los dispositivos esclavos, el dispositivo maestro y actuadores dividido en las siguientes etapas:

- Circuito de control (Ver figura 4.22).
- Lector de salida (Ver figura 4.23).
- Lector y teclado de ingreso (Ver figura 4.24).
- Actuador magnético, brazo hidráulico y botón de emergencia (Ver figura 4.25).

Figura 4.22 Placa de control.



Circuito de control del Sistema de Control de Acceso, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

Figura 4.23 Lector de Salida

Lector de Salida del Sistema de Control de Acceso, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

Figura 4.24 Sistema de ingreso.

Lector y Teclado de Ingreso del Sistema de Control de Acceso, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

Figura 4.25 Actuador magnético, brazo hidráulico y botón de emergencia

Actuadores de salida del Sistema de Control de Acceso, Elaborado por: Estefanía Olivo y Cristian Pazmiño.

CONCLUSIONES

El sistema de control de acceso utilizando tecnología RFID cumple con todos los objetivos establecidos en el presente proyecto y respeta las especificaciones planteadas; se agregó la opción de visualizar registros a través de un servidor web, diagramas de pastel que realizan un análisis estadístico de los eventos almacenados y descarga del reporte en formato PDF, lo cual brinda al administrador mayor facilidad de detectar inasistencias y posibles intentos de afectar la seguridad del área física del laboratorio.

El utilizar tarjetas MIFARE ofrece una gran variedad de aplicaciones, esto se debe tomar en cuenta para futuros proyectos, ya que estas tarjetas poseen una memoria interna integrada de 1 Kilobyte, donde se puede almacenar la información según la necesidad de la aplicación; un ejemplo de aquello puede ser el uso del carné institucional de docentes, estudiantes y personal administrativo con la funcionalidad de tener acceso a las instalaciones, aulas o laboratorios en el horario asignado brindando un control automático con la posibilidad de generar reportes al editar la aplicación web que se ejecuta en Windows 8 server.

Al emplear comunicación serial RS-485 para la transmisión de datos entre el Arduino Mega y el ordenador Raspberry Pi 3 se logró encapsular dentro de un paquete los datos y enviarlos por medio de una red TCP/IP, viajando de esta forma por un túnel de información transparente para el usuario y sin modificar el contenido original.

Al usar ordenadores de placa reducida como Raspberry Pi 3 se pueden obtener velocidades de procesamiento mayores en comparación a la de un controlador lógico programable; por ejemplo, al incorporar Arduino al momento de trabajar con flujo de datos grande, el procesador de 1.2 GHz de cuatro núcleos de la Raspberry Pi 3 facilita la multitarea al momento de realizar una consulta a una base de datos o la función de enviar y recibir datos de manera serial.

El uso de servidores web como herramienta de administración y monitoreo para el control de acceso, presenta muchas ventajas con respecto a las herramientas manejadas por software, ya que permite realizar conexión remota para brindar al usuario la facilidad de acceder a verificar el correcto funcionamiento del proceso.

RECOMENDACIONES

Al ser un plan piloto, el dimensionamiento del proyecto se reduce a las instalaciones del BLOQUE D del campus-sur específicamente el Laboratorio 3; debido a que el protocolo de comunicación SPI utilizado para el presente proyecto se adecuó para cumplir el funcionamiento dentro de dicho espacio físico, se debe tener en cuenta que para lograr escalabilidad del proyecto a un número mayor a 5 puertas se debe realizar una conversión de protocolos de SPI a RS485 para poder implementar el mismo diseño en distancias de hasta 100 m; esto se realiza para evitar el retraso o pérdida de los datos que se transmiten desde las lectoras RDIF y controlar los posibles errores de instalación del sistema.

Como alternativa a la utilización de identificadores MIFARE, también se pueden utilizar celulares con tecnología NFC para acceder a los diferentes tipos de control de acceso RFID; debido a que comparten en el mismo rango de frecuencia de operación correspondiente a 13,56 MHz.

REFERENCIAS

Á. H., J. G., J. M., & M. B. (2015). Teaching Control Engineering Concepts using Open Source tools on a Raspberry Pi Board. *IFAC*, 99-100.

A. J., A. A., M. Z., & A. S. (2010). Analysis of different techniques to define metadata estructure in NFC/RFID cards to reduce access latency, optimize capacity, and guarantee integrity. *Department Information and Communicatios Engineering, University of Murcia Science Faculty*, 193-195.

Andrearrs. (15 de mayo de 2014). *Hipertextual*. Obtenido de https://hipertextual.com/archivo/2014/05/que-es-api/

B. M., & J. R. (2015). Chapter 9 Serial Communications. En *Industrial Process Automation Systems* (págs. 307, 316, 319-322, 326-329). Kidlington, Oxford : Butterworth-Heinemann.

Benchoff, B. (28 de Febrero de 2016). *Hackaday*. Obtenido de http://hackaday.com/2016/02/28/introducing-the-raspberry-pi-3/

Celma Guiménez, M., Casamayor Ródenas, J. C., & Mota Herranz, L. (2003). Sistemas de gestión de Bases de Datos. En *Bases de datos Relacionales* (págs. 20,25,109,124-126,243). Madrid: PEARSON EDUCACIÓN S.A.

Foundation, P. S. (12 de Febrero de 2001-2017). *Phyton Documentación*. Obtenido de https://docs.python.org/3/license.html

Gonzales, D. R. (2006). Serial peripheral interfacing techniques. *Motorola Inc.*, 5,6,8.

González, A. (2011). Gestión de bases de datos. Bogotá: Ra-ma.

Google Maps. (Diciembre de 2016). *Google Maps*. Obtenido de https://www.google.com.ec/maps/place/Universidad+Polit%C3%A9cnica+Salesiana+(Sur)/@-0.2809629,-78.5506613,16z/data=!4m8!1m2!2m1!1suniversidad+politecnica+salesiana+campus+sur!3m4!1s0x0:0x71cbab6b6dcb5b6a!8m2!3d-0.2819741!4d-78.5496283

Kugelstadt, T. (2011). Extendig the SPI bus for long-distance communication. *Texas Instruments Incorporated*, 16.

Lockhart, T. (1996). Guía del Programador de PostgreSQL. Berkeley, California, USA.

López, J. G., Fernández, E. V., & García, A. A. (2010). Diseño y Creación de Portales Web. Madrid: Starbook.

Mandeep Kaur, M. S. (2011). RFID Technology Principles, Adavantages,Limitation and Its Applications. *International Journal of Computer and Electrical Egineering*, 151-154.

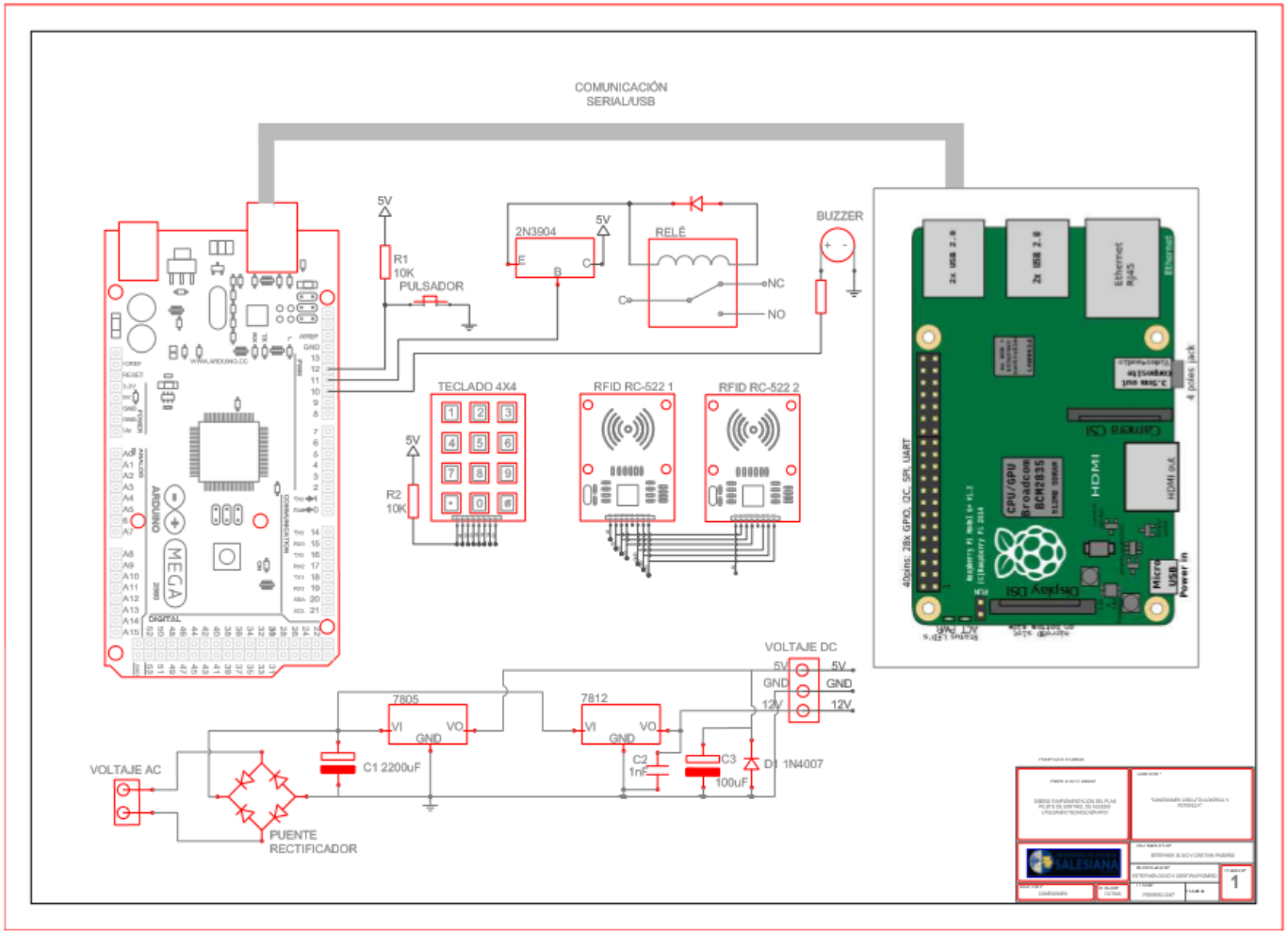
R. F. (2012). *RaspberryShop*. Obtenido de https://www.raspberryshop.es/

