UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

“MONITOREO DE SEÑALES DIGITALES Y ANALÓGICAS POR MEDIO DE MÓDULO GPRS CON UNA INTERFAZ A PC VIA TCP/IP”

Trabajo de graduación en modalidad de Tesis presentado por

Bryan Allan Roberto Hidalgo Montenegro

para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2016

“MONITOREO DE SEÑALES DIGITALES Y ANALÓGICAS POR MEDIO DE MÓDULO GPRS CON UNA INTERFAZ A PC VIA TCP/IP”

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

“MONITOREO DE SEÑALES DIGITALES Y ANALÓGICAS POR MEDIO DE MÓDULO GPRS CON UNA INTERFAZ A PC VIA TCP/IP”

Trabajo de graduación en modalidad de Tesis presentado por

Bryan Allan Roberto Hidalgo Montenegro

para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2016

Vo. Bo. :

(f) _____

MSc. Carlos Esquit

Tribunal Examinador:

(f) _____

MSc. Carlos Esquit

(f) _____

Ing. Julio Vásquez

(f) _____

Ing. Pablo Mazariegos

Fecha de aprobación: Guatemala, 15 de junio de 2016

ÍNDICE GENERAL

LISTA DE FIGURAS

Página

RESUMEN

En la industria existen sistemas que se encuentran a distancias considerables de una estación central. El monitoreo de estos equipos representa un reto dentro de la infraestructura física necesaria. Este proyecto consiste en resolver el problema utilizando tecnología de red celular y electrónica aplicada, para que sea posible monitorear y modificar el estado de cualquier sensor o dispositivo a distancia.

El diseño presentado utiliza como unidad central de procesamiento un sistema integrado llamado Raspberry PI. Utilizando sus diversos módulos de hardware se recolecta el valor de las señales y es enviada por medio de red celular. Esto permite una comunicación Cliente-Servidor que ofrece un enlace inalámbrico para conocer el estado de las señales de manera remota.

Las señales son monitoreadas mediante una página web, sirviendo de enlace entre el usuario y el dispositivo. Desde la web se pueden realizar cambios o ajustes en los sensores que estén en uso, cuyos valores monitoreados se almacenan en una base de datos, lo cual permite un control histórico de las señales medidas.

Este dispositivo está diseñado para uso industrial, de cuyas señales resultan 8 entradas y 8 salidas digitales, con un voltaje de 0 o 24 volts, que es lo estándar en esta aplicación; también posee 4 entradas y 4 salidas analógicas, que cuentan con un voltaje en el rango de 0 a 10 volts. El sistema cuenta con una interfaz GUI por medio de una pantalla táctil.

I. INTRODUCCIÓN

En la actualidad, se vive en un mundo donde la tecnología es imprescindible para el ser humano, sobre todo en la industria, donde diversos tipos de dispositivos electrónicos permiten controles automáticos y procesos que son cada día más rápidos y eficientes.

Sin embargo, en la mayoría de las industrias encontramos una cantidad enorme de cableado entre el sensor o actuador hacia el computador principal, de donde se resguardan las bases de datos de dicho sensores o dispositivos. Debido a que, en la industria se puede requerir varios sensores o actuadores por máquina, lo que lleva a una saturación de cableado, que trasladan la información hacia la base de datos y que pueden llegar a interferir u obstaculizar diferentes áreas en la empresa, repercutiendo en la en los mismos sistemas electrónicos y elevar los costos.

En este trabajo de graduación, se busca como objetivo principal elaborar un dispositivo electrónico, que opere a nivel industrial, a través de conexión de diferentes sensores o dispositivos, que sea capaz de guardar inalámbricamente subregistros de cada sensor o dispositivo conectado, con una base de datos almacenada en un servidor web. Para lograr la implementación de una plataforma de control y monitoreo remoto, lo cual es un importante potencial de resolución de varias limitantes en la automatización de la industria en Guatemala.

Este proyecto se elaboró, mediante un Raspberry Pi, haciendo uso de sus periféricos y librerías integradas dentro de la programación del mismo. Al dispositivo se integraron 8 entradas digitales de 24 volts, 8 salidas digitales de 24 volts, 4 entradas analógicas, 4 salidas analógicas, cada señal analógica cuenta con un voltaje en el rango de 0 a 10 volts. Para cada una de estas señales se utilizaron diferentes módulos electrónicos programados mediante Python (en el Raspberry PI) y PHP (en la web) para su completo funcionamiento. Adicionalmente tiene una pantalla táctil, para facilitar el uso del dispositivo.

Para la transmisión de datos inalámbricamente se utiliza un módulo de hardware que se conecta a la red celular. La conexión del módulo de hardware con el sistema

integrado es por medio del protocolo USB. Alternativamente se puede utilizar un módulo que transmite la información por medio del protocolo WIFI.

II. OBJETIVOS

A. General

Diseñar y programar un módulo que permita el control y monitoreo de diferentes señales de entradas utilizando la red celular.

B. Específicos

1. Programar en un Raspberry PI, una aplicación específica utilizando sus diferentes módulos integrados de hardware.
2. El dispositivo industrial se diseñará para 8 entradas digitales, 8 salidas digitales, 4 entradas analógicas y 4 salidas analógicas que cumplirán con los estándares a nivel industrial.
3. Diseñar una plataforma web, que almacene los datos monitoreados por el dispositivo y que establezca el enlace humano-máquina. El dispositivo será manuable desde la plataforma web.
4. Crear una interfaz sencilla, con pantalla táctil, de fácil operación para el usuario y este producto pueda ser utilizado por cualquier técnico eléctrico en el área industrial.

III. JUSTIFICACIÓN

En la industria muchas veces se utiliza una gran cantidad de sensores, y en la mayoría de las veces, la cantidad de cableado conectado hacia la estación de monitoreo, se puede convertir a parte de innecesario, un gasto considerable. Además, puede afectar la estética del lugar de trabajo e incluso incrementar el riesgo de algún accidente, que represente consecuencias para la empresa. Por todo esto, poder llevar el monitoreo y control de cada uno de los sensores puede volverse complicado debido a la gran cantidad de conexiones que pueden ser requeridas y que deben extenderse hasta la estación central de control.

Las tecnologías inalámbricas han evolucionado demasiado en los últimos años, abriendo la puerta a desarrolladores para hacer implementaciones más sencillas y prácticas. Sumando al hecho mencionado anteriormente, la infraestructura de comunicaciones celulares que brinda un servicio de cobertura amplia, facilitando la implementación de estas tecnologías incluso en áreas remotas.

Utilizando este dispositivo es posible organizar de manera más eficiente los diferentes nodos, e identificar cada dispositivo de manera independiente, lo que ayuda a que cualquier problema se puede diagnosticar de manera remota y eficaz. Incluso la aplicación puede crear alarmas, si alguna señal se pierde o sobrepasa los límites establecidos.

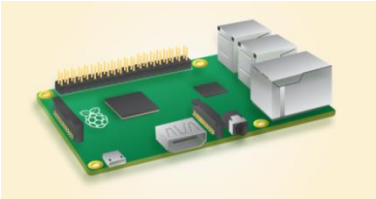
Además, se desea como resultado final que el proyecto pueda ser utilizado en la industria y de esa manera hacer que la industria en Guatemala evolucione, de manera paralela con la tecnología actual y no se quede aislada mundialmente.

IV. MARCO TEÓRICO

A. Raspberry PI

Es un ordenador en una placa de bajo costo, en el cual facilita el desarrollo de proyectos electrónicos. Este dispositivo se conecta a un monitor, un ratón y a un teclado para para su funcionamiento. El lenguaje de programación utilizado en este ordenador es Python. Este dispositivo fue fabricado por la empresa Raspberry Pi Foundation, que es una organización benéfica ubicada en el Reino unido (Raspberry Pi Foundation, 2013).

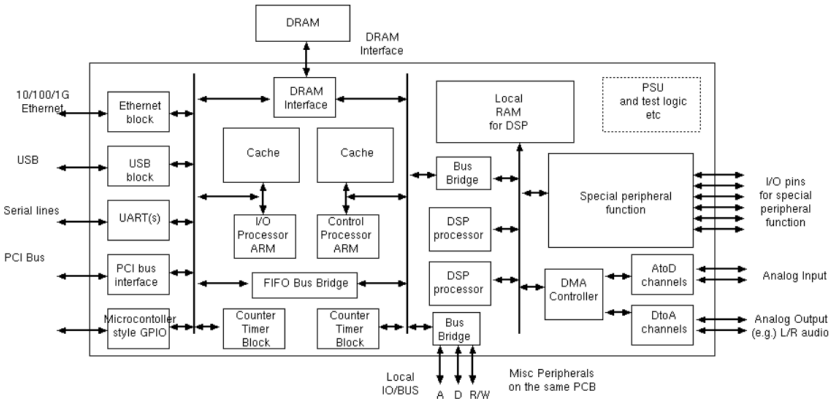
Figura 1. Raspberry Pi.



(Copyright 2015 por Raspberry Pi fundation.)

Este dispositivo tiene un diseño de SoC (System on chip) Broadcom BCM2835, que integra diferentes componentes en un sistema. La diferencia entre un SoC y un microcontrolador, es que los microcontroladores tienen pocos kilobytes de memoria RAM. La arquitectura de un SoC está basada por un microcontrolador, microprocesador, módulos de memoria (ROM, RAM, EEPROM), componentes periféricos, interfaces externas como SPI. Estos módulos interconectados están de acuerdo con estándares industriales de conexión de buses.

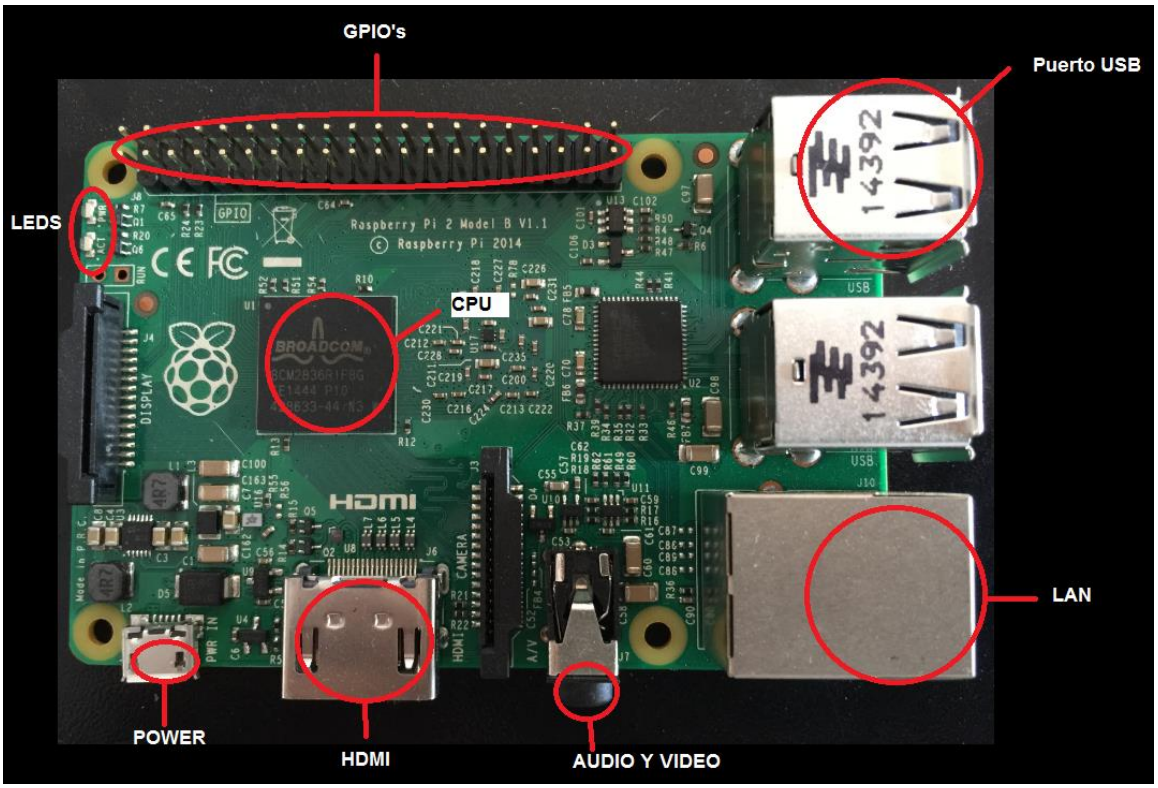
Figura 2. Arquitectura de un SoC.



(Copyright 2015 por Raspberry Pi fundation.

El Raspberry pi tiene un procesador central (CPU) ARM1176JZF-S a 700 MHz, un procesador gráfico (GPU) Video Core IV y una memoria RAM de 512 MB. Usa una tarjeta SD para el almacenamiento permanente, salidas de video, HDMI, audio y puerto Ethernet.

Figura 3. Componentes Raspberry Pi.



(“Elaboración Propia”)

Tiene un sistema operativo Raspbian, que es un sistema operativo gratuito Linux basado en Debian optimizado para el hardware de un Raspberry Pi.

B. Lenguaje Python

Es un lenguaje de programación creado por Guido van Rossum, tiene una sintaxis muy limpia y que favorece un código legible. Python es un lenguaje muy expresivo y sus programas suelen ser cortos. Es considerado como un lenguaje de programación de muy alto nivel. Es un lenguaje que permite facilidad en la lectura de los programas, así como también facilita la realización de pruebas ya que ofrece un entorno interactivo.

El entorno de ejecución detecta con facilidad los errores en la programación. Es un lenguaje de programación multiparadigma, ya que permite varios estilos en la programación, como programación orientada a objetos, programación imperativa procedimental y programación funcional. Además, posee un rico juego de estructura de datos que se pueden manipular de modo sencillo.

C. Lenguaje PHP (Hypertext Preprocessor)

Es un lenguaje de programación de código abierto diseñado adecuado para desarrollo web y puede ser incrustado en HTML (HyperText Markup Languaje). Fue creado por Rasmus Lerdfor. Lo que distingue a PHP es que el cliente recibirá el resultado de ejecutar el script. PHP se distingue por su simplicidad para el principiante, pero ofrece características avanzadas para los programadores profesionales, también es uno de los lenguajes más flexibles y de alto rendimiento. PHP puede emplearse en todos los sistemas operativos principales y admite la mayoría de los servidores web hoy en día.

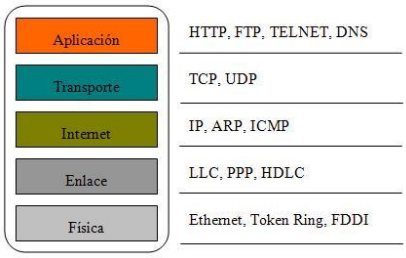
Una de las características más potentes de PHP es su soporte para una amplitud en la base de datos. Simplicidad en la escritura de una página web con acceso de una base de datos, cuenta con soporte para la comunicación con otros servicios usando protocolos como HTTP, LDAP, COM, POP3 y muchos otros. Así como también tiene soporte para la instalación de objetos Java.

D. Protocolo HTTP (Hypertext Transfer Protocol)

Es un protocolo sencillo cliente-servidor, el más utilizado en la Internet. Permite la transferencia de datos o archivos entre un navegador y un servidor web. Fue desarrollado por World Wide Web Consorium y IETF (Internet Engineering Task Force). La información que es transmitida se identifica mediante un URL (localizador uniforme de recursos).

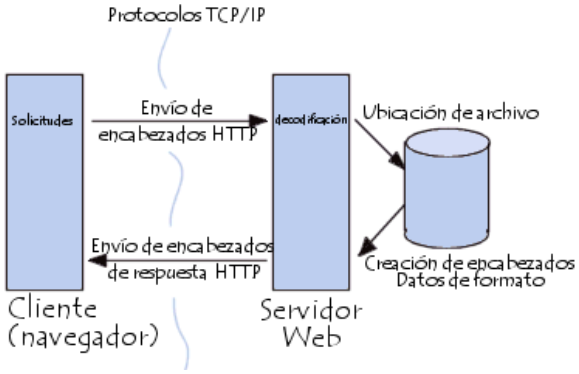
Este protocolo se basa en operaciones sencillas de solicitud/respuesta. Un cliente establece una conexión con un servidor y transmite un mensaje con los datos de la solicitud, el servidor responde con un mensaje que contiene el resultado y el estado de la operación. Es un protocolo en la capa de aplicación, un nivel arriba del TCP/IP como se observa en la Figura 4.

Figura 4. Capas de la red.



(Copyright por CCM Benchmark Group.)

Figura 5. Transferencia de HTTP.



(Copyright por CCM Benchmark Group.)

El protocolo POST (HTTP) envía los datos servidor ubicado en una URL específica. En la programación en PHP, los datos son codificados. Es uno de los métodos de petición soportado en HTTP utilizado por la World Wide Web. Los datos son enviados en el cuerpo del mensaje, se usa para proporcionar un bloque de datos, como un formulario HTML o un proceso de manejo de datos o actualización de datos.

La sintaxis del POST es la siguiente:

POST <URL> <VERSION>
<CABECERA>
<CRLF>
<CUERPO DEL MENSAJE>

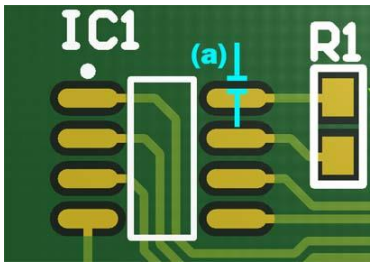
E. PCB (Tarjeta de Circuitos Impresos)

Es una tarjeta o placa de material conductor que está constituida por conexiones o buses que permiten la interconexión eléctrica entre dispositivos. La fabricación de estas placas o tarjetas impresas han ido evolucionando durante los años, debido a la disminución tamaño de los dispositivos eléctricos y la automatización al realizar el diseño de estos circuitos. También se han avanzado en los materiales y las capas que se tienen

en estas placas, debido a que ahora se puede realizar circuitos impresos con varias capas dentro de la tarjeta. La automatización de estas tarjetas, han servido para realizar la producción en masa y economizar los costos por tarjeta.

En los circuitos impresos se requiere de un proceso de ensamblado, este puede ser manual o automatizado. En este proceso de ensamblado se utiliza soldadura para que los componentes electrónicos queden fijos al circuito impreso. Se realiza una máscara a este soldado para evitar accidentes entre diferentes pistas, es un barniz que se coloca a los circuitos impresos que parece un borde a la impresión como se puede observar en la Figura 6.

Figura 6. Máscara de soldado.



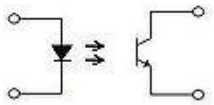
("Elaboración Propia")

F. Optoacoplador

Es un dispositivo electrónico que funciona como aislador eléctrico, está formado por un fotoemisor y un fotoreceptor. Estos dos elementos están acoplados para cuando el diodo LED emite luz, ilumina el fotoreceptor y conduzca. Esta señal luminosa es una ventaja para proteger los circuitos de entrada y salida. Por lo general este dispositivo viene en un encapsulado tipo DIP (Dual in line package).

La corriente de salida del optoacoplador es proporcional a la corriente de del diodo LED. La relación entre estas dos corrientes se llama transferencia de corriente y a mayor temperatura ambiente, la corriente del fototransistor es mayor. El optoacoplador es un dispositivo sensible a la frecuencia.

Figura 7. Optoacoplador.

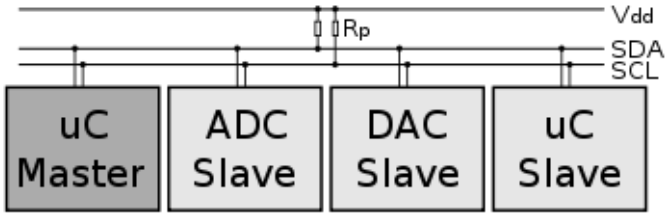


(Copyright por Unicrom)

G. Comunicación I2C (Inter-Integrated Circuit)

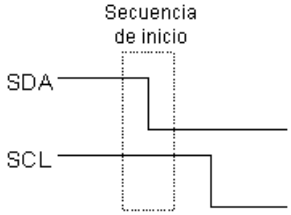
Es un protocolo que permite la conexión de múltiples circuitos integrados digitales llamados esclavos, que se comunican con un maestro. Fue desarrollado por Philips en 1982. Este protocolo utiliza dos líneas de señal, las cuales se llaman SDA (datos en serie) y SCL (sincroniza el sistema). En esta comunicación no hay necesidad de tener una selección de esclavo o chip. Para ello debe cumplir que cada esclavo conectado al bus, debe tener una dirección singular. La velocidad de los datos debe ser elegida entre 100kbps y 3.4Mbps, dependiendo si el usuario quiere usar la comunicación en modo normal o a alta velocidad.

