ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN PROTOTIPO PARA LA MONITORIZACIÓN INALÁMBRICA DE POTENCIA DE UN *DATA CENTER* UNIVERSITARIO

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

VERÓNICA MARCELA VELASTEGUÍ RUEDA

veronica.velasteguí@epn.edu.ec

DIRECTOR: ING. RICARDO XAVIER LLUGSI CAÑAR, MSc.

ricardo.llugsi@epn.edu.ec

Quito, diciembre 2019

AVAL

Certifico que el presente trabajo fue desarrollado por Verónica Marcela Velasteguí Rueda, bajo mi supervisión.

ING. RICARDO XAVIER LLUGSI CAÑAR, MSc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Verónica Marcela Velasteguí Rueda, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

VERÓNICA MARCELA VELASTEGUÍ RUEDA

DEDICATORIA

Dedico este estudio técnico a mi padre, mi amigo y mi consejero Dios que me ha dado la vida y gracias a él he llegado hasta el final.

AGRADECIMIENTO

Agradezco a Dios por guiar mi camino y a la Virgen María por ser guía y maestra en mi vida.

A mis padres, por su ayuda incondicional, brindándome todo el apoyo y la paciencia, durante todo mi trayecto educativo, hasta obtener mi título de Ingeniera.

A mi hermana, que con dulzura me supo guiar en la vida.

A la Escuela Politécnica Nacional, por abrirme sus puertas y permitir que su claustro docente desarrolle en mí, competencias profesionales con valores éticos y morales, para que el en campo laboral sea una digna representante de esta noble Institución.

Al Ing. Ricardo Llugsi MSc., quien, de manera profesional y carismática**,** supo guiarme de manera asertiva; y así, llegar a culminar este proyecto de titulación.

ÍNDICE DE CONTENIDO

AVAL I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO V
RESUMEN VII
ABSTRACT VIII
1. INTRODUCCIÓN 1
1.1 OBJETIVOS 2
1.2 ALCANCE 2
1.3 MARCO TEÓRICO 3
1.3.1 DATA CENTER Y SU FUNCIONAMIENTO 3
1.3.2 APP INVENTOR 5
1.3.3 SERVIDOR EN LA NUBE 7
1.3.4 RASPBERRY PI 3 B+ 9
1.3.5 RASPBIAN 12
1.3.6 PYTHON 12
1.3.7 ARDUINO UNO 13
1.3.8 LENGUAJE DE PROGRAMACIÓN PARA ARDUINO 15
1.3.9 MEDICIÓN DE LA CORRIENTE 15
1.3.10 MEDICIÓN DEL VOLTAJE 17
1.3.11 MEDICIÓN DE LA TEMPERATURA 18
1.3.12 SENSOR PARA CONTROLAR LA APERTURA Y CIERRE DEL PDU PRINCIPAL 19
1.3.13 ADMINISTRACIÓN DE BASES DE DATOS EN LA NUBE (BASE DE DATOS COMO SERVICIO) 20
1.3.14 PROTEUS 22
2. METODOLOGÍA 23
2.1 INTRODUCCIÓN 23
2.2 ARQUITECTURA GENERAL E IMPLEMENTACIÓN 23

2.2.1 ARQUITECTURA DE HARDWARE 24
2.3 IMPLEMENTACIÓN DE SOFTWARE 27
2.3.1 PROGRAMACIÓN EN ARDUINO UNO 29
2.3.2 PROGRAMACIÓN EN RASPBERRY PI 3 B+ 29
2.3.3 PROGRAMACIÓN WEB PHP 30
2.3.4 SERVIDOR WEB 32
2.3.5 PROGRAMACIÓN ANDROID CON APP INVENTOR 37
3. IMPLEMETACIÓN 39
3.1 IMPLEMENTACIÓN DEL PROTOTIPO 39
3.1.1 IMPLEMENTACIÓN DEL PROTOTIPO EN PCB 39
3.1.2 ARDUINO UNO. 43
3.1.3 RASPBERRY PI 3 B +. 43
3.1.4 BASE DE DATOS. 44
3.2 IMPLEMENTACIÓN DE LA APLICACIÓN ANDROID 44
3.3 IMPLEMENTACIÓN DEL PROTOTIPO EN EL *DATA CENTER*. 48
4. RESULTADOS Y DISCUSIÓN 51
4.1 VOLTAJE 51
4.2 CORRIENTE 54
4.3 POTENCIA 57
4.4 TEMPERATURA 60
4.5 CONTEO PARA CONTROLAR LA APERTURA Y CIERRE DEL PDU PRINCIPAL 61
5. CONCLUSIONES Y RECOMENDACIONES 63
5.1 CONCLUSIONES 63
5.2 RECOMENDACIONES 64
6. REFERENCIAS BIBLIOGRÁFICAS 66
ANEXOS 70
ANEXO A 71
ANEXO B 75
ANEXO C 79
ORDEN DE EMPASTADO 133

RESUMEN

En el presente trabajo se va a desarrollar un prototipo basado en hardware libre, para poder realizar mediciones de voltaje y corriente. La medición permitirá determinar la potencia consumida de un *Data center* Universitario, la misma que podrá ser visualizada y monitoreada en una aplicación para teléfonos móviles. El proyecto que se va a desarrollar abarca cinco capítulos los mismos que se presentan a continuación.

En el capítulo I, se estudiarán las características de un *Data Center*, seguidamente se revisarán los conceptos relacionados a los sensores y elementos requeridos para armar el prototipo, así como los recursos necesarios para el funcionamiento del software APP Inventor. Se consultará el funcionamiento de un servidor en la nube y cómo administrar la base de datos que se generará en este.

En el capítulo II, se va a implementar un prototipo basado en hardware libre y una aplicación que se desarrollará en APP Inventor. Se planificará la conexión inalámbrica, entre el prototipo y el servidor situado en la nube.

En el capítulo III, se elaborará el prototipo en base a hardware libre y los sensores seleccionados en la fase de diseño, con el propósito de realizar las mediciones de voltaje y corriente. Se realizará la conexión inalámbrica entre el prototipo diseñado y el servidor situado en la nube.

En el capítulo IV, se determinará la correcta operación del prototipo realizando dos tipos de pruebas de funcionamiento.

Finalmente, en el capítulo V, se presentarán las conclusiones y recomendaciones del presente trabajo.

PALABRAS CLAVE: *Data Center*, medición de potencia, hardware libre, sensores de voltaje, sensores de corriente, comunicación inalámbrica, servidor.

ABSTRACT

In the present work a prototype is going to be developed based on open source hardware, to be able to make measurements of voltage and current. The measurement will allow the visualization and monitoring of the consumed power of a University *data center* through the use of an application for mobile phones. The project that is going to be developed includes five chapters, which are presented below.

In chapter I the characteristics of a *data center* will be studied. Then the concepts related to the sensors and elements required to assemble the prototype will be reviewed. As well as the resources necessary for the operation of APP Inventor software. The operation of a server in the cloud will be consulted as well as the management of its database.

In Chapter II a prototype based on open source hardware and an application to be developed in APP Inventor will be implemented. The wireless connection between the prototype and the server located in the cloud will be planned.

In chapter III the prototype will be elaborated based on open source hardware and the sensors selected in the design phase. The wireless connection will be made between the designed prototype and the server located in the cloud.

In chapter IV the correct operation of the prototype will be determined by performing two types of tests: internal (operational) and general.

Finally, in chapter V the conclusions and recommendations of this work will be presented.

KEYWORDS: Data Center, power measurement, open source hardware, voltage sensors, current sensors, wireless communication, server.

1. INTRODUCCIÓN

Los centros de datos o *Data Centers* son estructuras o espacios dedicados dentro de un edificio para albergar varios tipos de sistemas computacionales potentes (servidores) y sus componentes, como los sistemas de telecomunicación y el almacenamiento. Actualmente, los centros de datos consumen mucha energía y requieren una infraestructura completa de refrigeración y de distribución de energía. Por otro lado, los *Data Centers* universitarios ayudan a un mejor funcionamiento de las redes localizadas en un campus las cuales dan servicio tanto al personal administrativo como a profesores y alumnos [1].

Los *data centers* son importantes y necesarios para que funcionen los sistemas de información dentro de la organización o universidad y que la comunicación entre los usuarios y sistemas fluya correctamente. Es por ello que, un buen mantenimiento, un adecuado uso de recursos y un propicio monitoreo de su potencia de consumo, permitirán que el *Data Center* presente un rendimiento apropiado para realizar su trabajo [1].

Uno de los controles que se debería llevar a cabo en un *Data Center* es el monitoreo de la potencia, ya que permite diagnosticar varias situaciones de energía dentro del mismo, por ejemplo, validar si existe un excesivo calentamiento de los equipos que componen el centro de datos, y por lo tanto un desproporcionado consumo de potencia en los equipos de enfriamiento [2].

Basado en mediciones de voltaje y corriente, es posible realizar un monitoreo básico de la potencia que consume un *Data Center* diariamente, y esto puede ayudar a tomar decisiones con respecto a disminuir el consumo de potencia. Por consecuente, el uso de hardware libre, transmisión inalámbrica de información y la generación de una aplicación para teléfono móvil, generará una herramienta eficaz que permitirá a las personas encargadas de un *Data Center* llevar a cabo un adecuado monitoreo de la potencia consumida en el mismo [2].

1.1 OBJETIVOS

El objetivo general de este Proyecto de Titulación es:

- Desarrollar un prototipo para monitorear la potencia de un *Data Center* universitario, basado en mediciones de voltaje y corriente que serán enviados a la nube mediante el uso de la red WLAN, las cuales serán receptadas y monitoreadas en un teléfono móvil mediante una aplicación.

Los objetivos específicos de este Proyecto de Titulación son:

- Analizar los fundamentos teóricos necesarios para el desarrollo e implementación del proyecto.

- Desarrollar un prototipo para mediciones de voltaje, corriente, temperatura y el número de veces que se abre la puerta del *PDU (Data Protocol Units,* Unidades de Protocolo de Datos*)* con el uso de Arduino uno, la minicomputadora Raspberry Pi 3 B+, sensores de voltaje, corriente, temperatura y un conmutador imantado, para calcular de la potencia consumida.

- Verificar el funcionamiento del prototipo en el *Data Center* de la Universidad Central del Ecuador.

- Establecer la comunicación inalámbrica mediante el uso de Raspberry y llevar a cabo la transmisión de los datos a un servidor que estará situado en la nube.

- Desarrollar la aplicación para teléfonos móviles mediante el uso del software APP Inventor para receptar y monitorear la información de potencia.

- Analizar los datos del consumo de potencia receptados diariamente a través de un informe de la aplicación móvil para recomendar una mejora del consumo.

1.2 ALCANCE

En este proyecto se pretende determinar el consumo de potencia en un *Data Center* universitario, mediante el uso de sensores de corriente y voltaje hardware libre. Estos datos de consumo serán almacenados en un servidor en la nube y se pueden visualizar y monitorear mediante una aplicación móvil desarrollada con la ayuda del APP Inventor del MIT.

Además, se pretende construir un voltímetro con el uso de sensores hardware libre, para determinar cuánto voltaje se está consumiente en los equipos del *Data Center* y en el

sistema de enfriamiento. Adicionalmente, se calcularán medidas estadísticas como la corriente media y el uso de la potencia total.

Mediante una aplicación para celulares Android se podrá obtener los datos de las mediciones del *Data Center*, esta aplicación será desarrollada con la herramienta APP Inventor del MIT. Además, esta permitirá visualizar gráficos como la potencia en función del tiempo alcanzado en el transcurso del día.

Finalmente, se realizará un análisis del consumo total de potencia para determinar el rendimiento del *Data Center*.

1.3 MARCO TEÓRICO

1.3.1 DATA CENTER Y SU FUNCIONAMIENTO

Un *Data Center* es considerado como el punto central de un sistema, ya que controla el tráfico de la información y las telecomunicaciones entre los usuarios o sistemas, posee varios servidores donde la información es almacenada, procesada y gestionada para realizar diferentes tareas que se requieren en la empresa o en donde este ubicado el *Data Center*. [3].

Las grandes compañías que manejan elevados bloques de información demandan el uso de *Data Center*, pero debido a que estos tienen un alto costo de mantenimiento, muchas compañías no pueden darse el lujo de tener uno propio. La utilidad de un *Data Center* se ve reflejado cada día ya que desempeñan una función fundamental en la vida garantizando el buen desempeño de las tareas diarias de una empresa [3].

Uno de los más relevantes *Data Centers* del Ecuador es el que está ubicado en la Universidad Central del Ecuador (UCE), el cual realiza una función muy importante para todo el campus ya que brinda servicio a más de 36839 usuarios que conforman la comunidad educativa [4].

Los *Data Centers* se pueden clasificar por niveles: Tier I, Tier II, Tier III y Tier IV. Las cuales se diferencian en qué tipo de servicio ofrecen y su calidad, siendo el nivel Tier IV el de mejor calidad y disponibilidad, es decir, en cada nivel se establecen los requerimientos de una empresa. A continuación, se describen cada uno de los niveles [5].

Los Tier I se denominan centros de datos básicos con una disponibilidad del 99.671 % son los de menor disponibilidad y ofrecen menos garantías en su uso. Estos *Data Centers* son muy susceptibles a interrumpirse por actividades no programadas y no posee redundancia en la refrigeración y distribución eléctrica. La instalación de este tipo de centro de datos no necesariamente se realiza en suelos elevados, generadores auxiliares o *UPS (Uninterruptible Power Supply,* Fuente de energía ininterrumpida). La implementación se ejecuta en aproximadamente 3 meses y al menos una vez al año se encuentra fuera de servicio por mantenimiento [6].

Los *Data Centers* Tier II tienen una disponibilidad aproximada de 99.741%, esto quiere decir que este puede brindar un mejor servicio sin demasiadas interrupciones como el Tier I. Este es considerado como un centro de datos redundante ya que tiene componentes (N+1), este posee menos susceptibilidad a las interrupciones por actividades no planeadas, y está construido con suelos elevados y tiene generadores auxiliares o UPS. En este sistema se tiene una línea única de distribución eléctrica y de refrigeración conectada a los equipos del *Data Center*. El tiempo estimado para la implementación de esta categoría de *Data Center* es de 3 a 6 meses. Además, cuando es requerido el mantenimiento de la línea de distribución o de cualquier parte de la instalación el servicio se ve totalmente interrumpido [5] [6].

Los Tier III se consideran concurrentemente mantenibles ya que la disponibilidad aumenta a un 99.982%, y permite la planificación de las operaciones de mantenimiento sin afectar el servicio por eventos que ocasionen interrupciones en el servicio. Este tipo de *Data Center* tiene componentes redundantes igual a N+1 que se conectan a múltiples líneas de distribución y refrigeración, pero solamente está activa una. El tiempo estimado para su implementación es de 15 a 20 meses. Para realizar el mantenimiento existen múltiples líneas de distribución y refrigeración por lo cual es seguro dar mantenimiento en una línea y el servicio se puede dar por otras [7].

Por último, los Tier IV son los más avanzados en la actualidad, lleva el nombre de centro tolerante a fallos con una disponibilidad en el servicio de 99.995%. Este tipo de *Data Center* permite planear actividades de mantenimiento sin tener que afectar al servicio brindado en las partes críticas, también puede soportar un evento no planeado en el peor escenario dado sin que exista un impacto muy crítico en la carga. Estos *Data Centers* están conectados a múltiples líneas de distribución y refrigeración con varios componentes redundantes iguales a 2(N+1), lo que significa que tienen 2 UPS con una redundancia de N+1. Su tiempo de construcción estimado es de 15 a 20 meses [7].

Para poder conocer cuántas horas al año el *Data Center* no estará disponible se puede realizar una regla de tres, considerando que el 100% son todas las horas del año. Por ejemplo, para un *Data Center* Tier III 99.982% estaría sin servicio 1.6 horas [6].

Por lo detallado anteriormente, el *Data Center* de la Universidad Central del Ecuador es designado como Tier II ya que cumple las características descritas anteriormente [5].

Las categorías mencionadas dan la garantía de funcionamiento de un *Data Center* y la fiabilidad en cada uno. Mientras mayor es el nivel del *Data Center,* mayor es el costo de su construcción y tiempo para su implementación [8].

Para garantizar la disponibilidad del servicio ante cualquier evento inesperado, un *Data Center* debe cumplir con los siguientes requerimientos y cantidades de *CDP (Center Data Protection*, Centro de Protección de Datos).

- Se debe asegurar que la conexión y el servicio sea las 24 horas del día y los 7 días a la semana [8].
- Para que las condiciones en las que trabaja el *Data Center* sean las adecuadas siempre debe haber un control del lugar en el que está situado y también se debe vigilar la temperatura que es un factor de gran importancia. Además, se debe considerar que la infraestructura debe tener equipos protegidos contra incendios por lo que sus materiales no deben ser inflamables.
- Para que el servicio sea ininterrumpido los *CDP* tienen un sistema para dar un apoyo en el caso de que los servidores caigan [8].
- Siempre se debe tener en cuenta que las personas que ingresan al *Data Center* deben ser identificadas y tener un conocimiento básico de su funcionamiento ya que podrían ocurrir incidentes de seguridad, independientemente de su tamaño [8].
- La administración del *Data Center* debe asegurar un alto grado de confiablidad y seguridad de la información ya que con el avance tecnológico pueden existir vulnerabilidades de acceso a información confidencial [8].

1.3.2 APP INVENTOR

Es una herramienta web de desarrollo que ayuda a la creación de aplicaciones para dispositivos Android usando un navegador web y un teléfono o emulador que se encuentre conectado, creada por el *MIT (Massachusetts Institute of Technology*, Instituto Tecnológico

de Massachusetts) y el equipo de Google *Education*. El lenguaje de App Inventor está basado en bloques y orientado a eventos, además que es de fácil uso, por medio de arrastrar y soltar construye aplicaciones para la plataforma Android [9].

Para crear aplicaciones en App Inventor se tiene las siguientes fases:

1.- En el proceso del diseño se muestra el display de un teléfono celular inteligente en el que se va a desarrollar una interfaz de usuario en la que se puede agregar distintas partes: botones, imágenes, texto, audio, etc. Además, esta herramienta toma en cuenta el aspecto gráfico, comportamiento, etc [9].

2.- El editor de bloques es el que permite mediante el uso de bloques realizar la programación de una forma visual e intuitiva con el flujo del funcionamiento de un programa [9].

3.- Cuando ya se ha terminado la aplicación se puede generar el instalador APK[1] para teléfonos celulares inteligentes Android y se obtiene un código *QR (Quick Response Code*, Código de Respuesta Rápida) para realizar su descarga [9].

Para poder utilizar App Inventor 2 son necesarios los siguientes requisitos:

Se requiere una Pc o Laptop con sistema operativo Windows, Mac o Linux y un navegador Google Chrome o Mozilla Firefox ya que Internet Explorer no está soportado [9].

Para proceder a utilizar el software App Inventor se tiene que acceder a la dirección: http://ai2.appinventor.mit.edu/ a través del navegador, si ya se tiene una cuenta Google, se procede a ingresar a la misma mediante el usuario y contraseña [10].

Esta herramienta de software permite escoger el idioma requerido y testear las aplicaciones directamente de dos formas:

- En la primera opción se conecta el dispositivo Android a la Pc o Laptop para realizar el testeo y consiste en la instalación de la aplicación *MIT AI2 Campanion* para establecer la conexión requerida con la página web http://ai2.appinventor.mit.edu/. Como requerimiento se necesita que la Pc o Laptop y el dispositivo Android se encuentren en la misma red [10].
- La segunda opción para testear es mediante una conexión USB desde el ordenador al dispositivo Android [10].

[1] *APK (Android Application Package*, Paquete de Aplicación Android), extensión de archivo usada para aplicaciones desarrolladas en App Inventor.

1.3.3 SERVIDOR EN LA NUBE

Figura 1.1. Servidor en la nube trabajando con Raspberry [11].

El servidor en la nube se define como una infraestructura que puede ser virtual o física para almacenar datos y procesar aplicaciones e información. Este está creado mediante la división de un servidor físico en varios servidores virtuales. Para procesar cargas de trabajo y guardar información se requiere la utilización de un modelo de Infraestructura como servicio (IaaS). Se puede tener acceso al servidor virtual mediante el uso de una interfaz de forma remota [11] [12].

Las características principales de un servidor en la nube son:

- Basándose en el uso de la infraestructura de computación, el servidor puede ser físico o virtual, o una combinación de ambas.
- Posee la capacidad de un servidor local.
- Puede almacenar información de grandes volúmenes y cargas de trabajo intensivos.

- Con el propósito de acceder bajo demanda a servicios automáticos es indispensable el uso de *APIs (Application Programming Interface,* Interfaz de programación de aplicaciones).
- Existe la posibilidad de realizar el pago mensualmente o por consumo.
- En función de las necesidades se puede escoger un plan de hosting compartido escalable [11].

La rentabilidad en los servidores en la nube permite que el usuario u organizaciones interesadas paguen por lo que requieren y por consiguiente el gasto de mantener el hardware del servidor se reduce [13].

Con el objetivo de satisfacer las necesidades cambiantes los usuarios pueden escalar los recursos de informática y almacenamiento para empresas que tienen necesidades cambiantes [11] [12].

Clasificación de los tipos de Nube:

Nube privada o dedicada: Usualmente esta nube es usada para una empresa u organización en la que se tiene hardware dedicado, tiene un mejor rendimiento en general y es una gran opción para la seguridad [14].

El despliegue de esta nube generalmente se realiza en un proveedor de servicios o en las oficinas de los usuarios o clientes. El uso de un *Data Center* externo es recomendado ya que es eficaz en temas de seguridad [14].

Las ventajas de esta nube es que la seguridad es mayor a cualquier otra nube que se utilice, ayudando a los procesos internos de seguridad y confiabilidad, el rendimiento de también es superior ya que este hardware no se comparte con otros clientes. El costo del uso de este servidor es mayor, pero es justificado frente a sus ventajas [13].

Nube pública o compartida: Esta nube da un servicio en donde el almacenamiento de la información y la memoria se utiliza de forma compartida entre todos los usuarios de la red. El usuario debe tener una gran habilidad técnica ya que realiza el trabajo por medio del despliegue de un portal. Las ventajas de este tipo de nube es que no existen contratos para tener permanencia, hay una autonomía para administrar recursos, existe la posibilidad de tener servicios híbridos y ahorrar en los costos [14] [13].

Las desventajas de este tipo de nube son que no se tienen políticas de privacidad y confidencialidad para sus procesos internos. Pueden existir brechas de seguridad ya que no hay destrucción inmediata de los datos al quitar el servicio, la información que yace en la nube queda expuesta. Si el cliente no es experto los costos son altos ya que se requiere soporte.

Nube comunitaria: Es una nube que funciona agrupando a empresas, organizaciones que tienen necesidades en común y por lo tanto se abaratan los costos [13].

Nube Híbrida: Con el fin de conmutar datos o extender funcionalidades diferentes las nubes se fusionan de dos o más grupos permitiendo la realización de los planes de continuidad de negocios y extender los servicios por muchas localizaciones geográficas. Es recomendada para *Data Centers* ya que requiere una red de alta velocidad [14].

1.3.4 RASPBERRY PI 3 B+

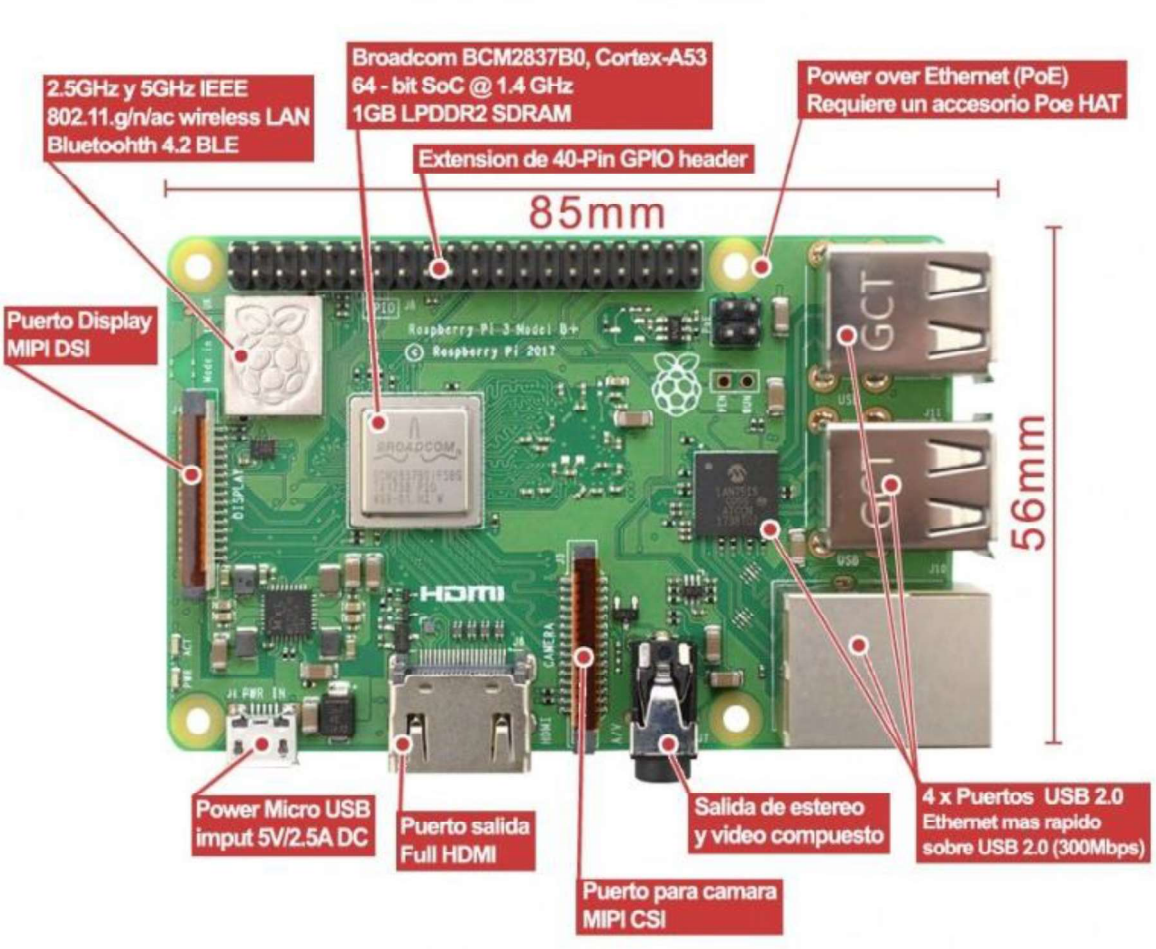


Figura 1.2. Raspberry pi 3 B+ [15].