ANEXOS

Anexo 1. Atributos de configuración de páginas Web.

ETIQUETA	SE VERÁ ASÍ
<B>Texto en Negrita	**Texto en Negrita**
<I>Itálica</i>	*Itálica*
<B><I>Negrita e Itálica</i>	***Negrita e Itálica***
<U>Subrayado</u>	Subrayado Solo Explorer
<EM>Enfatizado	*Enfatizado*
<STRONG>Fuerte	**Fuerte**
<CODE>Code Texto</code>	Code Texto
<CITE> Citation Text</cite>	*Citation Text*
<KBD>Keyboard Text</kbd>	Keyboard Text
<SAMP>Sample Text</samp>	Sample Text
<TT>Teletype Text</tt>	Teletype Text
<VAR>Variable Element Text</var>	*Variable Element Text*
<BIG>Texto grande</big>	Texto grande
<SMALL>Texto pequeño</small>	Texto pequeño
_{Subindice}	Subíndice Solo Explorer
^{Superíndice}	Superíndice Solo Explorer
<BLINK> Texto intermitente</blink>	Texto intermitente Solo Netscape
<STRIKE>Texto tachado</STRIKE>	~~Texto tachado~~

Anexo 2. Circuito de control, conexión entre Arduino Mega dispositivos esclavos.



Anexo 3. Código de control de Arduino Mega.

[code]

```
#include <SPI.h>                  // Incluye librería de la comuniación SPI al sketch
#include <MFRC522.h>                 // Incluye librería para ingresar, escribir y leer registros de las lectoras RFID-RC522
#include <Keypad.h>                // Incluye librería para uso de teclados matriciales
#include <EEPROM.h>                 // Incluye librería para  ingreso y lectura de registros en la memoria EEPROM del Uc atmega 2560
#define RST_PIN         9          // Configurable, see typical pin layout above
#define SS_1_PIN        42          // Configurable, take a unused pin, only HIGH/LOW required, must be diffrent to SS 2
#define SS_2_PIN        49          // Configurable, take a unused pin, only HIGH/LOW required, must be diffrent to SS 1
#define NR_OF_READERS   2

byte ssPins[] = {SS_1_PIN, SS_2_PIN}; // Almacena en un vector los valores logicos de los pines de selección de las lectoras

MFRC522 mfrc522[NR_OF_READERS];       // Creaa una instancia para las lectoras RC522.

const int buttonPin = 12;            // Entrada del pulsador de emergencia
const int ledPin =  11;             // Salida de activación para el relé
int buzzer=10;                      // Salida del buzzer
int buttonState = 0;                // variable para identificar el estado de pulsación del botón de emergencia
boolean isOpen=false;                // Bandera para detectar el estado de codigo de bloqueo del teclado
boolean isValid=false;              // Bandera para determinar la validación de la contraseña ingresada por el teclado
char entryCode[4]={'1','2','3','4'};  // Codigo necesario para sobreescribir la EEPROM
char inputB[4]={'*','*','*','*'};
int j;
long t=-1;

//Creamos varibles tipo byte para determinar el numero de columnas y filas para el teclado matricial
const byte ROWS = 4;
const byte COLS = 3;
//valores ASSCI segun la combinación de columnas y filas
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
byte rowPins[ROWS] = {22, 24, 26, 28}; //Pines de conexión filas del teclado matricial
byte colPins[COLS] = {30, 32, 34};     //Pines de conexión columnas teclado matricial
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup() {

  Serial.begin(9600); // Inicializar comunicación serial
  while (!Serial);    // Lazo para saber si el puerto serial esta abierto y no ralizar ninguna acción si no se recibe un dato

  SPI.begin();        // Iniciar bus SPI
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
//Lazo para leer el codigo pre-grabado en la memoria eeprom
  for (j=0;j<4;j++)
 {
    entryCode[j]=char(EEPROM.read(j));
 }
//Lazo para leer el numero de tarjetas RC522 conectadas al bus SPI
  for (uint8_t reader = 0; reader < NR_OF_READERS; reader++) {
    mfrc522[reader].PCD_Init(ssPins[reader], RST_PIN);

  }
}
byte ActualUID[4]; //Almacena el código del Tag leído
byte Usuario1[4]= {0x53, 0xCB, 0xCB, 0xDE} ; //código del Administrador 1
```