Figura 8. Conexión I2C con varios esclavos.



Esta comunicación es de tipo drenaje abierto, por la cual en la figura anterior se observa que SDA y SCL están polarizados en estado alto o también llamado pull-up. Media vez SDA y SCL permanezcan en estado lógico en alto, cualquier maestro puede ocupar la comunicación, cuando el maestro baja a tierra la línea de SDA, se presenta la secuencia de inicio.

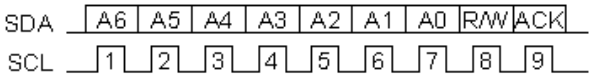
Figura 9. Secuencia de inicio en I2C.



El primer byte transmitido después de colocar el sistema en secuencia de inicio, empleando los primeros 7 bits, estos contienen la dirección del dispositivo (esclavo) que se desea comunicar, el octavo corresponde si se desea escribir o leer. Siendo interpretado como escritura si el nivel lógico es bajo y en lectura si el bit se encuentra en

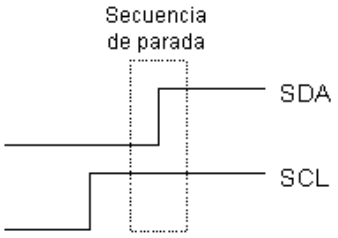
alto. Si se identifica escritura, el esclavo se mantiene esperando hasta que se hayan transmitidos todos los datos del maestro, pero si identifica lectura, el maestro manda pulsos para que el esclavo pueda enviar los datos.

Figura 10. Funcionamiento en intercambio de datos del maestro al esclavo.



Si se desea detener la comunicación del maestro, se debe colocar en alto las dos líneas de comunicación como se observa en la siguiente figura.

Figura 11. Secuencia de parada en la comunicación I2C.



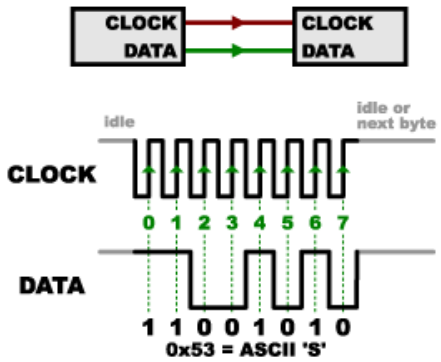
H. Comunicación SPI (Serial Peripheral Interface)

Es un protocolo de datos en serie síncrono utilizado para enviar los datos entre microcontroladores y pequeños periféricos. Este utiliza líneas de reloj y de datos separadas, a parte de una línea de selección para elegir el dispositivo que se desea comunicar. En esta conexión el maestro controla los dispositivos periféricos, en el cual se encuentran las diferentes líneas de comunicación a los esclavos. Los cuales son:

1. MISO (Master in Slave out). Línea de comunicación la cual el esclavo envía datos al maestro.
2. MOSI (Master out Slave in). Línea de comunicación la cual el maestro envía datos a los periféricos.
3. SCK (Serial clock). Línea que los impulsos de reloj sincronizan la transmisión de datos generado por el maestro.
4. SS (Slave Select). Línea que el maestro utiliza para activar o desactivar la comunicación con el esclavo seleccionado.

Esta comunicación utiliza líneas separadas para los datos y para el reloj, que se mantienen en sincronía. El reloj es una señal oscilante y al mismo tiempo manda los datos en cada flanco de la señal del reloj como se ve en la siguiente figura.

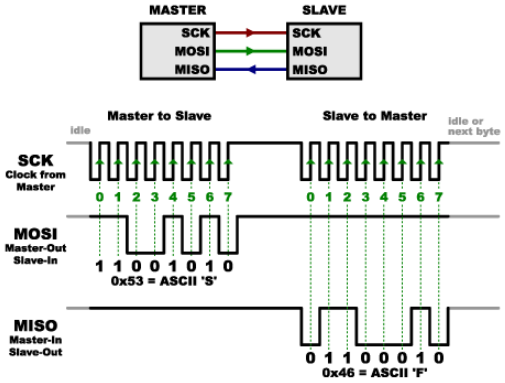
Figura 12. Funcionamiento de la sincronización de datos en SPI.



(Copyright por Prometec.)

Para la recepción de los datos del esclavo hacia el maestra, se utiliza el reloj que mantiene la sincronía de la comunicación conjunto a la línea MISO, que es la que lee los datos que envía el esclavo. En cambio si se quiere escribir al esclavo, se utiliza el mismo reloj síncrono pero con la línea de comunicación MOSI, esta línea es para mandar los datos del maestro hacia el esclavo.

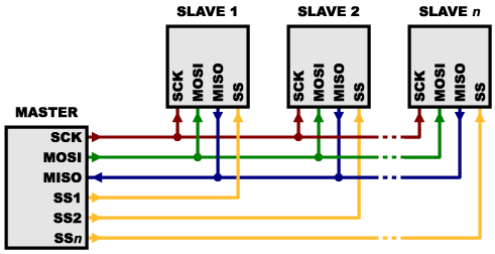
Figura 13. Funcionamiento de la lectura o escritura de datos en SPI.



(Copyright por Prometec.)

Este tipo de comunicación se puede colocar varios periféricos conectados al mismo maestro, esto dependiendo de los pines que tenga disponibles para la selección de los esclavos. No puede haber varios esclavos conectados al mismo tiempo, sino solo uno a la vez.

Figura 14. Conexión de varios esclavos al maestro.

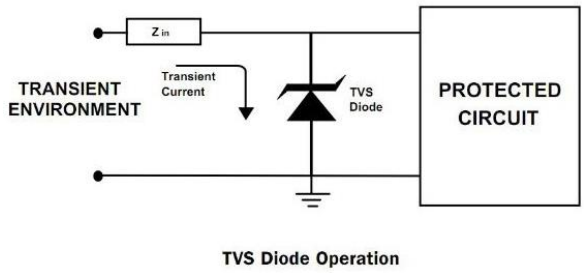


(Copyright por Prometec)

I. TVS

Es un tipo de diodo que permite la supresión de picos en tensiones elevadas. Sus siglas significan Transient Voltage Suppression. Es un protector electrónico sensible que funciona desviando el exceso de corriente cuando el voltaje inducido excede el voltaje que el diodo tiene establecido. Este diodo se restablece automáticamente cuando la sobretensión desaparece, este actúa como un rectificador de voltaje, pero con grandes corrientes de pico durante poco tiempo. El dispositivo se coloca de la entrada del voltaje a tierra como se ve en la siguiente figura.

Figura 15. Conexión del diodo TVS como protección.



(Copyright por semtech.)

V. METODOLOGÍA

Para este trabajo de graduación se realizó una profunda investigación de los tipos de señales y dispositivos más utilizados a nivel industrial. Debido a que este dispositivo tiene que cumplir con los estándares industriales.

Luego se investigó el protocolo HTTP (Protocolo de trasferencia de hipertexto) que es el permite la comunicación inalámbrica entre la base de datos del servidor y el Raspberry PI. Este protocolo utiliza los elementos de la arquitectura web para comunicarse con el servidor remoto.

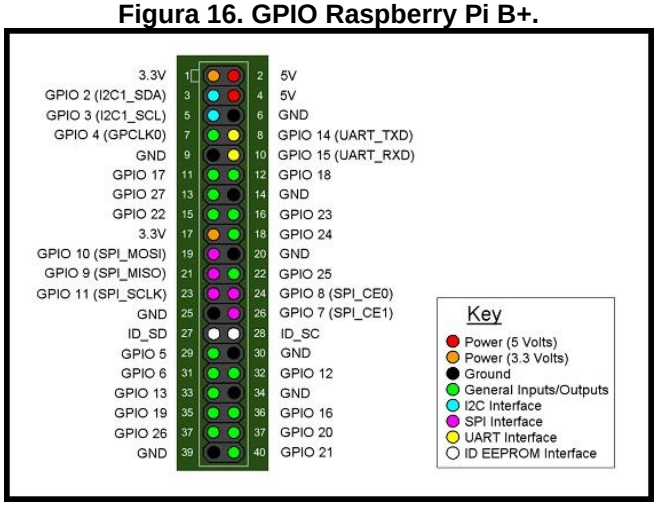
Se investigaron los métodos de petición del protocolo HTTP; sin embargo, el que se utilizó en este proyecto es el POST, que ayuda a mejorar la interacción y el intercambio de datos entre el dispositivo, la web y servidor remoto.

Para el almacenamiento de las señales y consulta de ellas en la base de datos se seleccionó el lenguaje de consulta estructurado (SQL), mediante este lenguaje se tiene acceso al servidor remoto desde la página web y se realizan consultas con fin de obtener el historial del sensor o dispositivo conectado al Raspberry PI.

Para la interacción del usuario en la página web, se investigó y se aplicó el lenguaje PHP, en el cual se incluyó los métodos de petición del protocolo HTTP, como el mencionado anteriormente, el POST.

Como cerebro del dispositivo se utilizó un Raspberry Pi, debido a su bajo consumo eléctrico que permite mantenerlo encendido las 24 horas del día, además de su variedad de comunicaciones en los diferentes pines como se ve en la Figura 16. El lenguaje que se utilizó en este dispositivo fue Python, ya que tiene una diversidad de librerías de ayuda para el Raspberry Pi.

Figura 16. GPIO Raspberry Pi B+.



(Copyright por Raspberry Pi fundation.)

La comunicación del Raspberry Pi hacia los sensores o dispositivos que son conectados, se divide en dos tipos de comunicación: señales digitales y señales analógicas.

En las señales digitales, se dividió en 8 entradas digitales y 8 salidas digitales. Para cada una de las salidas digitales se utilizó un puerto del Raspberry Pi, con la ayuda de la librería RPI-Gpio se configuraron cada una como salida, se colocó un circuito de acondicionamiento y aislamiento para que cada salida conduzca 24 volts o 0 volts, dependiendo de lo que el usuario requiera utilizar; el circuito de acondicionamiento fue diseñado para la protección para Raspberry Pi como se muestra en la Figura 18, esta protección se realizó con optoacoplador en cada una de las salidas, para que cada salida digital tenga su protección. Después se colocó un transistor mosfet para que sea capaz de proporcionar la corriente deseada en cada salida digital.

En las entradas digitales, se realizó un circuito de acondicionamiento para que el Raspberry Pi pueda tener lectura de los valores de los sensores o dispositivos digitales de 24 volts y permita aislamiento entre el Raspberry Pi y los dispositivos que se conectan como se ve en la Figura14. Para cada una de las entradas se utilizó un puerto del Raspberry Pi, establecido como entrada con la ayuda de la misma librería Rpi-Gpio. Se colocó un diodo TVS en cada una de las entradas para la protección contra picos mayores a 24 volts. Así como también se colocó una luz led en cada entrada para facilitar ópticamente el estado de cada entrada. Por último, se colocó un optoacoplador que permite la aislación del Raspberry Pi con el circuito de acondicionamiento.

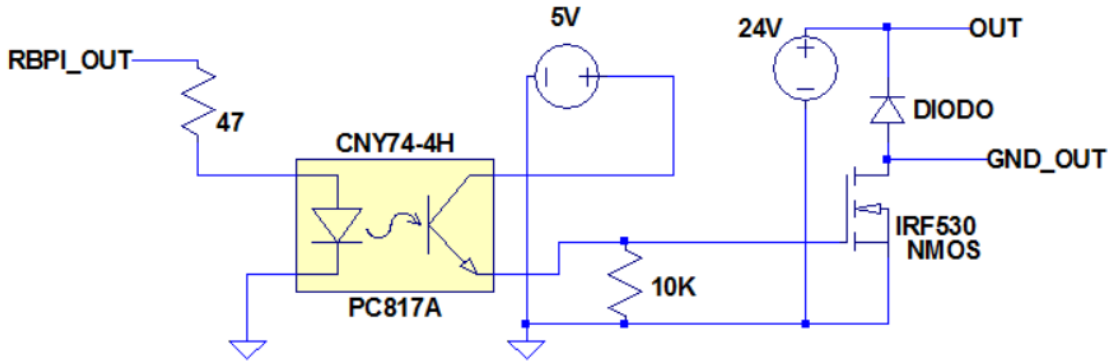
Figura 17. Diagrama de circuito de acondicionamiento para salidas digitales.

(“Elaboración propia”.)

Figura 18. Diagrama de circuito de acondicionamiento para entradas digitales.

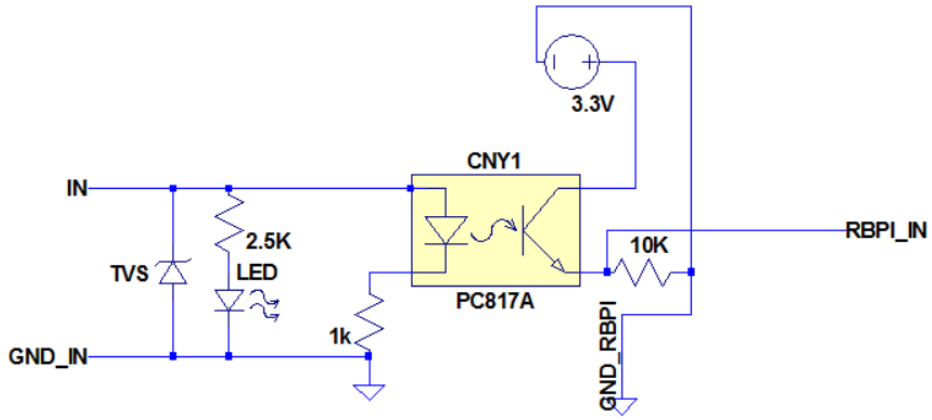
(“Elaboración propia”.)

Figura 19. Circuito de acondicionamiento y aislamiento de salidas digitales.



(“Elaboración propia”.)

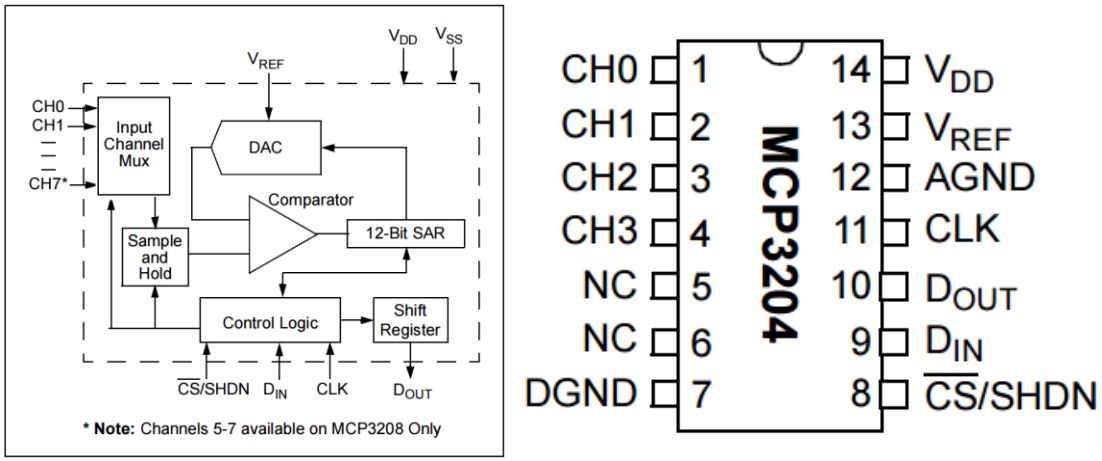
Figura 20. Circuito de acondicionamiento y aislamiento de entradas digitales.



(“Elaboración propia”.)

En las señales analógicas, se dividió en 4 entradas y 4 salidas analógicas. Para las entradas analógicas se utilizó un conversor analógico-digital de 4 canales, el cual es un integrado MCP3204 como se muestra en la Figura 21, conectado y manipulado mediante comunicación SPI con el Raspberry Pi. Se realizó un sistema de acondicionamiento para tener lectura en el Raspberry Pi, valores entre 0 a 10 volts de cada dispositivo conectado. Esta configuración se realizó con dos amplificadores operacionales. Para las salidas analógicas se utilizaron 4 conversores digital-analógico de un canal de voltaje de salida cada componente, el integrado que se utilizó es MCP4725, conectado mediante comunicación I2C con el Raspberry Pi, para cada una de las salidas se realizó un circuito de acondicionamiento que permite generar valores de 0 a 10 volts, el voltaje es seleccionado por el usuario. Para cada una de las señales analógicas se realizó un circuito de aislamiento del Raspberry Pi hacia los dispositivos analógicos a conectar, en la comunicación SPI se utilizó el aislador Adum1411, que es utilizado especialmente para esta comunicación, debido a la velocidad de interrupción. Para la comunicación I2C se utilizó el aislador iso1541, debido a que permite una comunicación bidireccional.

Figura 21. Diagrama de bloques del ADC (MCP3204).

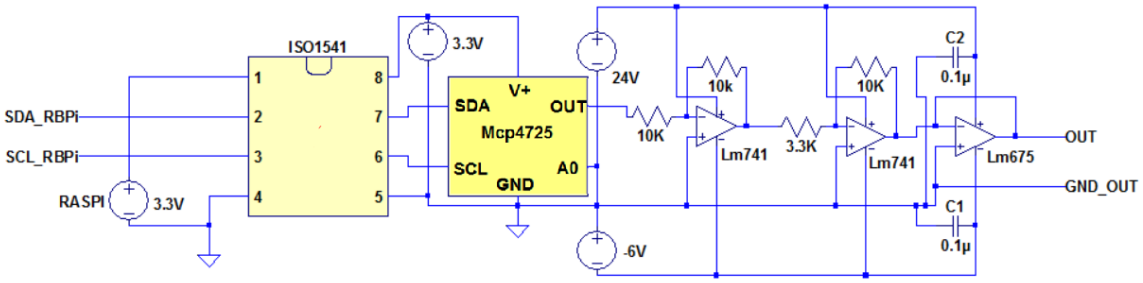


(Copyright por Texas Instrument.)

Figura 22. Diagrama de circuito de acondicionamiento para salidas analógicas.

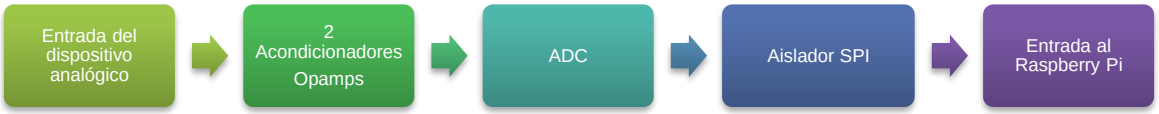
("Elaboración propia".)

Figura 23. Circuito de acondicionamiento para las salidas analógicas de 0 a 10 volts.



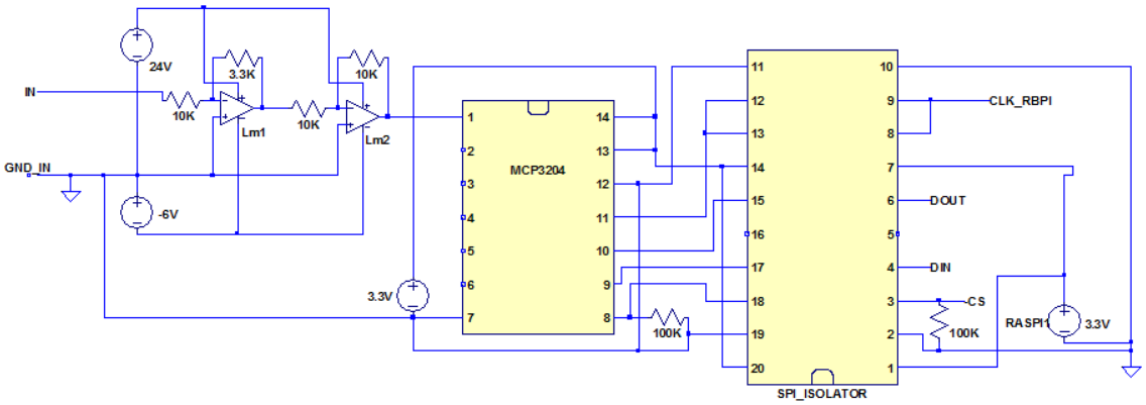
("Elaboración propia".)

Figura 24. Diagrama de circuito de acondicionamiento para entradas analógicas.



("Elaboración propia".)