Es una minicomputadora con una placa reducida SBC de costo bajo. Raspberry Pi usa hardware libre y tiene funcionamiento en base al microcontrolador ATmega644. Fue desarrollado en Reino Unido con el objetivo de ser un ordenador desnudo, es decir, que se pueden eliminar accesorios sin que se vea afectado el funcionamiento de este [16].

Raspberry pi 3 B+ viene con una capacidad inalámbrica mejor a sus antecesores, con Wifi 802.11ac de doble banda que funciona con 2.4 GHz y 5 GHz proporcionando un mejor alcance en entornos inalámbricos desafiantes [16].

El blindaje de metal que está pintado en el lado superior actúa como un disipador de calor y de presión, pero es recomendable adaptar otros disipadores para reducir el calentamiento de los circuitos. Con respecto a los modelos anteriores de Raspberry Pi el modelo B+ es tres veces más rápido que el Pi 2 y 3 por lo tanto es capaz de ejecutar múltiples funciones a un ritmo bastante aceptable [16].

El Raspberry Pi 3 Modelo B+ es el último modelo lanzado al mercado en el rango de Raspberry Pi 3.

Las partes de la Raspberry pi son:

- Broadcom BCM2837B0, Cortex-A53 64-bit SoC a 1.4GHz.
- 2.4GHz y 5GHz IEEE 802.11.b/g/n/ac wireless LAN.
- Bluetooth 4.2, BLE.
- RAM de 1GBLPDDR2 SDRAM.
- Gigabit Ethernet sobre USB 2.0 (máximo throughput 300Mbps).
- 4 buses USB 2.0.
- 1 puerto HDMI.
- Puerto para display MIPI DSI.
- Puerto para cámara MIPI CSI.
- 4 salidas estéreo polo y puerto de video compuesto.
- Pines para la entrada y salida con objetivo general.
- Salida analógica de video RCA.
- Alimentación por medio de un conector micro USB de 5V máximo.

- Lector de Tarjetas SD.
- H.264, decodificador MPEG-4 (1080p30); codificador H.264 (1080p30); OpenGL ES 1.1, 2.0 para gráficos multimedia [17].

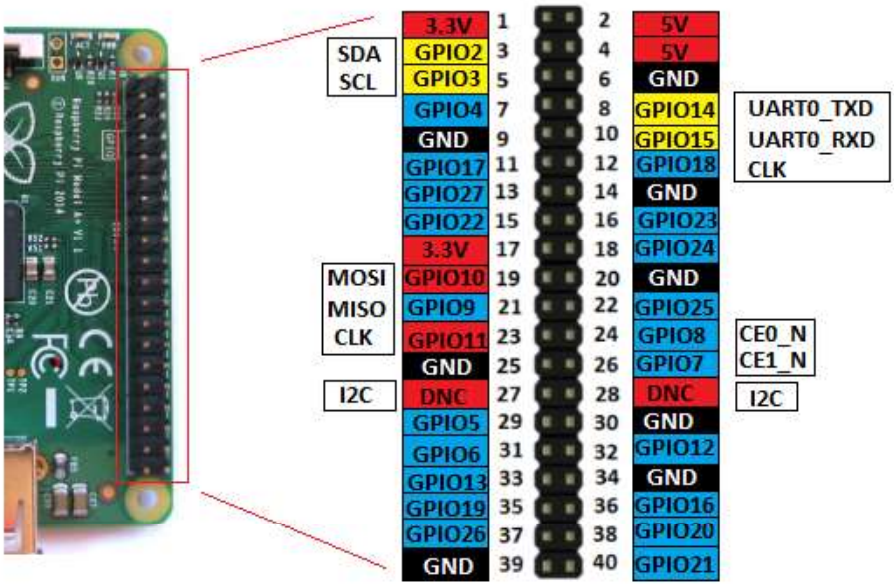


Figura 1.3. Pines de Raspberry Pi 3 B+ [17].

Raspberry pi 3 B+ posee un encabezado de 4 pines que está agregado al tablero situado cerca del encabezado de 40 pines. Este encabezado de 4 pines permite el uso de *PoE (Power Over Ethernet*, Alimentación a través de Ethernet) en otras palabras, brinda la corriente eléctrica que necesita el dispositivo ya que utiliza cables de datos en vez de cables de alimentación reduciendo la cantidad de cables que se necesitan la instalar un dispositivo en los proyectos que se realice con Raspberry [16] [17].

El encabezado poseedor de los 40 pines es utilizado para el realizar la conexión externa con la Raspberry y en la versión B+ se tiene que es igual a las anteriores.

De la cabecera de 40 pines se usan 26 para pines de entrada y salida E/S digitales y 9 de los 14 pines que quedan son denominados pines de E/S dedicados, lo que muestra que no vienen con una función alternativa [17] [16].

Los pines 3 y 5 poseen una resistencia de extracción de 1.8 KΩ y los pines 27 y 28 se encuentran dedicados a la *ID EEPROM (ID Electrically Erasable Programmable Read-Only Memory*, Identificación de la ROM programable y borrable eléctricamente) El cabezal *GPIO (General Purpose Input/Output,* Entrada/Salida de Propósito General) esta reposicionado para permitir un mayor espacio en el orificio de montaje extra [16].

Los sistemas operativos soportados por Raspberry son de software libre como Linux, que tiene muchas alternativas con sus diferentes sistemas operativos como Debian, Fedora Remix y Arch Linux [17].

1.3.5 RASPBIAN

Es un sistema operativo basado en Debian para la minicomputadora Raspberry Pi, es una distribución Linux y de software libre [18].

Este sistema operativo es un port no oficial de Debian para la Raspberry Pi, el cual posee un soporte destinado para realizar cálculos en coma flotante por hardware. Permitiendo dar mayor rendimiento la placa computadora [18].

Las versiones de este sistema operativo son: [18].

- Raspbian Píxel es la versión completa que posee un entorno gráfico con menús, iconos, fondos de pantalla, etc [18].
- Raspbian Lite es la versión carente de entorno gráfico es decir que solo se trabaja mediante la consola [18].

1.3.6 PYTHON

Es un lenguaje de programación que se caracteriza por su sencillez en su escritura y sintaxis. Tiene como filosofía hacer que se tenga un código legible. Además, que tiene estructuras de control que se organizan por medio de tabulaciones y no es necesario el uso de llaves, algo muy bueno en Python es que tiene limpieza y claridad sin igual. [19].

Gracias a que Python es un lenguaje interpretado[2], no se necesita compilar para que este sea ejecutado. El modo interactivo de Python permite la ejecución de instrucciones directo en el intérprete [19].

[22] Lenguaje Interpretado. – Es un lenguaje que no requiere que su código sea procesa por medio de un compilador, lo que significa que no es necesaria la traducción de todo el código ya que el computador puede ejecutar las instrucciones que da el programador.

Adicionalmente, es un lenguaje de programación multiplataforma y accede a la expansión de sus funciones con el uso de múltiples librerías como C/C++, estas se utilizan para dar servicio a páginas web, aplicaciones Android, videojuegos 3D, etc [19].

Python posee grandes facilidades para realizar programación orientada a objetos[3]. Está basado en lenguaje ABC e influenciado por otros como C, Modula-3, etc.

1.3.7 ARDUINO UNO

Es una plataforma que se encuentra basada en código abierto y su función principal receptar información del mundo físico. Su funcionamiento se encentra basado en una placa con un microcontrolador y el entorno de desarrollo para crear diferentes softwares para esta plataforma [20].

Con Arduino se puede realizar una variedad de proyectos ya que puede leer datos de muchas clases de interruptores, sensores, etc. y controlar múltiples actuadores físicos [21].

Esta construido con un microcontrolador ATmega328, su alimentación puede ocurrir de varias maneras por medio de su interfaz *USB (Universal Serial Bus*, Bus Universal en Serie) conectándolo a la computadora o a una fuente externa. Opera de 7-12V con la fuente externa, los pines digitales de entrada y salida son 14 y de estos 6 proveen una salida *PWM (Pulse-Width Modulation,* Modulación por Ancho de Pulso). También cuenta con 6 pines analógicos. Tiene memoria flash de 32 KB una *SRAM (Static Random Access Memory*, Memoria Estática de Acceso Aleatorio) de 2KB, *EEPROM (Electrically Erasable Programmable Read-Only Memory*, Memoria de solo Lectura Programable y Borrable Eléctricamente) de 1KB. La velocidad del reloj es de 16 MHz [21].

[3] Programación Orientada a Objetos. – Es una forma de programación que usa objetos y sus interacciones, requerida para el desarrollo de aplicaciones y programas informáticos.

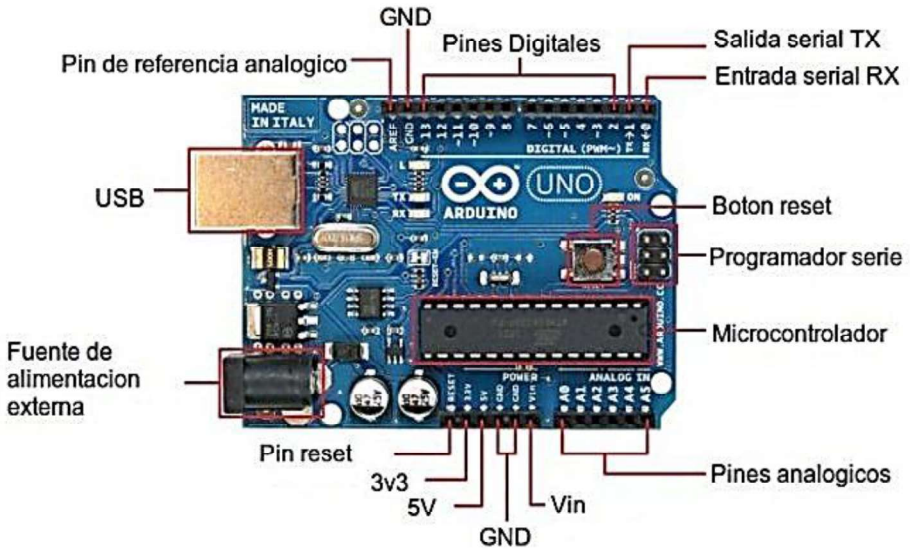


Figura 1.4. Arduino Uno [20].

El pin *VIN (Voltage Input*, Voltaje de Entrada) es la fuente de la tensión de entrada que contiene la tensión a la que se está alimentando al Arduino por la fuente externa [20].

El pin que indica 5V tiene una tensión regulada de 5V que pude venir del pin *VIN*, también pude venir de una fuente externa de 5V o a través de USB [21].

El pin que muestra 3.3V es un pin de fuente interna que ha sido generada por un regulador interno y su consumo de corriente es 50mA máximo [21].

El Pin *GND (Ground*, Puesta a Tierra) que indica el pin a tierra [21].

Arduino tiene 14 pines digitales que se pueden utilizar como una entrada o salida, estos pines pueden recibir como máximo 40 mA con resistencias pull-up de 20 a 50 K Ohm [21].

Los pines 0 (RX) y 1 (TX) son utilizados para transmisión y recepción de información o datos serie TTL [21].

Los Pines 2 y 3 consisten en las interrupciones externas.

Los pines 3, 5, 6, 9,10 y 11 Son pines por modulación de ancho de pulso que constituyen 8 bits de salida con la función analogWrite () [20].

Los pines 10 *SS (Slave Select*, Seleccionar Esclavo), 11 *MOSI(Master Out Slave In*, Maestro fuera y Esclavo Dentro), 12 *MISO (Master In Slave Out*, Maestro en y Esclavo

Fuera), 13 *SCK (Serial Clock*, Reloj serial), son pines de apoyo para realizar la comunicación *SPI (Serial Peripheral Interface*, Interfaz Periférica Serial) [21].

El pin 13 tiene un led conectado a el que indica si el pin el de valor alto encendiéndose y si el pin es de valor bajo este led se encuentra apagado [20].

Arduino tiene 6 entradas analógicas con etiquetas desde A0 hasta A5 que brindan cada uno 10 bits de resolución lo que quiere decir que se obtienen 1024 estados. Cada uno tiene un voltaje de 5V [21] [20].

1.3.8 LENGUAJE DE PROGRAMACIÓN PARA ARDUINO

El lenguaje estandarizado para programar en Arduino es C++, siendo posible la programación del mismo en otros lenguajes, pero no es un lenguaje puro C++, es una arreglo que viene de avr-libc la misma que brinda una librería de C con muy alta calidad para el uso con *GCC (GNU Compiler Collection*, Colección de Compiladores GNU) en los microcontroladores como el que posee Arduino ATmega328 [22].

Arduino brinda el uso de librerías facilitando así la programación del ATmega328 [22].

1.3.9 MEDICIÓN DE LA CORRIENTE

Para los propósitos indicados se tiene el uso de sensores de corriente adecuados con Raspberry pi para medir corriente alterna o continua, se debe producir una señal a la salida de corriente lineal que sea proporcional a la corriente que circula por el mismo, esto es muy usado para medir el consumo eléctrico de redes y equipos eléctricos [23].

Comúnmente se utilizan sensores de pinza amperométrica de efecto hall ya que no se invade la infraestructura en general [23].

Sensores efecto hall: Gracias a este sensor se puede realizar mediciones de corriente tan solo con colocar la pinza alrededor del cable y por lo cual es llamado sensor no invasivo.

El sensor escogido para realizar las mediciones será el SCT013 100 el cual puede medir en el rango de 1A – 60A, trabajando a temperaturas de -25 hasta + 70 grados centígrados. En la Figura 1.5 se muestra el sensor de efecto hall que se utiliza [24].

Figura 1.5. Sensor de corriente de efecto hall SCT013 100 [25].

En la Figura 1.5 a continuación se muestra la estructura interna del sensor de corriente.

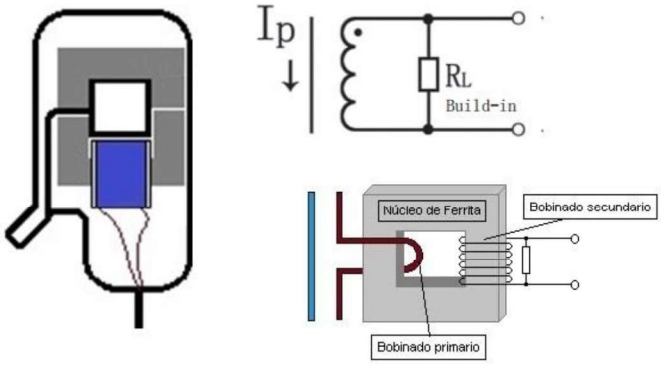


Figura 1.6. Estructura interna del sensor de corriente SCT-013 [26].

Estos sensores SCT-013 trabajan como transformadores de corriente AC a DC. Este sensor de corriente alterna tiene el devanado primario como un cable del equipo que se desea realizar la medición y tendríamos como número de vueltas 1. El devanado secundario tendría 2000 vueltas [24].

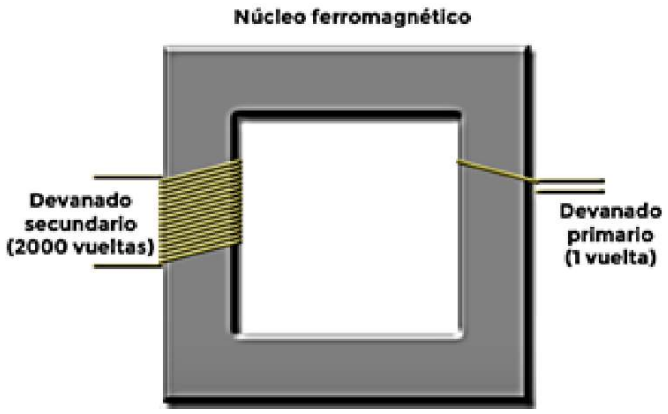


Figura 1.7. Principio de funcionamiento sensor SCT-013 [26].

Para medir una intensidad de 20 Amperios por ejemplo aplicando la relación de transformación (ecuación 1.1) se tiene: [26].

$$\frac{Np}{Ns} = \frac{Is}{Ip} => Is = \frac{Np * Ip}{Ns} = \frac{1 * 20}{2000} = 0.01\ A \quad (1.1)$$

Aquí se produce una transformación de energía ya que nunca se podría medir 20 A con una Raspberry que solo acepta 2.5 A de entrada.

El sensor que se va a utilizar va a ser SCT-013-000, este sensor no posee resistencia de carga y proporciona una corriente. La capacidad de medición es desde 50 mA a 100 A [24] [27].

1.3.10 MEDICIÓN DEL VOLTAJE

Para el propósito de medir el voltaje se va a utilizar el módulo transformador de voltaje alterno ZMPT101B que permite medir voltaje alterno, lo que hace este sensor es reducir el voltaje AC de entrada a un voltaje menor que pueda leer el Raspberry, ya que esta minicomputadora solo acepta de 0V a 5V [28].

Los sensores de voltaje son dispositivos que permiten medir, determinar y monitorear el suministro de voltaje, ésta magnitud física se transforma en señal eléctrica que se puede

leer fácilmente. En este trabajo se ha utilizado el sensor ZMPT101B descrito a continuación [28] [29].

El ZMPT101B (ver figura 1.8) es un transformador de corriente de tamaño pequeño con buena consistencia y aislamiento para las mediciones de voltaje. Además, este sensor contiene un circuito amplificador operacional que permite obtener una mejor precisión para la transformación de voltaje. Las especificaciones técnicas se pueden observar en la tabla siguiente: [28].

Tabla 1.1. Especificaciones ZMPT101b [28].

Parámetro	Valor
Relación de vueltas en el bobinado	1000:1000
Rango de frecuencia	50~60 Hz

Figura 1.8. Sensor de Voltaje Ac Zmpt101b [29].

1.3.11 MEDICIÓN DE LA TEMPERATURA

El sensor por utilizar será el DHT11 que es compatible con Arduino y Raspberry. El sensor de temperatura y humedad utilizado en este trabajo es el DHT11. Este es un sensor analógico diseñado para detectar el cambio físico en el calor y la humedad cuando se expone al aire con el cableado y la programación adecuados [30].

Su pequeño tamaño, su bajo precio, su bajo consumo de energía y sus rápidas respuestas son las características para ser una de las mejores opciones para muchos usuarios. El sensor DHT11 es aplicable en HVAC (calefacción, ventilación y aire acondicionado), se puede usar en Pruebas e inspección de equipos y bienes de consumo. El uso del sensor DHT11 ha demostrado su utilidad para medir y controlar la temperatura y la humedad en aparatos domésticos, médicos y en muchos otros sectores. La Figura 1.9 muestra el sensor

de humedad y temperatura DHT11, el rango de precisión y rendimiento se muestra en la tabla 1.2 [30].

Tabla 1.2. Precisión y rendimiento DHT11 [31].

	Temperatura	Humedad
Rango de medición	0 a 50 grados centígrados	20 a 90 RH
Precisión	±2%	±5%

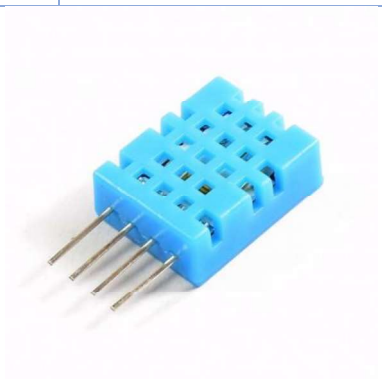


Figura 1.9. Sensor de Temperatura y Humedad DHT11 [30] [31].

1.3.12 SENSOR PARA CONTROLAR LA APERTURA Y CIERRE DEL PDU PRINCIPAL

Para obtener la medida de las veces que se abre la puerta del *PDU* de un centro de datos ha utilizado un interruptor de inducción magnética. Es un dispositivo que interrumpe o deja pasar la corriente eléctrica accionado magnéticamente mediante imanes de la puerta. Estos dispositivos tienen varias aplicaciones en motores, robots, entre otros. Para este trabajo se ha utilizado el sensor *Reed Switch* (ver figura 1.10), el cual tiene las siguientes especificaciones: [32].

Tabla 1.3. Especificaciones *Reed Switch* [32].

Parámetro	Valor
Voltaje máximo de conmutación	300 VDC
Clasificación de Contacto Máx .	10W
Corriente Máxima de Conmutación	0.55 A

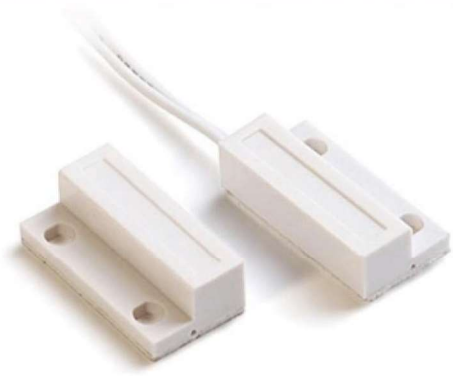


Figura 1.10. Interruptor de inducción magnética *Reed Switch* [32].

1.3.13 ADMINISTRACIÓN DE BASES DE DATOS EN LA NUBE (BASE DE DATOS COMO SERVICIO)

La *DBMS (Data Base Management System,* Administración de datos y Bases de datos) son la parte integral de varias aplicaciones. En particular, los DBMS se han utilizado masivamente debido a los beneficios que estos ofrecen: [33].

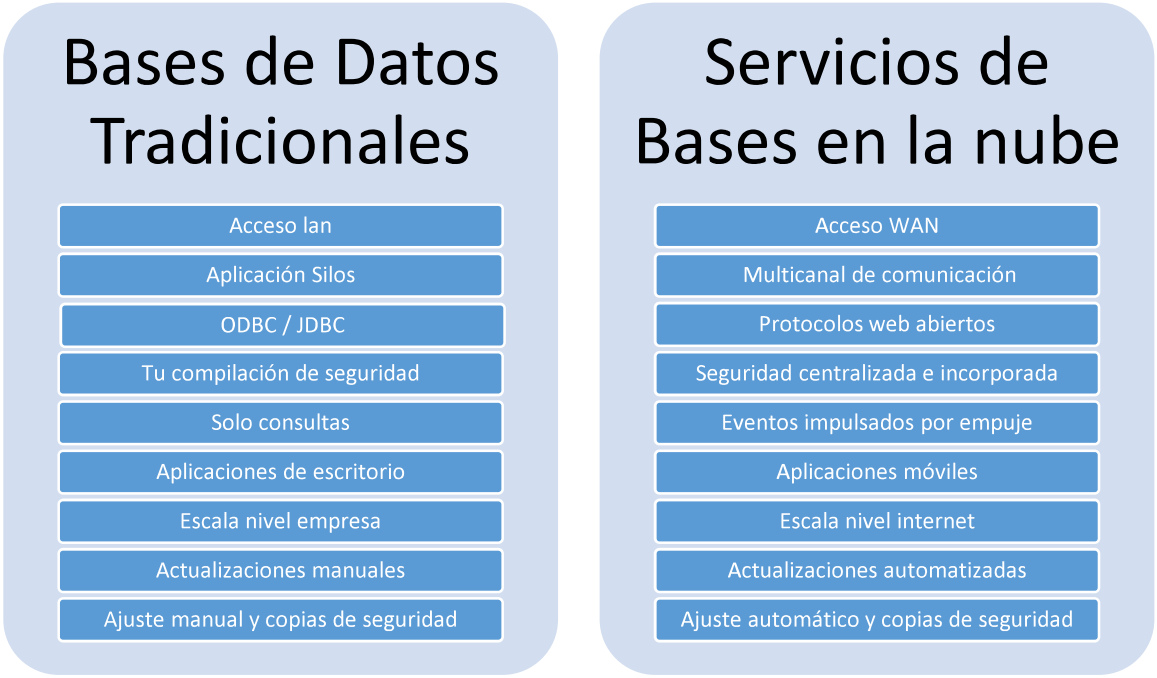
- Su funcionalidad en general ofrece un modelo intuitivo y simple para modelar diferentes tipos de aplicaciones.
- Consistencia: Datos siempre disponibles y sin errores sin importar las cargas de trabajo concurrentes y que los datos no estén sincronizados.
- Rendimiento: Baja latencia y alto funcionamiento de las bases de datos.
- Fiabilidad: Datos persistentes y seguros en presencia de diferentes tipos de fallas. [33].

Los DBMS tradicionales no están diseñados para ejecutarse sobre la arquitectura *shared-nothing* (máquinas independientes que realizan una tarea con un mínimo uso de recursos) y no proporcionan las herramientas necesarias para escalar a una gran cantidad de máquinas [33].

Líderes en tecnología como Google, Amazon y Microsoft han demostrado que los centros de datos que comprenden miles de nodos de cómputo ofrecen escala sin precedentes, ya que varias aplicaciones pueden compartir una infraestructura común. Las tres compañías han proporcionado marcos tales como *AWS (Amazon Web Services*, Servicios Web de

Amazon), App Engine[4] de Google y Microsoft Azure[5] para alojar aplicaciones de terceros en sus nubes (infraestructuras de centros de datos) [33].

A medida que el avance tecnológico va hacia el ámbito de la computación en la nube, que normalmente incluye centros de datos con miles de servidores, el enfoque manual de la administración de la base de datos ya no es viable. En cambio, existe una creciente necesidad de hacer que la capa de gestión de datos sea autónoma o auto gestionable, especialmente cuando se trata de la redistribución de la carga[6], la escalabilidad[7] y la elasticidad[8] [34].



9,10A

Figura 1.11. Tradicional VS Servicios de datos en la nube [34].

[4] App Engine. - Servicio de alojamiento web de Google para la ejecución de distintas aplicaciones sobre Google.
[5] Microsoft Azure. - Servicio en la nube para la creación y administración de aplicaciones en la red.
[6] Redistribución de la carga. - Quiere decir que no solo se concentre la carga en un servidor, sino que sea distribuida entre diferentes servidores.
[7] Escalabilidad. – El sistema puede crecer en datos y funcionalidad.
[8] Elasticidad. – Es la capacidad que tiene el centro de datos de ampliar y reducir sus recursos dependiendo de los requerimientos
[9] ODBC / JDBC. - Librerías que son utilizadas para la conexión a la base de datos.
[10] Aplicación Silos. - Aplicaciones para repositorios de información.

Como se puede observar en la Figura 1.11, la administración de los servicios de las bases de datos debe realizarse de manera automatizada ya sea las actualizaciones, los respaldos, las réplicas, entre otros. Además, la administración de seguridad de las bases de datos es centralizada [34].

1.3.14 PROTEUS

Proteus es un software para realizar simulaciones digitales y analógicas de una manera viable y práctica [35].

Es un programa muy completo que permite visualizar las diferentes conexiones y realizar diferentes diseños independientemente de los componentes que posea el circuito.