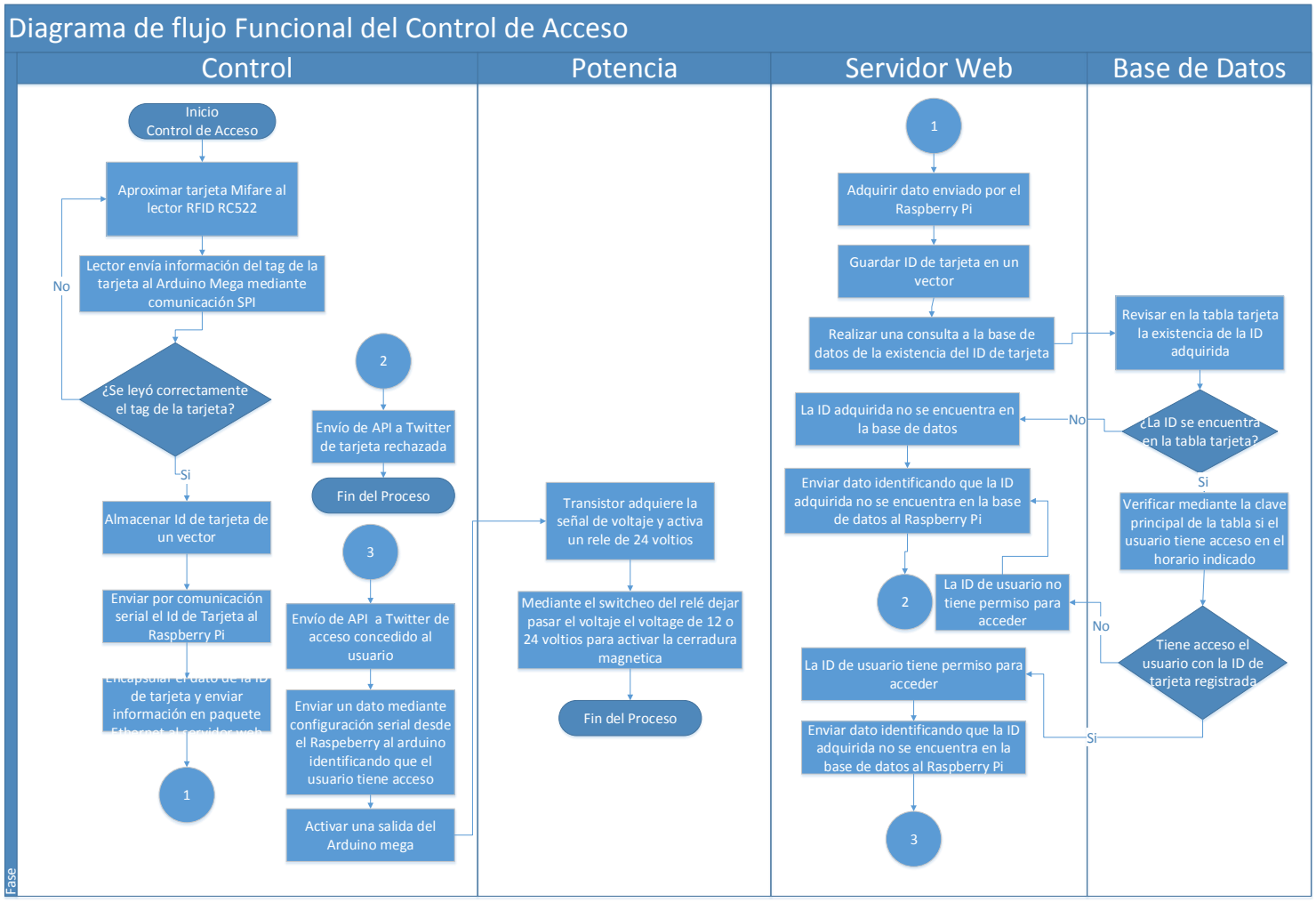
```
byte Usuario2[4]= {0x57, 0xCB, 0x84, 0xF4} ; //código del Administrador 2
/**
 * Lazo principal.
 */
void loop() {
  buttonState = digitalRead(buttonPin);
  //Lazo para determinar que lectora se encuentra activa
  for (uint8_t reader = 0; reader < NR_OF_READERS; reader++) {
    //Condicion si una de la lectoras se encuentran activas se determina cual y se envia por comunicación
serial el ID
    if (mfrc522[reader].PICC_IsNewCardPresent() && mfrc522[reader].PICC_ReadCardSerial()) {
      Serial.print(reader);
      dump_byte_array(mfrc522[reader].uid.uidByte, mfrc522[reader].uid.size);
      MFRC522::PICC_Type piccType = mfrc522[reader].PICC_GetType(mfrc522[reader].uid.sak);
      mfrc522[reader].PICC_HaltA();
      // Se detiene la encryptación del PCD(dispositivo de acoplamiento) es decir la lectora
      mfrc522[reader].PCD_StopCrypto1();
      //Condición que determina si es la lectora que se encuentra en el interior, si es verdadero se activa la
salida del relé
      if ( reader == 0)
          {
              tone(buzzer,500);
              delay(100);
              noTone(buzzer);
              tone(buzzer,600);
              delay(100);
              noTone(buzzer);
              tone(buzzer,800);
              delay(100);
              noTone(buzzer);
              digitalWrite(ledPin, HIGH);
              delay(7000);
              digitalWrite(ledPin, LOW);


          }
//Condición que determina si la ID recibida es igual a la del administrador 1, si es verdadero se activa la
salida del relé
            if(reader == 1 && compareArray(ActualUID,Usuario1))
    {
      tone(buzzer,500);
    delay(100);
    noTone(buzzer);
    tone(buzzer,600);
    delay(100);
    noTone(buzzer);
    tone(buzzer,800);
    delay(100);
    noTone(buzzer);
    digitalWrite(ledPin, HIGH);
    delay(7000);
    digitalWrite(ledPin, LOW);
      }
//Condición que determina si la ID recibida es igual a la del administrador 2, si es verdadero se activa la
salida del relé
      if(reader == 1 && compareArray(ActualUID,Usuario2))
    {
      tone(buzzer,500);
    delay(100);
    noTone(buzzer);
    tone(buzzer,600);
    delay(100);
    noTone(buzzer);
    tone(buzzer,800);
    delay(100);
    noTone(buzzer);
    digitalWrite(ledPin, HIGH);
    delay(7000);
    digitalWrite(ledPin, LOW);
      }
    }
```

```
  }
//Condición que permite el acceso mediente la pulsación del botón de emergencia
if (buttonState == HIGH) {
    // turn LED on:
    tone(buzzer,500);
    delay(100);
    noTone(buzzer);
    tone(buzzer,600);
    delay(100);
    noTone(buzzer);
    tone(buzzer,800);
    delay(100);
    noTone(buzzer);
    digitalWrite(ledPin, HIGH);
    delay(7000);
    digitalWrite(ledPin, LOW);
    Serial.print('c');
  }
//Condición que determina la validación de la ID de tarjeta por medio del Raspberry si es verdadero recibe
un caracter ASSCI "a" Y activa la salida del relé
if(Serial.read()=='a')
  {
    tone(buzzer,500);
    delay(100);
    noTone(buzzer);
    tone(buzzer,600);
    delay(100);
    noTone(buzzer);
    tone(buzzer,800);
    delay(100);
    noTone(buzzer);
    digitalWrite(ledPin, HIGH);
    delay(7000);
    digitalWrite(ledPin, LOW);

    }
//Manejo del teclado
  if (t>0) t--;
 if (t<=0)
 {
    isOpen=false;
 }

 char key = keypad.getKey(); // guarda la tecla presionada
  //Serial.print(key);
 if (key){
           tone(buzzer,350);     // tono de pulsacion
           delay(200);
           noTone(buzzer);

 if (key=='*') // Si '*' verifica las 4 últimas teclas presionadas
 {


    if(inputB[0]==entryCode[0] &&
    inputB[1]==entryCode[1] &&
    inputB[2]==entryCode[2] &&
    inputB[3]==entryCode[3] )
    {

      isOpen=true; // Si el codigo es correcto se pone la bandera "isOpen" en true

      inputB[0]='*'; // resetea buffer de entrada
      inputB[1]='*';
      inputB[2]='*';
      inputB[3]='*';

      t=1000000; // resetea contador
     }
     else
```

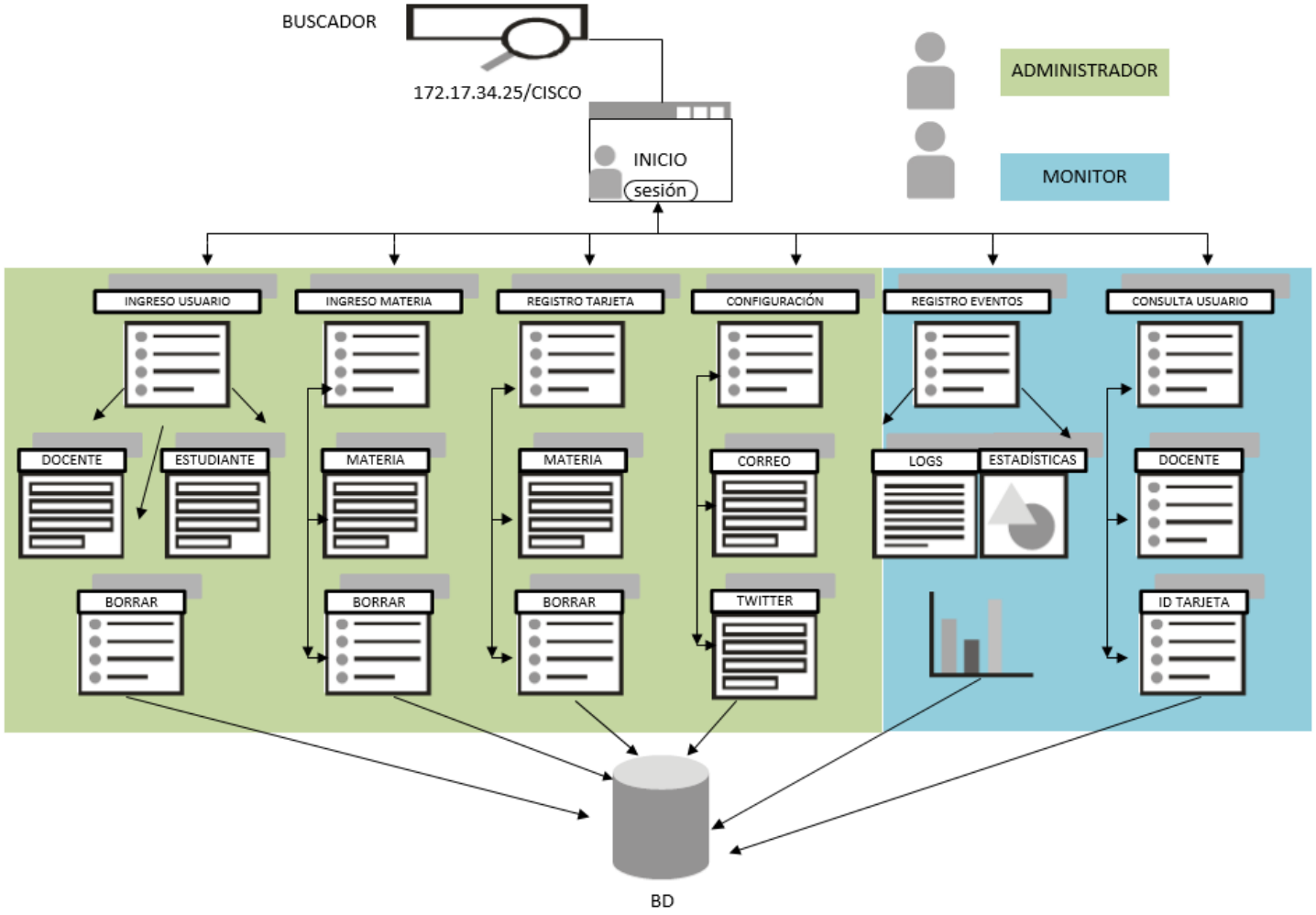
```
    {
        isOpen=false; // Si el codigo es incorrecto se pone la bandera "isOpen" en false

    }

     if(isOpen)
    {
      tone(buzzer,500);
        delay(100);
        noTone(buzzer);
        tone(buzzer,600);
        delay(100);
        noTone(buzzer);
        tone(buzzer,800);
        delay(100);
        noTone(buzzer);
        digitalWrite(ledPin, HIGH);
        delay(7000);
        digitalWrite(ledPin, LOW);
    }
    else
    {
      tone(buzzer,70,500); // para generar
         delay(200); // tono de error
         noTone(buzzer);
    }
 }
 else
 { // se presionó'*'

    if (isOpen && key=='#')
    {
 // verifica los 4 últimos digitos almacenados en el buffer
      isValid=true;
      for (j=0;j<4;j++) if(inputB[j]=='*' || inputB[j]=='#') isValid=false;

      if (isValid)
      {
        for (j=0;j<4;j++)
        {
          entryCode[j]=inputB[j]; // copia el contenido del buffer de entrada en el buffer del codigo de
seguridad
          EEPROM.write(j,entryCode[j]); // escribe los valores obtenidos en la memoria EEPROM
        }

         for (j=0;j<4;j++)
        {

          delay(100);
        }
    }
 }
 else
 {
    for (j=0;j<3;j++)
    {
      inputB[j]=inputB[j+1];
    }
    inputB[3]=key;
 }
 }
 }

}

//Función para imprimir la ID de tarjeta por comuniación serial
void dump_byte_array(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i], HEX);
    ActualUID[i]=buffer[i];
```

```
  }
}
//Función para comparar array de la ID de tarjeta con la almacenada en la variable de cada administrador
 boolean compareArray(byte array1[],byte array2[])
{
  if(array1[0] != array2[0])return(false);
  if(array1[1] != array2[1])return(false);
  if(array1[2] != array2[2])return(false);
  if(array1[3] != array2[3])return(false);
  return(true);
}

[/code]
```