Figura 25. Circuito de acondicionamiento para las entradas analógicas de 0 a 10 volts.

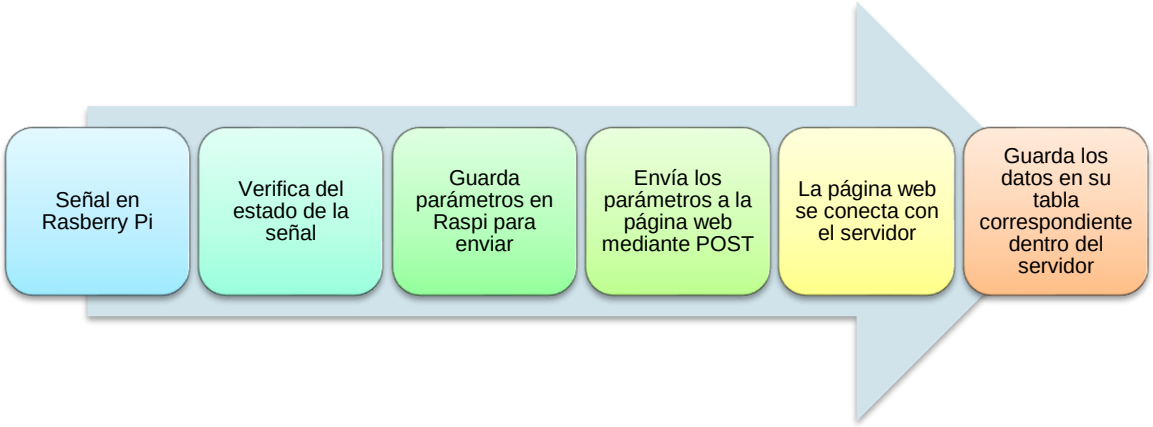


("Elaboración propia".)

Para los diferentes módulos del conversor análogo-digital, se utilizó la ayuda de la librería Webiopi, que es una librería dentro del lenguaje Python es fundamental para la ayuda de las señales analógicas y las señales digitales, así también para ayuda de los drivers para sensores. Esta librería se utilizó especialmente para la comunicación SPI e I2C entre el Raspberry Pi y los circuitos integrados que tenían este tipo de comunicación (I2C para cada DAC y SPI para el ADC).

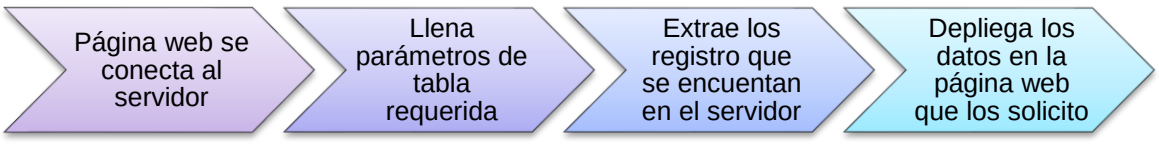
Para el almacenamiento de cada señal que se tiene conectado en el dispositivo, se envía por medio de parámetros que son proporcionados hacia una página web, donde toma estos parámetros y llena un formulario del método POST, este método hace la conexión con la base de datos. Para ello se requiere que el formulario que se tiene en la página web posea el host de la página donde se encuentra la base de datos, la contraseña, el nombre de la base de datos y el nombre de la tabla donde se guardan los parámetros proporcionados como se ve en la Figura 28. Para cada señal se tiene una página web con su formulario, y se tiene una tabla donde se coloca cada registro de cada entrada dentro de la base de datos en el servidor como se ve en la figura 26. Para el despliegue de los datos, la página se conecta al servidor y le realiza consultas de los datos que se encuentran almacenados en la base de datos, las almacena en una tabla y las despliega en la pantalla. Cada entrada/salida digital y analógica en la web tiene la opción de poder borrar el historial, observar gráficamente el comportamiento de la señal a través del tiempo, así como también poder descargar un archivo que contiene en tablas, la información de la señal que se desea.

Figura 26. Diagrama de proceso para almacenar los registros en el servidor.



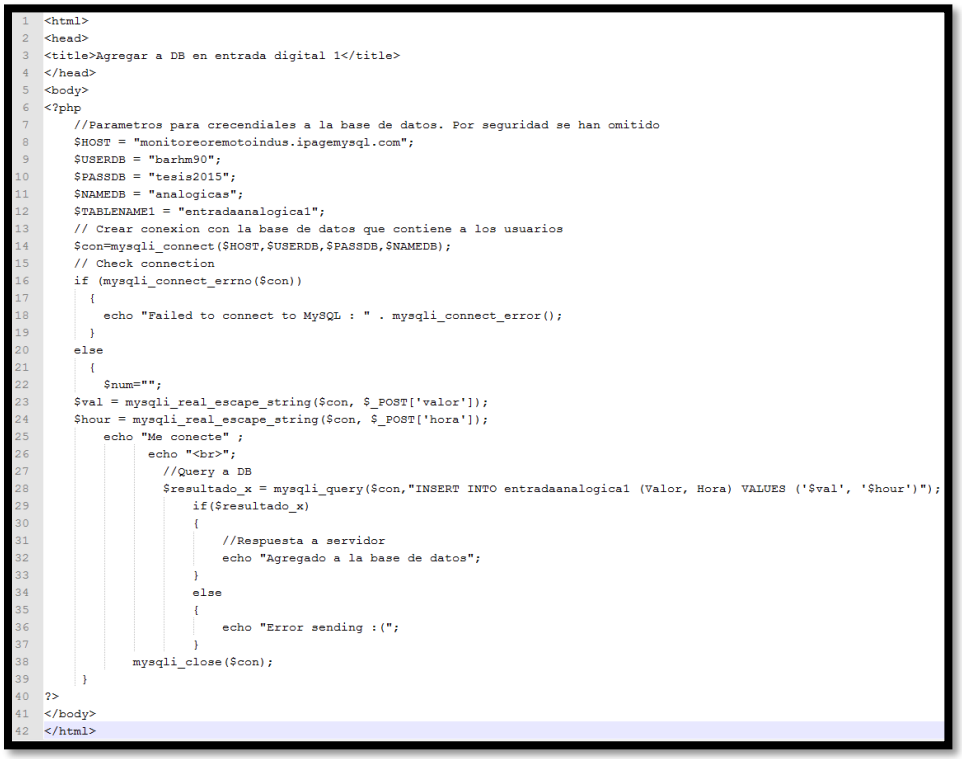
(“Elaboración propia”.)

Figura 27. Diagrama de proceso para desplegar los registros en la página web.



(“Elaboración propia”.)

Figura 28. Código que crea conexión con la base de datos, llena los formularios y almacena en la tabla de cada señal.



```
<html>
<head>
<title>Agregar a DB en entrada digital 1</title>
</head>
<body>
<?php
    //Parametros para crecendiales a la base de datos. Por seguridad se han omitido
    $HOST = "monitoreoremotoindus.ipagemysql.com";
    $USERDB = "barhm90";
    $PASSDB = "tesis2015";
    $NAMEDB = "analogicas";
    $TABLENAME1 = "entradaanalogica1";
    // Crear conexion con la base de datos que contiene a los usuarios
    $con=mysqli_connect($HOST,$USERDB,$PASSDB,$NAMEDB);
    // Check connection
    if (mysqli_connect_errno($con))
      {
        echo "Failed to connect to MySQL : " . mysqli_connect_error();
      }
    else
      {
        $num="";
    $val = mysqli_real_escape_string($con, $_POST['valor']);
    $hour = mysqli_real_escape_string($con, $_POST['hora']);
        echo "Me conecte" ;
             echo "<br>";
               //Query a DB
               $resultado_x = mysqli_query($con,"INSERT INTO entradaanalogica1 (Valor, Hora) VALUES ('$val', '$hour')");
                   if($resultado_x)
                   {
                       //Respuesta a servidor
                       echo "Agregado a la base de datos";
                   }
                   else
                   {
                       echo "Error sending :(";
                   }
           mysqli_close($con);
    }
?>
</body>
</html>
```

(“Elaboración propia”.)

Para este proyecto se rentó el dominio www.monitoreoremotoindustrial.com por medio de un “web hosting” llamado iPage, en el cual incluía una base de datos en conjunto a la renta del dominio. El tipo de base de datos que incluía es SQL, con el cual se realizaron las consultas a través del método POST del protocolo HTML. La página está diseñada para poder ver la URL ya sea desde un computador o desde un teléfono móvil.

El Raspberry Pi se conecta a la red celular a través de un sim de cualquier compañía te teléfono. Este debe ser colocado antes encender el Raspberry Pi, debido que en la iniciación del sistema, reconoce este sim e identifica la dirección IP con la cual se tendrá conexión al servidor. Para su correcta funcionalidad, este sim debe tener cobertura de la red, sino no se podrá tener conexión al servidor. Este reconocimiento de sim y de dirección IP se realizó con la ayuda de la librería usb-modeswitch, que detecta mediante el puerto USB, la conexión de la tarjeta de red celular. Como se ve en la siguiente figura.

Figura 29. Detección del modem 3G en el Raspberry Pi.

```
wwan0     Link encap:Ethernet  HWaddr 0c:5b:8f:27:9a:64
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5237 (5.1 KiB)  TX bytes:0 (0.0 B)
```

Y agregando las siguientes dos líneas en el archivo `/etc/network/interfaces.` Para así obtener la dirección IP de la conexión, en cada inicialización del sistema.

Figura 30. Código que activa el modem detectado para poder tomar el IP.

```
allow-hotplug wwan0
iface wwan0 inet dhcp
```

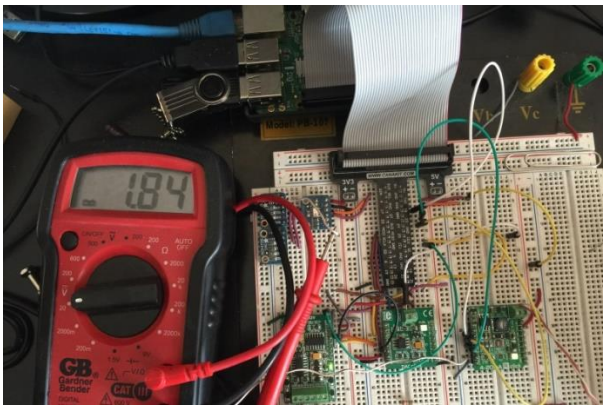
Al reiniciar el sistema, después del cambio realizado en el archivo especificado anteriormente. Se obtiene lo siguiente.

Figura 31. IP tomada del modem 3G.

```
wwan0     Link encap:Ethernet  HWaddr 0c:5b:8f:27:9a:64
          inet addr:192.168.1.101  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5237 (5.1 KiB)  TX bytes:0 (0.0 B)
```

Se realizaron pruebas en cada una de las diferentes señales, para estas pruebas se colocaron los diferentes componentes que se utilizaron en un protoboard como se ve en la Figura 32. Después haber realizado las pruebas, se realizó el diseño de la tarjeta de circuito impreso para mayor facilidad de la utilización y adaptación al usuario.

Figura 32. Pruebas en Protoboard.



(“Elaboración propia”.)

Para finalizar, se integró una pantalla táctil de 800x480 pixeles de resolución como se ve en la Figura 33, esta pantalla está conectada por vía HDMI hacia el Raspberry Pi, también está conectada por USB para obtener la corriente que necesita para su funcionamiento. Esta pantalla se adjuntó al circuito impreso del Proyecto de Graduación.

Figura 33. Pantalla táctil.

(“Elaboración propia”)

VI. RESULTADOS Y ANÁLISIS DE RESULTADOS

La pantalla principal en el dispositivo es manejada de manera táctil, además posee el estado de la conexión del dispositivo hacia el servidor, así también la temperatura que el dispositivo está siendo utilizado como se ve en la siguiente figura. El programa principal fue estandarizado para tener en todas las ventanas del programa, 800x460 pixeles, debido a que la pantalla táctil tiene esa resolución de imagen.

Figura 34. Pantalla principal del dispositivo con conexión a la red y temperatura trabaja.

("Elaboración propia")

Figura 35. Pantalla principal del dispositivo sin conexión a la red.

("Elaboración propia")

A. Entradas y salidas digitales

En las salidas digitales, como se utilizó un pin del Raspberry Pi para cada una, esto facilito el manejo en el encendido y apagado de cada salida. El Raspberry Pi proporciona 0 volts o 3.3 volts en cada salida. Debido a que en la industria se manejan voltajes de 24 volts se realizó el circuito de acondicionamiento mencionado anteriormente, para poder tener 0 volts o 24 volts en la salida. Los mosfet ayudaron para proporcionar hasta 1 amperio de corriente en cada salida, esto para que en la demanda de corriente que se requiera en la salida, no se tenga ningún problema. En la interfaz gráfica del dispositivo remoto, se indica una etiqueta en “ON” cuando proporciona un “1” lógico a la salida y un “OFF” cuando proporciona un “0” lógico como se ve en las figuras 36 y 37.

Figura 36. Interfaz gráfica de salidas digitales, proporcionando 0 lógico.

(“Elaboración propia”)

Figura 37. Interfaz gráfica de salidas digitales, proporcionando 1 lógico.

(“Elaboración propia”)

Para las entradas digitales con la utilización de un puerto del Raspberry Pi para cada entrada, esto simplificó las lecturas de cada entrada digital, ya que el sistema solo se mantiene leyendo el estado de cada entrada y las desplegaba en la pantalla. En el circuito lógico, solo se colocó una resistencia de 1kOhm entre la entrada del optoacoplador y el dispositivo a conectar, para que cada proporcione la corriente necesaria que el optoacoplador lea un uno lógico y lo envía al Raspberry Pi. En la interfaz gráfica del dispositivo remoto, indica ON cuando recibe un 1 lógico en la entrada e indica OFF cuando recibe un 0 lógico como se observa en las figuras 38 y 39.

Figura 38. Interfaz gráfica en las entradas digitales, recibiendo 1 lógico.

(“Elaboración propia”)

Figura 39. Interfaz gráfica en la entrada digital 1, recibiendo 0 lógico.

(“Elaboración propia”)

B. Entradas y salidas analógicas

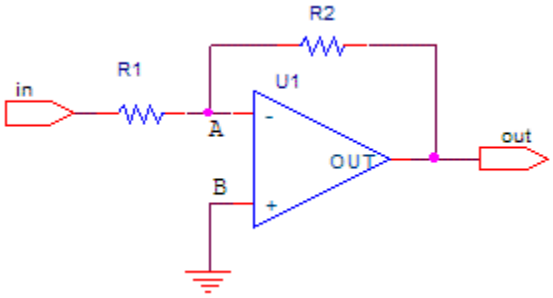
1. Salidas analógicas.

En las salidas analógicas, al utilizar un DAC (convertidor digital-análogo) por cada salida analógica, este convierte la señal de digital a señal analógica. Y estas pueden ser manipuladas desde el Raspberry Pi por medio de la comunicación I2C, variando el voltaje de salida en un rango desde 0 volts a 3 volts de salida en el Raspberry Pi.

Para cada salida analógica se utilizó un DAC y un circuito de acondicionamiento que permite la amplificación de 3 volts de la salida del Raspberry Pi a 10 volts de salida que se desea, se realizó con dos amplificadores operacionales. Para el primero se utilizó la configuración de amplificador inversor con ganancia uno, como se ve en la figura 40. Adicionalmente se colocó otro con configuración aplicador inversor con ganancia de 3.33*Vin para obtener la señal en la amplitud y el rango que se deseaba proporcionar en la salida del dispositivo.

Para las dos configuraciones se realizaron los siguientes cálculos:

Figura 40. Configuración utilizada para realizar amplificador inversor con Op-amp.



(“Elaboración propia”)

Ecuación de la configuración amplificador.

$$\frac{Vi * R2}{R1} = -Vo1$$

Partiendo de esta ecuación, se tiene como Vi igual 3 volts y se tiene que Vo1 es igual a -3 volts. A R1 se le colocó una resistencia de 10kOhm. Se realizó el inversor tomando como referencia la tierra general. Quedando como se ve en la siguiente ecuación.

Ecuación de la configuración de salida de -3 a 0 volts con los valores seleccionados.

$$\frac{3 * R2}{10kOhm} = -3$$

Al resolver la ecuación se obtiene la siguiente respuesta para R2:

$$R2 = 10kOhms$$

Se utilizó la misma configuración segundo amplificador, modificando la ganancia para obtener 10 volts en la salida del dispositivo. Se estableció R12 con una resistencia de 10kOhms y tomando como voltaje de entrada, el voltaje de salida del anterior amplificador operacional, este amplificador inversor se realizó con referencia de la tierra general.

Ecuación de la configuración del segundo amplificador.

$$\frac{Vo1 * R2}{R1} = -Vo2$$

Ecuación de la configuración de salida de 0 a 10 volts con los valores seleccionados.

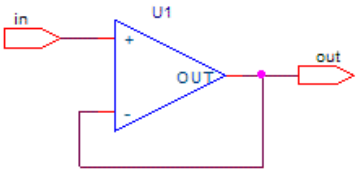
$$\frac{-3 * 10kOhm}{R1} = 10$$

Valor de R1 de la ecuación para salida de 10v en el dispositivo.

$$R1 = 3\ kOhms$$

Para la amplificación de corriente se utilizó un amplificador operacional de potencia. Configurado como seguidor o amplificador no inversor con ganancia unitaria. Como se ve en la Figura 41.

Figura 41. Seguidor de voltaje.



("Elaboración propia")

$$Vin = Vout$$

Tiene una impedancia de entrada Zin muy elevada, y una impedancia de salida Zout muy pequeña. Por este motivo se utiliza principalmente para aislar dos circuitos, de manera que el segundo no resulte una carga para el primero, pues la impedancia vista será la altísima Zin del operacional.

Así también la corriente de salida del amplificador depende solamente del él mismo debido a que no tiene ninguna resistencia para limitarlo. Se realizó un diseño para que cada salida analógica tenga una corriente máxima de 500 miliamperios, para ello se tomó un amplificador operacional con una corriente máxima mayor a 500 miliamperios. Se escogió el amplificador operacional lm675 ya que cumple con los requerimientos establecidos.

2. Entradas analógicas.

Para las entradas analógicas se realizó con dos amplificadores inversores, debido a que los sensores o dispositivos están en el rango de 0 volts a 10 volts. Y se tuvo que reducir a un rango de 0 volts a 3 volts que es el rango de lectura del ADC que transmite los datos al Raspberry Pi. Para cada entrada se realizaron las mismas configuraciones de los amplificadores operacionales, en el primero con un amplificador inversor, para convertir de 10 volts a -3 volts, haciendo los cálculos correspondientes para que el amplificador tenga una ganancia menor a 1. Y en el segundo se realizó la misma configuración de amplificador inversor, pero este con ganancia 1. Para cada uno de las configuraciones de entrada se realizaron los siguientes cálculos.

Para la primera configuración de ganancia menor a 1:

Ecuación de la configuración amplificador.

$$\frac{Vi * R2}{R1} = -Vo$$

Partiendo de esta ecuación, se tiene como V1 igual 10 volts, debido a que se hace el inversor con referencia a tierra general, solo se hace una reducción de voltaje de tres decimos de la entrada. A R1 se le colocó una resistencia de 10kOhms. Dando resultado como se ve en la siguiente ecuación.

Ecuación de la configuración de entrada de 0-10v con los valores seleccionados.

$$\frac{10 * R2}{10kOhm} = -3$$

Al resolver la ecuación se obtiene la siguiente respuesta para R2

$$R2 = 3KOhms$$

Para la segunda configuración de la entrada analógica, se utilizó la misma configuración que el anterior, solo que estableciendo la ganancia en 1 y haciendo reflejo en tierra para tener los valores en el rango de 0 a 3 volts. Para este se realizaron los siguientes cálculos:

Ecuación de la segunda configuración amplificador inversor con ganancia 1.

$$\frac{Vi * R2}{R1} = -Vo$$

Partiendo de esta ecuación, se tiene como Vi igual -3 volts y se tiene que Vo es igual a 3 volts. Se igualo R1 se colocó una resistencia de 10kOhms. Quedando como se ve en la siguiente ecuación.

Sustituyendo la ecuación de la segunda configuración con los valores seleccionados.