Permite la ejecución de diversos proyectos que tienen componentes electrónicos en todas sus diferentes etapas: [35].

- Diseño del esquema electrónico.
- Arquitectura del circuito impreso.
- Simulación total del diseño.

La ventaja de utilizar Proteus es que gracias a sus diferentes fases de prueba no existe necesidad de realizar nuevos prototipos [35].

Proteus posee de dos softwares principales: *ISIS (Intelligent Schematic Input System,* Sistema de Enrutado de Esquemas Inteligente) y *ARES (Advanced Routing and Editing Software*, Software de Edición y Ruteo Avanzado) [35].

ISIS es un software que da la facilidad de realizar esquemas o planos electrónicos de diferentes prototipos o conexiones de circuitos con los componentes necesarios para cada uno. Cada diseño que se ha desarrollado en *ISIS* está disponible para ser simulado en tiempo real con la ayuda del módulo *VSM (Virtual System Modeling*, *Sistema Virtual de Modelado*) que es una de las ventajas y partes de Proteus [35].

ARES es la herramienta utilizada para realizar el enrutamiento, edición y ubicación de los elementos del prototipo realizado. Con esta herramienta se pueden fabricar las láminas de circuito Impreso [35].

2. METODOLOGÍA

2.1 INTRODUCCIÓN

En este capítulo se especifican las arquitecturas e implementación del hardware y software utilizados para obtener un sistema de monitoreo de *Data Center*, este prototipo se ha realizado con el fin de no ser invasivo con ninguno de los componentes del *Data Center*. Toda la programación e instalación se realizó dentro del *Data Center* de la Universidad Central.

Además, se estudió las diferentes formas de medir la potencia en el *Data Center* y se decidió utilizar Arduino uno para receptar las mediciones analógicas de voltaje y corriente que estarían entregando los sensores, posteriormente estos valores de voltaje, corriente y potencia serán receptados en la tarjeta Raspberry pi 3 B+.

En las siguientes secciones se especifica la arquitectura utilizada para el desarrollo del proyecto.

2.2 ARQUITECTURA GENERAL E IMPLEMENTACIÓN

Todos los componentes principales del sistema se detallan en la Figura 2.1. Además, se especifica el proceso del sistema, en primer lugar, se mide la potencia mediante los sensores de corriente y voltaje localizados en el *Data Center* para posteriormente enviar la información de monitoreo a una tarjeta Raspberry Pi 3 B+, con esta tarjeta también se mide la temperatura interna del *Data Center*, y el número de veces que la puerta del *PDU* se abre. Toda la información obtenida es enviada a un servidor situado en la Universidad Central del Ecuador para un correcto análisis de los datos, es decir, se almacena esta información en una base de datos MySQL. Finalmente, se puede visualizar la información recopilada en una aplicación para dispositivos Android la cual está desarrollada mediante la herramienta APP Inventor del MTI.

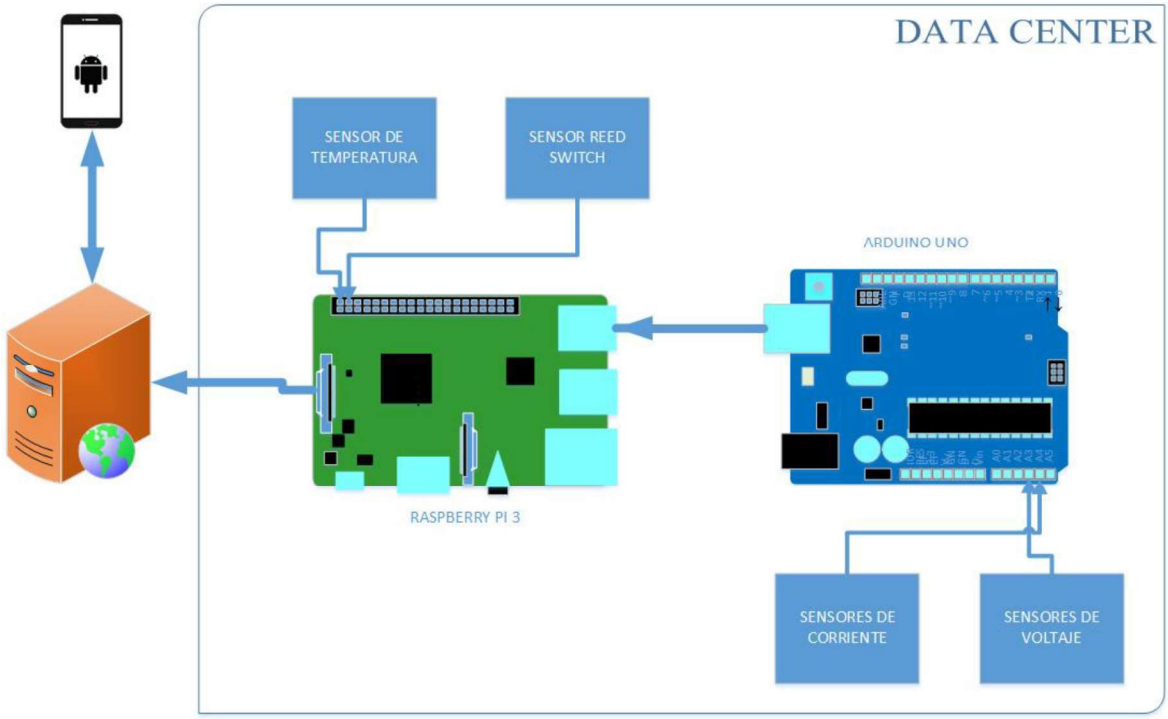


Figura 2.1. Arquitectura del sistema.

2.2.1 ARQUITECTURA DE HARDWARE

Todos los componentes de la arquitectura de hardware del prototipo están representados en la figura 2.2. Como se puede observar en la ilustración se ha utilizado una tarjeta Arduino UNO por su facilidad de implementación y desarrollo ya que tiene un lenguaje de programación muy similar al lenguaje C y C++. Adicionalmente, esta tarjeta contiene 6 entradas analógicas, las cuales permiten la lectura de los sensores de voltaje y corriente. Además, posee alimentación de 5V que puede ser adquirida mediante una fuente o con una alimentación directa de la computadora (por ejemplo, Raspberry PI 3+) mediante un cable *Universal Serial Bus* (USB).

Por otra parte, se ha hecho uso de la minicomputadora Raspberry Pi para realizar la lectura de los sensores de temperatura y magnético mediante los pines GPIO incorporados en la misma. Estos pines también realizan la alimentación de energía de los sensores. Adicionalmente, se utiliza la minicomputadora para enviar la información de todos los sensores (voltaje, corriente, temperatura y magnético) y la información de la potencia hacia un servidor ubicado en la Universidad Central del Ecuador. La transmisión de información se realiza en la red *Local Area Network* (LAN) de la Universidad mediante la tarjeta de red inalámbrica (Wireless) que posee la Raspberry.

A continuación, en la figura 2.2, se presenta el esquema de hardware general del diseño de todo el prototipo, las conexiones necesarias en cada pin de cada sensor y de los diferentes elementos.

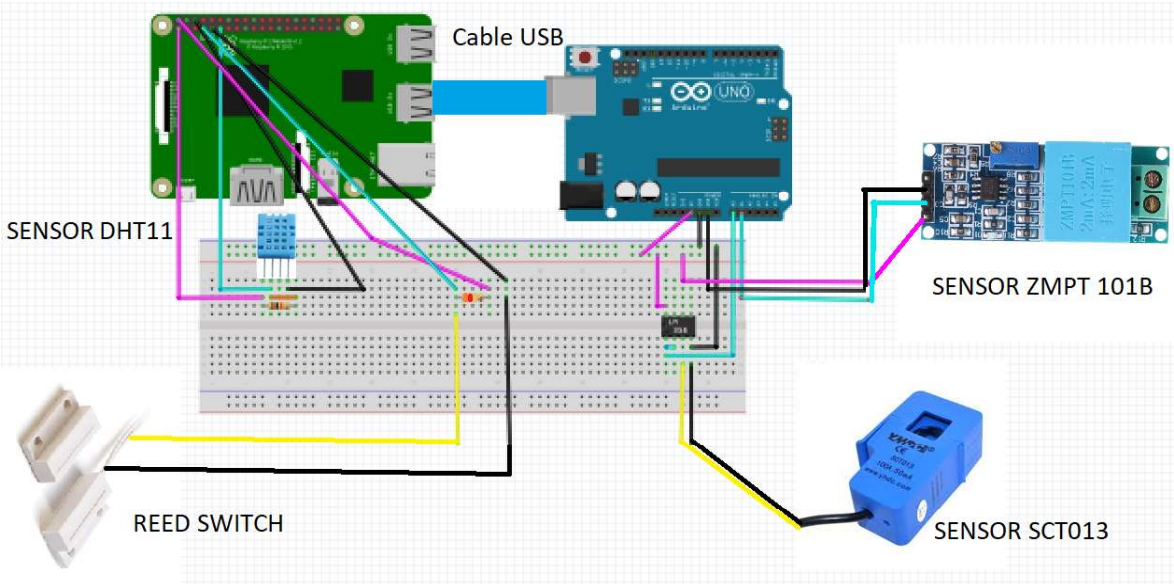


Figura 2.2. Diagrama de conexiones.

Como se puede ver en la Figura 2.2 la minicomputadora Raspberry Pi se conecta a Arduino mediante un cable de datos USB, el que sirve para poder enviar la toda información de los sensores de voltaje y corriente del Arduino a la Raspberry Pi mediante comunicación serial.

En general, se va a conectar un sensor de temperatura DHT11, un sensor imantado también llamado *Reed Switch* (Interruptor de Láminas), tres sensores de corriente efecto hall, conocidos como pinzas amperimétricas SCT 013 y tres sensores de voltaje ZMPT101B para el sistema trifásico que tiene el *Data Center*.

Teniendo en cuenta que el *Data Center* cuenta con un sistema trifásico para la alimentación de los equipos, se debe medir el voltaje y la corriente en cada una de las fases con relación al neutro para calcular la potencia que consume el *Data Center*.

Además, se mide la temperatura dentro del *Data Center*, y el número de veces que la puerta del *PDU* se abre para tener un control del ambiente ideal para el centro y de la seguridad de este.

La medición de corriente utiliza un circuito integrado LM358 ilustrado en la figura 2.3, este circuito posee dos amplificadores operacionales para usar con la pinza amperimétrica, estos amplificadores funcionan como rectificador de media onda eliminando la parte negativa, ya que el semi ciclo negativo de la onda que entrega la pinza amperimétrica podría dañar el pin analógico del Arduino UNO.

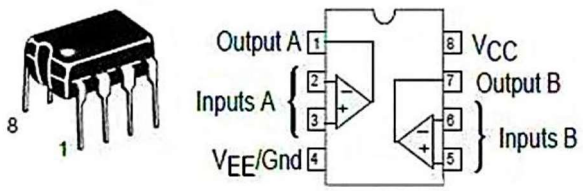


Figura 2.3. Circuito Integrado LM358 [35].

El amplificador operacional trabaja en una amplia gama de voltajes, proporciona un bajo consumo de corriente siendo independiente al voltaje que entrega la fuente de alimentación [35].

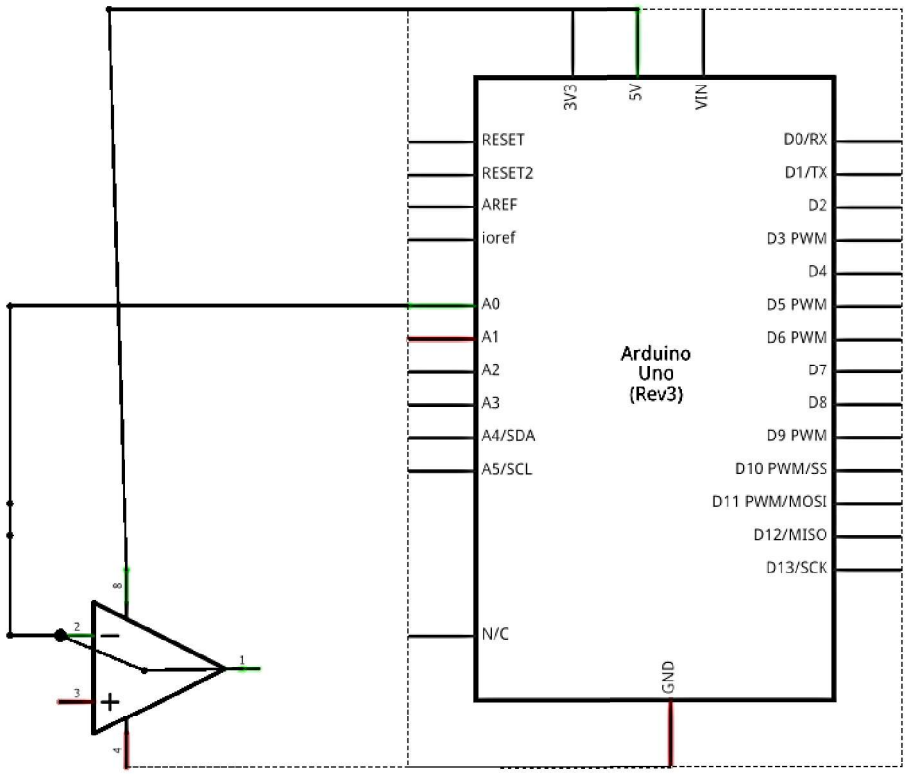


Figura 2.4. Conexión LM358 Arduino Uno.

La Figura 2.4 muestra la conexión del circuito operacional LM358 con Arduino uno, para posteriormente conectar la pinza amperimétrica en los pines 3 y 4 del circuito integrado LM358.

Como se muestra en la Figura 2.2 el sensor de voltaje ZMPT101B tiene el pin de la salida de la información conectado al segundo pin analógico de Arduino Uno, se alimenta al sensor de voltaje con 5V que entrega Arduino UNO. Las salidas del sensor permiten medir entre neutro y fase el valor del voltaje que se está consumiendo.

Las mediciones de corriente y voltaje se realizan para cada línea que alimenta al Data Center ya que es un sistema trifásico.

El sensor de temperatura está conectado a la tarjeta Raspberry Pi por medio del pin GPIO 11 y es alimentado con 3.3V que le entrega el Pin 1 de la tarjeta, se coloca una resistencia de 10K entre el pin 1 y 2 para tener una resistencia de pull up (4.7K – 10K) que evita el ruido o las medidas erróneas de la señal enviada por el sensor.

El sensor imantado está alimentado con 5 voltios que entrega la tarjeta Raspberry Pi por el pin 2, y de igual manera usa una resistencia de pull-up para evitar lo antes mencionado al receptar las medidas del paso de energía, se establece un contador que incrementa el valor en uno cuando existe dicho paso de energía.

2.3 IMPLEMENTACIÓN DE SOFTWARE

La arquitectura del software se detalla en la figura 2.5, todo el código de Arduino se ha desarrollado en el lenguaje C para obtener los datos de temperatura, corriente y potencia. Para la minicomputadora Raspberry PI 3 B+ su ha utilizado Python, la programación web en PHP y para la aplicación Andriod, se desarrolló en APP Inventor del MIT.

Como se puede observar en la figura 2.5, se ha desarrollado varios programas con diferentes lenguajes de programación. Cada uno de ellos se detalla a continuación:

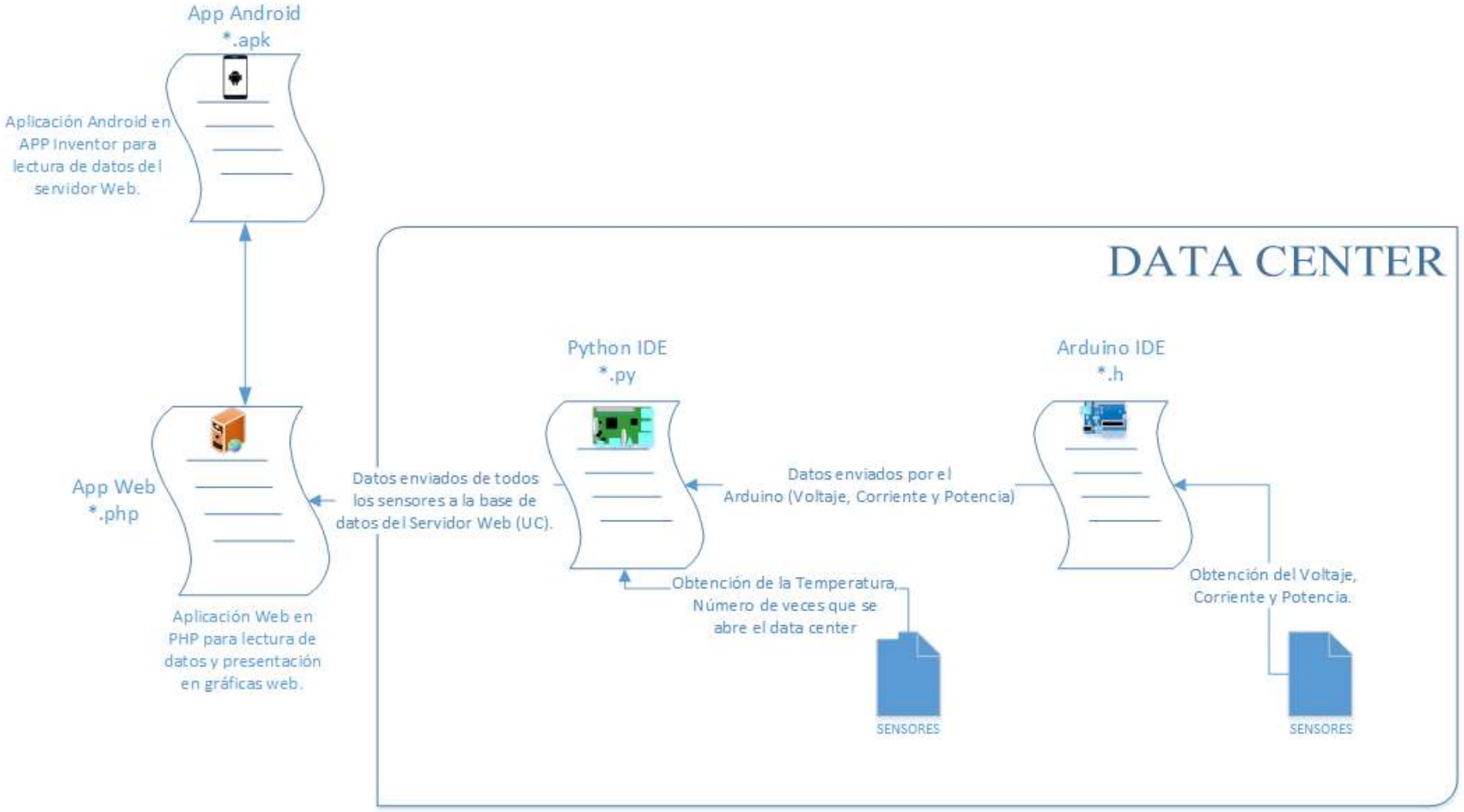


Figura 2.5. Arquitectura de Software.

2.3.1 PROGRAMACIÓN EN ARDUINO UNO

Toda la programación en Arduino se ha realizado utilizando el lenguaje C, para obtener los datos del sensor de corriente y voltaje. Además, se ha desarrollado una función para obtener la potencia mediante los datos de corriente y voltaje utilizando la ecuación 2.1. [37].

$$P = V\ x\ I \tag{2.1}$$

Donde:
P= Potencia (W)
V= Voltaje(V)
I= Corriente (A)

Adicional, se ha programado el microcontrolador en modo suspensión[11] y se activa mediante un caracter de lectura, cada 30 min se tiene las lecturas de voltaje y corriente de cada línea dentro del *Data Center* de la Universidad Central. El código se encuentra detallado en el Anexo B.

2.3.2 PROGRAMACIÓN EN RASPBERRY PI 3 B+

La programación en Raspberry PI 3 B+ se ha desarrollado utilizando el lenguaje de programación Python, este lenguaje se adapta correctamente a la minicomputadora y es muy fácil de entender y de codificar.

El programa realiza la lectura de los datos de los sensores de temperatura y de proximidad para obtener la cantidad de veces que se abre el Data Center. Además, este programa efectúa la lectura serial del Arduino Uno, lo que permite la obtención de datos de voltaje, corriente y potencia.

Estas lecturas se ejecutan cada 30 minutos para obtener un monitoreo más preciso y detallado y todos estos datos son enviados a una base de datos MySQL en el servidor web de la Universidad Central al mismo tiempo de la toma de datos. El código de este programa se encuentra en detalle en el Anexo B.

[11] Modo suspensión. - Programa en modo espera de ser activado mediante un agente externo, es decir, es un programa que no se ejecuta sin un activador.

2.3.3 PROGRAMACIÓN WEB PHP

Una vez almacenados los datos de lectura de todos los sensores del proyecto en la base de datos MySQL, se ha desarrollado un aplicativo Web utilizando el lenguaje de programación PHP, uno de los mejores lenguajes para el desarrollo en la nube. Este aplicativo se conecta a la base de datos para la lectura de los valores de voltaje, corriente, potencia, temperatura y número de veces que se abre la puerta del *PDU* del Data Center diariamente.

El desarrollo consiste en una conexión directa a la base de datos para consultar los valores medidos de voltaje, corriente, potencia, temperatura y número de veces que se abre la puerta del *PDU* del Data Center diariamente. Estos valores son presentados mediante gráficos lineales.

Para desplegar las gráficas se ha utilizado la librería CanvasJS, esta utiliza HTML5 [12]para el procesamiento de los gráficos y es 10 veces más rápido que las bibliotecas de gráficos basadas en *SVG* (*Scalable Vector Graphics*, Gráficos vectoriales escalables)[13].

El código fuente utilizado para este desarrollo se encuentra en el Anexo C.

A continuación, en la figura 2.6, se presenta la captura de pantalla del aplicativo *Responsive*, es decir, puede ser accedido mediante exploradores de Computadoras de escritorio, tablets, celulares y cualquier dispositivo que permita el acceso a portales web.

[12] HTML5.- Lenguaje de marcado (incorpora etiquetas) que se utiliza para presentar contenido en un explorador web

[13] Gráficos vectoriales escalables. - Formato de gráficos vectoriales bidimensionales.

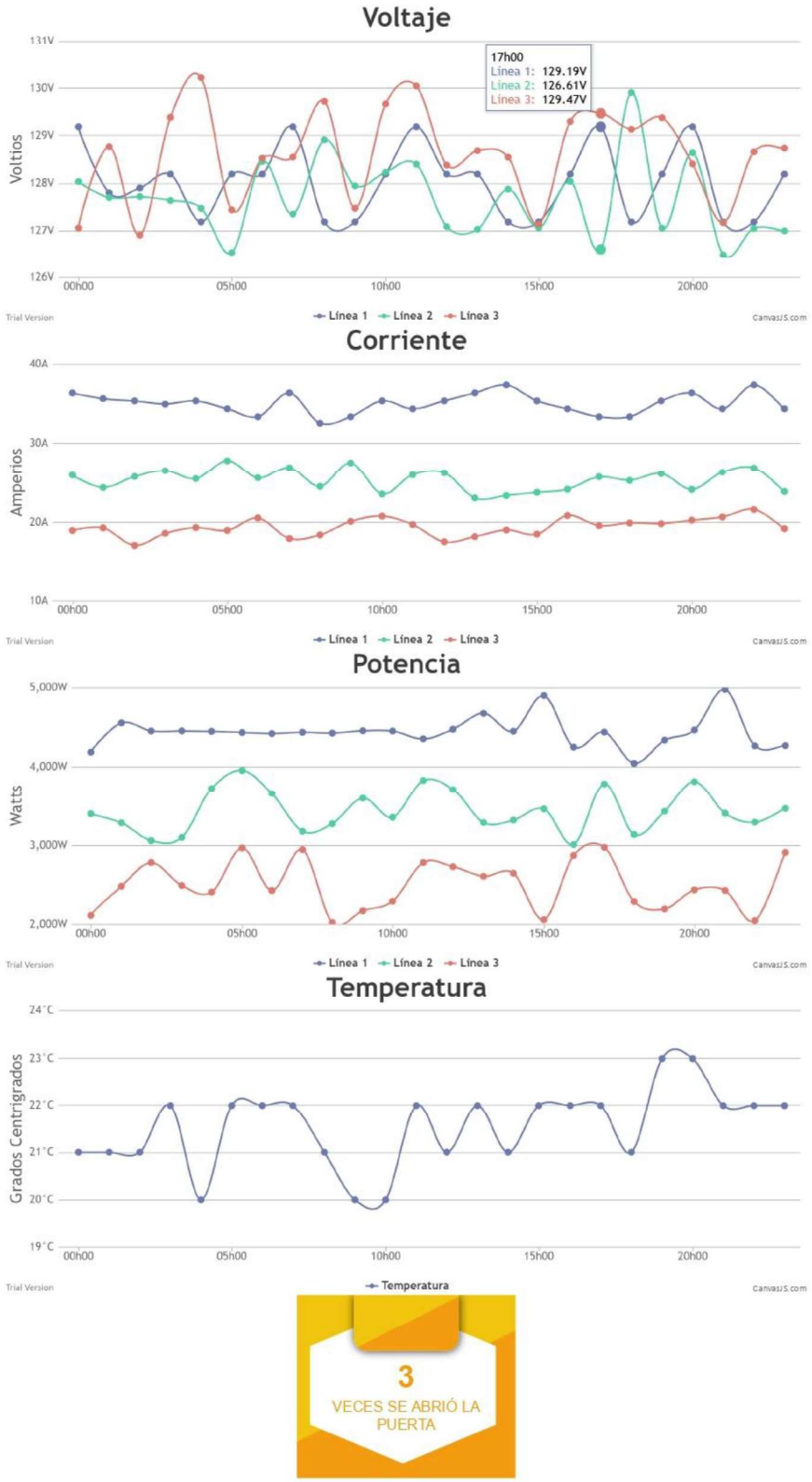


Figura 2.6. Aplicación Web del Proyecto

2.3.4 SERVIDOR WEB

El servidor web que se ha utilizado para el despliegue de la aplicación PHP, es un servidor con un sistema operativo Linux CentOS, el cual se encuentra en el Data Center de la Universidad Central del Ecuador. Esta distribución es de código abierto de Linux y se basa en *RHEL (Red Hat Enterprise Linux)*, que es considerado el más utilizado en el mundo corporativo de TI. Lo que hace que CentOS sea mucho más estable que otras distribuciones [37].