Anexo 4. Diagrama de flujo de información del Sistema de Control de Acceso



Anexo 5. Lógica de Aplicación Web



Anexo 6. Modos de consulta de la interfaz web.

COMANDOS TIPO DDL	
COMANDO	**DEFINICION**
CREATE	Crea nuevas bases de datos, tablas, campos e índices.
DROP	Elimina bases de datos, tablas, campos e índices.
ALTER	Modifica estructura, diseño de una base de datos

COMANDOS TIPO DML	
COMANDO	**DEFINICION**
SELECT	Consulta registros dentro de la base de datos
INSERT	Inserta o agrega datos en una tabla o entidad
DELETE	Elimina registros de una entidad bajo ciertos criterios.
UPDATE	Modifica o actualiza los datos existentes dentro de las entidades.

Anexo 7. Programación de mensajes de alerta en el ordenador Raspberry Pi 3

```
#!/usr/bin/python

import psycopg2
import serial
import time
import sys
import tweepy
import datetime
import smtplib
import random
from goto import with_goto

a=random.randint(0.1000)
@with_goto
def f():
label .get_input
try:
conn = psycopg2.connect(database="tesisdb", user="tesis", password="tesiscye",
host="192.168.100.37", port="5432")
except psycopg2.Error, e:
print "no se puede conectar"
goto .get_input
else:
print ("conexion a la base satisfactoriamente")
cur200=conn.cursor()
cur201=conn.cursor()
cur202=conn.cursor()
cur203=conn.cursor()
cur200.execute("SELECT con_key from twitter")
cur201.execute("SELECT con_secret from twitter")
cur202.execute("SELECT acc_key from twitter")
cur203.execute("SELECT con_secret from twitter")
conkey = cur200.fetchone()
consecret = cur201.fetchone()
acckey = cur202.fetchone()
accsecret = cur203.fetchone()
str200 = ''.join(conkey)
str201 = ''.join(consecret)
str202 = ''.join(acckey)
str203 = ''.join(accsecret)
CONSUMER_KEY = str200
CONSUMER_SECRET = str201
ACCESS_KEY = str202
ACCESS_SECRET = str203
auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_KEY, ACCESS_SECRET)
gettime = datetime.datetime.now().strftime("%Y-%m-%d a las %H:%M:%S")
cur = conn.cursor()
arduino=serial.Serial('/dev/ttyACM0',baudrate=9600, timeout = 3.0)
txt=''
cur.execute("SELECT id_tarjeta from tarjeta")
rows = cur.fetchall()
conn.close()
while True:

time.sleep(0.1)
while arduino.inWaiting() > 0:
txt += arduino.readline(1)
# print txt.rstrip('\n')
```

```
print txt


for row in rows:
if txt != '':

if txt == row[0]:
try:

conn = psycopg2.connect(database="tesisdb", user="tesis", password="tesiscye",
host="192.168.100.37", port="5432")
except psycopg2.Error, e:
print "no se puede conectar"
goto .get_input
else:
cur1 = conn.cursor()
cursor = conn.cursor()
cur7 = conn.cursor()
cur8 = conn.cursor()
cur9 = conn.cursor()
cur10 = conn.cursor()
cur15 = conn.cursor()
cur16 = conn.cursor()
cur17 = conn.cursor()
cur18 = conn.cursor()
cur19 = conn.cursor()
cursor.execute("SELECT EXTRACT(dow from dia1)::int as numdia from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur7.execute("SELECT EXTRACT(dow from dia2)::int as numdia1 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur8.execute("SELECT EXTRACT(dow from dia3)::int as numdia2 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur9.execute("SELECT EXTRACT(dow from dia4)::int as numdia3 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur10.execute("SELECT EXTRACT(dow from dia5)::int as numdia4 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur15.execute("SELECT EXTRACT(dow from dia6)::int as numdia5 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur16.execute("SELECT EXTRACT(dow from dia7)::int as numdia6 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur17.execute("SELECT EXTRACT(dow from dia8)::int as numdia7 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur18.execute("SELECT EXTRACT(dow from dia9)::int as numdia8 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur19.execute("SELECT EXTRACT(dow from dia10)::int as numdia9 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
numdiaa = cursor.fetchone()
numdiaa1 = cur7.fetchone()
numdiaa2 = cur8.fetchone()
numdiaa3 = cur9.fetchone()
numdiaa4 = cur10.fetchone()
```

```
numdiaa5 = cur15.fetchone()
numdiaa6 = cur16.fetchone()
numdiaa7 = cur17.fetchone()
numdiaa8 = cur18.fetchone()
numdiaa9 = cur19.fetchone()
cur1.execute("SELECT EXTRACT(dow from current_timestamp)::int as numdiac")
numdiac = cur1.fetchone()
conn.close()
print numdiaa
print numdiac
if numdiaa == numdiac or numdiaa1 == numdiac or numdiaa2 == numdiac or numdiaa3 ==
numdiac or numdiaa4 == numdiac or numdiaa5 == numdiac or numdiaa6 == numdiac or
numdiaa7 == numdiac or numdiaa8 == numdiac or numdiaa9 == numdiac:
try:
conn = psycopg2.connect(database="tesisdb", user="tesis", password="tesiscye",
host="192.168.100.37", port="5432")
except psycopg.Error, e:
print "no se puede conectar"
goto .get_input
else:
cur2 = conn.cursor()
cur11 = conn.cursor()
cur12 = conn.cursor()
cur13 = conn.cursor()
cur14 = conn.cursor()
cur20 = conn.cursor()
cur21 = conn.cursor()
cur22 = conn.cursor()
cur23 = conn.cursor()
cur24 = conn.cursor()
cur3 = conn.cursor()
cur2.execute("SELECT EXTRACT(hour from dia1)::int as horadia from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur11.execute("SELECT EXTRACT(hour from dia2)::int as horadia1 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur12.execute("SELECT EXTRACT(hour from dia3)::int as horadia2 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur13.execute("SELECT EXTRACT(hour from dia4)::int as horadia3 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur14.execute("SELECT EXTRACT(hour from dia5)::int as horadia4 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur20.execute("SELECT EXTRACT(hour from dia6)::int as horadia5 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur21.execute("SELECT EXTRACT(hour from dia7)::int as horadia6 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur22.execute("SELECT EXTRACT(hour from dia8)::int as horadia7 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur23.execute("SELECT EXTRACT(hour from dia9)::int as horadia8 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur24.execute("SELECT EXTRACT(hour from dia10)::int as horadia9 from materia, registro_tarjeta
where registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
horadia = cur2.fetchone()
```

```
horadia1 = cur11.fetchone()
horadia2 = cur12.fetchone()
horadia3 = cur13.fetchone()
horadia4 = cur14.fetchone()
horadia5 = cur20.fetchone()
horadia6 = cur21.fetchone()
horadia7 = cur22.fetchone()
horadia8 = cur23.fetchone()
horadia9 = cur24.fetchone()