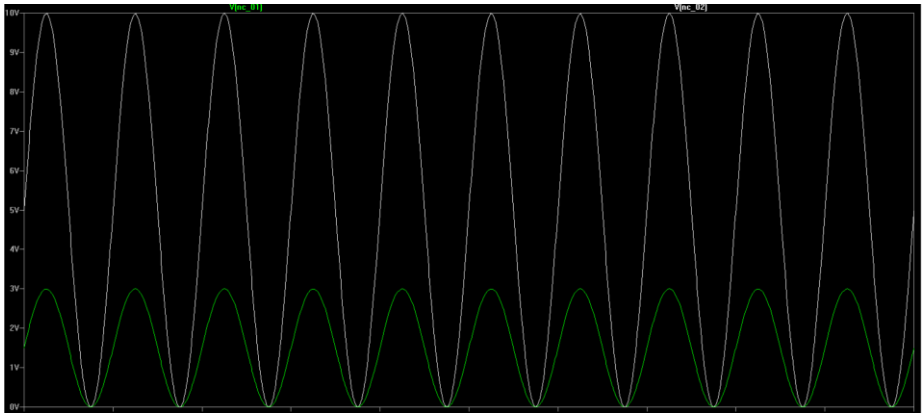
$$\frac{-3 * R2}{10kOhm} = -3$$

Al resolver la ecuación se obtiene la siguiente respuesta para R2:

$$R2 = 10kOhms$$

Después de tener los cálculos de cada uno de las diferentes configuraciones, se colocaron los dispositivos electrónicos correctos dentro de los circuitos de acondicionamiento detallados anteriormente. Correspondientes a las entradas y a las salidas analógicas de las dos configuraciones. Para cada una se obtuvieron las siguientes gráficas en cada la salida analógica como en cada entrada analógica.

Figura 42. Comparación del voltaje de salida del Raspberry Pi con el voltaje de salida para el sensor en configuración de 0v a 10v.



("Elaboración propia")

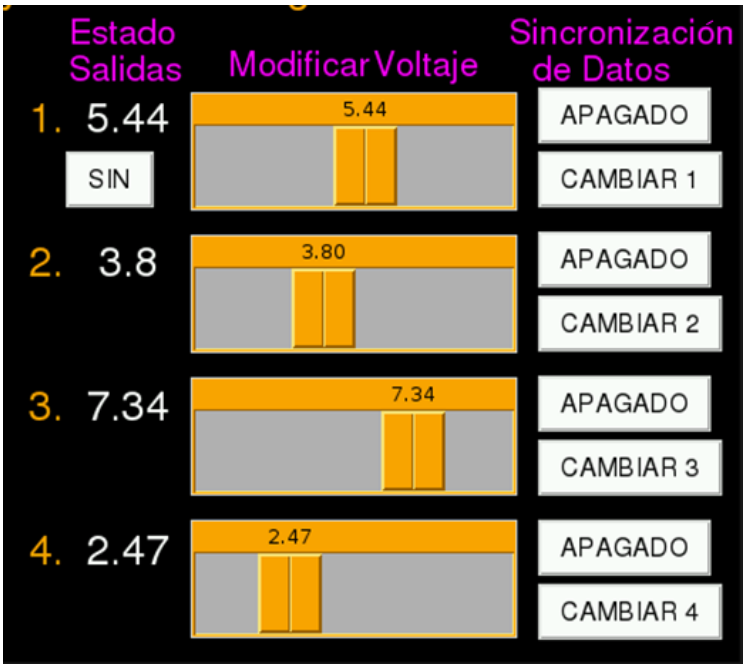
Cada una de estas gráficas, es el voltaje de la salida que proporciona el Raspberry Pi en el pin de salida, el otro voltaje es a la salida del sensor después del circuito de acondicionamiento proporcionado por los amplificadores operacionales. Se utiliza el mismo circuito en las diferentes configuraciones. Solo cambiando el valor de una resistencia para que el usuario pueda seleccionar que tipo de configuración se quiere, ya sea en las salidas analógicas como en las entradas analógicas.

En la interfaz gráfica del dispositivo remoto, para las entradas analógicas, en la ventana del dispositivo indica en volts del rango del Raspberry Pi cuando recibe el valor en cada una de las entradas del ADC, en un rango de 0 a 3 volts, y lo despliega en el dispositivo en el rango de 0 a 10 volts como se observa en la figura 43. Así como también para las salidas analógicas, indica el voltaje que proporciona a cada salida del DAC, en el mismo rango como se observa en la Figura 44.

Figura 43. Interfaz en el dispositivo que indica el voltaje en las entradas analógicas.

("Elaboración propia")

Figura 44. Interfaz en el dispositivo que indica el voltaje en las salidas analógicas.



("Elaboración propia")

C. Plataforma Web

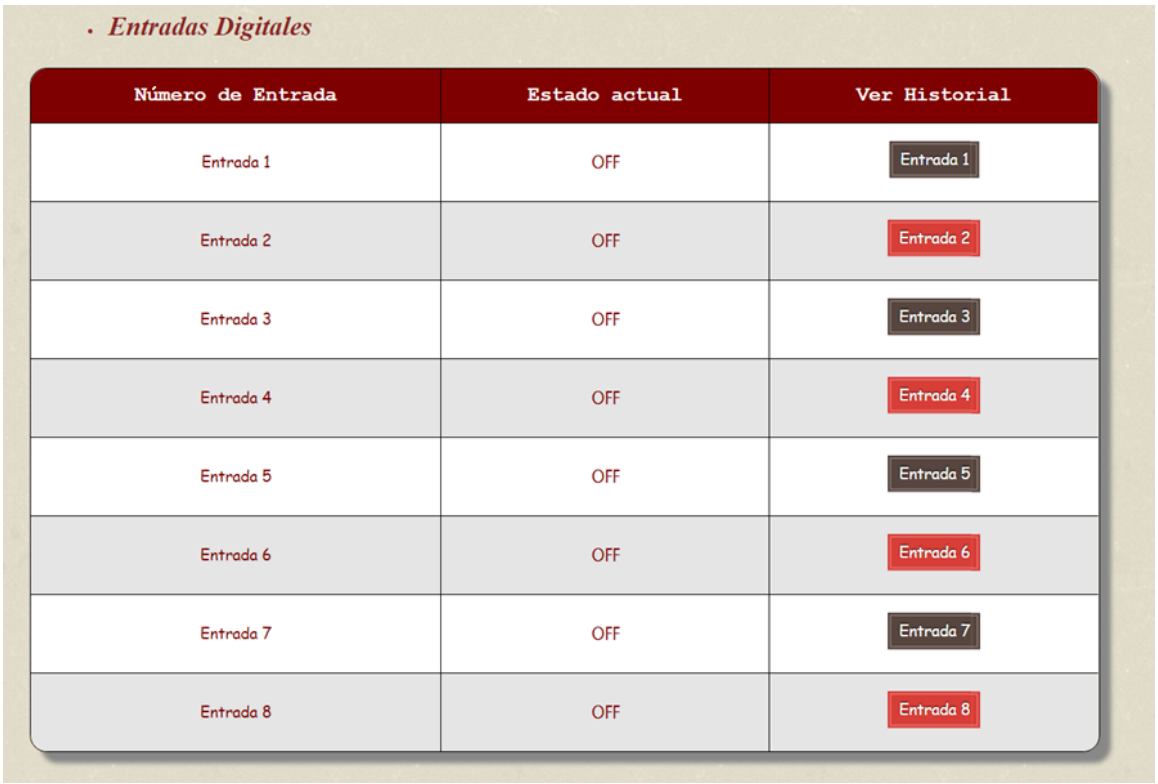
Para la plataforma Web se utilizó el lenguaje de PHP, debido a que en este tiene la ventaja de realizar consultas y conexiones hacia las bases de datos. Que era lo fundamental en este trabajo de graduación, este lenguaje facilitó el envío de los datos al servidor, así como también la facilidad al realizar las consultas a la base de datos para observar el historial de cada una de las señales.

Se diseñó la plataforma web, de manera sencilla para que el usuario le sea de fácil uso y así que sea funcional también. En la página web principal hay pestañas donde el usuario puede seleccionar el tipo de señal que desea ver su record histórico como se ve en la Figura 45. Cada una de estas pestañas se tiene las entradas y salidas correspondiendo al tipo de señal que seleccionó. Dentro de cada tipo de señal seleccionada, se tiene en tablas, los estados actuales de las diferentes señales. En una tabla se encuentran todas las entradas y en otra tabla las salidas. Cada entrada y salida posee un botón de historial, que desglosa en una nueva página, una tabla donde contiene una base de datos de todo el historial de la señal seleccionada. En las entradas se mantiene monitoreando el valor actual de dispositivo conectado, mientras que en las salidas, se puede cambiar el valor de desde la página web.

Figura 45. Página web principal.

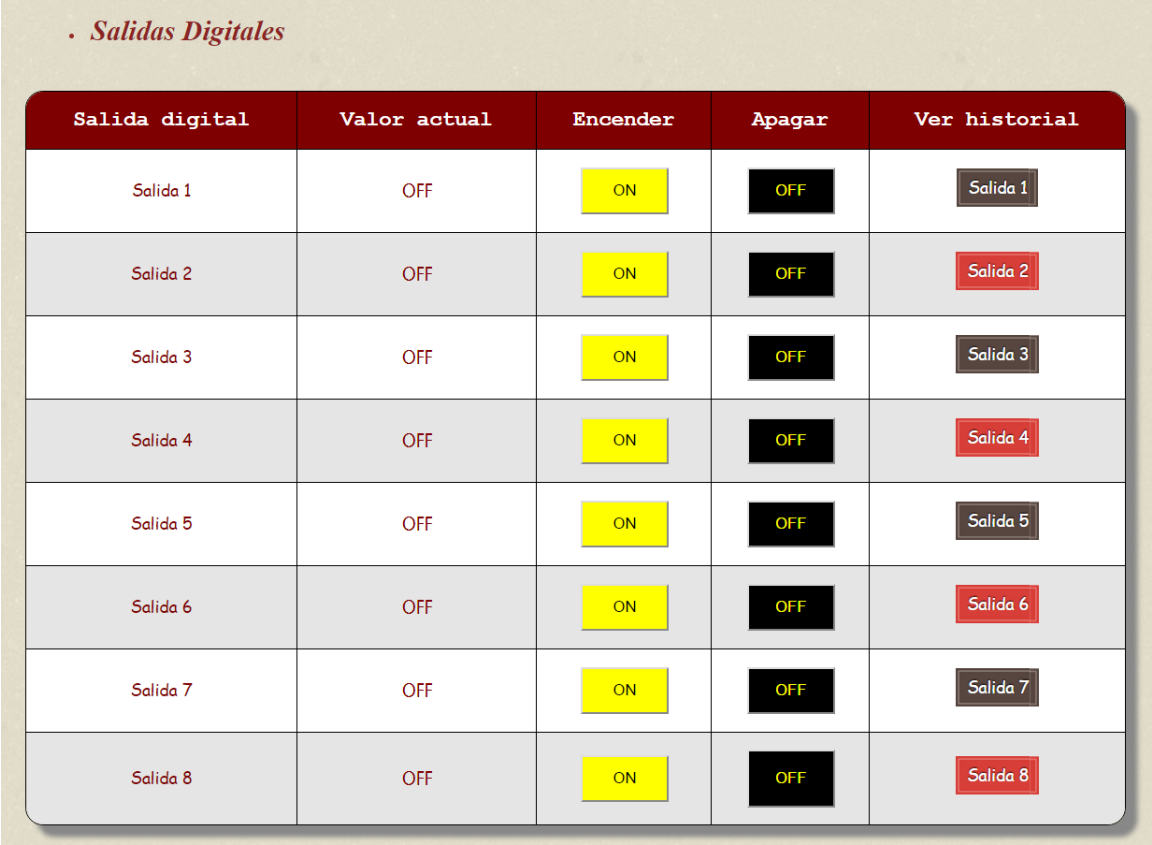
("Elaboración propia")

Figura 46. Tabla de las entradas digitales.



("Elaboración propia")

Figura 47. Tabla de las salidas digitales.

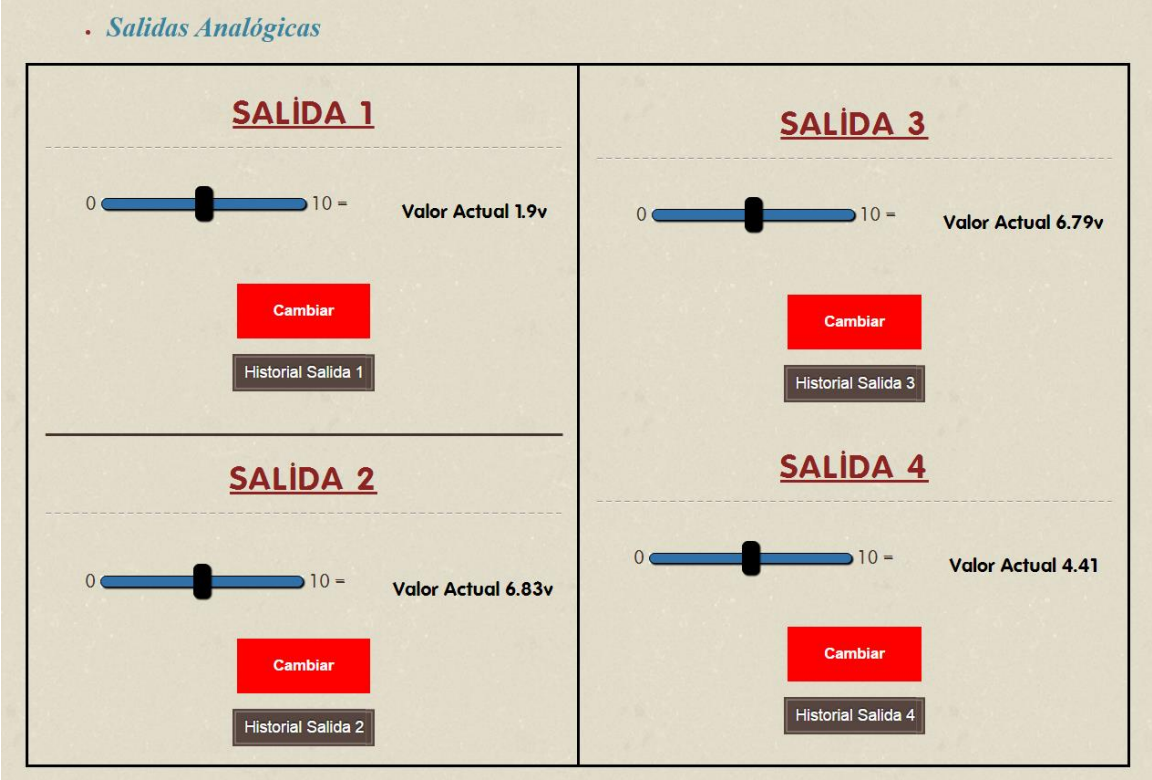


("Elaboración propia")

Figura 48. Tabla de las entradas analógicas.

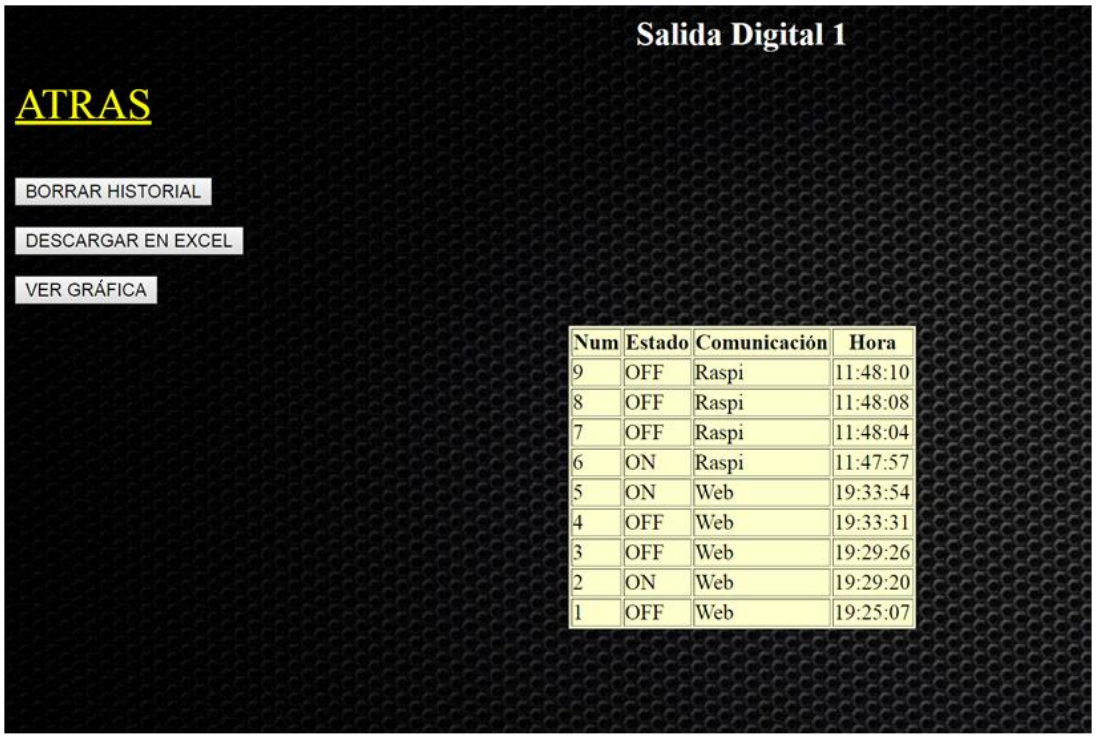
("Elaboración propia")

Figura 49. Tabla de las salidas analógicas.



(“Elaboración propia”)

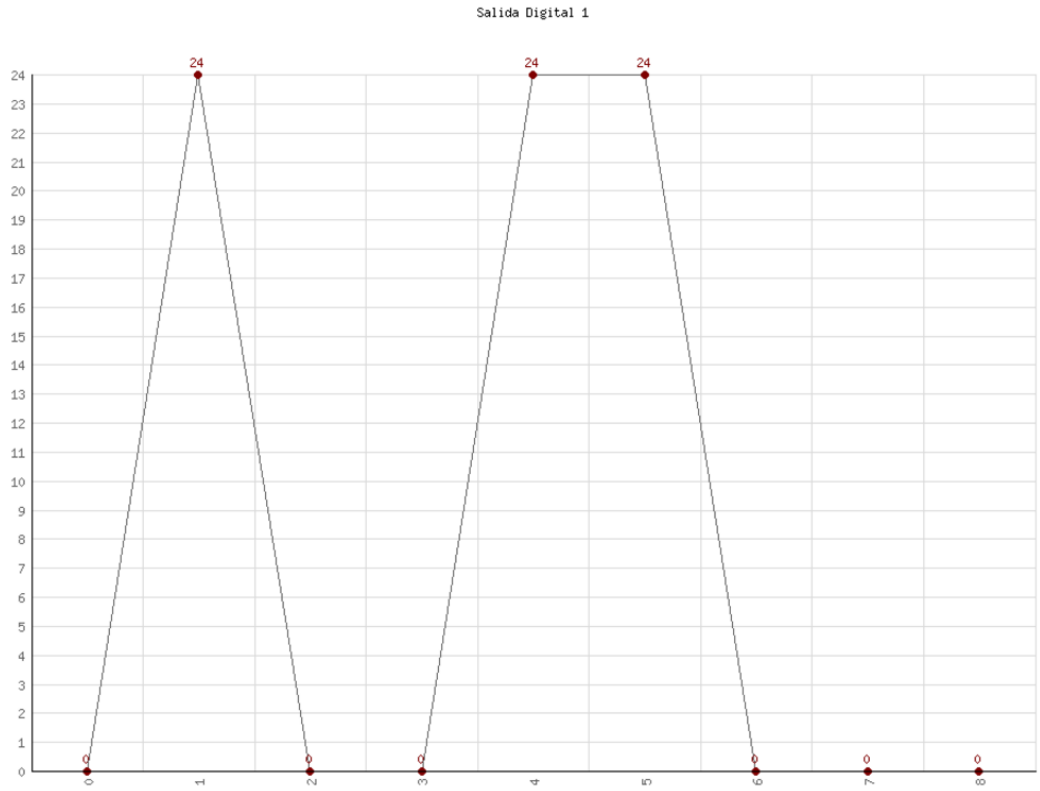
Figura 50. Página de la salida digital 1 que despliega su historial.