Instalación del Sistema Operativo CentOS:

A continuación, se muestra los comandos ejecutados para la instalación del sistema operativo CentOS, y los servidores de aplicación y base de datos. En la Figura 2.7 se puede apreciar la pantalla inicial de instalación básica de Centos 7 en donde se aloja la aplicación web y base de datos MariaDB.

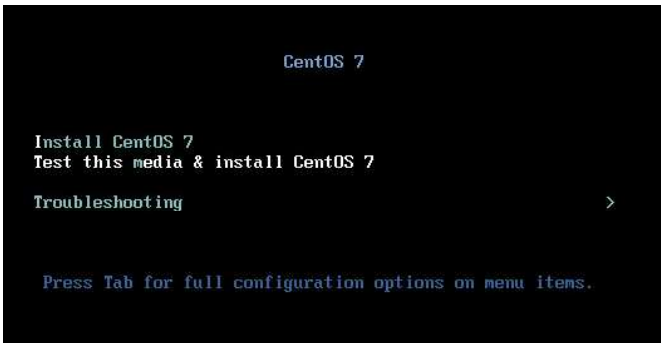


Figura 2.7. Pantalla de Bienvenida CentOS 7.

Posteriormente, en la Figura 2.8 se puede observar la selección del idioma para realizar el proceso de instalación, en este caso se ha seleccionado el idioma español para el sistema operativo CentOS.

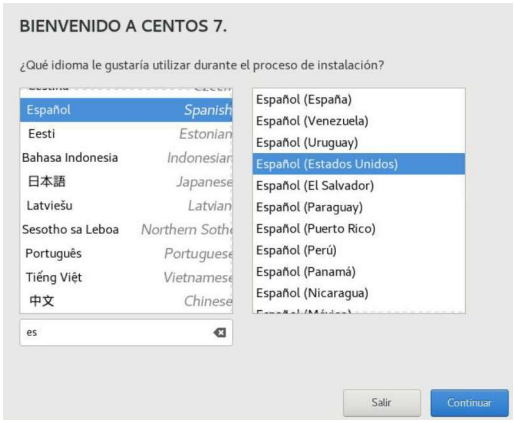


Figura 2.8. Selección del idioma.

El nombre de host del sistema CentOS se ha establecido como “tesisenergia” y la conexión a la red mediante cable Ethernet, esto se puede visualizar en la Figura 2.9.

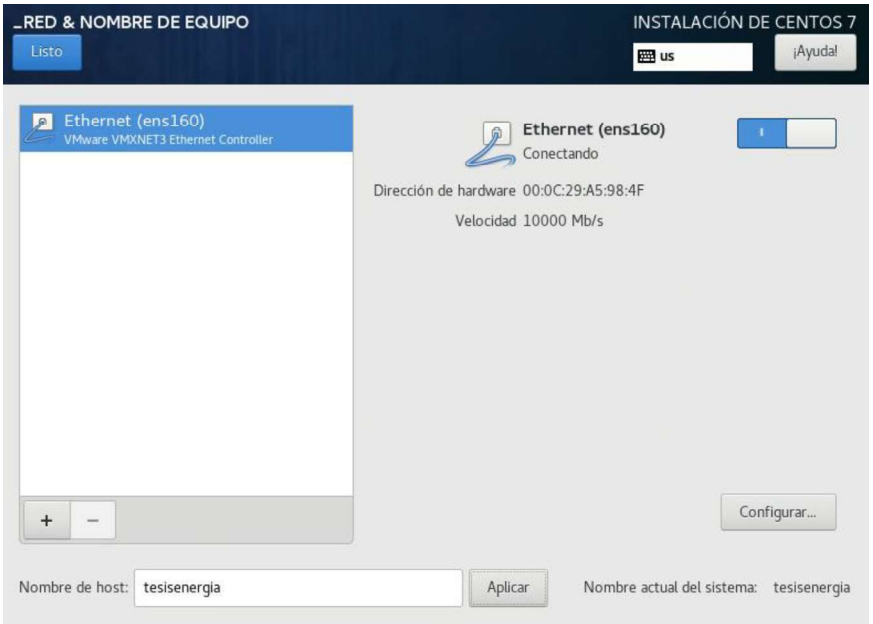


Figura 2.9. Configuración nombre de host y red.

Luego se selecciona el tipo de particionado para la instalación del sistema operativo CentOS, la Figura 2.10 muestra que para este proyecto se elige que el destino de la instalación sea por particionado automático, es decir que el instalador establece automáticamente el tamaño de cada partición de CentOS (swap, boot, raíz, y home).

Figura 2.10. Inicio de la instalación con particionado automático.

En la figura 2.11 se observa la configuración de usuarios, para este servidor no se ha creado ningún usuario y se ha establecido una contraseña fuerte para el super usuario de CentOS (root).

Figura 2.11. Asignación de contraseña al usuario root.

En la figura 2.12 se puede observar el proceso de instalación de sistema operativo CentOS.

Figura 2.12. Progreso de la instalación.

En la figura 2.13 se puede observar la instalación completa del sistema operativo CentOS, además, se observa la versión de este y del kernel.

Este servidor y los demás que se encuentran alojados en el Data Center se controlan mediante acceso remoto por el personal encargado del monitoreo del centro de datos. Es así, que la instalación del servidor Web se ha realizado mediante la terminal, en la Figura 2.13 se puede visualizar la descripción y versión del servidor.

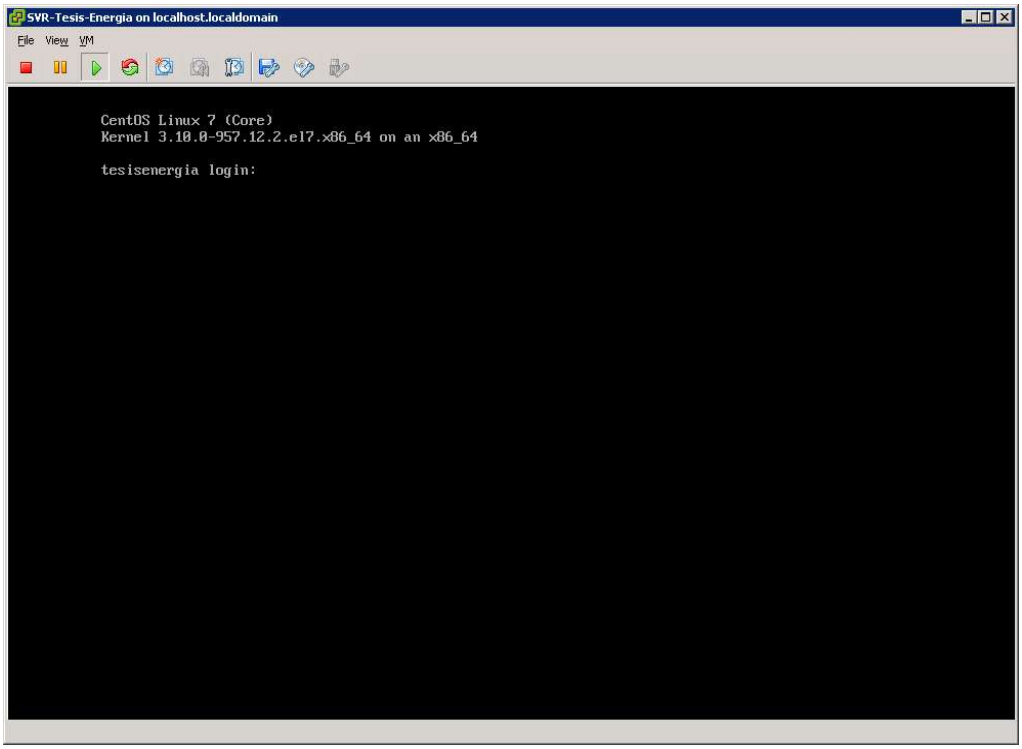


Figura 2.13. Sistema instalado, versión del Servidor web CentOS.

Una vez instalado el sistema operativo se procede a instalar los servidores de aplicación y de base de datos. Para ello se ha decidido utilizar la herramienta LAMP[14], la cual permite que la aplicación desarrollada en PHP se despliegue correctamente. A continuación, se muestra los pasos de instalación y comandos de los servidores.

[14] LAMP. - acrónimo de Linux, Apache, MySQL y PHP

Instalación de la Base de datos:

La instalación del motor de base de datos MariaDB versión 5 se realiza mediante el siguiente comando de la terminal de CentOS, este motor está basado en MySQL y tiene un mayor rendimiento y nuevas funcionalidades [39].

```
#yum -y install mariadb-server mariadb
```

Además, en este servidor se ha instalado y configurado la base de datos “tesis”. El script de creación de base de datos se muestra en el Anexo B.

Instalación del Servidor web apache:

La instalación del servidor Web Apache, se realiza mediante el siguiente comando, este servidor es de código abierto y es utilizado para el despliegue de aplicaciones web [39].

```
# yum -y install httpd
```

Se dejan los parámetros por defecto de la configuración del servidor web.

Instalación de PHP como intérprete y dependencias:

Para el correcto despliegue de la aplicación web desarrollada, se debe instalar el intérprete de PHP y todas las dependencias necesarias. Esto se realiza mediante el siguiente comando [39]:

```
# yum -y install php php-gd php-ldap php-odbc php-pear php-xml
php-xmlrpc php-mbstring php-soap curl curl-devel php-mysqlnd
php-pdo
```

Ejecución del servidor Web:

Una vez instalado el servidor web de apache, se ejecuta el siguiente comando para iniciar el servicio [39]:

```
#systemctl start httpd
```

Ejecución de MariaDB:

Además, se realiza el inicio del servicio del motor de base de datos, para esto se ejecuta el siguiente comando [39]:

```
#systemctl start mariadb
```

En la figura 2.14 se muestra el acceso al motor MariaDB y a la base de datos “tesis”.

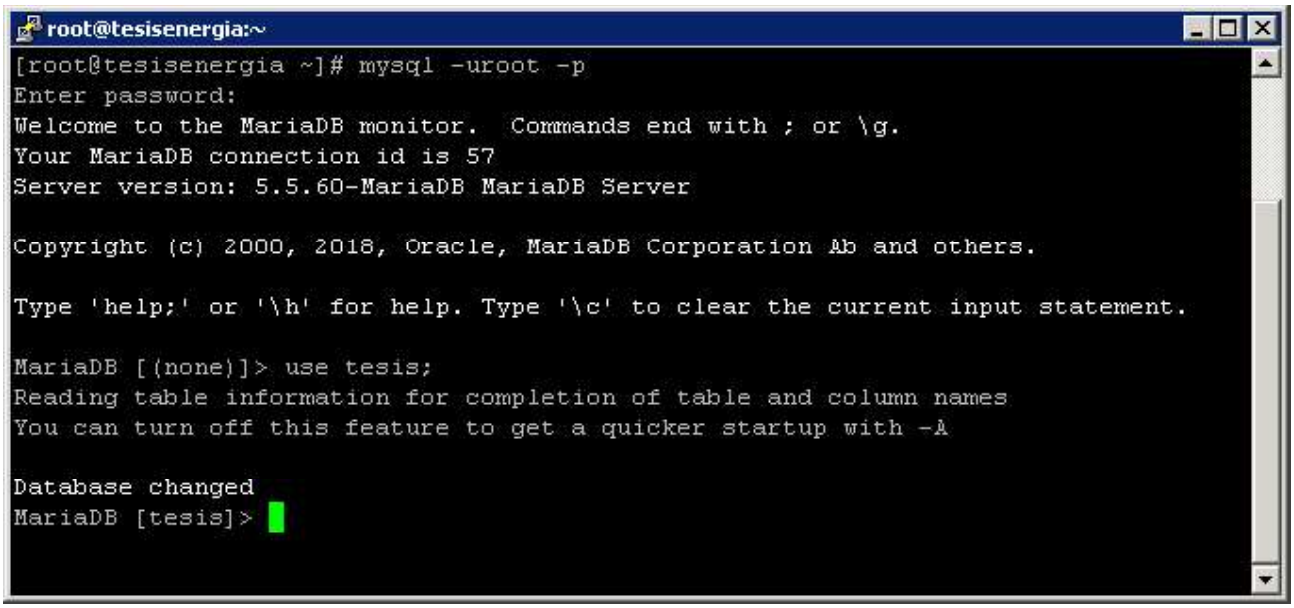


Figura 2.14. Acceso remoto al servidor y a la base de datos del sistema.

Acceso a la tarjeta Raspberry PI 3 B+:

Para acceder mediante acceso remoto al Raspberry Pi se ha utilizado la herramienta *VNC (Virtual Network Computing*, Computación Virtual), VNC es un programa para gestión remota de clientes mediante la estructura de cliente-servidor permitiendo el acceso remoto a un servidor por medio del cliente sin importar el sistema operativo del servidor con respecto al cliente, siempre y cuando el sistema operativo admita la utilización de VNC. [38].

2.3.5 PROGRAMACIÓN ANDROID CON APP INVENTOR

Para desarrollar la aplicación para teléfonos móviles se ha utilizado la herramienta APP Inventor del MIT. Esta herramienta permite la creación ágil de aplicaciones Android ya que su codificación es por bloques. En la figura 2.15 se presenta el diseño del prototipo que se va a diseñar dentro de la herramienta antes mencionada.

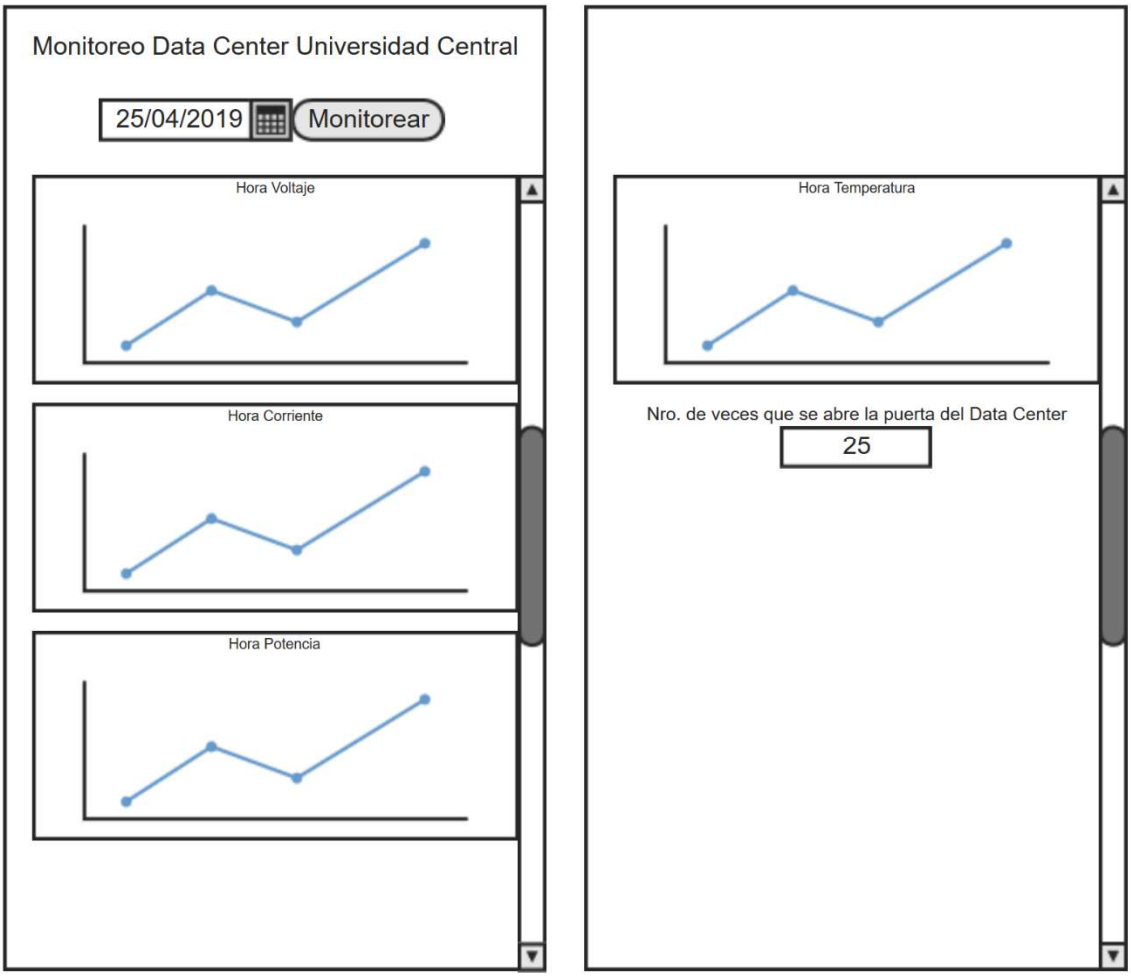


Figura 2.15. Diseño Prototipo Aplicación Android.

Esta aplicación se ha desarrollado con el fin de presentar los datos en una aplicación nativa de Android, la cual contiene la información de todos los sensores para un correcto monitoreo y control apropiado del *Data Center*.

3. IMPLEMETACIÓN

En este capítulo se presenta la implementación tanto del prototipo para medir la potencia, temperatura y el número de veces que se abre la puerta del *PDU* principal del centro de datos, así como la conexión inalámbrica entre el prototipo y el servidor situado en la nube.

Además, se detalla la implementación de la aplicación Android desarrollada, y su conexión al servidor de la Universidad Central.

3.1 IMPLEMENTACIÓN DEL PROTOTIPO

En esta sección se realiza el prototipo en una placa de cobre para la implementación en el *Data Center* de la Universidad Central del Ecuador.

3.1.1 IMPLEMENTACIÓN DEL PROTOTIPO EN PCB

A continuación, se presenta el diseño implementado con el programa Proteus del prototipo que se va a utilizar en el *Data Center*.

La Figura 3.1 detalla el prototipo implementado, el cual está separado por partes indicando sus diferentes conexiones, también se representan los sensores de corriente, voltaje, temperatura y el *Reed Switch*. Además, se ilustran las conexiones correspondientes a la minicomputadora Raspberry Pi y la tarjeta Arduino Uno.

Como se muestra en el diagrama del circuito, para la tarjeta Arduino Uno se tienen seis pines que representan las entradas analógicas. El puerto A0 es utilizado para el primer sensor de corriente, el puerto A1 es empleado para el primer sensor de voltaje, el puerto A2 es requerido para el segundo sensor de corriente, el puerto A3 se usa para el segundo sensor de voltaje, el puerto A4 se utiliza para el tercer sensor de corriente y el puerto A5 es usado para el tercer sensor de voltaje.

Para la minicomputadora Raspberry Pi se tiene 6 puertos que se conectan con el sensor de temperatura y el *Reed Switch,* el pin11 se conecta al sensor de temperatura y el pin 7 al *Reed switch*, El sensor de temperatura es alimentado con 3.3 V q vienen del pin 1 de la minicomputadora Raspberry Pi y el *Reed Switch* esta alimentado con 5 V entregados del

pin 2, y cada uno de los sensores que está conectado a la Raspberry Pi tiene su resistencia de pull-up de (4.7K – 10K) para que se complete el camino de la corriente y así evitar medidas erróneas de la señal enviada por los sensores.

En la figura 3.2 se muestra como está hecho el diseño de las conexiones en la placa de PBC y los diferentes sensores.

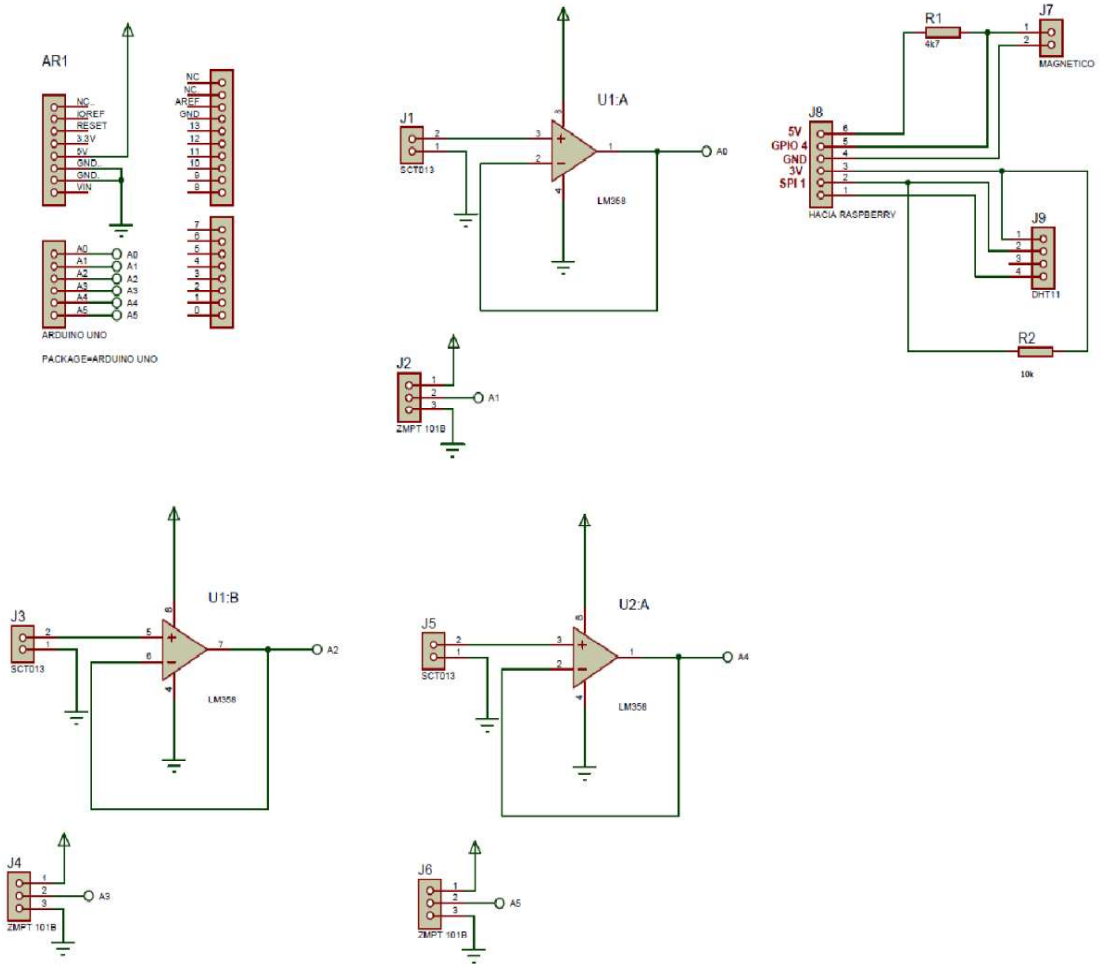


Figura 3.1. Prototipo Implementado en Proteus.

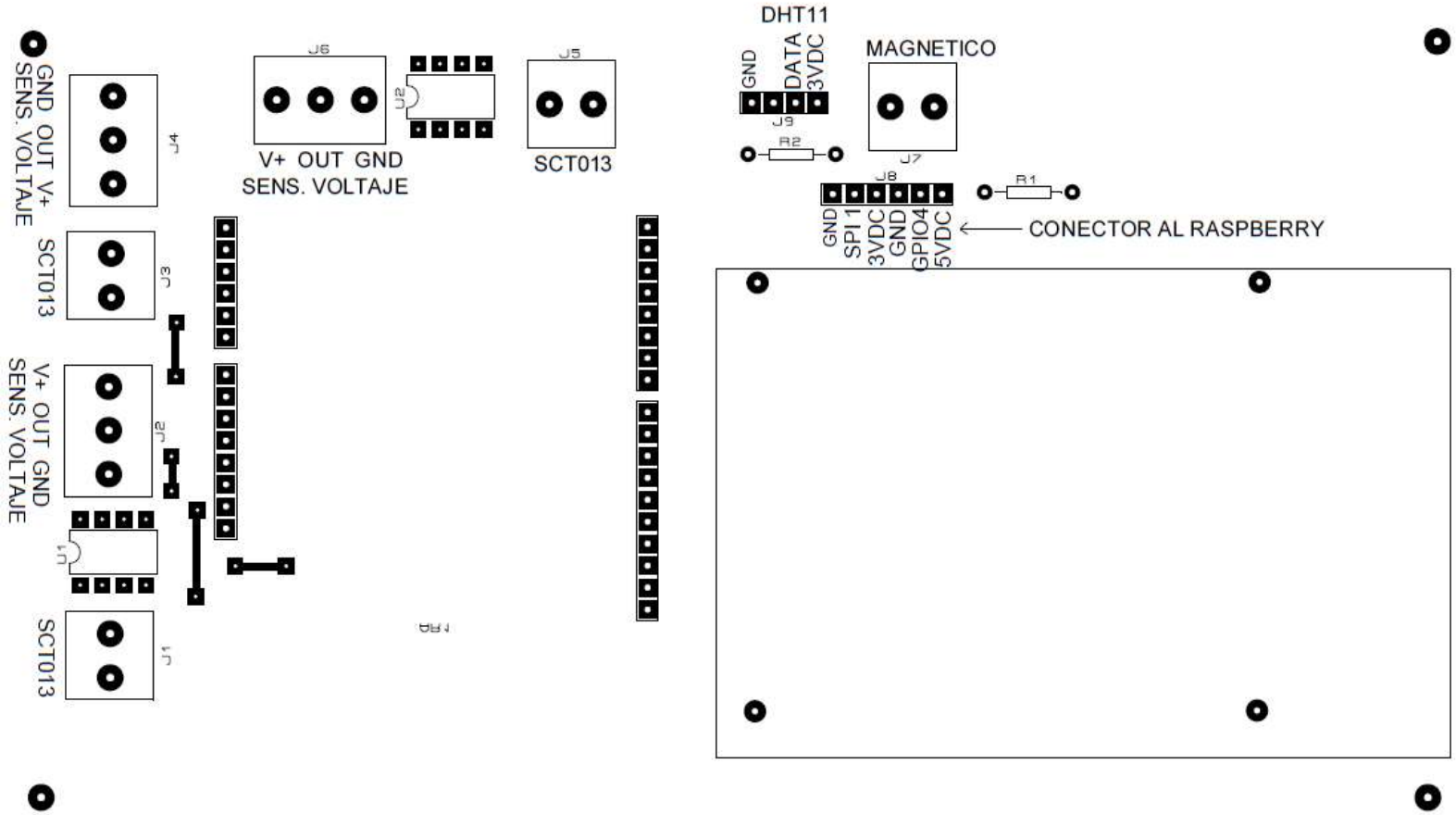


Figura 3.2 Esquema de conexión en la placa PCB.

En la Figura 3.3 se muestra el prototipo implementado, así como la conexión de todos los sensores y componentes.