cur3.execute("SELECT EXTRACT(hour from current_timestamp)::int as horadiac")
horadiac = cur3.fetchone()
print horadia
print horadiac
conn.close()
if horadia == horadiac or horadia1 == horadiac or horadia2 == horadiac or horadia3 == horadiac
or horadia4 == horadiac or horadia5 == horadiac or horadia6 == horadiac or horadia7 ==
horadiac or horadia8 == horadiac or horadia9 == horadiac:
try:
conn = psycopg2.connect(database="tesisdb", user="tesis", password="tesiscye",
host="192.168.100.37", port="5432")
except psycopg.Error, e:
print "no se puede conectar"
goto .get_input
else:
cur4 = conn.cursor()
cur26 = conn.cursor()
cur27 = conn.cursor()
cur108 = conn.cursor()
cur109 = conn.cursor()
cur110 = conn.cursor()
cur111 = conn.cursor()
cur4.execute("INSERT into registro_logs(id_tarjeta, nombre_log, apellido_log, fecha, comentario)
select registro_tarjeta.id_tarjeta, nombre_doc, apellido_doc, current_timestamp, 'si tiene acceso'
from docentes, tarjeta, registro_tarjeta, materia where
registro_tarjeta.codigo_materia=materia.codigo_materia and
docentes.cedula=registro_tarjeta.cedula and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
conn.commit()
print 'registro guardado exitosamente'
print 'tiene acceso'
cur26.execute("SELECT docentes.nombre_doc from docentes, materia, registro_tarjeta where
docentes.cedula=registro_tarjeta.cedula and
registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur27.execute("SELECT docentes.apellido_doc from docentes, materia, registro_tarjeta where
docentes.cedula=registro_tarjeta.cedula and
registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur108.execute("SELECT mailadmin from configuracion")
cur109.execute("SELECT passadmin from configuracion")
cur110.execute("SELECT mailto from configuracion")
cur111.execute("SELECT servidor_men from configuracion")
nombre = cur26.fetchone()
apellido = cur27.fetchone()
mail3 = cur108.fetchone()
pass3 = cur109.fetchone()
mailto3 = cur110.fetchone()
server3 = cur111.fetchone()
str1 = ''.join(nombre)
str2 = ''.join(apellido)
str108 = ''.join(mail3)
str109 = ''.join(pass3)
```

```
str110 = ''.join(mailto3)
str11 = ''.join(server3)
y = tweepy.API(auth)
tweet = "%s %s tiene acceso, su hora de acceso fue el %s" % (str1, str2, gettime)
y.update_status(status = tweet)
arduino.write('a')
fromaddr = str108
toaddr = str110
asunto = 'Informe de Ingreso Lab CISCO'
Cabecera = 'To:' + toaddrs + '\n'
Cabecera = 'From: ' + fromaddrs + '\n'
Cabecera = 'Subject:' + asunto + '\n' +
CuerpoMensaje = "%s %s tiene acceso, su hora de acceso fue el %s" % (str1, str2, gettime)
username = str109
password = str111
try:
enviotesis = smtplib.SMTP(str111)
enviotesis.starttls()
enviotesis.login(username,password)
enviotesis.sendmail(fromaddr, toaddr, Cabecera, CuerpoMensaje)
enviotesis.quit()
except smtplib.SMTPException:
print "fallo envio de correo"
goto .get_input
else:
y=''
txt=''
conn.close()
break
else:
try:
conn = psycopg2.connect(database="tesisdb", user="tesis", password="tesiscye",
host="192.168.100.37", port="5432")
except psycopg.Error, e:
print "no se puede conectar"
goto .get_input
else:
cur5 = conn.cursor()
cur28 = conn.cursor()
cur29 = conn.cursor()
cur5.execute("INSERT into registro_logs(id_tarjeta, nombre_log, apellido_log, fecha, comentario)
select registro_tarjeta.id_tarjeta, nombre_doc, apellido_doc, current_timestamp, 'no tiene acceso'
from docentes, tarjeta, registro_tarjeta, materia where
registro_tarjeta.codigo_materia=materia.codigo_materia and
docentes.cedula=registro_tarjeta.cedula and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
conn.commit()
print 'registro guardado exitosamente'
print 'no tiene acceso'
cur28.execute("SELECT docentes.nombre_doc from docentes, materia, registro_tarjeta where
docentes.cedula=registro_tarjeta.cedula and
registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur29.execute("SELECT docentes.apellido_doc from docentes, materia, registro_tarjeta where
docentes.cedula=registro_tarjeta.cedula and
registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
nombre1 = cur28.fetchone()
apellido1 = cur29.fetchone()
gettime1 = datetime.datetime.now().strftime("%Y-%m-%d a las %H:%M:%S")
str3 = ''.join(nombre1)
str4 = ''.join(apellido1)
```

```
x = tweepy.API(auth)
tweet1 = "%s %s no tiene acceso en esta hora, intento acceder el %s" % (str3, str4, gettime1)
x.update_status(status = tweet1)
conn.close()
break

else:
try:
conn = psycopg2.connect(database="tesisdb", user="tesis", password="tesiscye",
host="192.168.100.37", port="5432")
except psycopg.Error, e:
print "no se puede conectar"
goto .get_input
else:
cur6 = conn.cursor()
cur30 = conn.cursor()
cur31 = conn.cursor()
cur6.execute("INSERT into registro_logs(id_tarjeta, nombre_log, apellido_log, fecha, comentario)
select registro_tarjeta.id_tarjeta, nombre_doc, apellido_doc, current_timestamp, 'no tiene acceso'
from docentes, tarjeta, registro_tarjeta, materia where
registro_tarjeta.codigo_materia=materia.codigo_materia and
docentes.cedula=registro_tarjeta.cedula and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
conn.commit()
print 'registro guardado exitosamente'
cur30.execute("SELECT docentes.nombre_doc from docentes, materia, registro_tarjeta where
docentes.cedula=registro_tarjeta.cedula and
registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
cur31.execute("SELECT docentes.apellido_doc from docentes, materia, registro_tarjeta where
docentes.cedula=registro_tarjeta.cedula and
registro_tarjeta.codigo_materia=materia.codigo_materia and
registro_tarjeta.id_tarjeta=%(registro_tarjeta.id_tarjeta)s", {'registro_tarjeta.id_tarjeta': txt})
nombre2 = cur30.fetchone()
apellido2 = cur31.fetchone()
gettime2 = datetime.datetime.now().strftime("%Y-%m-%d a las %H:%M:%S")
str5 = ''.join(nombre2)
str6 = ''.join(apellido2)
z = tweepy.API(auth)
tweet2 = "%s %s no tiene acceso en este dia, intento acceder el %s" % (str5, str6, gettime2)
z.update_status(status = tweet2)
conn.close()
break

if txt == 'c':
try:
conn = psycopg2.connect(database="tesisdb", user="tesis", password="tesiscye",
host="192.168.100.37", port="5432")
except psycopg.Error, e:
print "no se puede conectar"
goto .get_input
else:
cur25 = conn.cursor()
cur25.execute("INSERT into registro_logs(id_tarjeta, nombre_log, apellido_log, fecha, comentario)
values ('ninguna', 'usuario', 'interno', current_timestamp, 'Se activó pulsador de emergencia')")
print 'registro guardado exitosamente'
conn.close()
for row1 in rows:
if txt != '':
if txt != row1[0]:
print 'no tiene acceso'
txt = ''
```