Num	Estado	Comunicación	Hora
9	OFF	Raspi	11:48:10
8	OFF	Raspi	11:48:08
7	OFF	Raspi	11:48:04
6	ON	Raspi	11:47:57
5	ON	Web	19:33:54
4	OFF	Web	19:33:31
3	OFF	Web	19:29:26
2	ON	Web	19:29:20
1	OFF	Web	19:25:07

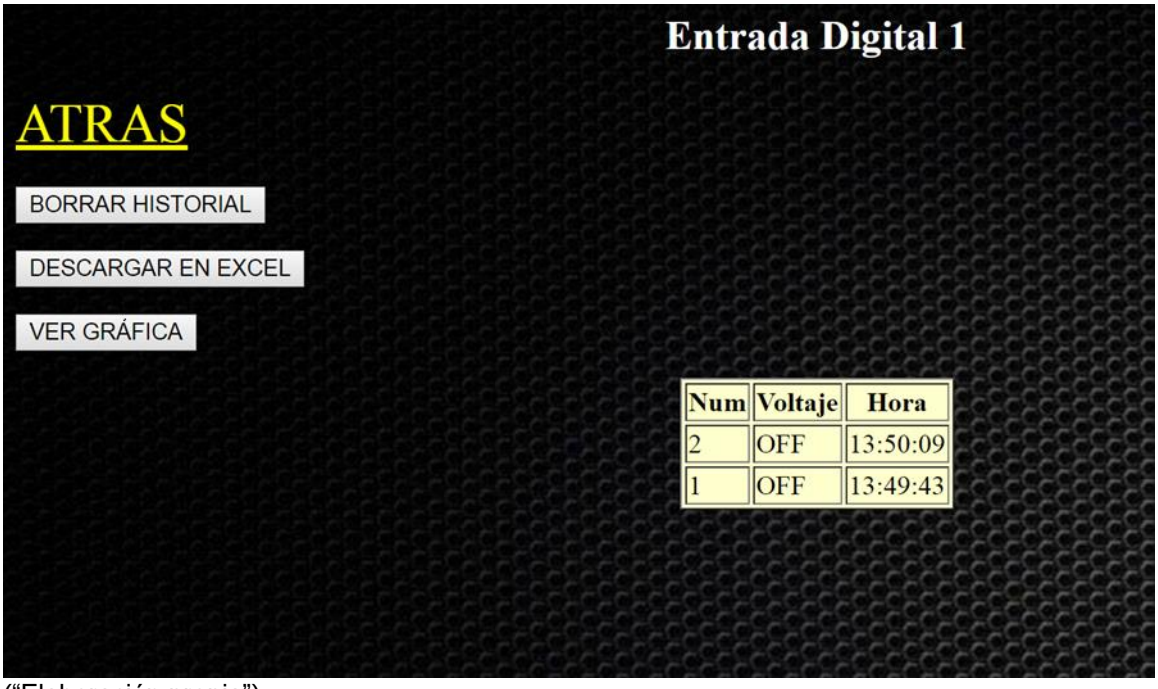
(“Elaboración propia”)

Figura 51. Gráfica de la consulta de la salida digital 1.



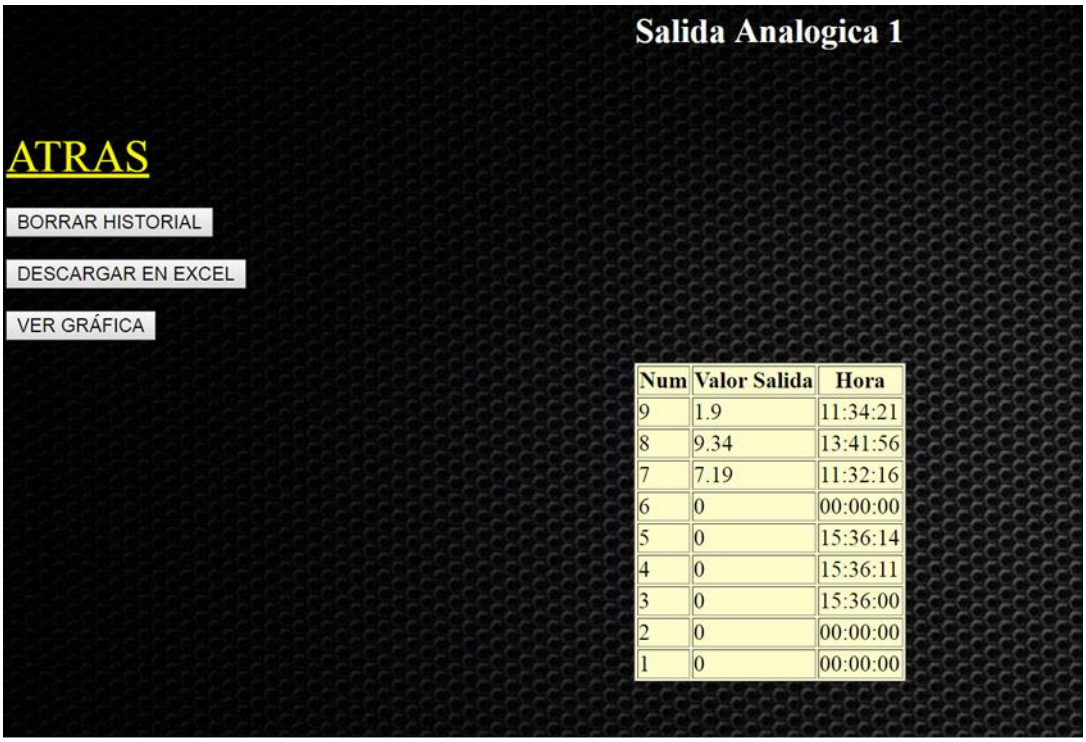
(“Elaboración propia”)

Figura 52. Página web de entrada digital 1 al realizar consulta de historial.



(“Elaboración propia”)

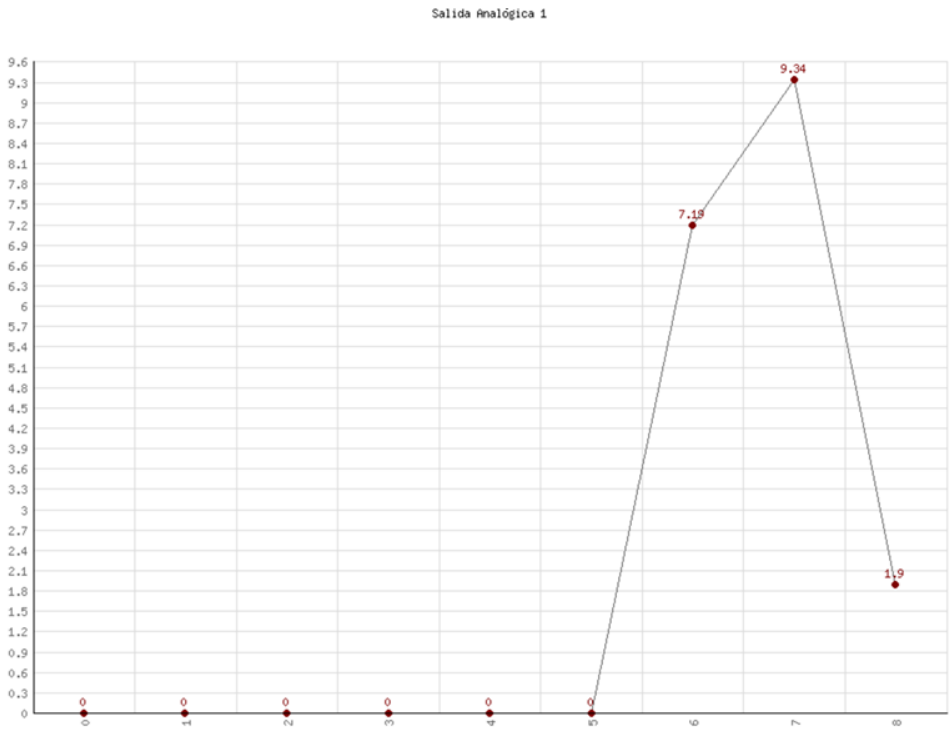
Figura 53. Página web de la salida analógica al realizar consulta de su historial.



Num	Valor Salida	Hora
9	1.9	11:34:21
8	9.34	13:41:56
7	7.19	11:32:16
6	0	00:00:00
5	0	15:36:14
4	0	15:36:11
3	0	15:36:00
2	0	00:00:00
1	0	00:00:00

(“Elaboración propia”)

Figura 54. Gráfica de historial de la salida analógica 1.



(“Elaboración propia”)

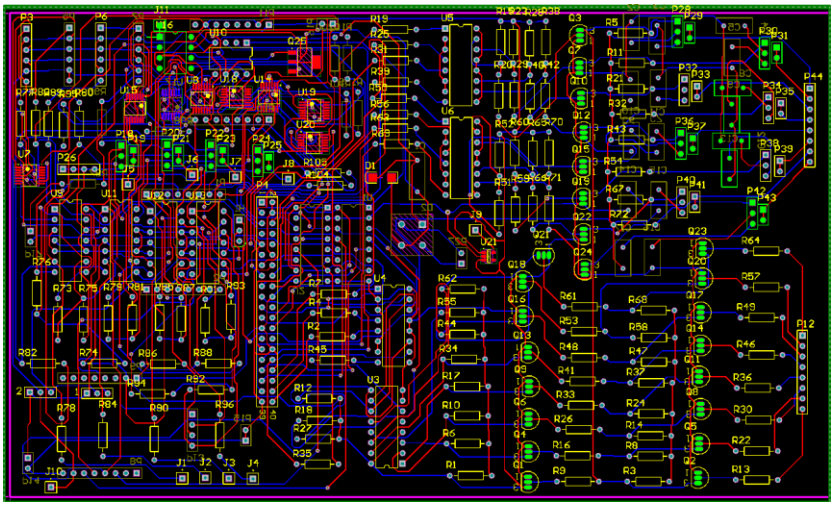
Figura 55. Página web de entrada analógica 1 al realizar consulta en base de datos.

(“Elaboración propia”)

D. PCB

Para el diseño de la PCB, se utilizó Altium Designer, ya que este programa tiene facilidades y librerías de varios componentes, así como también los tamaños correctos de estos. Se había diseñado para que todo estuviera incorporado en una placa como se ve en la Figura 56. Debido a que el sistema es muy complejo para realizarlo en una placa, se dividieron en cuatro módulos.

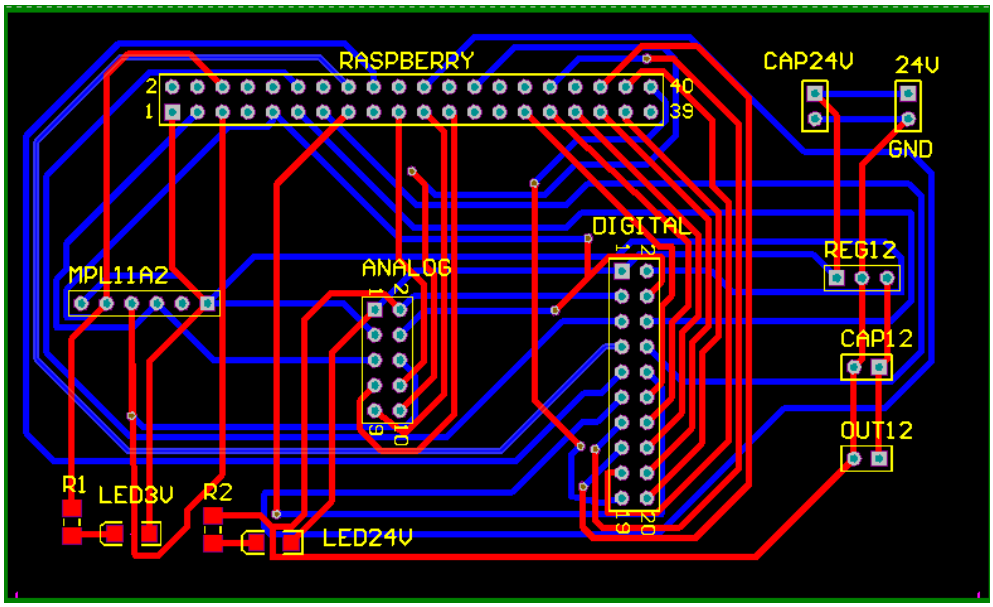
Figura 56. Primer diseño de la placa.



(“Elaboración propia”)

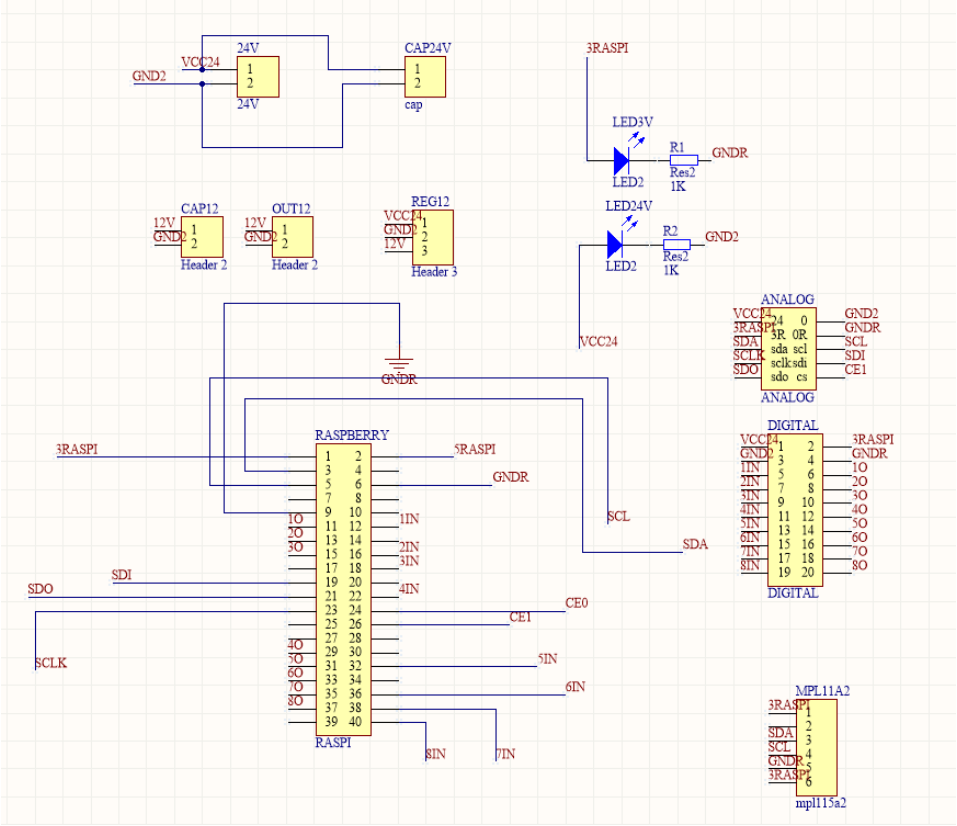
El módulo principal es el que recibe del Raspberry pi y proporciona los datos y voltajes en buses hacia los diferentes módulos. También realiza las regulaciones de los voltajes necesarios.

Figura 57. Diseño de módulo principal en PCB.



("Elaboración propia")

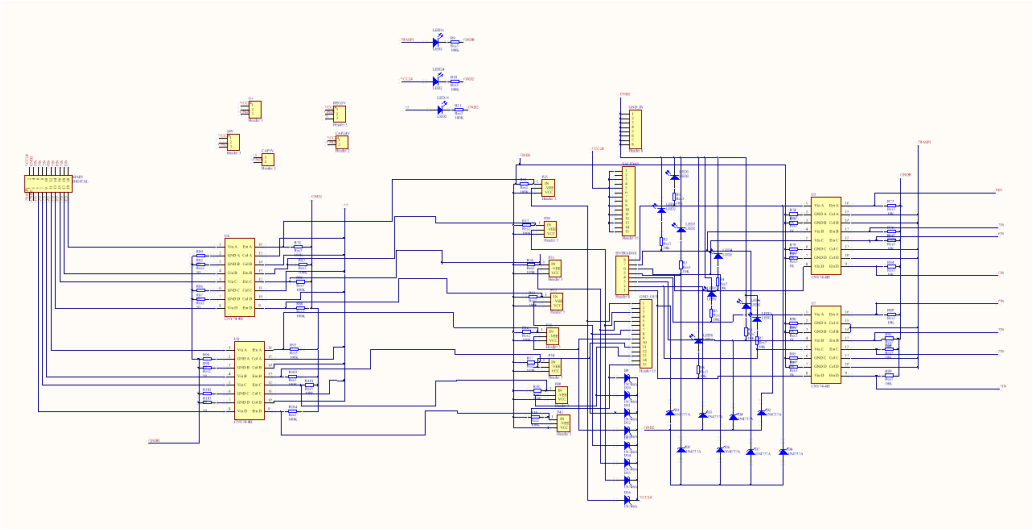
Figura 58. Diseño esquemático de módulo principal en Altium designer.



("Elaboración propia")

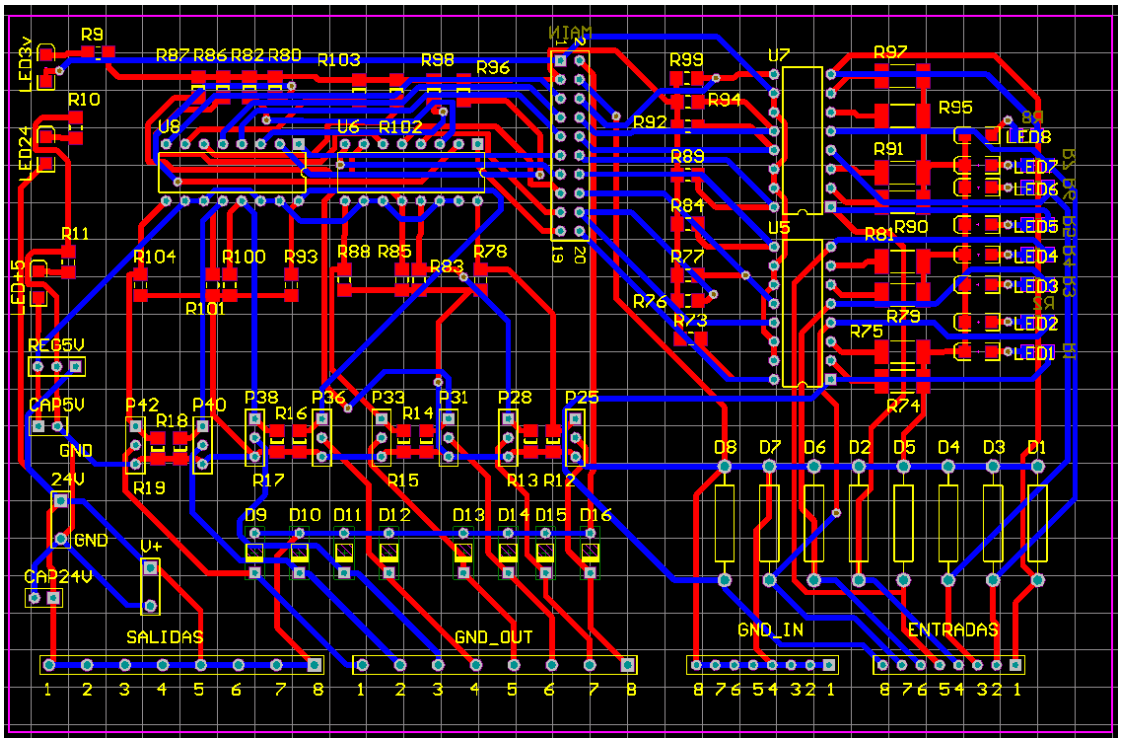
El segundo módulo se diseñó para las entradas y salidas digitales, para optimizar espacio, se utilizaron componentes de superficie, debido al tamaño de estos.

Figura 59. Diseño esquemático de salidas y entradas digitales en Altium designer.