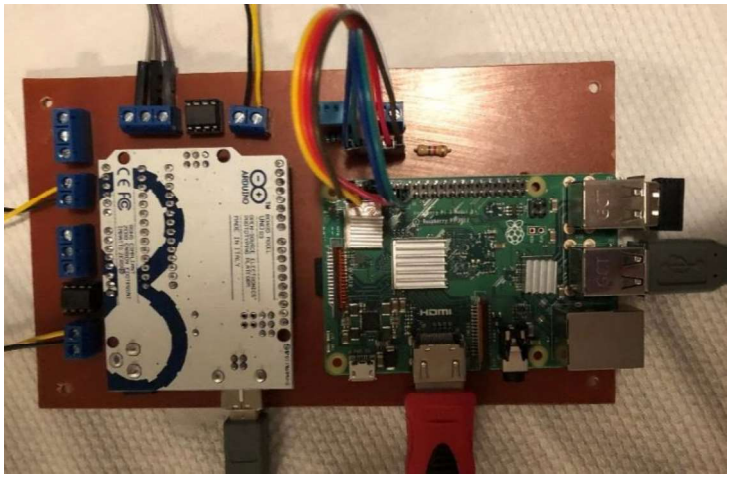


Figura 3.3. Prototipo Implementado.

La figura 3.3 muestra todas las conexiones del prototipo, este cuenta con tres sensores de corriente, tres sensores de voltaje, un sensor de temperatura y un *Reed Switch*.

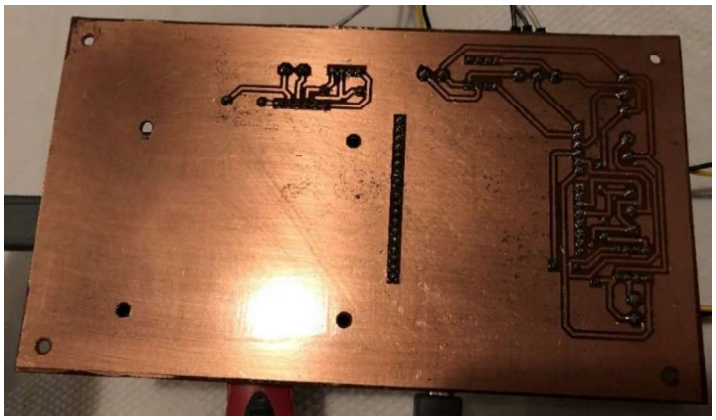


Figura 3.4. Diseño impreso en *PBC*.

En la figura 3.4 se muestra la parte posterior de la placa *PBC (Printed Circuit Board,* Placa de Circuito Impreso[15]) que fue impresa con las conexiones de los elementos del prototipo.

[15] Placa de Circuito Impreso. - es una superficie constituida con caminos o buses de material conductor laminadas sobre una base no conductora para circuitos eléctricos.

3.1.2 ARDUINO UNO.

Como se observa en la figura 3.1, el Arduino Uno tiene seis entradas analógicas para los tres sensores de voltaje y tres sensores de corriente, siendo estos conectados como se detalla a continuación:

1. Un sensor de corriente en el pin analógico A0.
2. Un sensor de voltaje está conectado en el pin A1
3. Un segundo sensor de corriente está conectado al pin A2
4. Un segundo sensor de voltaje será conectado al pin A3
5. El tercer sensor de corriente se ha conectado al pin A4
6. Finalmente, el tercer sensor de voltaje está conectado al pin A5.

También ha sido necesario un LM358 extra para las conexiones ya que los sensores de corriente requieren de amplificadores operacionales.

3.1.3 RASPBERRY PI 3 B +.

En la tarjeta Raspberry Pi se han conectado los siguientes sensores:

- El sensor de temperatura se encuentra conectado al pin GPIO 11 para obtener la información que entrega el sensor.
- El *Reed Switch* está conectado al GPIO 4 y se realiza un análisis de la conducción de energía dependiendo su estado.

Como se especifica en la sección 2.2, la alimentación del sensor de temperatura es de 3.3V y del sensor imantado es de 5V.

Además, se ha colocado los sensores de temperatura e imantado a una distancia de 20 metros aproximadamente ya que la placa *PBC* se encuentra instalada en el cuarto de telecomunicaciones en donde se puede discernir que líneas alimentan el *Data Center*.

Para almacenar los datos obtenidos por el prototipo implementado se ha utilizado un servidor de la Universidad Central del Ecuador, y se ha colocado la base de datos en este para realizar los procesos de envío y recepción de la información. A través de la aplicación en un dispositivo Android se puede visualizar y monitorear el *Data Center*.

3.1.4 BASE DE DATOS.

La base de datos ha sido diseñada, desarrollada e implementada utilizando el motor de base de datos MySQL, el modelo físico de la base de datos se muestra en la figura 3.2.

Figura 3.2. Diagrama físico de la base de datos.

Como se observa en la Figura 3.2 existen dos tablas para almacenar la información de los diferentes sensores, la tabla sensores generales almacena los datos que entregan los sensores de temperatura e imantado, la segunda tabla almacena el valor que entregan los sensores de voltaje, corriente y el valor de potencia obtenido del programa que se ejecuta en el Arduino Uno.

3.2 IMPLEMENTACIÓN DE LA APLICACIÓN ANDROID

En esta sección se detalla el desarrollo e implementación de la aplicación para dispositivos Android que se ha utilizado para el presente proyecto.

Una vez diseñado el prototipo que se muestra en la Figura 3.1 se realiza el diseño de la interfaz de usuario de la aplicación para dispositivos Android utilizando la herramienta App Inventor, como se observa en la Figura 3.3. Adicionalmente, se agregó la funcionalidad para tener un historial de monitoreo por día. La aplicación está conformada por dos botones, el primer botón es usado para seleccionar el día que se requiera consultar mediante un calendario de fechas, el segundo botón es utilizado para visualizar la

información de la fecha escogida. Finalmente, el tercer componente es utilizado para conectarse al servidor web.

Figura 3.3. Diseño de la interfaz de usuario con App Inventor.

A continuación, se presenta en la figura la programación realizada mediante bloques en la aplicación App inventor.

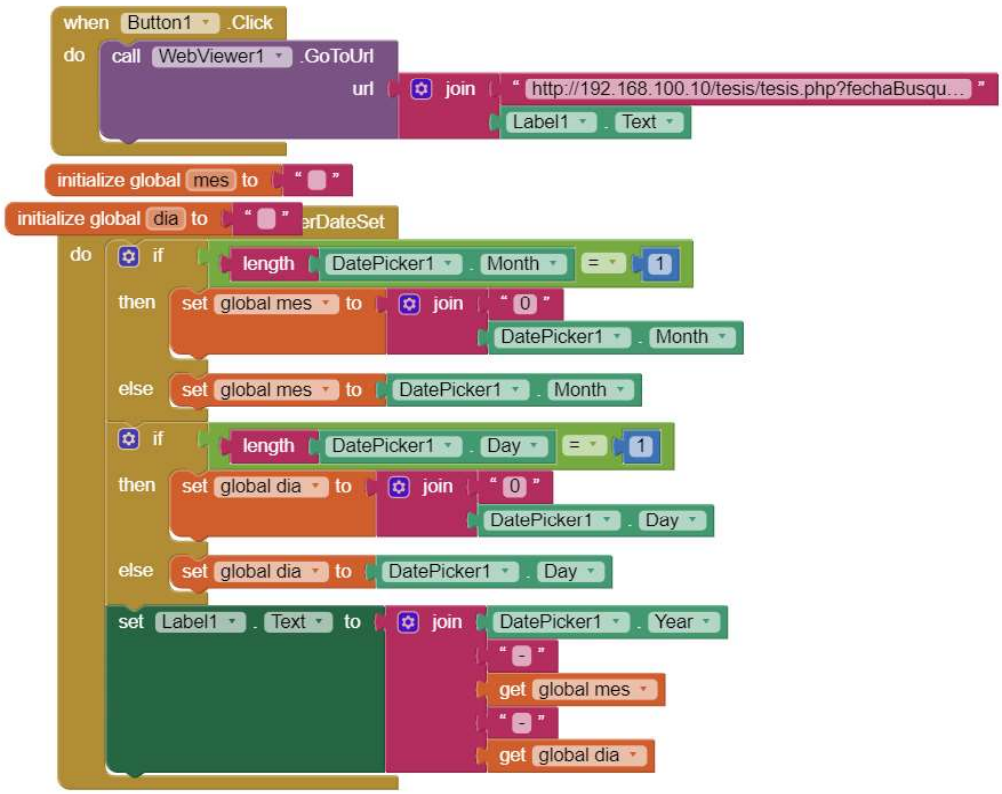


Figura 3.4. Diagrama de bloques de la aplicación de APP inventor.

Posteriormente en las Figuras 3.5, 3.6, 3.7 y 3.8 se muestra la aplicación instalada en un dispositivo Android.

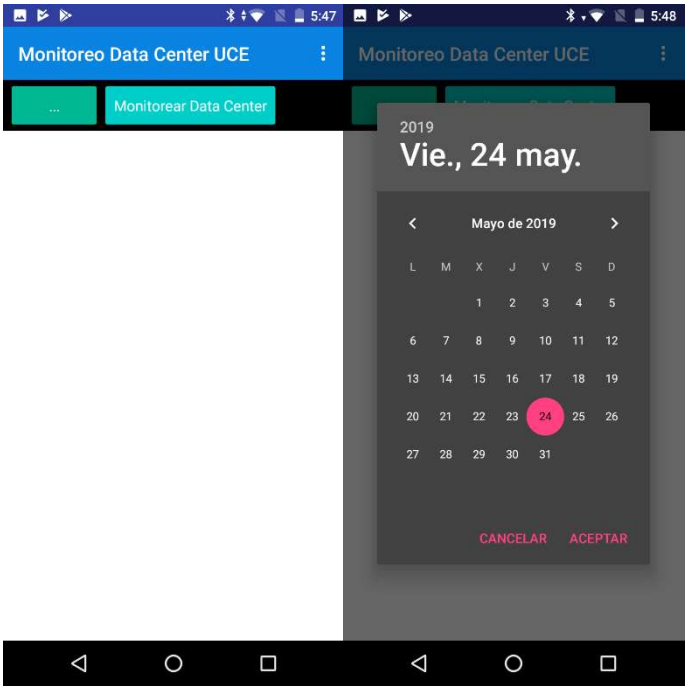


Figura 3.5. Captura de pantalla de la aplicación mostrando el calendario para escoger una fecha.

Figura 3.6. Captura de pantalla de la aplicación mostrando las gráficas de voltaje y corriente de la línea 1, 2 y3 a lo largo de un día.

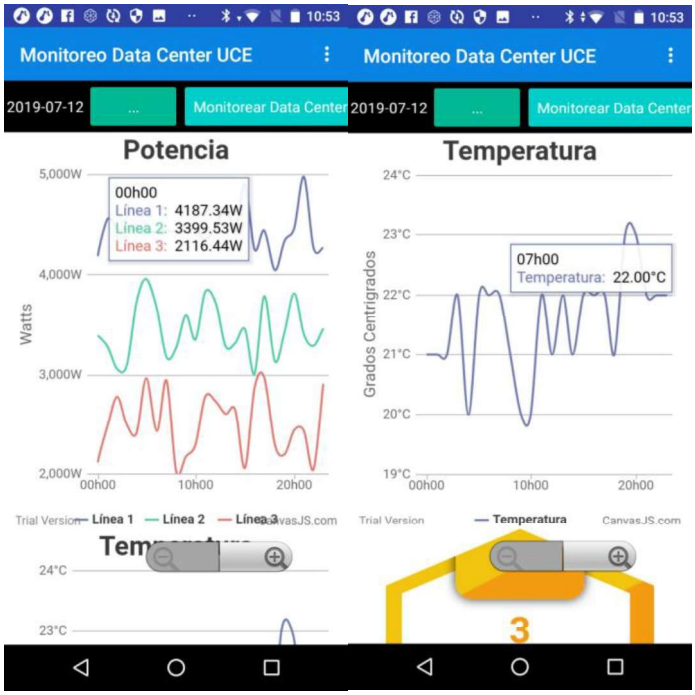


Figura 3.7. Captura de pantalla de la aplicación mostrando las gráficas de la potencia consumida y la temperatura medida a lo largo de un día.

Figura 3.8. Captura de pantalla de la aplicación mostrando el conteo de las veces que la puerta del UPC principal es abierta.

Como se muestra en las figuras anteriores todos los datos de los sensores están representados por gráficas que muestran las variaciones de voltaje, corriente, potencia y temperatura en el transcurso del día, además se muestra el pico máximo y mínimo en dichas variaciones.

Adicionalmente, con estos gráficos se puede determinar el consumo de potencia más alto en el día para un mejor monitoreo del *Data Center*.

También se realiza un control del número de veces que se abre la puerta del *PDU* principal del *Data Center* para evitar fallos en el sistema y como medida de seguridad, ya que se debe tener un control de las personas autorizadas para ingresar al *Data Center*.

3.3 IMPLEMENTACIÓN DEL PROTOTIPO EN EL *DATA CENTER*.

A continuación, se muestra la instalación del prototipo en el Data Center de la Universidad Central del Ecuador.

Los materiales utilizados son:

- 3 *breakers*.
- Cable flexible número 14.
- Cable UTP.
- Taipe.
- Cautín y estaño.
- Caja para colocar el prototipo.
- Cinta doble faz.
- Prototipo con todos los sensores necesarios.
- Tomacorriente.

En la Figura 3.9 se observa el prototipo colocado en la caja con las conexiones a los diferentes sensores necesarios para medir corriente, voltaje, temperatura y el número de veces que la puerta del *PDU* se abre.

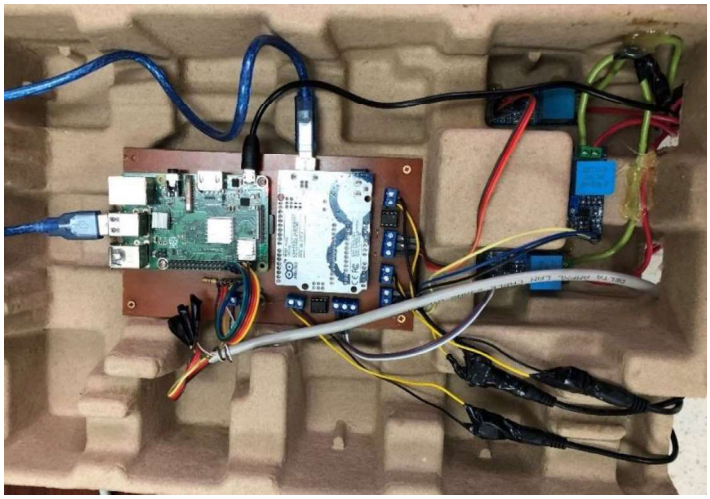


Figura 3.9. Prototipo conectado a los diferentes sensores y colocado en la caja.

En la Figura 3.10 se observa la ubicación del prototipo en una caja negra dentro del *PDU*, además, este está asegurado con cinta doble faz. Existen tres líneas de alimentación para el *Data Center:* "línea A" o 1, "línea B" o 2 y "línea C" o 3. Los sensores de voltaje y corriente están conectados a estas líneas, las cuales se han extendido mediante el uso de *breakers*.

Asimismo, se puede observar el sensor de temperatura colocado de manera que pueda medir la temperatura en el *Data Center* y el sensor para monitorear la apertura de la puerta en el *PDU* principal.

Figura 3.10. Prototipo implementado con los sensores conectados a las líneas que alimentan el *Data Center*.

La Figura 3.10 muestra la conexión de los sensores de voltaje mediante cables rojos representados como las fases y un cable verde que representa el neutro, el cual se encuentra puenteado entre los 3 sensores de voltaje. De igual manera los sensores de corriente están colocados directamente en las 3 líneas que alimentan al *Data Center*.

Se utiliza un cable UTP para conectar el sensor de temperatura y el *Reed Switch*, estos se ubican en la parte superior de la puerta como se observa en la Figura 3.12, los cables de estos sensores se han empalmado en el cable UTP.

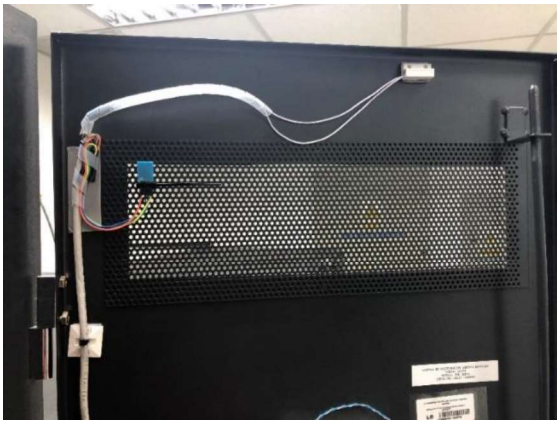


Figura 3.11. Sensores de Temperatura y *Reed Switch*.

La Figura 3.12 ilustra los diferentes sensores de voltaje, estos se han colocado mediante el uso de *breakers* para poder medir las líneas de alimentación. Además, se utiliza un tomacorriente para la alimentación de la tarjeta Raspberry Pi al lado izquierdo de la puerta.

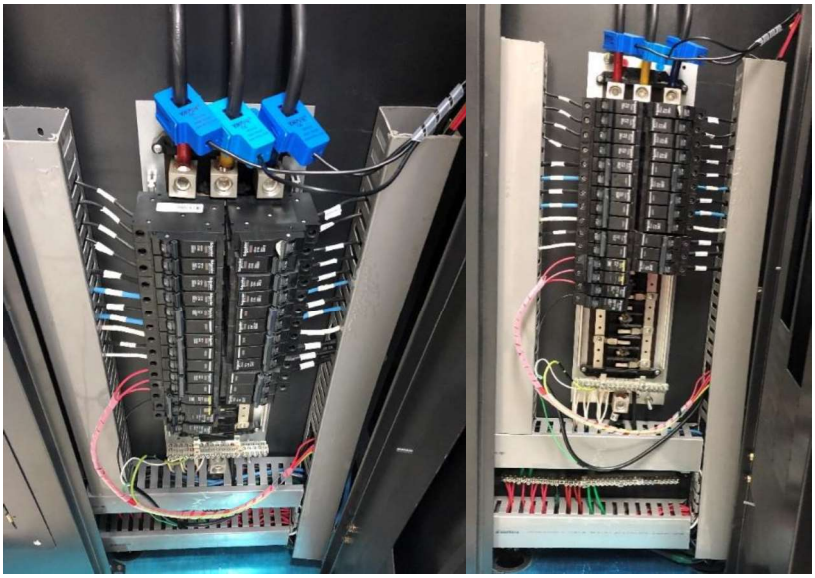


Figura 3.12. Conexión de los sensores de Corriente y los sensores de voltaje.

4. RESULTADOS Y DISCUSIÓN

En este capítulo se analizan los resultados obtenidos de la implementación del prototipo.

Luego de realizar la correcta implementación del prototipo, se ha obtenido los siguientes valores de corriente, voltaje, potencia, temperatura y el número de veces que se abre la puerta del UPC. Este análisis se realiza para el mes de junio de 2019.

4.1 VOLTAJE

Fecha de inicio: 01 de junio del 2019

Fecha de fin: 30 de junio del 2019

Valores máximos:

- **Por día:**
    - Línea 1: 129.19 V
    - Línea 2: 129.91 V
    - Línea 3: 130.22 V
- **Por mes:**
    - Línea 1:130.22 V
    - Línea 2: 131.11 V
    - Línea 3: 130.97 V

Valores mínimos:

- **Por día:**
    - Línea 1: 127.15 V
    - Línea 2: 126.49 V
    - Línea 3: 126.91 V

- **Por mes:**
    - Línea 1: 126.68 V
    - Línea 2: 125.79 V
    - Línea 3: 125.97 V

Media por día:

- Línea 1: 128.0371 V
- Línea 2: 127.6789 V
- Línea 3: 128.6188V

Media por mes:

- Línea 1: 128.9978 V
- Línea 2: 128.7564 V
- Línea 3: 129,2527V

A continuación, los gráficos 4.1, 4.2 y 4.3 muestran el consumo de voltaje en el mes de junio de 2019.

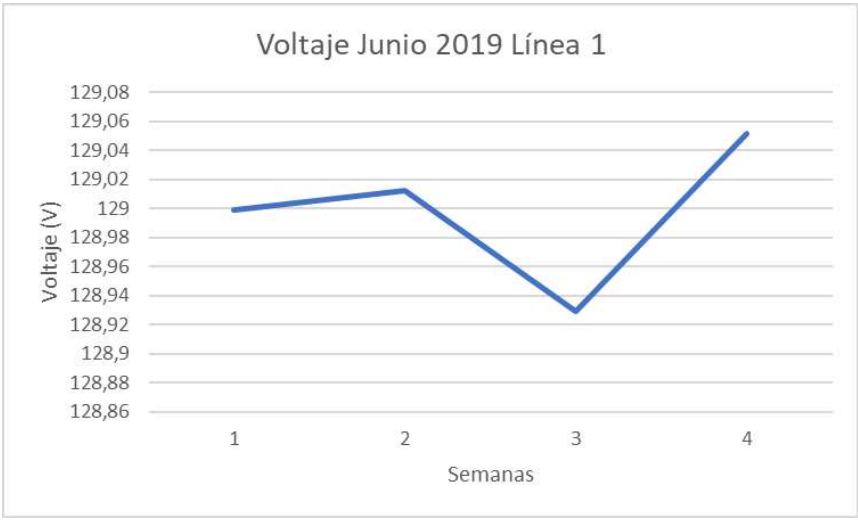


Figura 4.1. Línea 1 gráfico del mes del voltaje.

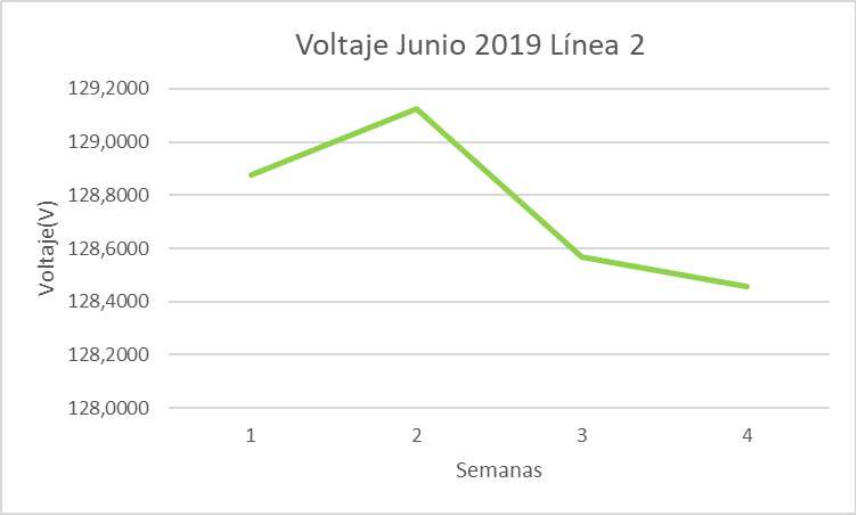


Figura 4.2. Línea 2 gráfico del mes del voltaje.

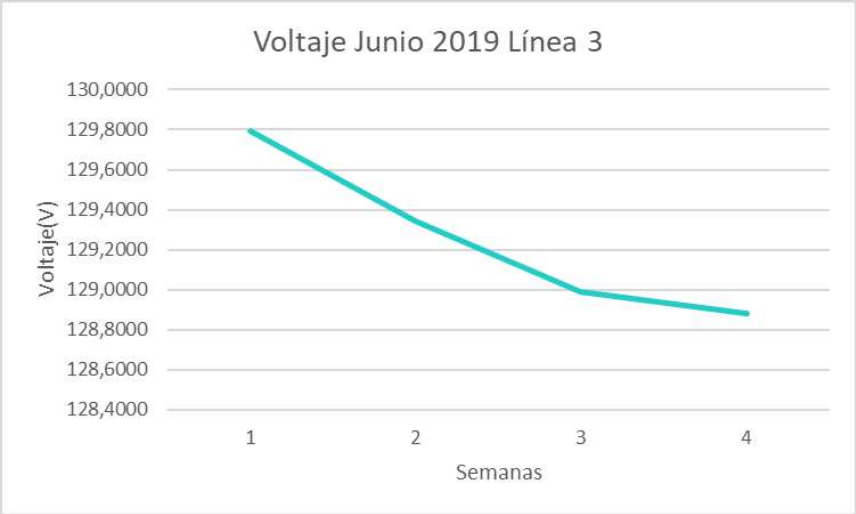


Figura 4.3. Línea 3 gráfico del mes del voltaje.

Los valores de voltaje medidos en el mes de junio de 2019 se encuentran dentro de un rango estable y no existen variaciones muy notorias, por lo que se considera que en el mes de junio no existieron fallas de voltaje en los equipos ni mantenimientos dentro del *Data Center*.

4.2 CORRIENTE

Fecha de inicio: 01 de junio del 2019

Fecha de fin: 30 de junio del 2019

Valores máximos:

- **Por día:**
    - Línea 1: 37.38 A
    - Línea 2: 27.82 A
    - Línea 3: 21.61 A
- **Por mes:**
    - Línea 1: 38.56 A
    - Línea 2: 28.37 A
    - Línea 3: 22.32 A

Valores mínimos:

- **Por día:**
    - Línea 1: 32.56 A
    - Línea 2: 23.05 A
    - Línea 3: 17.11 A
- **Por mes:**
    - Línea 1: 31.76 A
    - Línea 2: 22.67 A
    - Línea 3: 16.19 A

Media por día:

- Línea 1: 35.0491 A
- Línea 2: 25.3612 A
- Línea 3: 19.3811 A

Media por mes:

- Línea 1: 36.1170 A
- Línea 2: 26.2811 A
- Línea 3: 20.1398 A

A continuación, los gráficos 4.4, 4.5 y 4.6 muestran el consumo de corriente en el mes de junio de 2019.

Figura 4.4. Línea 1 gráfico del mes de junio de la corriente.

Figura 4.5. Línea 2 gráfico del mes de junio de la corriente.

Figura 4.6. Línea 3 gráfico del mes de junio de la corriente.

Los valores de corriente medidos en el mes de junio de 2019 se encuentran dentro de un rango estable y no existen variaciones muy notorias, por lo que se considera que en el mes de junio no existieron fallas de corriente en los equipos ni mantenimientos dentro del *Data Center*.

En las figuras 4.4, 4.5 y 4.6, se muestran las mediciones de corriente con diferentes valores porque cada línea alimenta a equipos distintos. La línea 1 es la que tiene la mayor carga de corriente con un valor medio de 36.1170 amperios en el mes de junio de 2019.