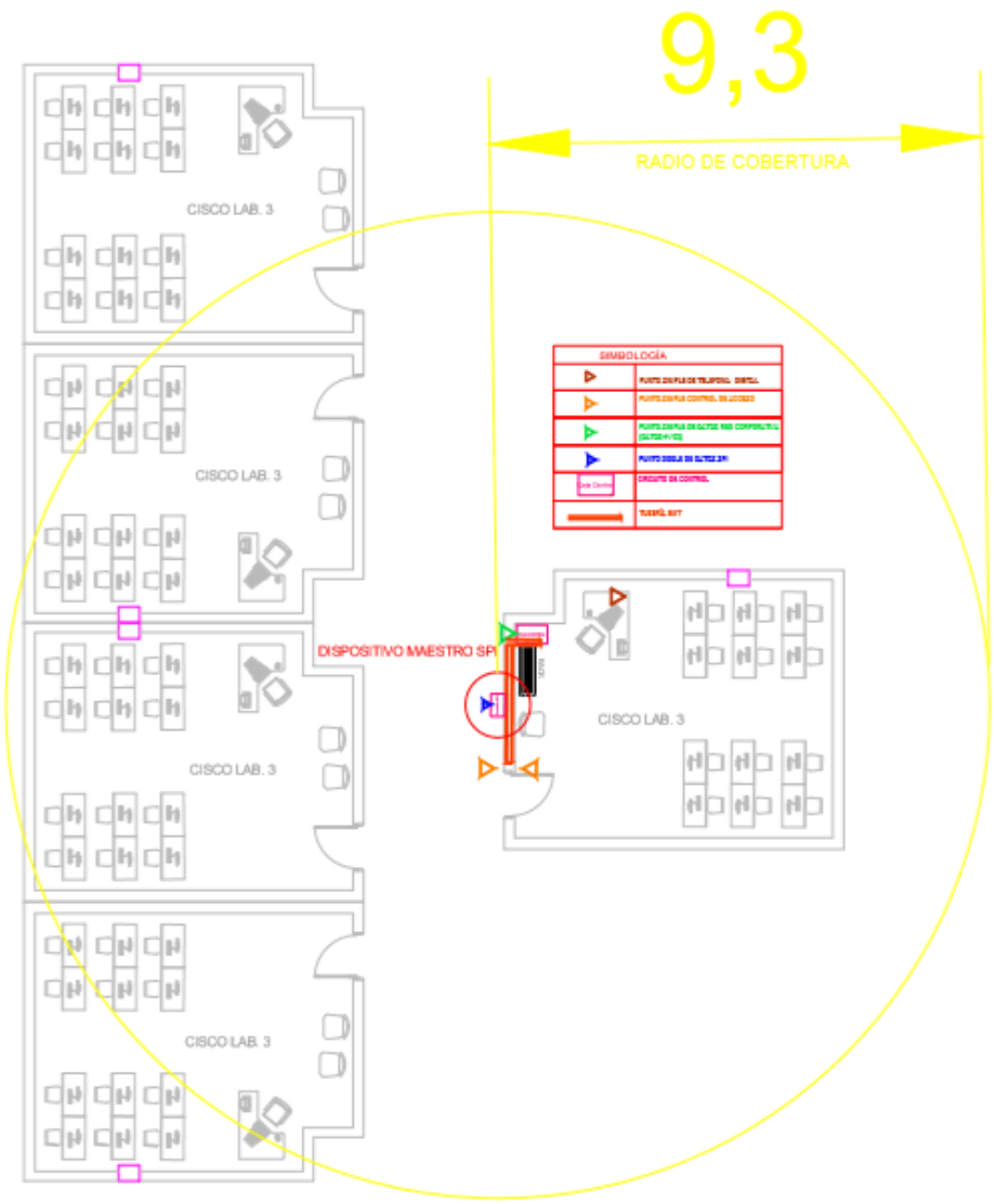
```
txt = ''

txt = ''
arduino.close()


f()
```

Anexo 8. Relación entre frecuencia SPI y la frecuencia de oscilador.

Table 21-5. Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

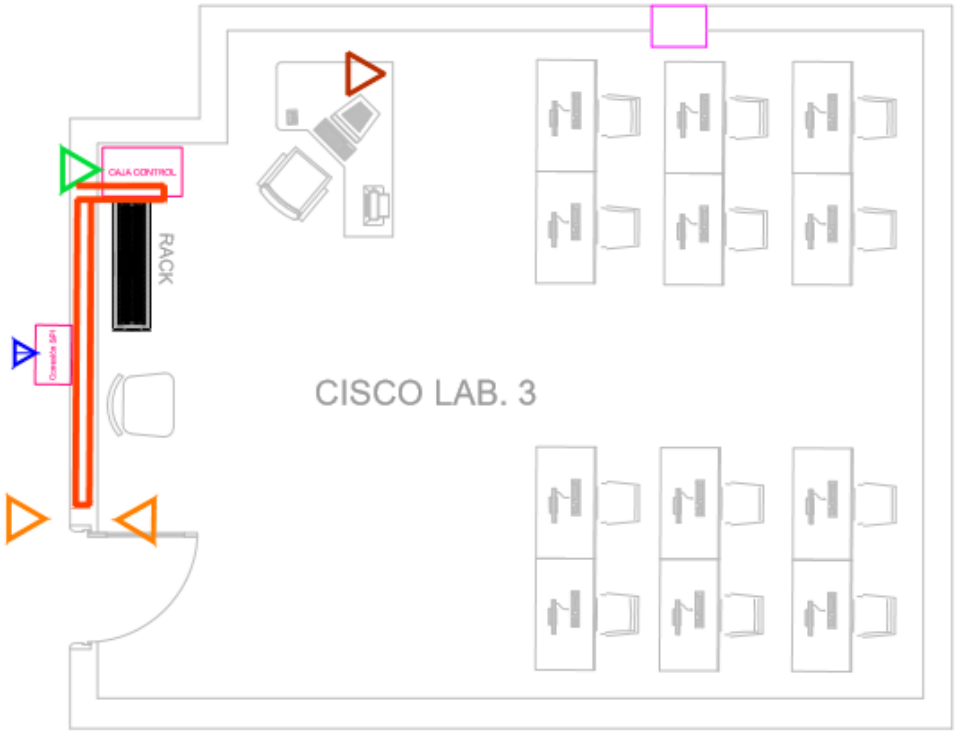
Anexo 9. Longitud máxima de cobertura del bus SPI.



Anexo 10. Presupuesto del proyecto

ADECUACION DEL SISTEMA DE CONTROL DE ACCESO DEL LABORATORIO 3 BLOQUE D CAMPUS SUR					
		PRESUPUESTO			
Rubro	DESCRIPCIÓN DEL RUBRO	Uni	CANTIDAD	P.U	Subtotal
	INSTALACIÓN DE PUNTOS DE CONTROL DE ACCESO				**$ 16.786,90**
TECNOLOGÍA	PLACA ARDUINO MEGA 2560	u	1,00	$57,00	$ 57,00
TECNOLOGÍA	RASPBERRY PI 3 MODEL B	u	1,00	$75,00	$ 75,00
ARQUITECTURA	CERRADURA ELECTROMAGNÉTICA ELOCK 12V/24V	u	1,00	$69,00	$ 69,00
ARQUITECTURA	BRAZO HIDRAULICO VIRO	u	1,00	$65,00	$ 65,00
ARQUITECTURA	CABLEADO ESTRUCTURADO CAT 6	rollo	1,00	$280,28	$ 280,28
ARQUITECTURA	TUBERÍA EMT 1 pulg	m	6,00	$3,50	$ 21,00
TECNOLOGÍA	MÓDULO LECTOR DE PRÓXIMIDAD RFID RC522	u	2,00	$10,00	$ 20,00
ARQUITECTURA	TECLADO MATRICIAL 4*3	u	1,00	$5,49	$ 5,49
ARQUITECTURA	BOTÓN PARO DE EMERGENCIA	u	1,00	$5,00	$ 5,00
TECNOLOGÍA	UPS (LABORATORIO 3 DE CISCO)	u	1,00	$80,00	$ 80,00
TECNOLOGÍA	SERVIDOR (SOPORTE TÉCNICO CAMPUS SUR)	u	1,00	$16.000,00	$ 16.000,00
ARQUITECTURA	CABLE GEMELO	m	8,00	$0,45	$ 3,60
ARQUITECTURA	CAJA DE ALUMINIO 20*20	u	2,00	$20,00	$ 40,00
TECNOLOGÍA	PLACA DE FIBRA DE VIDRIO A4	u	1,00	$6,90	$ 6,90
TECNOLOGÍA	RESISTENCIAS	u	5,00	$0,02	$ 0,10
ARQUITECTURA	LÁMINA DE ACRÍLICO 30cm*30cm	u	1,00	$12,72	$ 12,72
TECNOLOGÍA	CAPACITORES	u	5,00	$0,15	$ 0,75
TECNOLOGÍA	DIODO 1N4007	u	1,00	$0,20	$ 0,20
TECNOLOGÍA	TRANSISTOR 2N3904	u	1,00	$0,20	$ 0,20
TECNOLOGÍA	TARJETAS MIFARE	u	3,00	$1,80	$ 5,40
TECNOLOGÍA	LLAVERO MIFARE	u	4,00	$2,49	$ 9,96
TECNOLOGÍA	FUENTE 12VDC	u	1,00	$6,00	$ 6,00
TECNOLOGÍA	BUZZER	u	1,00	$2,90	$ 2,90
TECNOLOGÍA	REGULADOR DE VOLTAJE 5VDC	u	1,00	$0,55	$ 0,55
TECNOLOGÍA	REGULADOR DE VOLTAJE 12VDC	u	1,00	$0,55	$ 0,55
TECNOLOGÍA	BORNERA	u	6,00	$0,40	$ 2,40
ARQUITECTURA	ESPADINES	u	3,00	$0,40	$ 1,20
ARQUITECTURA	CAJA EMT	u	3,00	$3,10	$ 9,30
ARQUITECTURA	CODO EMT 1 pulg	u	2,00	$3,20	$ 6,40