("Elaboración propia")

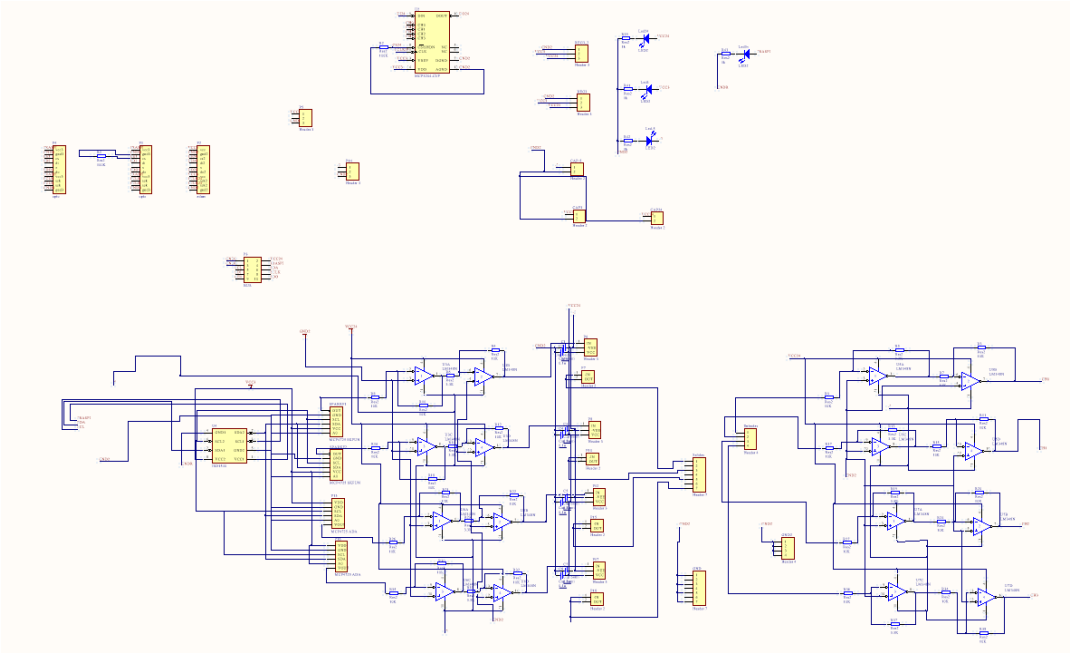
Figura 60. Diseño de módulo de salidas y entradas digitales para PCB.



(“Elaboración propia”)

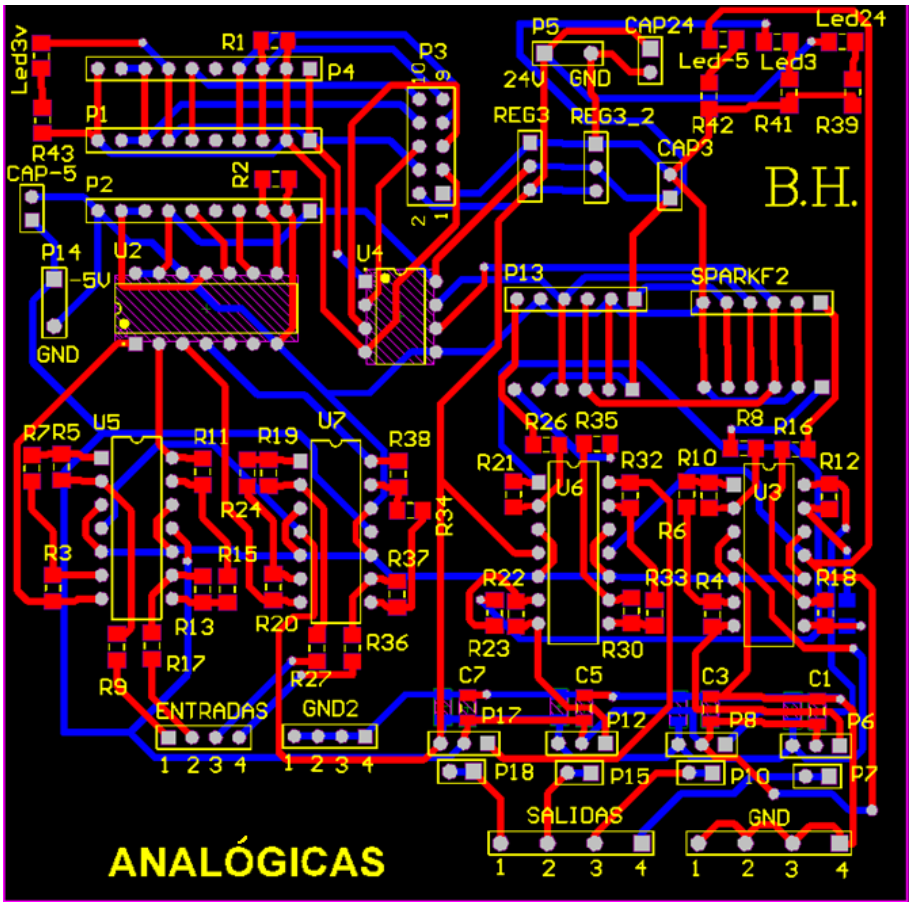
En el tercer módulo se diseñó para las entradas y salidas analógicas. Se tienen 4 entradas analógicas y 4 salidas analógicas.

Figura 61. Diseño esquemático de entradas y salidas analógicas.



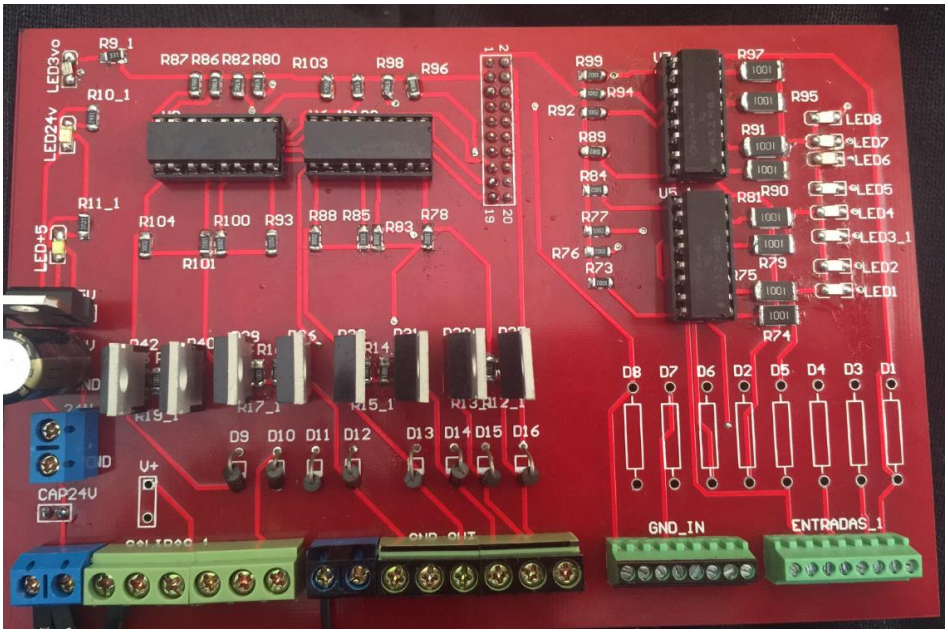
(“Elaboración propia”)

Figura 62. Diseño de módulo de entradas y salidas analógicas en PCB.



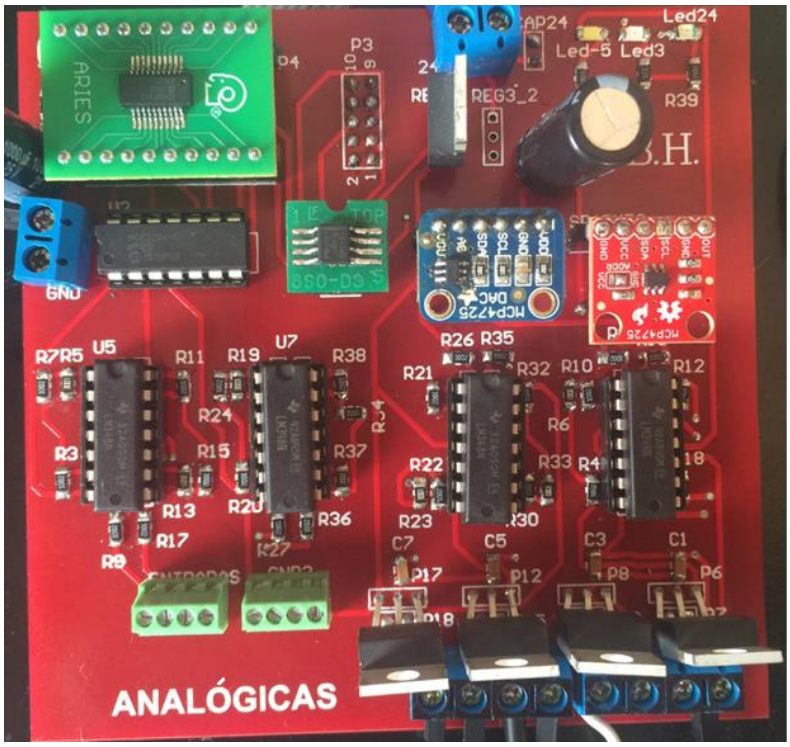
(“Elaboración propia”)

Figura 63. Placa de señales digitales ya elaborada y ensamblada.



(“Elaboración propia”)

Figura 64. Placa de señales analógicas ya ensamblada.

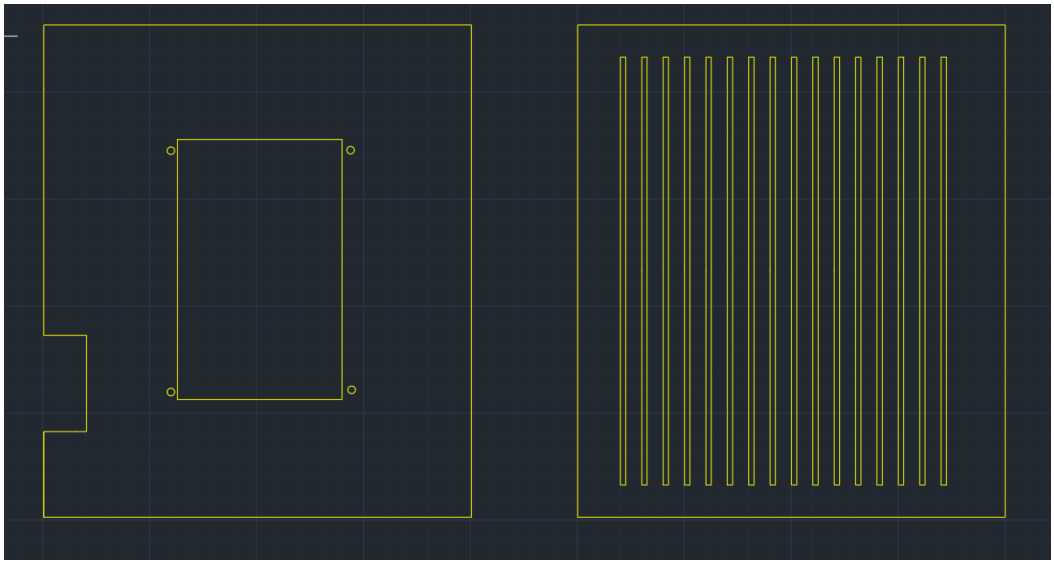


(“Elaboración propia”)

E. Case del dispositivo

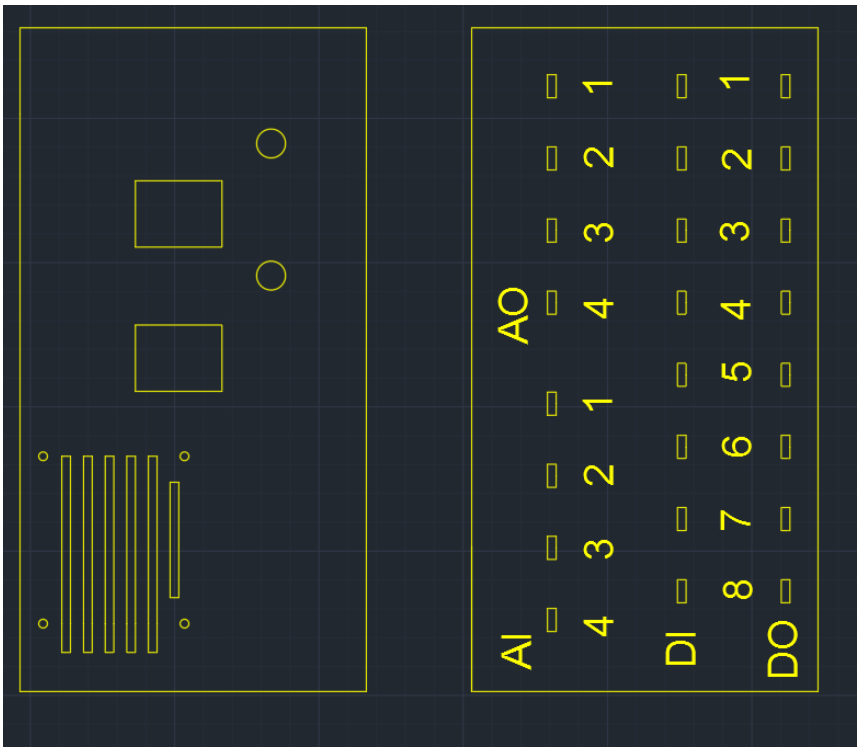
En el diseño del case, se realizó en Autocad con las medidas exactas para este dispositivo. Como se ve en las siguientes figuras.

Figura 65. Diseño de base y tapadera de case.



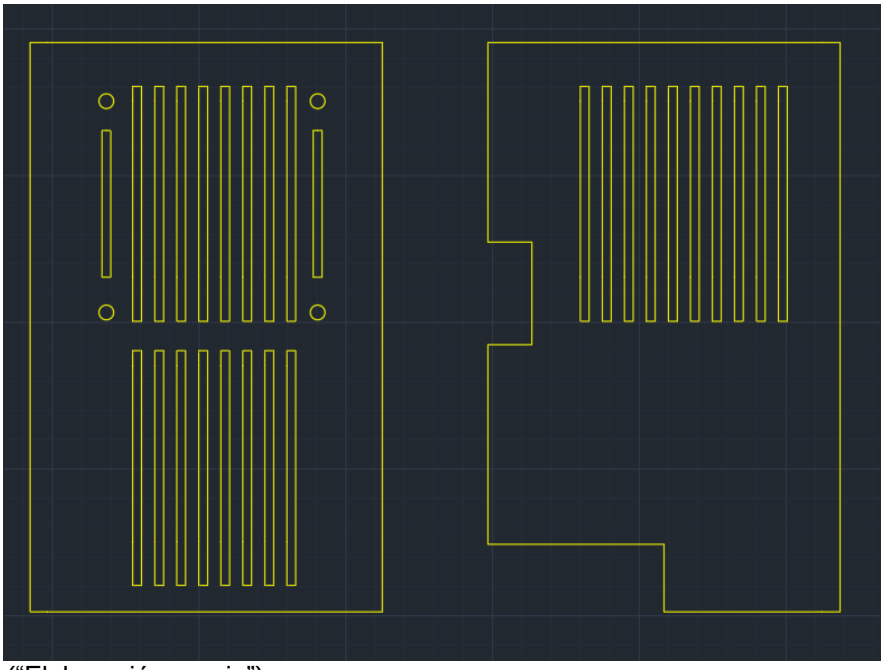
(“Elaboración propia”)

Figura 66. Diseño lateral de las salidas y entradas de los dispositivos a conectar.



("Elaboración propia")

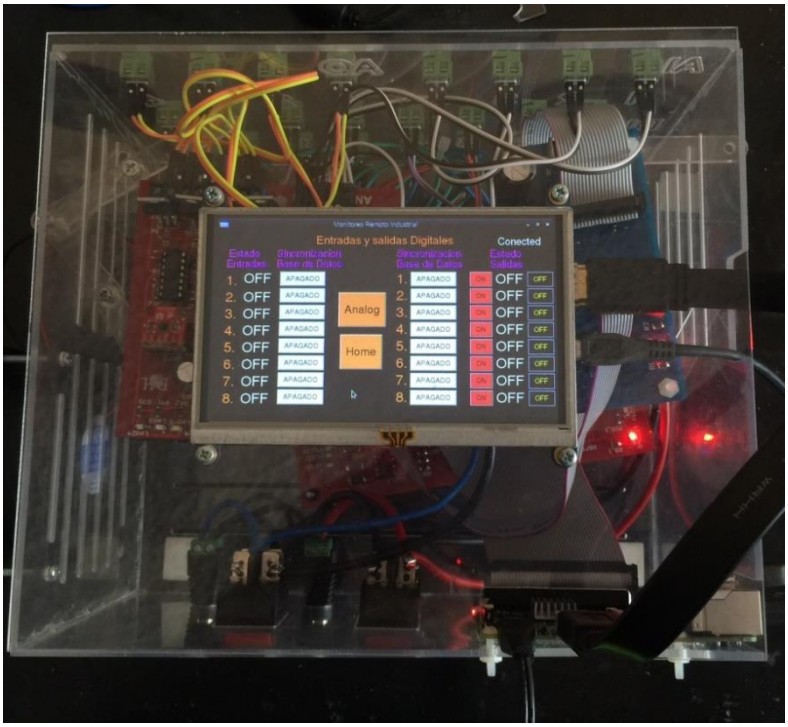
Figura 67. Diseño de los laterales restantes del case.



("Elaboración propia")

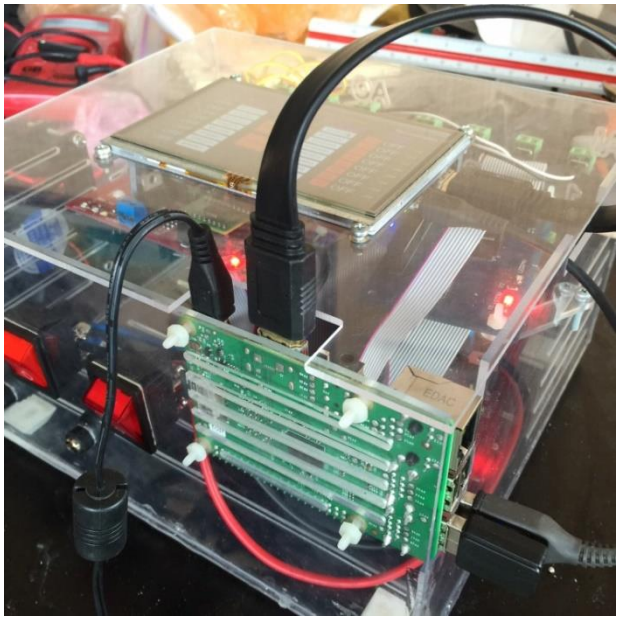
Al realizar el ensamblaje del Raspberry Pi, las placas de dispositivos digitales y analógicos, así como también de la pantalla táctil dentro del case. El dispositivo quedó ensamblado como se ve en las siguientes figuras.

Figura 68. Vista aérea del dispositivo ya ensamblado.



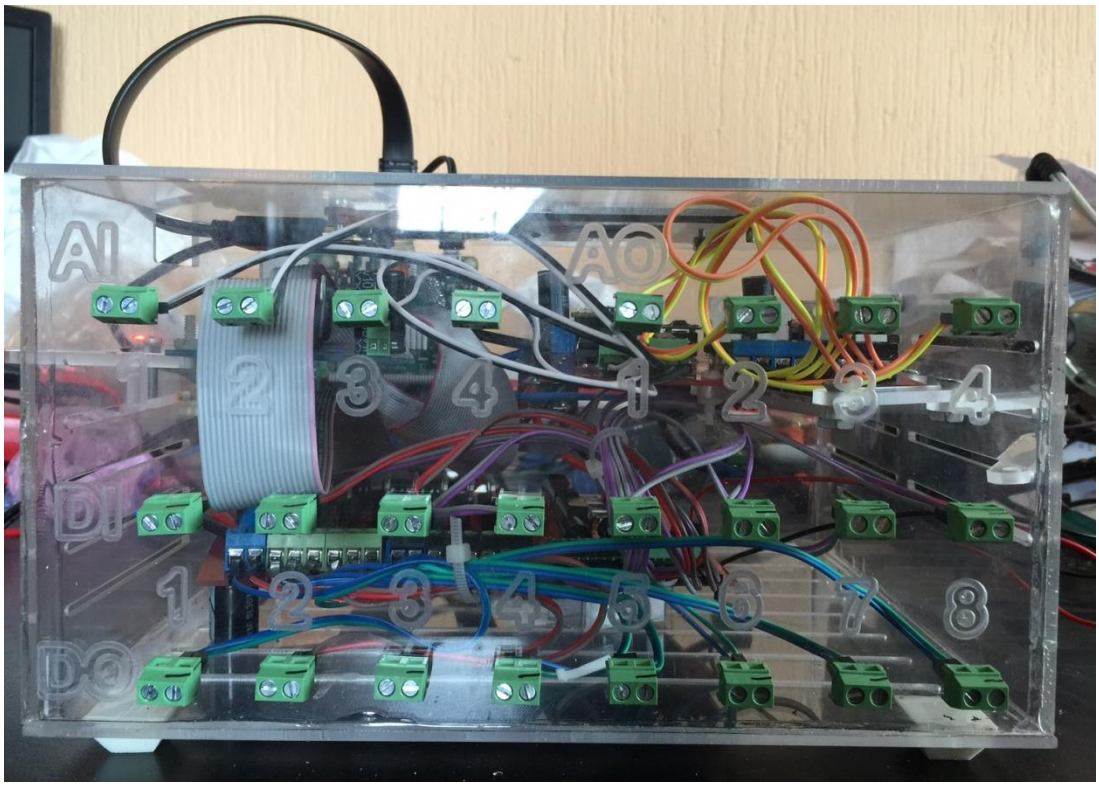
(“Elaboración propia”)

Figura 69. Vista lateral del dispositivo.