4.3 POTENCIA

Fecha de inicio: 01 de junio del 2019

Fecha de fin: 30 de junio del 2019

Valores máximos:

- **Por día:**
    - Línea 1: 4978.97 W
    - Línea 2: 3958.53 W
    - Línea 3: 2978.17 W
- **Por mes:**
    - Línea 1: 5032.65 W
    - Línea 2: 4045.68 W
    - Línea 3: 3893,18 W

Valores mínimos:

- **Por día:**
    - Línea 1: 4047.47 W
    - Línea 2: 3012.13 W
    - Línea 3: 2016.56 W
- **Por mes:**
    - Línea 1: 3985.32 W
    - Línea 2: 2934.56 W
    - Línea 3: 1934.56 W

Media por día:

- Línea 1: 4445.4291 W
- Línea 2: 3440,2955 W
- Línea 3: 2506,3804 W

Media por mes:

- Línea 1: 4577.2834 W
- Línea 2: 3579.0565 W
- Línea 3: 2651.9508W

A continuación, los gráficos 4.7, 4.8 y 4.9 muestran el consumo de potencia en el mes de junio de 2019.

Figura 4.7. Línea 1 gráfico del mes de junio de la potencia.

Figura 4.8. Línea 2 gráfico del mes de junio de la potencia.

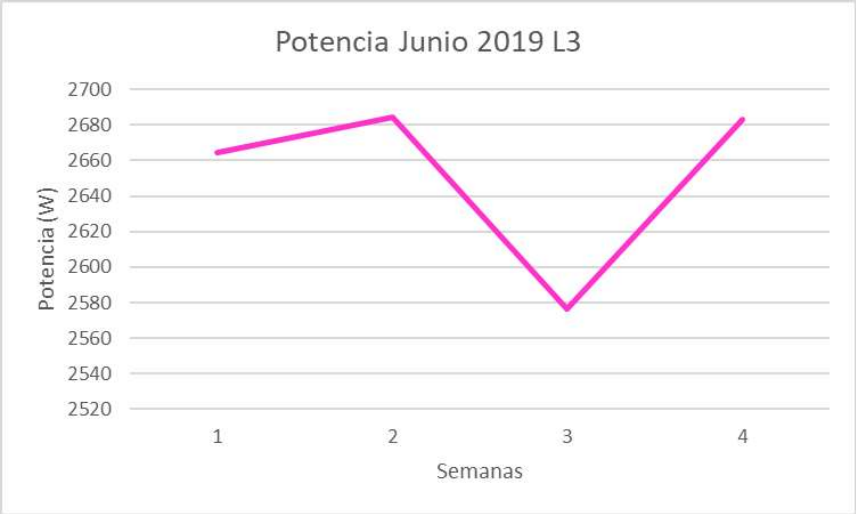


Figura 4.9. Línea 3 gráfico del mes de Potencia.

Los valores de potencia medidos en el mes de junio de 2019 se encuentran dentro de un rango estable y no existen variaciones muy notorias, por lo que se considera que en el mes de junio no existieron fallas de potencia en los equipos ni mantenimientos dentro del *Data Center*.

Dado que la corriente consumida es diferente en las 3 líneas que alimentan al *Data Center* se tienen diferentes valores de Potencia consumida para cada línea, el mayor valor de potencia consumida es para la línea 1 con un valor de 4577.2834 W.

La fluctuación de los valores de potencia puede ser debido a que existe doble alimentación del *Data Center* en caso de que uno de estos falle, también pueden variar los valores de

potencia por mantenimiento de los equipos y existen variaciones en la potencia consumida en el *Data Center.*

4.4 TEMPERATURA

Fecha de inicio: 01 de junio del 2019

Fecha de fin: 30 de junio del 2019

Valores máximos:

Por día: 23 °C

Por mes: 22 °C

Valores mínimos:

Por día: 20 °C

Por mes: 19 °C

Media por día: 21.5 °C

Media por mes: 21 °C

A continuación, el gráfico 4.10 muestra la temperatura en el *Data Center* en el mes de junio de 2019.

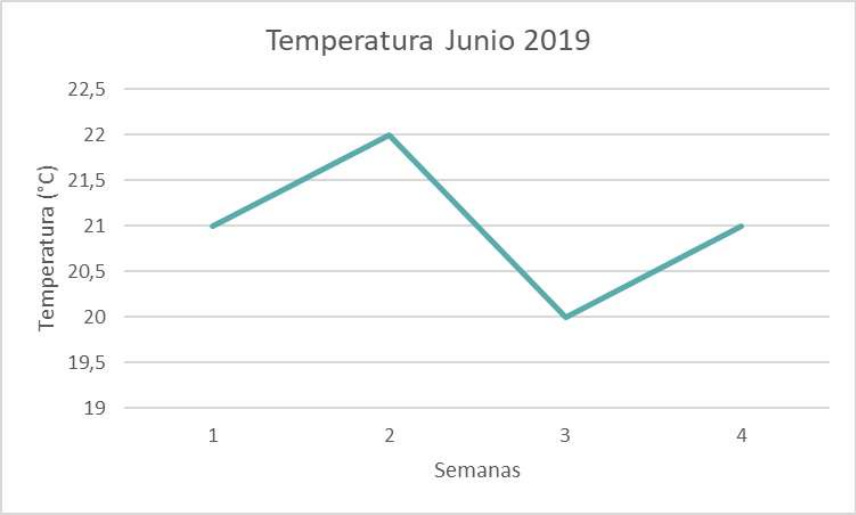


Figura 4.10. Gráfico de la temperatura del mes de junio.

Por lo que se observa en la Figura 4.10, el valor de la temperatura se mantiene en un rango de mediciones es de 19-22 grados centígrados entonces se considera que no hay errores de medida. También se considera que el *Data Center* mantiene la temperatura de acuerdo con los requerimientos de éste ya que como se conoce el rango de temperatura óptimo al cual debe operar un Data Center varía entre 15 y 25 grados centígrados [2].

4.5 CONTEO PARA CONTROLAR LA APERTURA Y CIERRE DEL PDU PRINCIPAL

Fecha de inicio: 01 de junio del 2019

Fecha de fin: 30 de junio del 2019

Valores máximos:

Por día: 5 veces.

Por mes: 7 veces.

Valores mínimos:

Por día: 0 veces.

Por mes: 0 veces.

Media por día: 1.16 veces.

Media por mes: 1.91 veces.

A continuación, el gráfico 4.11 muestra el número de veces que la puerta del *PDU* del *Data Center* en el mes de junio de 2019.

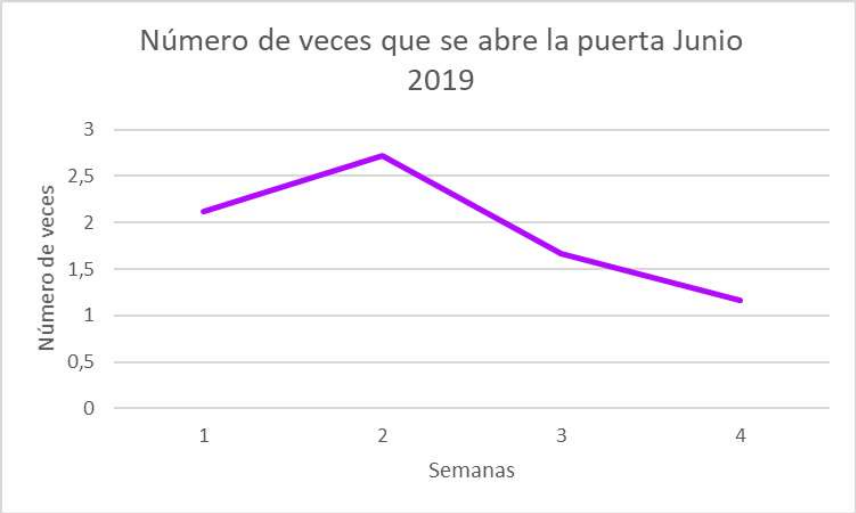


Figura 4.11. Gráfico del número de veces que se abre la puerta en el mes de junio.

Como se puede observar en la Figura 4.11 la puerta del *PDU* principal no se abre con frecuencia por motivos de seguridad ya que aquí se encuentran las 3 líneas que alimentan al *Data Center*.

5. CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

Los Data Centers son sistemas importantes para cualquier tipo de organización, ya que de ellos dependen los sistemas de información dentro de la entidad. Es por ello por lo que un correcto monitoreo de estos disminuye el riesgo de seguridad de los sistemas, además que permiten tener sistemas mucho más confiables y disponibles, y a una organización con alta comunicación.

Con los datos reunidos de la potencia consumida fue posible identificar las horas en las que existió un consumo de potencia alto con base en el monitoreo realizado. Además, el monitoreo de la temperatura ayudó al personal encargado del Data Center a tener un mejor control del funcionamiento de los sistemas de enfriamiento y considerar posibles mejoras en su mantenimiento.

La potencia se calculó a través de la programación en la tarjeta Arduino Uno, esta se obtuvo multiplicando la corriente por el voltaje de cada línea que entra en la alimentación del Data Center obtenida mediante la lectura de los sensores. Estos y los datos de temperatura y número de veces que se abre la puerta del *PDU*, se almacenan cada media hora en la base de datos ubicada en el servidor web, para posteriormente, ser visualizados en la aplicación para dispositivos Android.

La aplicación Android está diseñada con la ayuda de la herramienta APP Inventor del MIT lo que permitió un desarrollo eficaz y eficiente. Además, se tiene un monitoreo del Data Center en tiempo real y desde cualquier lugar dentro del Campus de la Universidad Central del Ecuador.

Se utilizó un servidor con sistema operativo CentOS para la administración del aplicativo web y la base de datos, el cual permite la centralidad de los datos y la aplicación para ser accedidos dentro del campus de la universidad. Además, la administración y configuración del servidor se realiza mediante acceso remoto para facilidad de mantenimiento de este.

Lo más complejo con respecto a la implementación del prototipo y la comunicación inalámbrica ha sido la conexión con la base de datos ya que se deben tomar en cuenta las diferentes versiones del lenguaje que se utiliza para la programación de la conexión que

en este caso ha sido Python 3, por lo que se requiere usar los comandos necesarios e indicados para las distintas versiones.

El servidor utilizado debe aceptar las diferentes características de la base de datos como en este caso utf-8, que es un conjunto de caracteres utilizado para intercambiar datos con la base de datos de MySQL.

Se decidió el uso de la red interna de la Universidad Central del Ecuador ya que es mucho más factible para los Ingenieros encargados del mantenimiento y monitoreo de este, también se ha escogido este método para el envió de la información ya que solo se requiere el monitoreo dentro del campus de la UCE y con el uso de red LAN interna del campus la rapidez será mayor.

Se ha utilizado un servidor interno de la UCE ya que por medio de éste se tendrá más capacidad en el almacenamiento de la información y sin costos extras a lo largo del monitoreo por parte del prototipo implementado.

5.2 RECOMENDACIONES

El uso de la tarjeta Arduino Uno ayuda a las mediciones de señales analógicas ya que la minicomputadora Raspberry Pi 3 B+ no posee pines de entrada analógicos, para otros sensores es recomendable el uso de la misma Raspberry Pi ya que no se requiere el script de envío de la información.

El aplicativo web desarrollado se realizó con una arquitectura *backend*[16] en el lenguaje de programación PHP y *frontend*[17] utilizando html5 y css3, lo que permite tener una aplicación *Responsive* que permite la visualización de las gráficas en cualquier dispositivo móvil, Tablet, computador, etc. Se recomienda el uso de html5 para que exista un fácil acceso al aplicativo, para que sea más amigable para el usuario y que la visualización sea la mejor independiente del dispositivo en el que se despliegue la información [39].

Las mediciones realizadas de voltaje deben ser respecto a la fase y el neutro en cada línea del sistema trifásico para evitar inconvenientes con los valores de voltaje ya que puede leerse un valor erróneo. De igual manera la lectura de la corriente solo debe ser tomada en

[16] Backend. - Lenguaje de programación utilizado para las transacciones realizadas con la base de datos.

[17] Frontend. - Lenguaje de programación utilizado para realizar la presentación del programa con respecto al usuario.

la fase ya que de otro modo se anula la medida, por lo tanto, se debe abrazar con la pinza amperométrica solo al cable de fase.

Para el uso de servidores y administración de la base de datos en empresas de gran escala es recomendable el uso de servidores Linux, como en este caso se utilizan servidores en el sistema operativo CentOS server, para hacer un mejor uso de los recursos que este tiene [39].

Sería recomendable hacer un balance de las cargas de corriente puesto que las tres líneas tienen diferentes valores de corriente, para así controlar de mejor manera la fluctuación de corriente mejorando la eficiencia del *Data Center*.

Para resolver el problema del consumo de potencia elevado en ciertas horas pico, existen varias estrategias que disminuyen la carga consumida por el enfriamiento del *Data Center,* a continuación, se mencionan 4 opciones de mejorar el problema. La estrategia del pasillo caliente y pasillo frio que consiste en que el escape que tiene cada rack de aire caliente y las tomas de aire frio se enfrenten en una línea de racks de servidores. Otra alternativa es la contención que consisten en encerrar a los racks en estructuras para que no se mezcle el aire frio con el aire caliente ya que si se hace uso de la contención de pasillo frio, el aire caliente va a retornar en los racks con temperaturas de 26ºC y 28ºC y de igual manera cuando se utiliza la contención de aire caliente el aire retorna directamente a los equipos con temperaturas de 35ºC o 40ºC. Los beneficios que se obtiene al utilizar estas opciones de enfriamiento son más eficiencia en el enfriamiento al evitar que se entremezclen las corrientes de aire, mayor fiabilidad ya que evitar que los equipos se expongan a temperaturas altas del aire caliente y finalmente menos gasto de energía y por ende de los recursos por evitar el efecto de recirculación del escape del aire ya que los sistemas de enfriamiento lo enfrían a 12.78ºC pero los sistemas basados en la contención aíslan el aire para que este retorne y sea entregado a 18,34ºC y por consiguiente el consumo de energía se reduce en un 16% de promedio aproximadamente [39].

El uso de ventiladores de frecuencia variable también ayuda con la disminución del gasto de la energía en el *Data Center* ya que el uso de estos permite disminuir la velocidad de los ventiladores y por ende el consumo de potencia. La optimización del proceso de refrigeración es del 5% y esto ayuda a reducir el 1% de los costos totales [39].

El sistema de enfriamiento inteligente es una opción muy adecuada ya que el *Data Center* tiene diferentes sistemas de mayor o menor magnitud y capacidad, por lo que sería ideal un sistema que pueda analizar, prever y ajustar la refrigeración para reducir el consumo de energía y los costos [40].

6. REFERENCIAS BIBLIOGRÁFICAS

[1] K. G. Kamble, «U.S. Patent Application No. 10/193,889.,» 2019.

[2] C. Spera, «Las claves en la administración de energía del Data Center,» *Logicalis Now,* pp. 13-14, 2012.

[3] Guo, Chuanxiong and Lu, Guohan and Li, Dan and Wu, Haitao and Zhang, Xuan and Shi, Yunfeng and Tian, Chen and Zhang, Yongguang and Lu, Songwu, «BCube: a high performance, server-centric network architecture for modular data centers,» *ACM SIGCOMM Computer Communication Review,* vol. 39, nº 4, pp. 63-74, 2009.

[4] Lin, Minghong and Wierman, Adam and Andrew, Lachlan LH and Thereska, Eno, «Dynamic right-sizing for power-proportional data centers,» *IEEE/ACM Transactions on Networking (TON),* vol. 21, nº 5, pp. 1378-1391, 2013.

[5] W. P. a. S. J. H. a. B. K. G. Turner, *Industry standard tier classifications define site infrastructure performance,* Santa Fe: Santa Fe, NM: Uptime Institute, 2005.

[6] M. Wiboonrat, «2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing},» de *An optimal data center availability and investment trade-offs*, Madrid, 2008.

[7] J. a. M. J. E. a. M. J. Karidis, «Proceedings of the 6th ACM conference on Computing frontiers,» de *True value: assessing and optimizing the cost of computing at the data center level*, NY, USA, 2009.

[8] Greenberg, Albert and Hamilton, James and Maltz, David A and Patel, Parveen, «The cost of a cloud: research problems in data center networks,» *ACM SIGCOMM computer communication review,* vol. 39, nº 1, pp. 68-73, 2008.

[9] D. Wolber, «SIGCSE,» de *App inventor and real-world motivation*, San Francisco, 2011.

[10] Gray, Jeff and Abelson, Hal and Wolber, David and Friend, Michelle, «Teaching CS principles with app inventor,» de *Proceedings of the 50th Annual Southeast Regional Conference*, Alabama USA, 2012.

[11] Prahlad, Anand and Muller, Marcus S and Kottomtharayil, Rajiv and Kavuri, Srinivas and Gokhale, Parag and Vijayan, Manoj, *Data object store and server for a cloud storage environment, including data deduplication and data management across multiple cloud storage sites,* Los Angeles, USA: Google Patents, 2012.

[12] McCrory, Dave D and Hirschfeld, Robert A, *Virtual server cloud interfacing,* Washington DC, USA: Google Patents, 2009.

[13] Chen, Rongmao and Mu, Yi and Yang, Guomin and Guo, Fuchun and Wang, Xiaofen, «Dual-server public-key encryption with keyword search for secure cloud storage,» *IEEE transactions on information forensics and security,* vol. 11, nº 4, pp. 789-798, 2016.

[14] Yin, Yonghua and Wang, Lan and Gelenbe, Erol, «Multi-layer neural networks for quality of service oriented server-state classification in cloud servers,» de *2017 International Joint Conference on Neural Networks (IJCNN)*, New York, USA, 2017.

[15] R. Alchemister, *Raspberry The New Release,* Liverpool: Smart Home, 2016.

[16] Upton, Eben and Halfacree, Gareth, Raspberry Pi user guide, Washington DC, USA: John Wiley \& Sons, 2014.

[17] Zhao, Cheah Wai and Jegatheesan, Jayanand and Loon, Son Chee, «Exploring iot application using raspberry pi,» *International Journal of Computer Networks and Applications,* vol. 2, nº 1, pp. 27-34, 2015.

[18] Smith, Bruce, Raspberry Pi Assembly Language RASPBIAN Beginners: Hands On Guide, Los Angeles, USA: CreateSpace Independent Publishing Platform, 2013.

[19] Marzal, Andr{\'e}s and Luengo, Isabel Gracia, Introducci{\'o}n a la Programaci{\'o}n con Python y C, Barcelona, España: Publicacions de la Universitat Jaume I, 2002.

[20] Arduino, Store Arduino, «Arduino,» *Arduino LLC,* vol. 1, nº 1, pp. 20-50, 2015.

[21] Badamasi, Yusuf Abdullahi, «The working principle of an Arduino,» de *2014 11th International Conference on Electronics, Computer and Computation (ICECCO)*, New York, USA, 2014.

[22] B. Evans, Beginning Arduino Programming, Orlando, USA: Apress, 2011.

[23] Guam{\'a}n-Molina, Jes{\'u}s and Vargas-Guevara, Carlos and Nogales-Portero, Rub{\'e}n and Guevara-Aulestia, David and Garc{\'\i}a-Carrillo, Mario and R{\'\i}os-Villacorta, Alberto, «Solar manager: plataforma cloud de adquisici{\'o}n, tratamiento y visualizaci{\'o}n de informaci{\'o}n de sistemas fotovoltaicos aislados,» *Ingenius. Revista de Ciencia y Tecnolog{\'\i}a,* vol. 3, nº 4, pp. 5-16, 2016.

[24] Shajahan, Altaf Hamed and Anand, A, «Data acquisition and control using Arduino-Android platform: Smart plug,» de *2013 International Conference on Energy Efficient Technologies for Sustainability*, Nagercoil, India, 2013.

[25] Gabriele, T and Pantoli, L and Stornelli, V and Chiulli, D and Muttillo, M, «Smart power management system for home appliances and wellness based on wireless sensors network and mobile technology,» de *2015 XVIII AISEM Annual Conference*, Trento, Italy, 2015.

[26] {\'A}lvarez-Alvarado, J and R{\'\i}os-Moreno, GJ and Ronquillo, G and Trejo-Perea, M, «Medidor inteligente para las variables de energ{\'\i}a el{\'e}ctrica basado en un sistema embebido.,» *Research in Computing Science,* vol. 3, nº 2, pp. 107-116, 2016.

[27] Gabriele, T and Pantoli, L and Stornelli, V and Chiulli, D and Muttillo, M, «Smart power management system for home appliances and wellness based on wireless sensors network and mobile technology,» de *2015 XVIII AISEM Annual Conference*, Trento, Italy, 2015.

[28] Hajar, Ibnu and Hafizd, Muhammad and Dani, Akhmad Wahyu and Miharno, Satriyo, «Monitoring of Electrical System Using Internet of Things with Smart Current Electric Sensors,» *Sinergi: Jurnal Teknik Mercu Buana,* vol. 22, nº 3, pp. 211-218, 2012.

[29] Utami, Sentagi Sesotya and Na'im A, Azizi and Kencanawati, Erlin and Tanjung, M Akbar and Achmad, Balza, «Energy Monitoring System for Existing Buildings in Indonesia,» de *E3S Web of Conferences*, Yakarta, 2018.

[30] Saptadi, Arief Hendra, «Perbandingan Akurasi Pengukuran Suhu dan Kelembaban Antara Sensor DHT11 dan DHT22,» *Jurnal Infotel,* vol. 6, nº 2, pp. 49-56, 2014.

[31] Herutomo, Anton and Abdurohman, Maman and Suwastika, Novian Anggis and Prabowo, Sidik and Wijiutomo, Catur Wirawan, «Forest fire detection system reliability test using wireless sensor network and OpenMTC communication platform,» *2015 3rd International conference on information and communication technology (ICoICT),* vol. 4, nº 3, pp. 87-91, 2015.

[32] T. Gray, «KEYENCE CORPORATION.,» 29 Nobiembre 2010. [En línea]. Available: https://www.keyence.com.mx/ss/products/sensor/sensorbasics/proximity/info/?fbclid=IwAR2zHIh_0xC3akci3JaWzr-x-2Iuj0hxwuWTDOJwCTHcfyPAHqtCuK_9NWQ. [Último acceso: 21 Abril 2019].

[33] Xiong, Pengcheng and Chi, Yun and Zhu, Shenghuo and Moon, Hyun Jin and Pu, Calton and Hacig{\"u}m{\"u}{\c{s}}, Hakan, «Intelligent management of virtualized resources for database systems in cloud environment,» de *2011 IEEE 27th International Conference on Data Engineering*, Shanghai, China, 2011.

[34] Kuang, Wenhui and Liu, Jiyuan and Dong, Jinwei and Chi, Wenfeng and Zhang, Chi, «The rapid and massive urban and industrial land expansions in China between 1990 and 2010: A CLUD-based analysis of their trajectories, patterns, and drivers,» *Landscape and Urban Planning,* vol. 1, nº 2, pp. 21-33, 2016.

[35] Breijo, Eduardo Garc{\'\i}a, Compilador C CCS y simulador PROTEUS para microcontroladores PIC, New York, USA: Marcombo, 2012.

[36] Instruments, Texas and Sensors, LM35 Precision Centigrade Temperature, «LM358 Datasheet,» *Texas Instruments Incorporated,* vol. 2, nº 1, pp. 21-35, 1999.

[37] Harper, Gilberto Enr{\'\i}quez, ABC de Las Intalaciones Electricas Industriales, México: Editorial Limusa, 2002.

[38] J. Hobson, CentOS 6 Linux Server Cookbook, Packt Publishing Ltd, 2013.

[39] Castro, Edwin Geovanny Flores, «Implementaci{\'o}n de una base de datos heterog{\'e}nea distribuida entre los SGBDs ORACLE, MySQL y PostgreSQL con replicaci{\'o}n, mediante un

script bash implementado en el sistema operativo CentOS usando software libre,» *INNOVA Research Journal,* vol. 3, nº 2.1, pp. 59-66, 2018.

[40] Jadhav, Archana and Oswal, Vipul and Madane, Sagar and Zope, Harshal and Hatmode, Vishal, «Vnc architecture based remote desktop access through android mobile phones,» *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 1, nº 2, p. 98, 2012.

[41] Welling, Luke and Thomson, Laura, «Desarrollo web con php y mysql php 5 y mysqul 4.1 y 5: disco compacto,» *Madrid, Espa{\~n}a: Anaya Multimedia,* vol. 5, nº 12, pp. 29-34, 2005.

[42] C. Spera, «Las claves en la administraci{\'o}n de energ{\'\i}a del Data Center,» *Logicalis Now,* vol. 14, nº 2, pp. 13-14, 2012.

[43] Carbonell Morales, Tania and Salgado {\'A}lvarez, Ibis, «Sistema de enfriamiento con desecante para reducir consumo de energ{\'\i}a en restaurante caso de estudio,» *Ingenier{\'\i}a Energ{\'e}tica,* vol. 37, nº 1, pp. 55-62, 2016.

ANEXOS

A continuación, se presentan los anexos que han sido necesarios implementar para la programación de la tarjera Arduino Uno y para la tarjera Raspberry Pi 3 B+, para el desarrollo del prototipo implementado en el *Data Center* de la Universidad Central del Ecuador.

ANEXO A. Código fuente del programa implementado en la tarjeta Arduino Uno.

ANEXO B. Código fuente del programa implementado en la tarjeta Raspberry Pi 3 B+.

ANEXO C. Código fuente del programa implementado para la aplicación web y la aplicación de App Inventor.