Anexo 11. Instalación del Sistema de Control de Acceso



SIMBOLOGÍA	
	PUNTO SIMPLE DE TELEFONIA DIGITAL
	PUNTO SIMPLE CONTROL DE ACCESO
	PUNTO SIMPLE DE DATOS RED CORPORATIVA (DATOS+VOZ)
	PUNTO DOBLE DE DATOS SPI
Caja Contro	CIRCUITO DE CONTROL
	TUBERÍA EMT

Anexo 12. Manual de configuración de alertas a Twitter

MANUAL DE CONFIGURACIÓN DE API KEY, CONSUMER TOKEN Y ACCESS KEY PARA LA AUTENTIFICACIÓN DE TWITTER

ELABORADOR POR:

OLIVO BARCIA ESTEFANÍA ESMERALDA

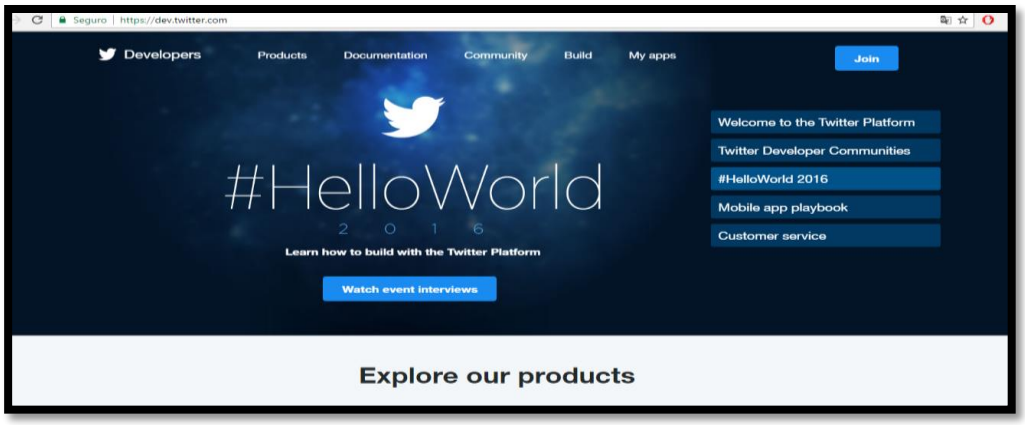
PAZMIÑO CORTEZ CRISTIAN ADRIAN

AÑO: 2017

Paso 1:

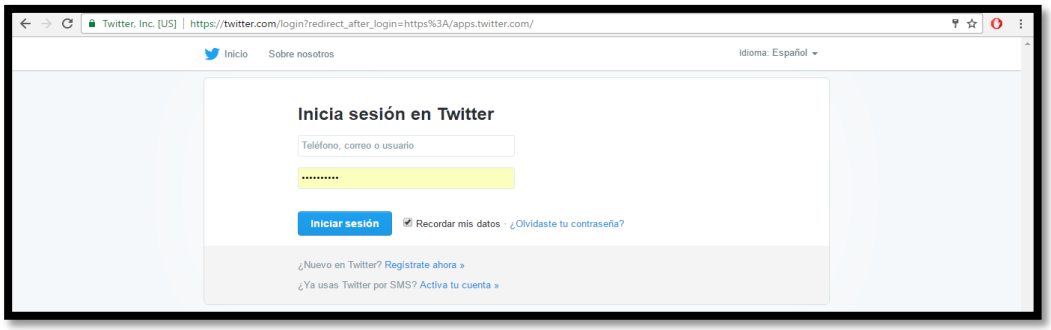
Utilizando el navegador de tu preferencia ingresar a la siguiente URL:

https://dev.twitter.com/ En la cual se podrá crear las credenciales para obtener los API Tokens.

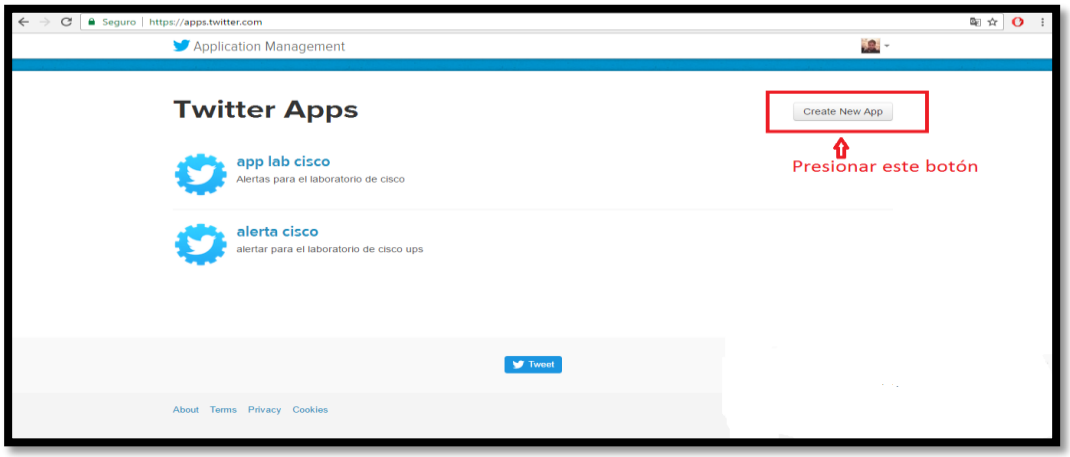


Paso 2:

Ingresar en la pestaña con el nombre "**My apps**" e iniciar sección con la cuenta personal que posee en twitter.

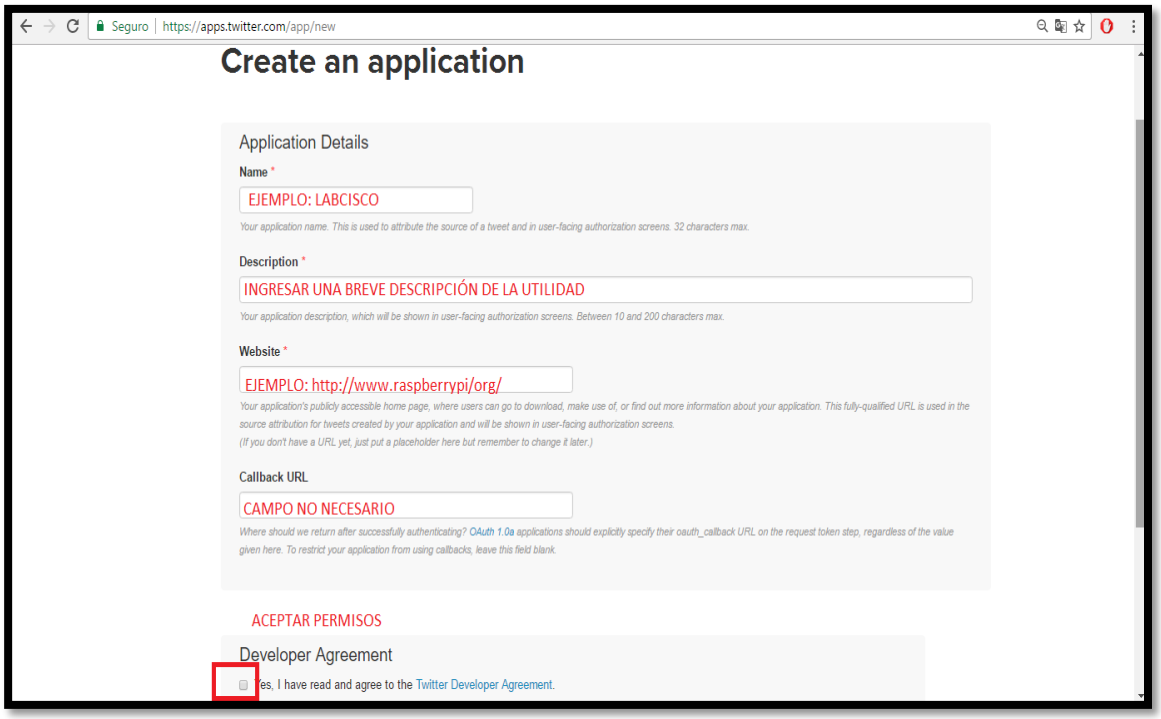


Una vez iniciada sesión aparecerá todas las aplicaciones que tenemos creadas en caso de no tener ninguna dar click en el botón **"Create New App"** como se muestra en la siguiente captura.