(“Elaboración propia”)

Figura 70. Vista lateral donde se encuentran las salidas y/o entradas digitales y analógicas.



(“Elaboración propia”)

VII. CONCLUSIONES

- El diseño y programación realizados permitió la construcción de un módulo capaz de monitorear y controlar señales analógicas y digitales, interconectado a través de la red celular.

- La utilización del Raspberry Pi y sus módulos de hardware como dispositivo central en el proyecto funcionó satisfactoriamente en el monitoreo remoto de las señales de los diferentes sensores.

- Al dispositivo se integraron diferentes módulos, 8 entradas digitales y 8 salidas digitales de 0 volts y 24 volts, el estándar en sensores o actuadores digitales a nivel industrial. 4 entradas analógicas y 4 salidas analógicas cada una con una configuración estándar a nivel industrial de 0 volts a 10 volts diseñada para sensores o actuadores analógicos.

- La plataforma Web diseñada logró almacenar el historial de datos recopilado de las señales monitoreadas, mostrando que es un método más eficiente y ordenado para recabar informaciones de cada dispositivo conectado.

- Se diseñó y programó una interfaz gráfica de uso sencillo para el usuario, para esto se realizó en una pantalla táctil.

VIII. RECOMENDACIONES

- Se recomienda agregar un circuito integrado que permita la expansión de los pines en el Raspberry Pi, para poder tener más entradas y salidas de señales digitales o más entradas y salidas analógicas.

- Se recomienda adjuntar un teclado alternativo para poder escribir dentro de la plataforma del dispositivo electrónico.

IX. BIBLIOGRAFÍA

- Aguilar, Domingo. *Optoacopladores*. http://www.ugr.es/~amroldan/enlaces/dispo_potencia/opto.htm#AA [Consultado Febrero 2015]
- Barragán, Javier. *Características de la Automatización Industrial.* http://uhu.es/antonio.barragan/content/caracteristicas-principales [Consultado Febrero 2015]
- Bartolomé, Jordi. *Modbus.* http://www.tolaemon.com/docs/modbus.htm [Consultado Febrero 2015]
- Broadcom Corporation. *BCM2835.* https://www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM-Peripherals.pdf [Consultado enero 2015]
- Debian. *Acerca de Raspbian*. http://www.raspbian.org/RaspbianAbout [Consultado Enero 2015]
- Electrosoft Ingeniería. *PCB*. http://www.pcb.electrosoft.cl/04-articulos-circuitos-impresos-desarrollo-sistemas/01-conceptos-circuitos-impresos/conceptos-circuitos-impresos-pcb.html [Consultado Febrero 2015]
- González, Raúl. *Python para Todos*. https://launchpadlibrarian.net/18980633/Python%20para%20todos.pdf [Consultado Enero 2015]
- Google Project Hosting. *Webiopi.* https://code.google.com/p/webiopi/wiki/INSTALL [Consulta Enero 2015]
- Greaves, David. *System on Chip.* http://www.cl.cam.ac.uk/teaching/1011/SysOnChip/socdam-notes1011.pdf [Consultado Enero 2015]
- Hart Comunication Fundation. *Acerca de protocolo Hart*. http://en.hartcomm.org/hcp/tech/aboutprotocol/aboutprotocol_specs.html [Consultado Febrero 2015]
- IST. *Protocolo HTTP.* http://informatica.uv.es/iiguia/IST/Tema2.pdf [Consultado Febrero 2015]
- Kamal, Raj. *System on Chip.*

http://www.dauniv.ac.in/downloads/EmbsysRevEd_PPTs/Chap01Lesson_7Emsys.pdf [Consultado Enero 2015]

- López, Fran. *Usos del Raspberry Pi.* http://arduprojects.blogspot.com/2013/10/definicion-y-usos-de-raspberry-pi.html [Consultado Enero 2015]
- PHP Group. *Post-Http*. http://php.net/manual/es/function.http-post-fields.php [Consultado Febrero 2015]
- Python TM. *Guía de principiantes para Python.* https://wiki.python.org/moin/BeginnersGuide [Consultado Enero 2015]
- Python. *Tutorial Python*. http://www.tutorialspoint.com/python/index.htm [Consultado Febrero 2015]
- Raspberry Pi Fundation. *Acerca de Raspberry Pi*. https://www.raspberrypi.org/about/ [Consultado Enero 2015]
- Raspberry Pi Fundation. Documentación de Python. https://www.raspberrypi.org/documentation/usage/python/ [Consultado Enero 2015]
- Raspberry Pi Fundation. *Que es un Raspberry Pi*. http://www.raspberrypi.org/help/what-is-a-raspberry-pi/ [Consultado Enero 2015]
- Raspberry Pi Fundation. *Raspberry Pi 2 modelo b.* http://www.raspberrypi.org/products/raspberry-pi-2-model-b/ [Consultado Febrero 2015]
- Raspberry Pi Fundation*. Raspberry Pi modelo b+.* http://www.raspberrypi.org/products/model-b-plus/ [Consultado Enero 2015]
- W3schools. *HTML*. http://www.w3schools.com/html/default.asp [Consultado Enero 2015]
- W3schools. *PHP*. http://www.w3schools.com/php/default.asp [Consultado Enero 2015]
- http://www.littelfuse.com/~/media/electronics/product_catalogs/littelfuse_tvs_diode_catalog.pdf.pdf

X. ANEXO

A. Código PHP para guardar datos en servidor

```
<html>
<Head>
<title>Agregar a DB en entrada analógica 1</title>
</head>
<Body>
<?php
        //Parámetros para credenciales a la base de datos. Por seguridad se han omitido
        $HOST = "monitoreoremotoindus.ipagemysql.com";
        $USERDB = "barhm90";
        $PASSDB = "tesis2015";
        $NAMEDB = "analógicas";
        $TABLENAME1 = "entradaanalogica1";
        $TABLENAME2 = "";
        // Créate conexión con la base de datos que contiene a los usuarios
        $con=mysqli_connect($HOST,$USERDB,$PASSDB,$NAMEDB);
        // Check connection
        if (mysqli_connect_errno($con))
         {
                echo "Failed to connect to MySQL : " . mysqli_connect_error();
         }
        else
         {
    $num="";
        $val = mysqli_real_escape_string($con, $_POST['valor']);
        $hour = mysqli_real_escape_string($con, $_POST['hora']);


                echo "Me conecte" ;
      echo "<br>";

                                        //Query a DB
                                        $resultado_x = mysqli_query($con,"INSERT INTO entradaanalogica1 (Valor, Hora)
VALUES ('$val', '$hour')");

                                    if($resultado_x)
                                    {
                                                                //Respuesta a servidor
                                        echo "Agregado a la base de datos";
                                    }
                                    else
                                    {
                                        echo "Error sending :(";
                                    }
            mysqli_close($con);
         }
?>
</body>
</html>
```

B. Código PHP para consulta de historial

```
<html>
<head>
<title>Entrada Analógica 1</title>
</head>
<body>
```

```
<h2>Entrada Analógica 1</h2>
<a align='center' href="http://monitoreoremotoindustrial.com/entradas-y-salidas-analogicas.html">back</a>
<br><br>
<select align='center'>
 <option value="v1">0v a 10v</option>
 <option value="v2">-10v a 10v</option>
</select>
<?php
//Parámetros para credenciales a la base de datos. Por seguridad se han omitido
        $HOST = "monitoreoremotoindus.ipagemysql.com";
        $USERDB = "barhm90";
        $PASSDB = "tesis2015";
        $NAMEDB = "analógicas";
        $TABLENAME1 = "entradaanalogica1";
        $TABLENAME2 = "";
$con=mysqli_connect($HOST,$USERDB,$PASSDB,$NAMEDB);
// Check connection
if (mysqli_connect_errno()) {
 echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
$result = mysqli_query($con,"SELECT * FROM entradaanalogica1");
echo "<table border='1' align='center' 'display=none'>
<tr>
<th>Num</th>
<th>Valor Raspi</th>
<th>Valor Entrada</th>
<th>Hora</th>
</tr>";
while($row = mysqli_fetch_array($result)) {
 echo "<tr>";
 echo "<td>" . $row['Num'] . "</td>";
 echo "<td>" . $row['Valor'] . "</td>";
 echo"<td>" . ($row['Valor'])*(3.03). "</td>";
 echo "<td>" . $row['Hora'] . "</td>";
 echo "</tr>";
}
echo "</table>";
mysqli_close($con);
?>
</body>
</html>
```

C. Código PHP para limpiar historial.

```
<?php
//Parámetros para credenciales a la base de datos. Por seguridad se han omitido
$HOST = "monitoreoremotoindus.ipagemysql.com";
$USERDB = "barhm90";
$PASSDB = "tesis2015";
$NAMEDB = "analógicas";
$TABLENAME1 = "entradaanalogica1";
$TABLENAME2 = "";
// Create conexión con la base de datos que contiene a los usuarios
$con=mysqli_connect($HOST,$USERDB,$PASSDB,$NAMEDB);
// Check connection
if ($con->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
// sql to delete a record
$sql = "TRUNCATE  entradaanalogica1";

if ($con->query($sql) === TRUE) {
   echo "Record deleted successfully";
} else {
   echo "Error deleting record: " . $conn->error;
```

```
}
$con->close();
?>
```