ANEXO A

```
#include "EmonLib.h"                   // Emon Library= libreria de monitoreo de energia arduino


#define CALIBRACION_DE_VOLTAJE 125 //Contanste de calibracion del voltaje

#define CALIBRACION_DE_CORRIENTE 70 //Contanste de calibracion del corriente


EnergyMonitor linea1;                  // Creacion de la instacion EMON (crear objeto)

EnergyMonitor linea2;

EnergyMonitor linea3;


void setup()

{

  Serial.begin(9600);// se inicializa la serial


  linea1.voltage(A1, CALIBRACION_DE_VOLTAJE, 1.7); //Llamado a la funcion y definicion de parametros(Voltaje: pin analogico de entrada, calibración, cambio de fase)


  linea1.current(A0, CALIBRACION_DE_CORRIENTE);         // Llamado a la funcion y definicion de parametros(Corriente: ping analogico de entrada, calibracion.)


  linea2.voltage(A3, CALIBRACION_DE_VOLTAJE, 1.7); //Llamado a la funcion y definicion de parametros(Voltaje: pin analogico de entrada, calibración, cambio de fase)
```

```
  linea2.current(A2, CALIBRACION_DE_CORRIENTE);         // Llamado a la funcion y definicion de parametros(Corriente: ping analogico de entrada, calibracion)


  linea3.voltage(A5, CALIBRACION_DE_VOLTAJE, 1.7); //Llamado a la funcion y definicion de parametros(Voltaje: pin analogico de entrada, calibración, cambio de fase)


  linea3.current(A4, CALIBRACION_DE_CORRIENTE);         // Llamado a la funcion y definicion de parametros(Corriente: ping analogico de entrada, calibracion)

}


void loop()

{

  Serial.flush();

  while(Serial.available()){

//delay(10);

    char entrada=Serial.read();

    if(entrada=='M'){

  linea1.calcVI(20,2000);                                    //    Calcula realPower,apparentPower,powerFactor,Vrms,Irms,kWh   increment   con   los paramtreos de (numeros de logitudes de onda media, time-out)

  linea2.calcVI(20,2000);                                    //    Calcula realPower,apparentPower,powerFactor,Vrms,Irms,kWh   increment   con   los paramtreos de (numeros de logitudes de onda media, time-out)
```

```
  linea3.calcVI(20,2000);                                    //    Calcula
realPower,apparentPower,powerFactor,Vrms,Irms,kWh   increment   con   los
paramtreos de (numeros de logitudes de onda media, time-out)



  float corrienteActual1 = linea1.Irms;             //extrae la corriente
(Irms) en Variable
  float voltajeActual1   = linea1.Vrms;                      //extrae el
voltaje (Vrms) en Variable
  float corrienteActual2 = linea2.Irms;             //extrae la corriente
(Irms) en Variable
  float voltajeActual2   = linea2.Vrms;                      //extrae el
voltaje (Vrms) en Variable
  float corrienteActual3 = linea3.Irms;             //extrae la corriente
(Irms) en Variable
  float voltajeActual3   = linea3.Vrms;                      //extrae el
voltaje (Vrms) en Variable


  // impresion de valores en monitor de serie
  float potencia1=corrienteActual1 * voltajeActual1;
  float potencia2=corrienteActual2 * voltajeActual2;
  float potencia3=corrienteActual3 * voltajeActual3;


  Serial.print("[[");
  Serial.print(voltajeActual1);
  Serial.print(",");
  Serial.print(corrienteActual1);
```

```
  Serial.print(",");
  Serial.print(potencia1);
  Serial.print("],[");
  Serial.print(voltajeActual2);
  Serial.print(",");
  Serial.print(corrienteActual2);
  Serial.print(",");
  Serial.print(potencia2);
  Serial.print("],[");
  Serial.print(voltajeActual3);
  Serial.print(",");
  Serial.print(corrienteActual3);
  Serial.print(",");
  Serial.print(potencia3);
  Serial.print("]]");
  }
  //else{
   // Serial.print("ERROR");
  //}
  }
}.
```

ANEXO B

```
import RPi.GPIO as GPIO

import dht11

import time

import datetime

from time import sleep

import serial

import mysql.connector

import ast

ser = serial.Serial(port='/dev/ttyACM0',baudrate = 9600, timeout=2)

ser.flushInput()

s = [0,1]

# initialize GPIO

GPIO.setwarnings(False)# que no se mande ninguna alerta

GPIO.setmode(GPIO.BOARD)

GPIO.cleanup()

button1=7

GPIO.setup(button1,GPIO.IN,pull_up_down=GPIO.PUD_UP)

contador=0

bandera1=0

bandera2=0

# read data using pin 17

instanciaDHT = dht11.DHT11(pin=11)

print(datetime.datetime.now().time())

while True:
```

```
    db=mysql.connector.connect(host="10.20.1.189",          user="root",
passwd="tesis", db="tesis")

    cursor = db.cursor()

    insertarSensoresGenerales = ("INSERT INTO sensoresGenerales (Fecha,
valorTemperatura, valorNumeroVecesPuerta) VALUES (%s, %s, %s)")

    insertarSensoresCorrieteVoltaje        =        ("INSERT        INTO
sensoresCorrienteVoltaje    (Fecha,    valorCorriente,    valorVoltaje,
valorPotencia, linea) VALUES (%s, %s, %s, %s, %s)")


    LecturaActual=GPIO.input(button1)

    if LecturaActual!=bandera1:

        bandera2=1

    if bandera2==1 and LecturaActual==bandera1:

        bandera2=2

    if bandera2==2:

        bandera2=0

        contador=contador+1

    hora= str(datetime.datetime.now().time()).split(":")[0]

    minuto=str(datetime.datetime.now().time()).split(":")[1]


segundo=str(datetime.datetime.now().time()).split(":")[2].split(".")[0]


    if((minuto=="00" or minuto=="30") and segundo=="00"):

        resultado = instanciaDHT.read()

        if resultado.is_valid():

            print("Fecha de lectura:",datetime.datetime.now())

            print("Temperatura (°C):",resultado.temperature)
```

```
        print("contador= ",contador)
        ser.write(b'M')
        time.sleep(4)
        read_serial=ser.readline()
        print(read_serial==b"")

        if(read_serial==b""):
            ser.write(b'M')
            time.sleep(4)
            read_serial=ser.readline()

        print(ast.literal_eval(read_serial.decode("utf-
8").replace("\\x00","").replace("\\n","")))


datosSensoresGenerales=(datetime.datetime.now(),resultado.temperature,
contador)
        datosSensoresCorrienteVoltaje=(datetime.datetime.now(),
float(ast.literal_eval(read_serial.decode("utf-
8").replace("\\x00","").replace("\\n",""))[0][1]),float(ast.literal_eval
(read_serial.decode("utf-
8").replace("\\x00","").replace("\\n",""))[0][0]),float(ast.literal_eval
(read_serial.decode("utf-
8").replace("\\x00","").replace("\\n",""))[0][2]),1)

        cursor.execute(insertarSensoresGenerales,
datosSensoresGenerales)
```

```
        cursor.execute(insertarSensoresCorrieteVoltaje,
datosSensoresCorrienteVoltaje)


        datosSensoresCorrienteVoltaje=(datetime.datetime.now(),
float(ast.literal_eval(read_serial.decode("utf-
8").replace("\\x00","").replace("\\n",""))[1][1]),float(ast.literal_eval
(read_serial.decode("utf-
8").replace("\\x00","").replace("\\n",""))[1][0]),float(ast.literal_eval
(read_serial.decode("utf-
8").replace("\\x00","").replace("\\n",""))[1][2]),2)

        cursor.execute(insertarSensoresCorrieteVoltaje,
datosSensoresCorrienteVoltaje)


        datosSensoresCorrienteVoltaje=(datetime.datetime.now(),
float(ast.literal_eval(read_serial.decode("utf-
8").replace("\\x00","").replace("\\n",""))[2][1]),float(ast.literal_eval
(read_serial.decode("utf-
8").replace("\\x00","").replace("\\n",""))[2][0]),float(ast.literal_eval
(read_serial.decode("utf-
8").replace("\\x00","").replace("\\n",""))[2][2]),3)

        cursor.execute(insertarSensoresCorrieteVoltaje,
datosSensoresCorrienteVoltaje)

        db.commit()

    cursor.close()

    db.close()

    if hora == "00" and minuto=="00" and segundo == "00":

        contador=0

class SensoresGenerales:

    def __init__(self, Fecha, ValorTemperatura, ValorNumeroVecesPuerta):

        self.fecha = Fecha
```

```
        self.valortemperatura = ValorTemperatura
        self.valornuemrovecespuerta = ValorNumeroVecesPuerta
class SensoresCorrienteVoltaje:
    def __init__(self, Fecha, ValorCorriente, ValorVolatje, ValorPotecia,
Linea ):
        self.fecha = Fecha
        self.valorcorriente = ValorCorriente
        self.valorvoltaje = ValorVoltaje
        self.valorpotencia = ValorPotencia
        self.linea = Linea
```