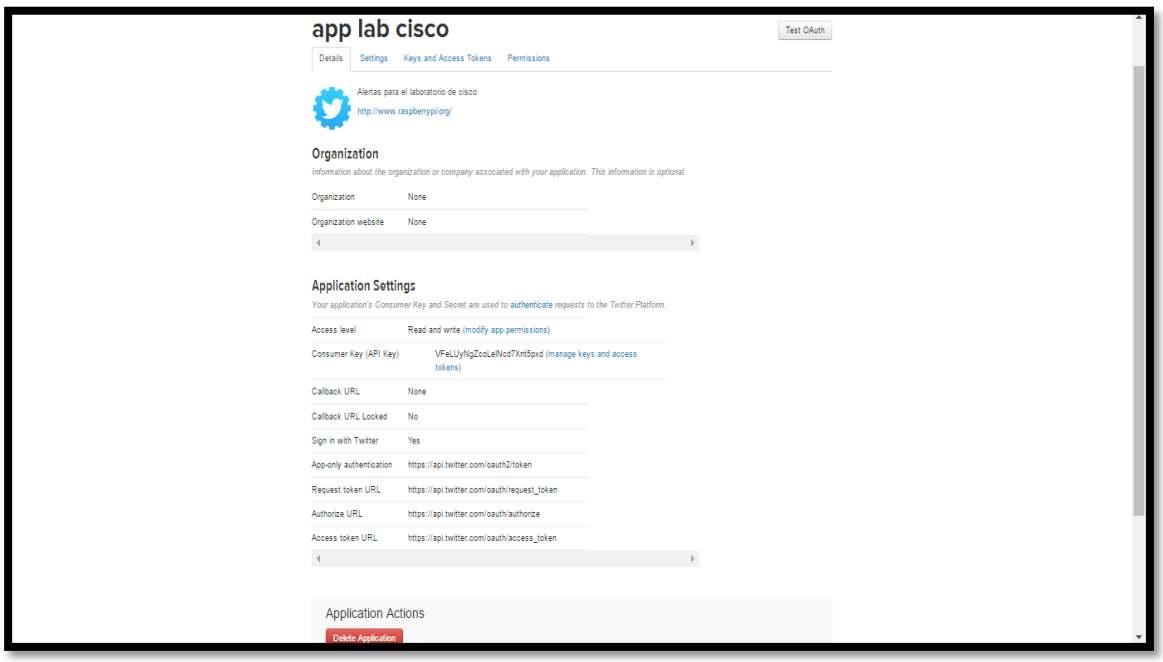


Paso 3:

Introducir los 3 primeros campos los cuales son obligatorios para la creación de la aplicación como se muestra en la siguiente ilustración:

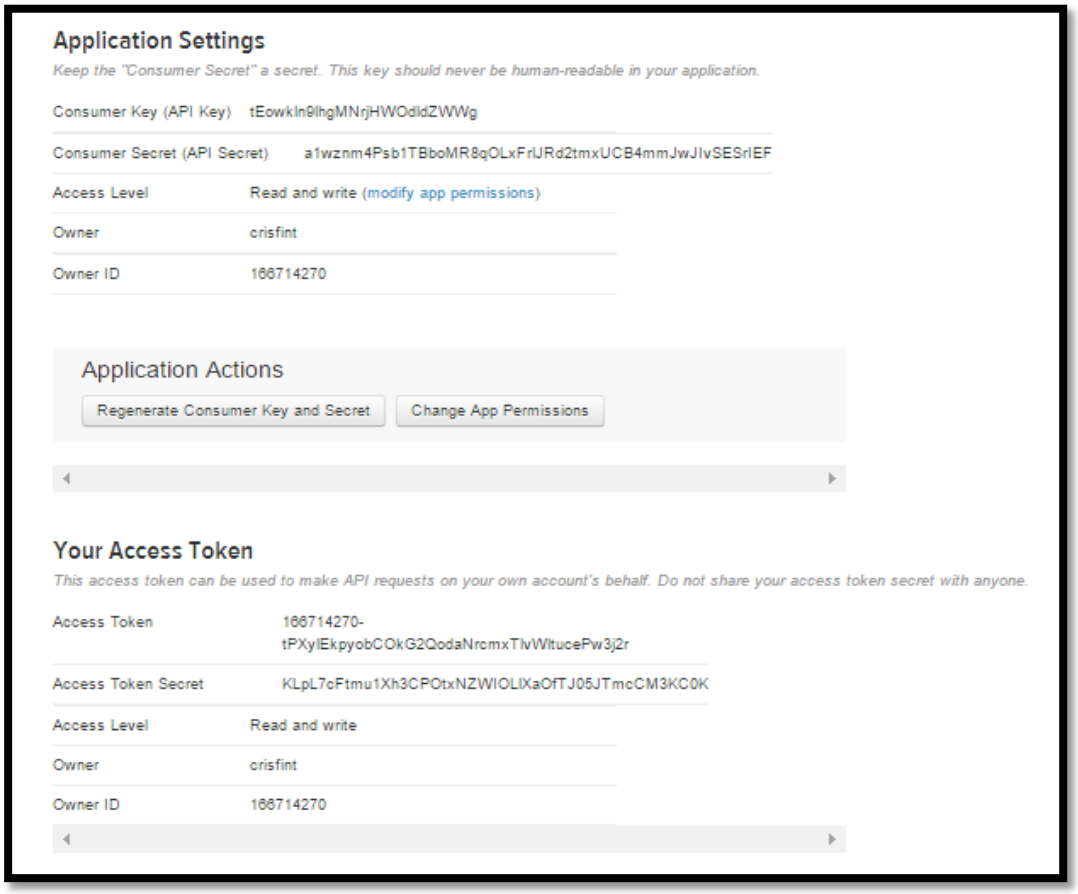


Una vez creada la aplicación nos aparecerá toda la información como se muestra a continuación



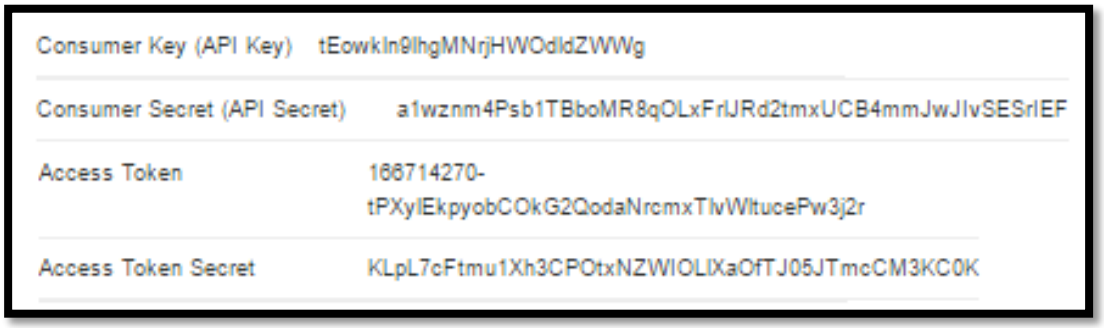
Paso 4:

Se debe dirigir en la pestaña que con el nombre **"Key and Access token"** en la cual encontraremos en la parte superior la información del Consumer Key y el Consumer Secret para la autentificación del API. En la parte inferior tenemos la información **"Your Access Token"** la primera vez que se crea el API se debe dar click en el botón "Create Token", inmediatamente se obtendrá la información del Acces Token y el Access Token Secret.



Paso 5:

Anotar las siguientes keys que se muestra a continuación:



Paso 6:

En la parte superior dar click en el botón de **“permissions”** y marcar **“Read and Write”**

Paso 7:

Ingresar a 172.17.34.25/CISCO autentificarse como administrador e ingresar a la pestaña de configuración de Twitter e ingresar los parámetros del token.