ANEXO C

```
<?php
$fechaBusqueda = trim($_GET['fechaBusqueda']);
$mysqli = new mysqli('127.0.0.1', 'root', 'tesis', 'tesis');
$mysqli->set_charset("utf8");
$res = $mysqli->query("SELECT * FROM sensorescorrientevoltaje where Fecha
like '$fechaBusqueda%'");
$vl1h0=0;
$vl1h0=0;
$vl1h1=0;
$vl1h2=0;
$vl1h3=0;
$vl1h4=0;
```

```
$vl1h5=0;

$vl1h6=0;

$vl1h7=0;

$vl1h8=0;

$vl1h9=0;

$vl1h10=0;

$vl1h11=0;

$vl1h12=0;

$vl1h13=0;

$vl1h14=0;

$vl1h15=0;

$vl1h16=0;

$vl1h17=0;

$vl1h18=0;

$vl1h19=0;

$vl1h20=0;

$vl1h21=0;

$vl1h22=0;

$vl1h23=0;


$vl2h0=0;

$vl2h0=0;

$vl2h1=0;

$vl2h2=0;

$vl2h3=0;
```

```
$vl2h4=0;

$vl2h5=0;

$vl2h6=0;

$vl2h7=0;

$vl2h8=0;

$vl2h9=0;

$vl2h10=0;

$vl2h11=0;

$vl2h12=0;

$vl2h13=0;

$vl2h14=0;

$vl2h15=0;

$vl2h16=0;

$vl2h17=0;

$vl2h18=0;

$vl2h19=0;

$vl2h20=0;

$vl2h21=0;

$vl2h22=0;

$vl2h23=0;


$vl3h0=0;

$vl3h0=0;

$vl3h1=0;

$vl3h2=0;
```

```
$vl3h3=0;

$vl3h4=0;

$vl3h5=0;

$vl3h6=0;

$vl3h7=0;

$vl3h8=0;

$vl3h9=0;

$vl3h10=0;

$vl3h11=0;

$vl3h12=0;

$vl3h13=0;

$vl3h14=0;

$vl3h15=0;

$vl3h16=0;

$vl3h17=0;

$vl3h18=0;

$vl3h19=0;

$vl3h20=0;

$vl3h21=0;

$vl3h22=0;

$vl3h23=0;


$cl1h0=0;

$cl1h0=0;

$cl1h1=0;
```

```
$cl1h2=0;

$cl1h3=0;

$cl1h4=0;

$cl1h5=0;

$cl1h6=0;

$cl1h7=0;

$cl1h8=0;

$cl1h9=0;

$cl1h10=0;

$cl1h11=0;

$cl1h12=0;

$cl1h13=0;

$cl1h14=0;

$cl1h15=0;

$cl1h16=0;

$cl1h17=0;

$cl1h18=0;

$cl1h19=0;

$cl1h20=0;

$cl1h21=0;

$cl1h22=0;

$cl1h23=0;


$cl2h0=0;

$cl2h0=0;
```

```
$cl2h1=0;

$cl2h2=0;

$cl2h3=0;

$cl2h4=0;

$cl2h5=0;

$cl2h6=0;

$cl2h7=0;

$cl2h8=0;

$cl2h9=0;

$cl2h10=0;

$cl2h11=0;

$cl2h12=0;

$cl2h13=0;

$cl2h14=0;

$cl2h15=0;

$cl2h16=0;

$cl2h17=0;

$cl2h18=0;

$cl2h19=0;

$cl2h20=0;

$cl2h21=0;

$cl2h22=0;

$cl2h23=0;


$cl3h0=0;
```

```
$cl3h0=0;
$cl3h1=0;
$cl3h2=0;
$cl3h3=0;
$cl3h4=0;
$cl3h5=0;
$cl3h6=0;
$cl3h7=0;
$cl3h8=0;
$cl3h9=0;
$cl3h10=0;
$cl3h11=0;
$cl3h12=0;
$cl3h13=0;
$cl3h14=0;
$cl3h15=0;
$cl3h16=0;
$cl3h17=0;
$cl3h18=0;
$cl3h19=0;
$cl3h20=0;
$cl3h21=0;
$cl3h22=0;
$cl3h23=0;
```

```
$pl1h0=0;

$pl1h0=0;

$pl1h1=0;

$pl1h2=0;

$pl1h3=0;

$pl1h4=0;

$pl1h5=0;

$pl1h6=0;

$pl1h7=0;

$pl1h8=0;

$pl1h9=0;

$pl1h10=0;

$pl1h11=0;

$pl1h12=0;

$pl1h13=0;

$pl1h14=0;

$pl1h15=0;

$pl1h16=0;

$pl1h17=0;

$pl1h18=0;

$pl1h19=0;

$pl1h20=0;

$pl1h21=0;

$pl1h22=0;

$pl1h23=0;
```

```
$pl2h0=0;

$pl2h0=0;

$pl2h1=0;

$pl2h2=0;

$pl2h3=0;

$pl2h4=0;

$pl2h5=0;

$pl2h6=0;

$pl2h7=0;

$pl2h8=0;

$pl2h9=0;

$pl2h10=0;

$pl2h11=0;

$pl2h12=0;

$pl2h13=0;

$pl2h14=0;

$pl2h15=0;

$pl2h16=0;

$pl2h17=0;

$pl2h18=0;

$pl2h19=0;

$pl2h20=0;

$pl2h21=0;

$pl2h22=0;
```

```
$pl2h23=0;


$pl3h0=0;

$pl3h0=0;

$pl3h1=0;

$pl3h2=0;

$pl3h3=0;

$pl3h4=0;

$pl3h5=0;

$pl3h6=0;

$pl3h7=0;

$pl3h8=0;

$pl3h9=0;

$pl3h10=0;

$pl3h11=0;

$pl3h12=0;

$pl3h13=0;

$pl3h14=0;

$pl3h15=0;

$pl3h16=0;

$pl3h17=0;

$pl3h18=0;

$pl3h19=0;

$pl3h20=0;

$pl3h21=0;
```

```
$pl3h22=0;
$pl3h23=0;

$th0=0;
$th0=0;
$th1=0;
$th2=0;
$th3=0;
$th4=0;
$th5=0;
$th6=0;
$th7=0;
$th8=0;
$th9=0;
$th10=0;
$th11=0;
$th12=0;
$th13=0;
$th14=0;
$th15=0;
$th16=0;
$th17=0;
$th18=0;
$th19=0;
$th20=0;
```

```
$th21=0;

$th22=0;

$th23=0;


while($f = $res->fetch_object()){

      $fechaSCV=$f->Fecha;

      $lineaSCV=$f->linea;

      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="00"     and
$lineaSCV=="1"){

            $vl1h0=$f->valorVoltaje;

            $cl1h0=$f->valorCorriente;

            $pl1h0=$f->valorPotencia;

      }

      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="01"     and
$lineaSCV=="1"){

            $vl1h1=$f->valorVoltaje;

            $cl1h1=$f->valorCorriente;

            $pl1h1=$f->valorPotencia;

      }

      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="02"     and
$lineaSCV=="1"){

            $vl1h2=$f->valorVoltaje;

            $cl1h2=$f->valorCorriente;

            $pl1h2=$f->valorPotencia;

      }
```

```
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="03"     and
$lineaSCV=="1"){
            $vl1h3=$f->valorVoltaje;
            $cl1h3=$f->valorCorriente;
            $pl1h3=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="04"     and
$lineaSCV=="1"){
            $vl1h4=$f->valorVoltaje;
            $cl1h4=$f->valorCorriente;
            $pl1h4=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="05"     and
$lineaSCV=="1"){
            $vl1h5=$f->valorVoltaje;
            $cl1h5=$f->valorCorriente;
            $pl1h5=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="06"     and
$lineaSCV=="1"){
            $vl1h6=$f->valorVoltaje;
            $cl1h6=$f->valorCorriente;
            $pl1h6=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="07"     and
$lineaSCV=="1"){
            $vl1h7=$f->valorVoltaje;
```

```
            $cl1h7=$f->valorCorriente;
            $pl1h7=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="08"     and
$lineaSCV=="1"){
            $vl1h8=$f->valorVoltaje;
            $cl1h8=$f->valorCorriente;
            $pl1h8=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="09"     and
$lineaSCV=="1"){
            $vl1h9=$f->valorVoltaje;
            $cl1h9=$f->valorCorriente;
            $pl1h9=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="10"     and
$lineaSCV=="1"){
            $vl1h10=$f->valorVoltaje;
            $cl1h10=$f->valorCorriente;
            $pl1h10=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="11"     and
$lineaSCV=="1"){
            $vl1h11=$f->valorVoltaje;
            $cl1h11=$f->valorCorriente;
            $pl1h11=$f->valorPotencia;
```

```
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="12"     and
$lineaSCV=="1"){
            $vl1h12=$f->valorVoltaje;
            $cl1h12=$f->valorCorriente;
            $pl1h12=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="13"     and
$lineaSCV=="1"){
            $vl1h13=$f->valorVoltaje;
            $cl1h13=$f->valorCorriente;
            $pl1h13=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="14"     and
$lineaSCV=="1"){
            $vl1h14=$f->valorVoltaje;
            $cl1h14=$f->valorCorriente;
            $pl1h14=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="15"     and
$lineaSCV=="1"){
            $vl1h15=$f->valorVoltaje;
            $cl1h15=$f->valorCorriente;
            $pl1h15=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="16"     and
$lineaSCV=="1"){
```

```
            $vl1h16=$f->valorVoltaje;
            $cl1h16=$f->valorCorriente;
            $pl1h16=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="17"     and
$lineaSCV=="1"){
            $vl1h17=$f->valorVoltaje;
            $cl1h17=$f->valorCorriente;
            $pl1h17=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="18"     and
$lineaSCV=="1"){
            $vl1h18=$f->valorVoltaje;
            $cl1h18=$f->valorCorriente;
            $pl1h18=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="19"     and
$lineaSCV=="1"){
            $vl1h19=$f->valorVoltaje;
            $cl1h19=$f->valorCorriente;
            $pl1h19=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="20"     and
$lineaSCV=="1"){
            $vl1h20=$f->valorVoltaje;
            $cl1h20=$f->valorCorriente;
```

```
            $pl1h20=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="21"     and
$lineaSCV=="1"){
            $vl1h21=$f->valorVoltaje;
            $cl1h21=$f->valorCorriente;
            $pl1h21=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="22"     and
$lineaSCV=="1"){
            $vl1h22=$f->valorVoltaje;
            $cl1h22=$f->valorCorriente;
            $pl1h22=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="23"     and
$lineaSCV=="1"){
            $vl1h23=$f->valorVoltaje;
            $cl1h23=$f->valorCorriente;
            $pl1h23=$f->valorPotencia;
      }

      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="00"     and
$lineaSCV=="2"){
            $vl2h0=$f->valorVoltaje;
            $cl2h0=$f->valorCorriente;
            $pl2h0=$f->valorPotencia;
```

```
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="01"     and
$lineaSCV=="2"){
            $vl2h1=$f->valorVoltaje;
            $cl2h1=$f->valorCorriente;
            $pl2h1=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="02"     and
$lineaSCV=="2"){
            $vl2h2=$f->valorVoltaje;
            $cl2h2=$f->valorCorriente;
            $pl2h2=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="03"     and
$lineaSCV=="2"){
            $vl2h3=$f->valorVoltaje;
            $cl2h3=$f->valorCorriente;
            $pl2h3=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="04"     and
$lineaSCV=="2"){
            $vl2h4=$f->valorVoltaje;
            $cl2h4=$f->valorCorriente;
            $pl2h4=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="05"     and
$lineaSCV=="2"){
```

```
            $vl2h5=$f->valorVoltaje;
            $cl2h5=$f->valorCorriente;
            $pl2h5=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="06"     and
$lineaSCV=="2"){
            $vl2h6=$f->valorVoltaje;
            $cl2h6=$f->valorCorriente;
            $pl2h6=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="07"     and
$lineaSCV=="2"){
            $vl2h7=$f->valorVoltaje;
            $cl2h7=$f->valorCorriente;
            $pl2h7=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="08"     and
$lineaSCV=="2"){
            $vl2h8=$f->valorVoltaje;
            $cl2h8=$f->valorCorriente;
            $pl2h8=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="09"     and
$lineaSCV=="2"){
            $vl2h9=$f->valorVoltaje;
            $cl2h9=$f->valorCorriente;
```

```
            $pl2h9=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="10"     and
$lineaSCV=="2"){
            $vl2h10=$f->valorVoltaje;
            $cl2h10=$f->valorCorriente;
            $pl2h10=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="11"     and
$lineaSCV=="2"){
            $vl2h11=$f->valorVoltaje;
            $cl2h11=$f->valorCorriente;
            $pl2h11=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="12"     and
$lineaSCV=="2"){
            $vl2h12=$f->valorVoltaje;
            $cl2h12=$f->valorCorriente;
            $pl2h12=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="13"     and
$lineaSCV=="2"){
            $vl2h13=$f->valorVoltaje;
            $cl2h13=$f->valorCorriente;
            $pl2h13=$f->valorPotencia;
      }
```

```
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="14"     and
$lineaSCV=="2"){
            $vl2h14=$f->valorVoltaje;
            $cl2h14=$f->valorCorriente;
            $pl2h14=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="15"     and
$lineaSCV=="2"){
            $vl2h15=$f->valorVoltaje;
            $cl2h15=$f->valorCorriente;
            $pl2h15=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="16"     and
$lineaSCV=="2"){
            $vl2h16=$f->valorVoltaje;
            $cl2h16=$f->valorCorriente;
            $pl2h16=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="17"     and
$lineaSCV=="2"){
            $vl2h17=$f->valorVoltaje;
            $cl2h17=$f->valorCorriente;
            $pl2h17=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="18"     and
$lineaSCV=="2"){
            $vl2h18=$f->valorVoltaje;
```

```
            $cl2h18=$f->valorCorriente;
            $pl2h18=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="19"     and
$lineaSCV=="2"){
            $vl2h19=$f->valorVoltaje;
            $cl2h19=$f->valorCorriente;
            $pl2h19=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="20"     and
$lineaSCV=="2"){
            $vl2h20=$f->valorVoltaje;
            $cl2h20=$f->valorCorriente;
            $pl2h20=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="21"     and
$lineaSCV=="2"){
            $vl2h21=$f->valorVoltaje;
            $cl2h21=$f->valorCorriente;
            $pl2h21=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="22"     and
$lineaSCV=="2"){
            $vl2h22=$f->valorVoltaje;
            $cl2h22=$f->valorCorriente;
            $pl2h22=$f->valorPotencia;
```

```
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="23"     and
$lineaSCV=="2"){
            $vl2h23=$f->valorVoltaje;
            $cl2h23=$f->valorCorriente;
            $pl2h23=$f->valorPotencia;
      }

      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="00"     and
$lineaSCV=="3"){
            $vl3h0=$f->valorVoltaje;
            $cl3h0=$f->valorCorriente;
            $pl3h0=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="01"     and
$lineaSCV=="3"){
            $vl3h1=$f->valorVoltaje;
            $cl3h1=$f->valorCorriente;
            $pl3h1=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="02"     and
$lineaSCV=="3"){
            $vl3h2=$f->valorVoltaje;
            $cl3h2=$f->valorCorriente;
            $pl3h2=$f->valorPotencia;
      }
```

```
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="03"     and
$lineaSCV=="3"){
            $vl3h3=$f->valorVoltaje;
            $cl3h3=$f->valorCorriente;
            $pl3h3=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="04"     and
$lineaSCV=="3"){
            $vl3h4=$f->valorVoltaje;
            $cl3h4=$f->valorCorriente;
            $pl3h4=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="05"     and
$lineaSCV=="3"){
            $vl3h5=$f->valorVoltaje;
            $cl3h5=$f->valorCorriente;
            $pl3h5=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="06"     and
$lineaSCV=="3"){
            $vl3h6=$f->valorVoltaje;
            $cl3h6=$f->valorCorriente;
            $pl3h6=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="07"     and
$lineaSCV=="3"){
            $vl3h7=$f->valorVoltaje;
```

```
            $cl3h7=$f->valorCorriente;
            $pl3h7=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="08"     and
$lineaSCV=="3"){
            $vl3h8=$f->valorVoltaje;
            $cl3h8=$f->valorCorriente;
            $pl3h8=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="09"     and
$lineaSCV=="3"){
            $vl3h9=$f->valorVoltaje;
            $cl3h9=$f->valorCorriente;
            $pl3h9=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="10"     and
$lineaSCV=="3"){
            $vl3h10=$f->valorVoltaje;
            $cl3h10=$f->valorCorriente;
            $pl3h10=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="11"     and
$lineaSCV=="3"){
            $vl3h11=$f->valorVoltaje;
            $cl3h11=$f->valorCorriente;
            $pl3h11=$f->valorPotencia;
```

```
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="12"     and
$lineaSCV=="3"){
            $vl3h12=$f->valorVoltaje;
            $cl3h12=$f->valorCorriente;
            $pl3h12=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="13"     and
$lineaSCV=="3"){
            $vl3h13=$f->valorVoltaje;
            $cl3h13=$f->valorCorriente;
            $pl3h13=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="14"     and
$lineaSCV=="3"){
            $vl3h14=$f->valorVoltaje;
            $cl3h14=$f->valorCorriente;
            $pl3h14=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="15"     and
$lineaSCV=="3"){
            $vl3h15=$f->valorVoltaje;
            $cl3h15=$f->valorCorriente;
            $pl3h15=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="16"     and
$lineaSCV=="3"){
```

```
            $vl3h16=$f->valorVoltaje;
            $cl3h16=$f->valorCorriente;
            $pl3h16=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="17"     and
$lineaSCV=="3"){
            $vl3h17=$f->valorVoltaje;
            $cl3h17=$f->valorCorriente;
            $pl3h17=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="18"     and
$lineaSCV=="3"){
            $vl3h18=$f->valorVoltaje;
            $cl3h18=$f->valorCorriente;
            $pl3h18=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="19"     and
$lineaSCV=="3"){
            $vl3h19=$f->valorVoltaje;
            $cl3h19=$f->valorCorriente;
            $pl3h19=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="20"     and
$lineaSCV=="3"){
            $vl3h20=$f->valorVoltaje;
            $cl3h20=$f->valorCorriente;
```

```
            $pl3h20=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="21"     and
$lineaSCV=="3"){
            $vl3h21=$f->valorVoltaje;
            $cl3h21=$f->valorCorriente;
            $pl3h21=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="22"     and
$lineaSCV=="3"){
            $vl3h22=$f->valorVoltaje;
            $cl3h22=$f->valorCorriente;
            $pl3h22=$f->valorPotencia;
      }
      if(explode(":",explode("     ",     $fechaSCV)[1])[0]=="23"     and
$lineaSCV=="3"){
            $vl3h23=$f->valorVoltaje;
            $cl3h23=$f->valorCorriente;
            $pl3h23=$f->valorPotencia;
      }
}
$res = $mysqli->query("SELECT * FROM sensoresgenerales where Fecha like
'$fechaBusqueda%'");
while($f = $res->fetch_object()){
      $fechaSG=$f->Fecha;
      if(explode(":",explode(" ", $fechaSG)[1])[0]=="00"){
```

```
      $th0=$f->valorTemperatura;
}
if(explode(":",explode(" ", $fechaSG)[1])[0]=="01"){
      $th1=$f->valorTemperatura;
}
if(explode(":",explode(" ", $fechaSG)[1])[0]=="02"){
      $th2=$f->valorTemperatura;
}
if(explode(":",explode(" ", $fechaSG)[1])[0]=="03"){
      $th3=$f->valorTemperatura;
}
if(explode(":",explode(" ", $fechaSG)[1])[0]=="04"){
      $th4=$f->valorTemperatura;
}
if(explode(":",explode(" ", $fechaSG)[1])[0]=="05"){
      $th5=$f->valorTemperatura;
}
if(explode(":",explode(" ", $fechaSG)[1])[0]=="06"){
      $th6=$f->valorTemperatura;
}
if(explode(":",explode(" ", $fechaSG)[1])[0]=="07"){
      $th7=$f->valorTemperatura;
}
if(explode(":",explode(" ", $fechaSG)[1])[0]=="08"){
      $th8=$f->valorTemperatura;
```

```
}
if(explode(":",explode(" ", $fechaSG)[1])[0]=="09"){
      $th9=$f->valorTemperatura;
}
if(explode(":",explode(" ", $fechaSG)[1])[0]=="10"){
      $th10=$f->valorTemperatura;
}
if(explode(":",explode(" ", $fechaSG)[1])[0]=="11"){
      $th11=$f->valorTemperatura;
}
if(explode(":",explode(" ", $fechaSG)[1])[0]=="12"){
      $th12=$f->valorTemperatura;
}
if(explode(":",explode(" ", $fechaSG)[1])[0]=="13"){
      $th13=$f->valorTemperatura;
}
if(explode(":",explode(" ", $fechaSG)[1])[0]=="14"){
      $th14=$f->valorTemperatura;
}
if(explode(":",explode(" ", $fechaSG)[1])[0]=="15"){
      $th15=$f->valorTemperatura;
}
if(explode(":",explode(" ", $fechaSG)[1])[0]=="16"){
      $th16=$f->valorTemperatura;
}
```

```
	if(explode(":",explode(" ", $fechaSG)[1])[0]=="17"){
		$th17=$f->valorTemperatura;
	}
	if(explode(":",explode(" ", $fechaSG)[1])[0]=="18"){
		$th18=$f->valorTemperatura;
	}
	if(explode(":",explode(" ", $fechaSG)[1])[0]=="19"){
		$th19=$f->valorTemperatura;
	}
	if(explode(":",explode(" ", $fechaSG)[1])[0]=="20"){
		$th20=$f->valorTemperatura;
	}
	if(explode(":",explode(" ", $fechaSG)[1])[0]=="21"){
		$th21=$f->valorTemperatura;
	}
	if(explode(":",explode(" ", $fechaSG)[1])[0]=="22"){
		$th22=$f->valorTemperatura;
	}
	if(explode(":",explode(" ", $fechaSG)[1])[0]=="23"){
		$th23=$f->valorTemperatura;
	}
}
$res = $mysqli->query("SELECT max(valorNumeroVecesPuerta) as maximo FROM
sensoresgenerales where Fecha like '$fechaBusqueda%'");
$puertaSG=0;
```

```
while($f = $res->fetch_object()){
	$puertaSG=$f->maximo;
}
?>
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<script>
window.onload = function () {

var chart = new CanvasJS.Chart("chartContainer", {
	theme:"light2",
	animationEnabled: true,
	title:{
		text: "Voltaje"
	},
	axisY :{
		includeZero: false,
		title: "Voltios",
		suffix: "V"
	},
	toolTip: {
		shared: "true"
	},
```

```
legend:{
      cursor:"pointer",
      itemclick : toggleDataSeries
},
data: [{
      type: "spline",
      visible: true,
      showInLegend: true,
      yValueFormatString: "##.00V",
      name: "Línea 1",
      dataPoints: [
            { label: "00h00", y: <?php echo $vl1h0;?> },
            { label: "01h00", y: <?php echo $vl1h1;?> },
            { label: "02h00", y: <?php echo $vl1h2;?> },
            { label: "03h00", y: <?php echo $vl1h3;?> },
            { label: "04h00", y: <?php echo $vl1h4;?> },
            { label: "05h00", y: <?php echo $vl1h5;?> },
            { label: "06h00", y: <?php echo $vl1h6;?> },
            { label: "07h00", y: <?php echo $vl1h7;?> },
            { label: "08h00", y: <?php echo $vl1h8;?> },
            { label: "09h00", y: <?php echo $vl1h9;?> },
            { label: "10h00", y: <?php echo $vl1h10;?> },
            { label: "11h00", y: <?php echo $vl1h11;?> },
            { label: "12h00", y: <?php echo $vl1h12;?> },
            { label: "13h00", y: <?php echo $vl1h13;?> },
```

```
            { label: "14h00", y: <?php echo $vl1h14;?> },
            { label: "15h00", y: <?php echo $vl1h15;?> },
            { label: "16h00", y: <?php echo $vl1h16;?> },
            { label: "17h00", y: <?php echo $vl1h17;?> },
            { label: "18h00", y: <?php echo $vl1h18;?> },
            { label: "19h00", y: <?php echo $vl1h19;?> },
            { label: "20h00", y: <?php echo $vl1h20;?> },
            { label: "21h00", y: <?php echo $vl1h21;?> },
            { label: "22h00", y: <?php echo $vl1h22;?> },
            { label: "23h00", y: <?php echo $vl1h23;?> }
      ]
},
{
      type: "spline",
      showInLegend: true,
      visible: true,
      yValueFormatString: "##.00V",
      name: "Línea 2",
      dataPoints: [
            { label: "00h00", y: <?php echo $vl2h0;?> },
            { label: "01h00", y: <?php echo $vl2h1;?> },
            { label: "02h00", y: <?php echo $vl2h2;?> },
            { label: "03h00", y: <?php echo $vl2h3;?> },
            { label: "04h00", y: <?php echo $vl2h4;?> },
            { label: "05h00", y: <?php echo $vl2h5;?> },
```

```
            { label: "06h00", y: <?php echo $vl2h6;?> },
            { label: "07h00", y: <?php echo $vl2h7;?> },
            { label: "08h00", y: <?php echo $vl2h8;?> },
            { label: "09h00", y: <?php echo $vl2h9;?> },
            { label: "10h00", y: <?php echo $vl2h10;?> },
            { label: "11h00", y: <?php echo $vl2h11;?> },
            { label: "12h00", y: <?php echo $vl2h12;?> },
            { label: "13h00", y: <?php echo $vl2h13;?> },
            { label: "14h00", y: <?php echo $vl2h14;?> },
            { label: "15h00", y: <?php echo $vl2h15;?> },
            { label: "16h00", y: <?php echo $vl2h16;?> },
            { label: "17h00", y: <?php echo $vl2h17;?> },
            { label: "18h00", y: <?php echo $vl2h18;?> },
            { label: "19h00", y: <?php echo $vl2h19;?> },
            { label: "20h00", y: <?php echo $vl2h20;?> },
            { label: "21h00", y: <?php echo $vl2h21;?> },
            { label: "22h00", y: <?php echo $vl2h22;?> },
            { label: "23h00", y: <?php echo $vl2h23;?> }
        ]
},
{
        type: "spline",
        visible: true,
        showInLegend: true,
        yValueFormatString: "##.00V",
```

```
name: "Línea 3",
dataPoints: [
      { label: "00h00", y: <?php echo $vl3h0;?> },
      { label: "01h00", y: <?php echo $vl3h1;?> },
      { label: "02h00", y: <?php echo $vl3h2;?> },
      { label: "03h00", y: <?php echo $vl3h3;?> },
      { label: "04h00", y: <?php echo $vl3h4;?> },
      { label: "05h00", y: <?php echo $vl3h5;?> },
      { label: "06h00", y: <?php echo $vl3h6;?> },
      { label: "07h00", y: <?php echo $vl3h7;?> },
      { label: "08h00", y: <?php echo $vl3h8;?> },
      { label: "09h00", y: <?php echo $vl3h9;?> },
      { label: "10h00", y: <?php echo $vl3h10;?> },
      { label: "11h00", y: <?php echo $vl3h11;?> },
      { label: "12h00", y: <?php echo $vl3h12;?> },
      { label: "13h00", y: <?php echo $vl3h13;?> },
      { label: "14h00", y: <?php echo $vl3h14;?> },
      { label: "15h00", y: <?php echo $vl3h15;?> },
      { label: "16h00", y: <?php echo $vl3h16;?> },
      { label: "17h00", y: <?php echo $vl3h17;?> },
      { label: "18h00", y: <?php echo $vl3h18;?> },
      { label: "19h00", y: <?php echo $vl3h19;?> },
      { label: "20h00", y: <?php echo $vl3h20;?> },
      { label: "21h00", y: <?php echo $vl3h21;?> },
      { label: "22h00", y: <?php echo $vl3h22;?> },
```

```
                { label: "23h00", y: <?php echo $vl3h23;?> }
            ]
        }]
});
var chart2 = new CanvasJS.Chart("chartContainer2", {
        theme:"light2",
        animationEnabled: true,
        title:{
            text: "Corriente"
        },
        axisY :{
            includeZero: false,
            title: "Amperios",
            suffix: "A"
        },
        toolTip: {
            shared: "true"
        },
        legend:{
            cursor:"pointer",
            itemclick : toggleDataSeries
        },
        data: [{
            type: "spline",
            visible: true,
```

```
showInLegend: true,
yValueFormatString: "##.00A",
name: "Línea 1",
dataPoints: [
      { label: "00h00", y: <?php echo $cl1h0;?> },
      { label: "01h00", y: <?php echo $cl1h1;?> },
      { label: "02h00", y: <?php echo $cl1h2;?> },
      { label: "03h00", y: <?php echo $cl1h3;?> },
      { label: "04h00", y: <?php echo $cl1h4;?> },
      { label: "05h00", y: <?php echo $cl1h5;?> },
      { label: "06h00", y: <?php echo $cl1h6;?> },
      { label: "07h00", y: <?php echo $cl1h7;?> },
      { label: "08h00", y: <?php echo $cl1h8;?> },
      { label: "09h00", y: <?php echo $cl1h9;?> },
      { label: "10h00", y: <?php echo $cl1h10;?> },
      { label: "11h00", y: <?php echo $cl1h11;?> },
      { label: "12h00", y: <?php echo $cl1h12;?> },
      { label: "13h00", y: <?php echo $cl1h13;?> },
      { label: "14h00", y: <?php echo $cl1h14;?> },
      { label: "15h00", y: <?php echo $cl1h15;?> },
      { label: "16h00", y: <?php echo $cl1h16;?> },
      { label: "17h00", y: <?php echo $cl1h17;?> },
      { label: "18h00", y: <?php echo $cl1h18;?> },
      { label: "19h00", y: <?php echo $cl1h19;?> },
      { label: "20h00", y: <?php echo $cl1h20;?> },
```

```
            { label: "21h00", y: <?php echo $cl1h21;?> },
            { label: "22h00", y: <?php echo $cl1h22;?> },
            { label: "23h00", y: <?php echo $cl1h23;?> }
      ]
},
{
      type: "spline",
      showInLegend: true,
      visible: true,
      yValueFormatString: "##.00A",
      name: "Línea 2",
      dataPoints: [
            { label: "00h00", y: <?php echo $cl2h0;?> },
            { label: "01h00", y: <?php echo $cl2h1;?> },
            { label: "02h00", y: <?php echo $cl2h2;?> },
            { label: "03h00", y: <?php echo $cl2h3;?> },
            { label: "04h00", y: <?php echo $cl2h4;?> },
            { label: "05h00", y: <?php echo $cl2h5;?> },
            { label: "06h00", y: <?php echo $cl2h6;?> },
            { label: "07h00", y: <?php echo $cl2h7;?> },
            { label: "08h00", y: <?php echo $cl2h8;?> },
            { label: "09h00", y: <?php echo $cl2h9;?> },
            { label: "10h00", y: <?php echo $cl2h10;?> },
            { label: "11h00", y: <?php echo $cl2h11;?> },
            { label: "12h00", y: <?php echo $cl2h12;?> },
```

```
            { label: "13h00", y: <?php echo $cl2h13;?> },
            { label: "14h00", y: <?php echo $cl2h14;?> },
            { label: "15h00", y: <?php echo $cl2h15;?> },
            { label: "16h00", y: <?php echo $cl2h16;?> },
            { label: "17h00", y: <?php echo $cl2h17;?> },
            { label: "18h00", y: <?php echo $cl2h18;?> },
            { label: "19h00", y: <?php echo $cl2h19;?> },
            { label: "20h00", y: <?php echo $cl2h20;?> },
            { label: "21h00", y: <?php echo $cl2h21;?> },
            { label: "22h00", y: <?php echo $cl2h22;?> },
            { label: "23h00", y: <?php echo $cl2h23;?> }
      ]
},
{
      type: "spline",
      visible: true,
      showInLegend: true,
      yValueFormatString: "##.00A",
      name: "Línea 3",
      dataPoints: [
            { label: "00h00", y: <?php echo $cl3h0;?> },
            { label: "01h00", y: <?php echo $cl3h1;?> },
            { label: "02h00", y: <?php echo $cl3h2;?> },
            { label: "03h00", y: <?php echo $cl3h3;?> },
            { label: "04h00", y: <?php echo $cl3h4;?> },
```

```
                { label: "05h00", y: <?php echo $cl3h5;?> },
                { label: "06h00", y: <?php echo $cl3h6;?> },
                { label: "07h00", y: <?php echo $cl3h7;?> },
                { label: "08h00", y: <?php echo $cl3h8;?> },
                { label: "09h00", y: <?php echo $cl3h9;?> },
                { label: "10h00", y: <?php echo $cl3h10;?> },
                { label: "11h00", y: <?php echo $cl3h11;?> },
                { label: "12h00", y: <?php echo $cl3h12;?> },
                { label: "13h00", y: <?php echo $cl3h13;?> },
                { label: "14h00", y: <?php echo $cl3h14;?> },
                { label: "15h00", y: <?php echo $cl3h15;?> },
                { label: "16h00", y: <?php echo $cl3h16;?> },
                { label: "17h00", y: <?php echo $cl3h17;?> },
                { label: "18h00", y: <?php echo $cl3h18;?> },
                { label: "19h00", y: <?php echo $cl3h19;?> },
                { label: "20h00", y: <?php echo $cl3h20;?> },
                { label: "21h00", y: <?php echo $cl3h21;?> },
                { label: "22h00", y: <?php echo $cl3h22;?> },
                { label: "23h00", y: <?php echo $cl3h23;?> }
            ]
        }]
});
var chart3 = new CanvasJS.Chart("chartContainer3", {
        theme:"light2",
        animationEnabled: true,
```

```
title:{
      text: "Potencia"
},
axisY :{
      includeZero: false,
      title: "Watts",
      suffix: "W"
},
toolTip: {
      shared: "true"
},
legend:{
      cursor:"pointer",
      itemclick : toggleDataSeries
},
data: [{
      type: "spline",
      visible: true,
      showInLegend: true,
      yValueFormatString: "##.00W",
      name: "Línea 1",
      dataPoints: [
            { label: "00h00", y: <?php echo $pl1h0;?> },
            { label: "01h00", y: <?php echo $pl1h1;?> },
            { label: "02h00", y: <?php echo $pl1h2;?> },
```

```
            { label: "03h00", y: <?php echo $pl1h3;?> },
            { label: "04h00", y: <?php echo $pl1h4;?> },
            { label: "05h00", y: <?php echo $pl1h5;?> },
            { label: "06h00", y: <?php echo $pl1h6;?> },
            { label: "07h00", y: <?php echo $pl1h7;?> },
            { label: "08h00", y: <?php echo $pl1h8;?> },
            { label: "09h00", y: <?php echo $pl1h9;?> },
            { label: "10h00", y: <?php echo $pl1h10;?> },
            { label: "11h00", y: <?php echo $pl1h11;?> },
            { label: "12h00", y: <?php echo $pl1h12;?> },
            { label: "13h00", y: <?php echo $pl1h13;?> },
            { label: "14h00", y: <?php echo $pl1h14;?> },
            { label: "15h00", y: <?php echo $pl1h15;?> },
            { label: "16h00", y: <?php echo $pl1h16;?> },
            { label: "17h00", y: <?php echo $pl1h17;?> },
            { label: "18h00", y: <?php echo $pl1h18;?> },
            { label: "19h00", y: <?php echo $pl1h19;?> },
            { label: "20h00", y: <?php echo $pl1h20;?> },
            { label: "21h00", y: <?php echo $pl1h21;?> },
            { label: "22h00", y: <?php echo $pl1h22;?> },
            { label: "23h00", y: <?php echo $pl1h23;?> }
        ]
    },
    {
        type: "spline",
```

```
showInLegend: true,
visible: true,
yValueFormatString: "##.00W",
name: "Línea 2",
dataPoints: [
      { label: "00h00", y: <?php echo $pl2h0;?> },
      { label: "01h00", y: <?php echo $pl2h1;?> },
      { label: "02h00", y: <?php echo $pl2h2;?> },
      { label: "03h00", y: <?php echo $pl2h3;?> },
      { label: "04h00", y: <?php echo $pl2h4;?> },
      { label: "05h00", y: <?php echo $pl2h5;?> },
      { label: "06h00", y: <?php echo $pl2h6;?> },
      { label: "07h00", y: <?php echo $pl2h7;?> },
      { label: "08h00", y: <?php echo $pl2h8;?> },
      { label: "09h00", y: <?php echo $pl2h9;?> },
      { label: "10h00", y: <?php echo $pl2h10;?> },
      { label: "11h00", y: <?php echo $pl2h11;?> },
      { label: "12h00", y: <?php echo $pl2h12;?> },
      { label: "13h00", y: <?php echo $pl2h13;?> },
      { label: "14h00", y: <?php echo $pl2h14;?> },
      { label: "15h00", y: <?php echo $pl2h15;?> },
      { label: "16h00", y: <?php echo $pl2h16;?> },
      { label: "17h00", y: <?php echo $pl2h17;?> },
      { label: "18h00", y: <?php echo $pl2h18;?> },
      { label: "19h00", y: <?php echo $pl2h19;?> },
```

```
            { label: "20h00", y: <?php echo $pl2h20;?> },
            { label: "21h00", y: <?php echo $pl2h21;?> },
            { label: "22h00", y: <?php echo $pl2h22;?> },
            { label: "23h00", y: <?php echo $pl2h23;?> }
      ]
},
{
      type: "spline",
      visible: true,
      showInLegend: true,
      yValueFormatString: "##.00W",
      name: "Línea 3",
      dataPoints: [
            { label: "00h00", y: <?php echo $pl3h0;?> },
            { label: "01h00", y: <?php echo $pl3h1;?> },
            { label: "02h00", y: <?php echo $pl3h2;?> },
            { label: "03h00", y: <?php echo $pl3h3;?> },
            { label: "04h00", y: <?php echo $pl3h4;?> },
            { label: "05h00", y: <?php echo $pl3h5;?> },
            { label: "06h00", y: <?php echo $pl3h6;?> },
            { label: "07h00", y: <?php echo $pl3h7;?> },
            { label: "08h00", y: <?php echo $pl3h8;?> },
            { label: "09h00", y: <?php echo $pl3h9;?> },
            { label: "10h00", y: <?php echo $pl3h10;?> },
            { label: "11h00", y: <?php echo $pl3h11;?> },
```

```
                { label: "12h00", y: <?php echo $pl3h12;?> },
                { label: "13h00", y: <?php echo $pl3h13;?> },
                { label: "14h00", y: <?php echo $pl3h14;?> },
                { label: "15h00", y: <?php echo $pl3h15;?> },
                { label: "16h00", y: <?php echo $pl3h16;?> },
                { label: "17h00", y: <?php echo $pl3h17;?> },
                { label: "18h00", y: <?php echo $pl3h18;?> },
                { label: "19h00", y: <?php echo $pl3h19;?> },
                { label: "20h00", y: <?php echo $pl3h20;?> },
                { label: "21h00", y: <?php echo $pl3h21;?> },
                { label: "22h00", y: <?php echo $pl3h22;?> },
                { label: "23h00", y: <?php echo $pl3h23;?> }
            ]
        }]
});
var chart4 = new CanvasJS.Chart("chartContainer4", {
        theme:"light2",
        animationEnabled: true,
        title:{
            text: "Temperatura"
        },
        axisY :{
            includeZero: false,
            title: "Grados Centrigrados",
            suffix: "°C"
```

```
},
toolTip: {
      shared: "true"
},
legend:{
      cursor:"pointer",
      itemclick : toggleDataSeries
},
data: [{
      type: "spline",
      visible: true,
      showInLegend: true,
      yValueFormatString: "##.00°C",
      name: "Temperatura",
      dataPoints: [
            { label: "00h00", y: <?php echo $th0;?> },
            { label: "01h00", y: <?php echo $th1;?> },
            { label: "02h00", y: <?php echo $th2;?> },
            { label: "03h00", y: <?php echo $th3;?> },
            { label: "04h00", y: <?php echo $th4;?> },
            { label: "05h00", y: <?php echo $th5;?> },
            { label: "06h00", y: <?php echo $th6;?> },
            { label: "07h00", y: <?php echo $th7;?> },
            { label: "08h00", y: <?php echo $th8;?> },
            { label: "09h00", y: <?php echo $th9;?> },
```

```
                { label: "10h00", y: <?php echo $th10;?> },
                { label: "11h00", y: <?php echo $th11;?> },
                { label: "12h00", y: <?php echo $th12;?> },
                { label: "13h00", y: <?php echo $th13;?> },
                { label: "14h00", y: <?php echo $th14;?> },
                { label: "15h00", y: <?php echo $th15;?> },
                { label: "16h00", y: <?php echo $th16;?> },
                { label: "17h00", y: <?php echo $th17;?> },
                { label: "18h00", y: <?php echo $th18;?> },
                { label: "19h00", y: <?php echo $th19;?> },
                { label: "20h00", y: <?php echo $th20;?> },
                { label: "21h00", y: <?php echo $th21;?> },
                { label: "22h00", y: <?php echo $th22;?> },
                { label: "23h00", y: <?php echo $th23;?> }
            ]
      }]
});
chart.render();
chart2.render();
chart3.render();
chart4.render();

function toggleDataSeries(e) {
      if     (typeof(e.dataSeries.visible)     ===     "undefined"     ||
e.dataSeries.visible ){
```

```
            e.dataSeries.visible = false;
      } else {
            e.dataSeries.visible = true;
      }
      chart.render();
      chart2.render();
      chart3.render();
      chart4.render();
}



}

$(document).ready(function(){
    $('.counter-value').each(function(){
        $(this).prop('Counter',0).animate({
            Counter: $(this).text()
        },{
            duration: 3500,
            easing: 'swing',
            step: function (now){
                $(this).text(Math.ceil(now));
            }
        });
    });
```

```
});
</script>
<script    type="text/javascript"    src="https://code.jquery.com/jquery-1.12.0.min.js"></script>
 <!-- Latest compiled and minified JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"                                                integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous"></script>

<style>
#body {
  font-family: Arial;
  padding: 25px;
  background-color: #f5f5f5;
  color: #808080;
      text-align: center;
}
.counter{
    color: #4834d4;
    background: linear-gradient(-27deg,#4834d4 50%,#686de0 51%);
    font-family: 'Ubuntu', sans-serif;
    text-align: center;
    height: 210px;
    width: 210px;
    padding: 0 20px;
```

```
    margin: 0 auto;
    position: relative;
    -webkit-clip-path: polygon(50% 0%, 100% 25%, 100% 75%, 50% 100%, 0% 75%, 0% 25%);
    clip-path: polygon(50% 0%, 100% 25%, 100% 75%, 50% 100%, 0% 75%, 0% 25%);
    transition: all 0.3s;
}
.counter:after{
    content: "";
    background: #fff;
    width: 85%;
    height: 85%;
    position: absolute;
    top: 15px;
    left: 15px;
    z-index: -1;
    -webkit-clip-path: polygon(50% 0%, 100% 25%, 100% 75%, 50% 100%, 0% 75%, 0% 25%);
    clip-path: polygon(50% 0%, 100% 25%, 100% 75%, 50% 100%, 0% 75%, 0% 25%);
}
.counter .counter-icon{
    color: #fff;
    background: linear-gradient(-27deg,#4834d4 49%,#686de0 50%);
    font-size: 40px;
```

```
    line-height: 70px;
    width: 120px;
    height: 65px;
    margin: 0 auto 20px;
    border-radius: 0 0 15px 15px;
    box-shadow: 0 0 10px rgba(0,0,0,0.8);
}
.counter .counter-value{
    font-size: 35px;
    font-weight: 600;
    line-height: 20px;
    margin: 0 0 15px;
    display: block;
    transition: all 0.3s ease;
}
.counter:hover .counter-value{ text-shadow: 2px 2px 0 #999; }
.counter h3{
    font-size: 17px;
    font-weight: 500;
    text-transform: uppercase;
    margin: 0;
    display: inline-block;
}
.counter.yellow{ color:#f39c12; }
.counter.yellow,
```

```
.counter.yellow .counter-icon{
    background: linear-gradient(-27deg,#f39c12 49%,#f1c40f 50%);
}
.counter.orange{ color:#d35400; }
.counter.orange,
.counter.orange .counter-icon{
    background: linear-gradient(-27deg,#d35400 49%,#e67e22 50%);
}
.counter.pink{ color:#be2edd; }
.counter.pink,
.counter.pink .counter-icon{
    background: linear-gradient(-27deg,#be2edd 49%,#e056fd 50%);
}
@media screen and (max-width:990px){
    .counter{ margin-bottom: 40px; }
}
</style>
</head>
<body>
<div id="chartContainer" style="height: 370px; max-width: 920px; margin: 0px auto;"></div>
<div id="chartContainer2" style="height: 370px; max-width: 920px; margin: 0px auto;"></div>
<div id="chartContainer3" style="height: 370px; max-width: 920px; margin: 0px auto;"></div>
```

```
<div id="chartContainer4" style="height: 370px; max-width: 920px; margin: 0px auto;"></div>
<div class="container">
    <div class="row">
        <div class="col-md-3 col-sm-6">
            <div class="counter yellow">
                <div class="counter-icon">
                    <i class="fa fa-briefcase"></i>
                </div>
                <span class="counter-value"><?php echo $puertaSG;?></span>
                <h3>Veces se abrió la puerta</h3>
            </div>
        </div>
    </div>
</div>
<script src="canvasjs.min.js"></script>
</body>
</html>
```

ORDEN DE EMPASTADO