D. Código Python en el Raspberry Pi.

```
#Universidad del Valle de Guatemala
#Bryan Allan Roberto Hidalgo Montenegro
#Carne 08222
#Trabajo de Graduación

#se importan las librerías necesarias
import sys
import smbus
from smbus import SMBus
import threading
from Adafruit_I2C import Adafruit_I2C
from Adafruit_MCP4725 import MCP4725
#se importan los drivers del ADC y DAC
from webiopi.devices.analog import MCP3204
from webiopi.devices.analog import MCP4921
#se importa la librería para los GPIO's
import RPi.GPIO as GPIO
#se importa librería para la interfaz
from Tkinter import *
import time
from datetime import datetime
import requests



dac = MCP4725(0x62)           #se configura el dac en la memoria de I2C
dac2 = MCP4725(0X63)          #se configura el dac2 en la memoria de I2C
dac3= MCP4725(0X60)
dac4= MCP4725(0X61)
adc0= MCP3204(1)         #se configura el ADC en SPI
mcp0= MCP4921(0)
hora = datetime.now().time()    #configuración de la hora

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)     #Configuración de los GPIO'S

GPIO.setup(17, GPIO.OUT)    #salida digital 1
GPIO.output(17,False)
GPIO.setup(27, GPIO.OUT)    #salida digital 2
GPIO.output(27,False)
GPIO.setup(22, GPIO.OUT)    #salida digital 3
GPIO.output(22,False)
GPIO.setup(5, GPIO.OUT)    #salida digital 4
GPIO.output(5,False)
GPIO.setup(6, GPIO.OUT)    #salida digital 5
GPIO.output(6,False)
GPIO.setup(13, GPIO.OUT)    #salida digital 6
GPIO.output(13,False)
GPIO.setup(19, GPIO.OUT)    #salida digital 7
GPIO.output(19,False)
GPIO.setup(26, GPIO.OUT)    #salida digital 8
GPIO.output(26,False)

GPIO.setup(12, GPIO.IN)     #Entrada digital INDIG
GPIO.setup(16, GPIO.OUT)     #Entrada digital S0
GPIO.output(16,True)
GPIO.setup(20, GPIO.OUT)     #Entrada digital S1
GPIO.output(20,True)
GPIO.setup(21, GPIO.OUT)     #Entrada digital S2
```

```
GPIO.output(21,True)

global CS
CS=18
global DOUT
DOUT=23
global CLK
CLK=24
global VREFINHART
VREFINHART=2.042
GPIO.setup(CS, GPIO.OUT)    #4-20mA Entrada
GPIO.setup(DOUT, GPIO.IN)
GPIO.setup(CLK,GPIO.OUT)


root = Tk()               #configuración de ventanas para la interfaz
miframe=Frame(root)       #configuración de la ventana principal
miframeADC1=Frame(root)    #configuración para cada ventana adicional
miframeADC2=Frame(root)
miframeADC3=Frame(root)
miframeADC4=Frame(root)
miframeDAC=Frame(root)
miframeDAC2=Frame(root)
miframeDAC3=Frame(root)
miframeDAC4=Frame(root)

miframeINHART=Frame(root)
miframeOUTHART=Frame(root)
miframeDIGITALES=Frame(root)
miframeANALOG=Frame(root)
miframeHART=Frame(root)
miframeDI1=Frame(root)
miframeDI2=Frame(root)
miframeDI3=Frame(root)
miframeDI4=Frame(root)
miframeDI5=Frame(root)
miframeDI6=Frame(root)
miframeDI7=Frame(root)
miframeDI8=Frame(root)
miframeDO1=Frame(root)
miframeDO2=Frame(root)
miframeDO3=Frame(root)
miframeDO4=Frame(root)
miframeDO5=Frame(root)
miframeDO6=Frame(root)
miframeDO7=Frame(root)
miframeDO8=Frame(root)

root.geometry("800x480")       #se configura los pixeles de la ventana


#configuración variables de sincronización con base de datos
global configadc1
configadc1=5
global banderaadc1
banderaadc1=0
global configadc2
configadc2=5
global banderaadc2
banderaadc2=0
global configadc3
configadc3=5
global banderaadc3
banderaadc3=0
global configadc4
configadc4=5
```

```
global banderaadc4
banderaadc4=0
global t1
global finished

def VALOR10ADC1():     #método para colocar configuracion adc1 en 10v
  global configadc1
  configadc1=10
  ADC1()

def VALOR5ADC1():      #método para colocar configuracion adc1 en 5v
  global configadc1
  configadc1=5
  ADC1()

def VALOR10ADC2():     #método para colocar configuracion adc2 en 10v
  global configadc2
  configadc2=10
  ADC2()

def VALOR5ADC2():      #método para colocar configuracion adc2 en 5v
  global configadc2
  configadc2=5
  ADC2()

def VALOR10ADC3():     #método para colocar configuracion adc3 en 10v
  global configadc3
  configadc3=10
  ADC3()

def VALOR5ADC3():      #método para colocar configuracion adc3 en 5v
  global configadc3
  configadc3=5
  ADC3()

def VALOR10ADC4():     #método para colocar configuracion adc3 en 10v
  global configadc4
  configadc4=10
  ADC4()

def VALOR5ADC4():      #método para colocar configuracion adc3 en 5v
  global configadc4
  configadc4=5
  ADC4()

def MANDARDB():        #manda datos a base de datos
  global configadc1
  configadc1=int((requests.get('http://monitoreoremotoindustrial.com/tesis/entradaanalogica1json.php')).content.strip("\n"))
  global configadc2
  configadc2=int((requests.get('http://monitoreoremotoindustrial.com/tesis/entradaanalogica2json.php')).content.strip("\n"))
  global configadc3
  configadc3=int((requests.get('http://monitoreoremotoindustrial.com/tesis/entradaanalogica3json.php')).content.strip("\n"))
  global configadc4
  configadc4=int((requests.get('http://monitoreoremotoindustrial.com/tesis/entradaanalogica3json.php')).content.strip("\n"))
  while not t1.shutdown:
    hora = datetime.now().time()
    if (configadc1==5 and banderaadc1==1):
      displayADC1 = "%.4f" %(1.67*adc0.analogReadVolt(0))          #lee voltaje del ADC
      parameters = {'valor':displayADC1,'hora':hora,'configadc1':configadc1}  #establece parámetros para enviar
      #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
      request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradaanalogica1.php', data=parameters)
    if(configadc1==10 and banderaadc1==1):
      displayADC1 = "%.4f" %(3.3*adc0.analogReadVolt(0))          #lee voltaje del ADC
      parameters = {'valor':displayADC1,'hora':hora,'configadc1':configadc1}  #establece parámetros para enviar
      #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
      request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradaanalogica1.php', data=parameters)
```

```
        if (configadc2==5 and banderaadc2==1):
            displayADC2 = "%.4f" %(1.67*adc0.analogReadVolt(1))         #lee voltaje del ADC 2
            parameters = {'valor':displayADC2,'hora':hora,'configadc2':configadc2}  #establece parámetros para enviar
            #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
            request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradaanalogica2.php', data=parameters)
        if (configadc2==10 and banderaadc2==1):
            displayADC2 = "%.4f" %(3.3*adc0.analogReadVolt(1))         #lee voltaje del ADC 2
            parameters = {'valor':displayADC2,'hora':hora, 'configadc2':configadc2}  #establece parámetros para enviar
            #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
            request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradaanalogica2.php', data=parameters)
        if (configadc3==5 and banderaadc3==1):
            displayADC3 = "%.4f" %(1.67*adc0.analogReadVolt(2))  #lee voltaje del ADC 3
            parameters = {'valor':displayADC3,'hora':hora,'configadc3':configadc3}  #establece parámetros para enviar
            #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
            request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradaanalogica3.php', data=parameters)
        if (configadc3==10 and banderaadc3==1):
            displayADC3 = "%.4f" %(3.3*adc0.analogReadVolt(2))  #lee voltaje del ADC 3
            parameters = {'valor':displayADC3,'hora':hora,'configadc3':configadc3}  #establece parámetros para enviar
            #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
            request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradaanalogica3.php', data=parameters)
        if (configadc4==10 and banderaadc4==1):
            displayADC4 = "%.4f" %(1.67*adc0.analogReadVolt(3))         #lee voltaje del ADC 4
            parameters = {'valor':displayADC4,'hora':hora, 'configadc4':configadc4}  #establece parámetros para enviar
            #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
            request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradaanalogica4.php', data=parameters)
        elif (configadc4==10 and banderaadc4==1):
            displayADC4 = "%.4f" %(1.67*adc0.analogReadVolt(3))         #lee voltaje del ADC 4
            parameters = {'valor':displayADC4,'hora':hora, 'configadc4':configadc4}  #establece parámetros para enviar
            #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
            request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradaanalogica4.php', data=parameters)

        time.sleep(20)

def ADC1():                 #comando para la entrada analógica 1
    limpiar()               #llama al comando para limpiar la ventana
    global banderaadc1
    banderaadc1=1
    root.title("ADC1")      #coloca título a ventana
    root["bg"]="white"      #cambia de color a ventana
    miframeADC1.pack()      #llama a ventana de entrada analógica 1
    miframeADC1["bg"]="white"   #coloca color adentro de ventana
    if (configadc1==5):
        displayADC1 = "%.4f" %(1.67*adc0.analogReadVolt(0))         #lee voltaje del ADC
        hora = datetime.now().time()
        parameters = {'valor':displayADC1,'hora':hora,'configadc1':configadc1}  #establece parámetros para enviar
        #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradaanalogica1.php', data=parameters)
        labadc1disp =Label( miframeADC1, bg="white",text= "       ", font=("Arial","24")).grid(row=1, column =2)
        labadc1disp =Label( miframeADC1, bg="white",text= displayADC1, font=("Arial","24")).grid(row=1, column =2)

    else:
        displayADC1 = "%.4f" %(3.3*adc0.analogReadVolt(0))         #lee voltaje del ADC
        hora = datetime.now().time()
        parameters = {'valor':displayADC1,'hora':hora,'configadc1':configadc1}  #establece parámetros para enviar
        #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradaanalogica1.php', data=parameters)
        #escribe en la ventana el título, despliega el valor que se lee
        labadc1disp =Label( miframeADC1, bg="white",text= "       ", font=("Arial","24")).grid(row=1, column =2)
        labadc1disp =Label( miframeADC1, bg="white",text= displayADC1, font=("Arial","24")).grid(row=1, column =2)


    #escribe en la ventana el título, despliega el valor que se lee
    labadc1title = Label( miframeADC1, bg="white",text="ENTRADA ANALOGICA 1", font=("Arial","24")).grid(row=0, column =2)
    #botones para la navegación en la interfaz
    homebutton = Button(miframeADC1,text="HOME",fg="yellow",bg="black", font=("Arial","24"),command=main).grid(row = 2,
column = 2)
```

```
   refresh = Button(miframeADC1,text="REFRESH",fg="Red",bg="black", font=("Arial","24"),command=ADC1).grid(row = 3, column =
2)
   backbutton = Button(miframeADC1,text="BACK",fg="blue",bg="black", font=("Arial","24"),command=ANALOG).grid(row = 4,
column = 2)
   R1=Radiobutton(miframeADC1, text="0v-5v", command=VALOR5ADC1).grid(row=5, column=1)
   R2=Radiobutton(miframeADC1, text="0v-10v", command=VALOR10ADC1).grid(row=5, column=3)




def ADC2():                  #comando para la entrada analógica 2
   limpiar()                 #llama al comando para limpiar la ventana
   global banderaadc2
   banderaadc2=1
   root.title("ADC2")        #coloca título a ventana
   root["bg"]="white"
   miframeADC2.pack()        #llama a ventana de entrada analogica 2
   miframeADC2["bg"]="white"    #coloca color adentro de ventana
   if (configadc2==5):
      displayADC2 = "%.4f" %(1.67*adc0.analogReadVolt(1))         #lee voltaje del ADC 2
      hora = datetime.now().time()
      parameters = {'valor':displayADC2,'hora':hora,'configadc2':configadc2}  #establece parámetros para enviar
      #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
      request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradaanalogica2.php', data=parameters)
      labadc1disp =Label( miframeADC2,bg="white", text= displayADC2, font=("Arial","24")).grid(row=1, column =2)
   else:
      displayADC2 = "%.4f" %(3.3*adc0.analogReadVolt(1))         #lee voltaje del ADC 2
      hora = datetime.now().time()
      parameters = {'valor':displayADC2,'hora':hora, 'configadc2':configadc2}  #establece parámetros para enviar
      #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
      request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradaanalogica2.php', data=parameters)
      #escribe en la ventana el título, despliega el valor que se lee
      labadc1disp =Label( miframeADC2,bg="white", text= displayADC2, font=("Arial","24")).grid(row=1, column =2)

   #escribe en la ventana el título, despliega el valor que se lee
   labadc2title = Label( miframeADC2,bg="white", text="ADC2", font=("Arial","24")).grid(row=0, column =2)
   #botones para la navegación en la interfaz
   homebutton = Button(miframeADC2,text="HOME",fg="yellow" ,bg="black",font=("Arial","24"),command=main).grid(row = 2,
column = 2)
   refresh = Button(miframeADC2,text="REFRESH",fg="Red",bg="black", font=("Arial","24"),command=ADC2).grid(row = 3, column =
2)
   backbutton = Button(miframeADC2,text="BACK",fg="blue",bg="black", font=("Arial","24"),command=ANALOG).grid(row = 4,
column = 2)
   R1=Radiobutton(miframeADC2, text="0v-5v", command=VALOR5ADC2).grid(row=5, column=1)
   R2=Radiobutton(miframeADC2, text="0v-10v", command=VALOR10ADC2).grid(row=5, column=3)


def ADC3():                        #comando para la entrada analogica 3
   limpiar()
   global banderaadc3
   banderaadc3=1
   root.title("ADC3")                 #coloca título a ventana
   root["bg"]="white"
   miframeADC3.pack()
   miframeADC1["bg"]="white"             #coloca color adentro de ventana
   if (configadc3==5):
      displayADC3 = "%.4f" %(1.67*adc0.analogReadVolt(2))  #lee voltaje del ADC 3
      hora = datetime.now().time()
      parameters = {'valor':displayADC3,'hora':hora,'configadc3':configadc3} #establece parámetros para enviar
      #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
      request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradaanalogica3.php', data=parameters)
      labadc1disp =Label( miframeADC3, bg="white",text= displayADC3, font=("Arial","24")).grid(row=1, column =2)
   else:
      displayADC3 = "%.4f" %(3.3*adc0.analogReadVolt(2))  #lee voltaje del ADC 3
      hora = datetime.now().time()
```

```
    parameters = {'valor':displayADC3,'hora':hora,'configadc3':configadc3}  #establece parámetros para enviar
    #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
    request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradaanalogica3.php', data=parameters)
    labadc1disp =Label( miframeADC3, bg="white",text= displayADC3, font=("Arial","24")).grid(row=1, column =2)

  #escribe en la ventana el título, despliega el valor que se lee
  labadc3title = Label( miframeADC3,bg="white", text="ADC3", font=("Arial","24")).grid(row=0, column =2)
  #botones para la navegación en la interfaz
  homebutton = Button(miframeADC3,text="HOME",fg="Yellow", bg="black",font=("Arial","24"),command=main).grid(row = 2,
column = 2)
  refresh = Button(miframeADC3,text="REFRESH",fg="Red",bg="black", font=("Arial","24"),command=ADC3).grid(row = 3, column =
2)
  backbutton = Button(miframeADC3,text="BACK",fg="blue",bg="black", font=("Arial","24"),command=ANALOG).grid(row = 4,
column = 2)
  R1=Radiobutton(miframeADC3, text="0v-5v", command=VALOR5ADC3).grid(row=5, column=1)
  R2=Radiobutton(miframeADC3, text="0v-10v", command=VALOR10ADC3).grid(row=5, column=3)


def ADC4():                    #comando para la entrada analogica 4

  limpiar()
  global banderaadc4
  banderaadc4=1
  root.title("ADC4")             #coloca título a ventana
  root["bg"]="white"
  miframeADC4.pack()
  miframeADC4["bg"]="white"        #coloca color adentro de ventana
  if (configadc3==5):
    displayADC4 = "%.4f" %(1.67*adc0.analogReadVolt(3))         #lee voltaje del ADC 4
    hora = datetime.now().time()
    parameters = {'valor':displayADC4,'hora':hora, 'configadc4':configadc4}  #establece parámetros para enviar
    #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
    request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradaanalogica4.php', data=parameters)
    labadc1disp =Label( miframeADC4, bg="white",text= displayADC4, font=("Arial","24")).grid(row=1, column =2)
  else:
    displayADC4 = "%.4f" %(3.3*adc0.analogReadVolt(3))         #lee voltaje del ADC 4
    hora = datetime.now().time()
    parameters = {'valor':displayADC4,'hora':hora, 'configadc4':configadc4}  #establece parámetros para enviar
    #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
    request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradaanalogica4.php', data=parameters)
    labadc1disp =Label( miframeADC4, bg="white",text= displayADC4, font=("Arial","24")).grid(row=1, column =2)

  #escribe en la ventana el título, despliega el valor que se lee
  labadc4title = Label( miframeADC4,bg="white", text="ADC4", font=("Arial","24")).grid(row=0, column =2)
  #botones para la navegación en la interfaz
  homebutton = Button(miframeADC4,text="home",fg="yellow", bg="black", font=("Arial","24"),command=main).grid(row = 2,
column = 2)
  refresh = Button(miframeADC4,text="REFRESH",fg="Red", bg="black", font=("Arial","24"),command=ADC4).grid(row = 3, column =
2)
  backbutton = Button(miframeADC4,text="BACK",fg="blue", bg="black", font=("Arial","24"),command=ANALOG).grid(row = 4,
column = 2)
  R1=Radiobutton(miframeADC4, text="0v-5v", command=VALOR5ADC4).grid(row=5, column=1)
  R2=Radiobutton(miframeADC4, text="0v-10v", command=VALOR10ADC4).grid(row=5, column=3)


def DAC():                   #comando para la salida analogica 1
  print "boton DAC1"
  limpiar()
  root.title("DAC")            #coloca título a ventana
  root["bg"]="white"
  miframeDAC.pack()            #coloca paquete de ventana
  miframeDAC["bg"]="white"
  vdac1= 2030
  dac.setVoltage(vdac1)         #Coloca voltaje en la salida analogica 1 y hace operaciones para mandar los bits correctos
  v2dac1=(vdac1*3.29)/4096
  parameters = {'valor':v2dac1,'hora':hora}  #establece los parámetros para envió
```

```
   #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
   request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarsalidaanalogica1.php', data=parameters)
   #escribe en la ventana el título, despliega el valor que se lee
   labdactitle = Label( miframeDAC, bg="white",text="DAC1", font=("Arial","24")).grid(row=0, column =2)
   labadc1disp =Label( miframeDAC, bg="white",text= v2dac1, font=("Arial","24")).grid(row=1, column =2)
   #botones para la navegación en la interfaz
   homebutton = Button(miframeDAC,text="HOME",fg="yellow", bg="black",font=("Arial","24"),command=main).grid(row = 2,
column = 2)
   backbutton = Button(miframeDAC,text="BACK",fg="Red", bg="black",font=("Arial","24"),command=ANALOG).grid(row = 3, column
= 2)

def DAC2():                #comando para la salida analogica 2
   print "boton DAC2"
   limpiar()
   root.title("DAC2")       #coloca título a ventana
   root["bg"]="white"
   miframeDAC2.pack()
   miframeDAC2["bg"]="white"
   vdac2= 2000
   dac2.setVoltage(vdac2)     #Coloca voltaje en la salida analogica 2 y hace operaciones para mandar los bits correctos
   v2dac2=(vdac2*3.29)/4096
   parameters = {'valor':v2dac2,'hora':hora}  #establece los parámetros para envió
   #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
   request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarsalidaanalogica2.php', data=parameters)
   #escribe en la ventana el título, despliega el valor que se lee
   labdactitle = Label( miframeDAC2, bg="white", text="DAC2", font=("Arial","24")).grid(row=0, column =2)
   labadc1disp =Label( miframeDAC2, bg="white",text= v2dac2, font=("Arial","24")).grid(row=1, column =2)
   #botones para la navegación en la interfaz
   homebutton = Button(miframeDAC2,text="HOME",fg="Red", font=("Arial","24"),command=main).grid(row = 2, column = 2)
   backbutton = Button(miframeDAC2,text="BACK",fg="Red", font=("Arial","24"),command=ANALOG).grid(row = 3, column = 2)

def DAC3():                #comando para la salida analogica 3
   print "boton DAC3"
   limpiar()
   root.title("DAC3")       #coloca título a ventana
   miframeDAC3.pack()
   vdac3= 1000
   #dac3.setVoltage(vdac3)          #Coloca voltaje en la salida analogica 3 y hace operaciones para mandar los bits correctos
   v2dac3=(vdac3*3.29)/4096
   parameters = {'valor':v2dac3,'hora':hora}
   #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
   request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarsalidaanalogica3.php', data=parameters)
   #escribe en la ventana el título, despliega el valor que se lee
   labdactitle = Label( miframeDAC3, bg="white", text="DAC3", font=("Arial","24")).grid(row=0, column =2)
   labadc1disp =Label( miframeDAC3, bg="white",text= v2dac3, font=("Arial","24")).grid(row=1, column =2)
   #botones para la navegación en la interfaz
   homebutton = Button(miframeDAC3,text="HOME",fg="yellow", bg="black", font=("Arial","24"),command=main).grid(row = 2,
column = 2)
   backbutton = Button(miframeDAC3,text="BACK",fg="Red", bg="black",font=("Arial","24"),command=ANALOG).grid(row = 3,
column = 2)

def DAC4():              #comando para la salida analogica 4
   print "boton DAC3"
   limpiar()
   root.title("DAC3")       #coloca título a ventana
   miframeDAC4.pack()
   vdac4= 1500
   #dac4.setVoltage(vdac4)    #Coloca voltaje en la salida analogica 3 y hace operaciones para mandar los bits correctos
   v2dac4=(vdac4*3.29)/4096
   parameters = {'valor':v2dac4,'hora':hora}
   #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
   request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarsalidaanalogica4.php', data=parameters)
   #escribe en la ventana el título, despliega el valor que se lee
   labdactitle = Label( miframeDAC4, text="DAC4", bg="white", font=("Arial","24")).grid(row=0, column =2)
   labadc1disp =Label( miframeDAC4, bg="white",text= v2dac4, font=("Arial","24")).grid(row=1, column =2)
   #botones para la navegación en la interfaz
```

```
   homebutton = Button(miframeDAC4,text="HOME",fg="yellow", bg="black", font=("Arial","24"),command=main).grid(row = 2,
column = 2)
   backbutton = Button(miframeDAC4,text="BACK",fg="Red", bg="black", font=("Arial","24"),command=ANALOG).grid(row = 3,
column = 2)



def DI1():                    #comando para entrada digital 1
   limpiar()
   root.title("ENTRADA DIGITAL 1") #coloca título en la ventana
   miframeDI1.pack()
   GPIO.output(16,True)
   GPIO.output(20,True)
   GPIO.output(21,True)
   if GPIO.input(12) == 1:        #evalúa si en la entrada 1 hay voltaje
      DI1var= "24"
      DI1var2="ON"
      hora = datetime.now().time()
      parameters = {'valor':DI1var,'hora':hora,'estado':DI1var2}
      #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
      request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradadigital1.php', data=parameters)
   else:                      #si no, indica que está apagado
      DI1var= " 0 Volts "
      DI1var2="OFF"
      hora = datetime.now().time()
      parameters = {'valor':DI1var,'hora':hora,'estado':DI1var2}
      #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
      request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradadigital1.php', data=parameters)

   #escribe en la ventana el valor que se lee
   labdis =Label( miframeDI1, text= DI1var, font=("Arial","24")).grid(row=2, column =2)
   #botones para la navegación en la interfaz
   homebutton = Button(miframeDI1,text="HOME",fg="Red", font=("Arial","24"),command=main).grid(row = 3, column = 2)
   backbutton = Button(miframeDI1,text="BACK",fg="Red", font=("Arial","24"),command=DIGITAL).grid(row = 4, column = 2)
   refresh= Button(miframeDI1, text="REFRESH", fg="blue",font=("Arial","24"),command=DI1).grid(row = 5, column = 2)

def DI2():                    #comando para entrada digital 2
   limpiar()
   root.title("ENTRADA DIGITAL 2") #coloca título en la ventana
   miframeDI2.pack()
   GPIO.output(16,True)
   GPIO.output(20,True)
   GPIO.output(21,False)
   if GPIO.input(12) == 1:    #evalúa si en la entrada 2 hay voltaje
      DI2var= "24"
      DI2var2="ON"
      hora = datetime.now().time()
      parameters = {'valor':DI2var,'hora':hora,'estado':DI2var2}
      #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
      request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradadigital2.php', data=parameters)

   else:                      #si no, indica que está apagado
      DI2var= "0"
      DI2var2="OFF"
      hora = datetime.now().time()
      parameters = {'valor':DI2var,'hora':hora,'estado':DI2var2}
      #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
      request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradadigital2.php', data=parameters)

   #escribe en la ventana el valor que se lee
   labdis =Label( miframeDI2, text= DI2var, font=("Arial","24")).grid(row=2, column =2)
   #botones para la navegación en la interfaz
   homebutton = Button(miframeDI2,text="HOME",fg="Red", font=("Arial","24"),command=main).grid(row = 3, column = 2)
   backbutton = Button(miframeDI2,text="BACK",fg="Red", font=("Arial","24"),command=DIGITAL).grid(row = 4, column = 2)
   refresh= Button(miframeDI2, text="REFRESH", fg="blue",font=("Arial","24"),command=DI2).grid(row = 5, column = 2)
```

```
def DI3():              #comando para entrada digital 3
    limpiar()
    root.title("ENTRADA DIGITAL 3")
    miframeDI3.pack()
    GPIO.output(16,True)
    GPIO.output(20,False)
    GPIO.output(21,True)
    if GPIO.input(12) == 1:    #evalúa si en la entrada 3 hay voltaje
        DI3var= "24"
        DI3var2="ON"
        hora = datetime.now().time()
        parameters = {'valor':DI3var,'hora':hora,'estado':DI3var2}
        #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradadigital3.php', data=parameters)

    else:                   #si no, indica que está apagado
        DI3var= "0"
        DI3var2="OFF"
        hora = datetime.now().time()
        parameters = {'valor':DI3var,'hora':hora,'estado':DI3var2}
        #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradadigital3.php', data=parameters)

    #escribe en la ventana el valor que se lee
    labdis =Label( miframeDI3, text= DI3var, font=("Arial","24")).grid(row=2, column =2)
    #botones para la navegación en la interfaz

    homebutton = Button(miframeDI3,text="HOME",fg="Red", font=("Arial","24"),command=main).grid(row = 3, column = 2)
    backbutton = Button(miframeDI3,text="BACK",fg="Red", font=("Arial","24"),command=DIGITAL).grid(row = 4, column = 2)
    refresh= Button(miframeDI3, text="REFRESH", fg="blue",font=("Arial","24"),command=DI3).grid(row = 5, column = 2)

def DI4():              #comando de entrada digital 4
    limpiar()
    root.title("ENTRADA DIGITAL 4")
    miframeDI4.pack()
    GPIO.output(16,True)
    GPIO.output(20,False)
    GPIO.output(21,False)
    if GPIO.input(12) == 1:    #evalúa si en la entrada 4 hay voltaje
        DI4var="24"
        DI4var2="ON"
        hora = datetime.now().time()
        parameters = {'valor':DI4var,'hora':hora,'estado':DI4var2}
        #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradadigital4.php', data=parameters)

    else:                   #si no, indica que está apagado
        DI4var= "0"
        DI4var2="OFF"
        hora = datetime.now().time()
        parameters = {'valor':DI4var,'hora':hora,'estado':DI4var2}
        #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradadigital4.php', data=parameters)

    #escribe en la ventana el valor que se lee
    labdis =Label( miframeDI4, text= DI4var, font=("Arial","24")).grid(row=2, column =2)
    #botones para la navegación en la interfaz
    homebutton = Button(miframeDI4,text="HOME",fg="Red", font=("Arial","14"),command=main).grid(row = 3, column = 2)
    backbutton = Button(miframeDI4,text="BACK",fg="Red", font=("Arial","14"),command=DIGITAL).grid(row = 4, column = 2)
    refresh= Button(miframeDI4, text="REFRESH", fg="blue",font=("Arial","24"),command=DI4).grid(row = 5, column = 2)

def DI5():
    limpiar()
    root.title("ENTRADA DIGITAL 5")
    miframeDI5.pack()
```

```
    GPIO.output(16,False)
    GPIO.output(20,True)
    GPIO.output(21,True)
    if GPIO.input(12) == 1:    #evalúa si en la entrada 4 hay voltaje
        DI5var="24"
        DI5var2="ON"
        hora = datetime.now().time()
        parameters = {'valor':DI5var,'hora':hora,'estado':DI5var2}
        #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradadigital5.php', data=parameters)

    else:                      #si no, indica que está apagado
        DI5var= "0"
        DI5var2="OFF"
        hora = datetime.now().time()
        parameters = {'valor':DI5var,'hora':hora,'estado':DI5var2}
        #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradadigital5.php', data=parameters)

    #escribe en la ventana el valor que se lee
    labdis =Label( miframeDI5, text= DI5var, font=("Arial","24")).grid(row=2, column =2)
    #botones para la navegación en la interfaz

    homebutton = Button(miframeDI5,text="HOME",fg="Red", font=("Arial","24"),command=main).grid(row = 3, column = 2)
    backbutton = Button(miframeDI5,text="BACK",fg="Red", font=("Arial","24"),command=DIGITAL).grid(row = 4, column = 2)
    refresh= Button(miframeDI5, text="REFRESH", fg="blue",font=("Arial","24"),command=DI5).grid(row = 5, column = 2)

def DI6():
    limpiar()
    root.title("ENTRADA DIGITAL 6")
    miframeDI6.pack()
    GPIO.output(16,False)
    GPIO.output(20,True)
    GPIO.output(21,False)
    if GPIO.input(12) == 1:    #evalúa si en la entrada 4 hay voltaje
        DI6var="24"
        DI6var2="ON"
        hora = datetime.now().time()
        parameters = {'valor':DI6var,'hora':hora,'estado':DI6var2}
        #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradadigital6.php', data=parameters)

    else:                      #si no, indica que está apagado
        DI6var= "0"
        DI6var2="OFF"
        hora = datetime.now().time()
        parameters = {'valor':DI6var,'hora':hora,'estado':DI6var2}
        #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradadigital6.php', data=parameters)

    #escribe en la ventana el valor que se lee
    labdis =Label( miframeDI6, text= DI6var, font=("Arial","24")).grid(row=2, column =2)
    #botones para la navegación en la interfaz

    homebutton = Button(miframeDI6,text="HOME",fg="Red", font=("Arial","24"),command=main).grid(row = 3, column = 2)
    backbutton = Button(miframeDI6,text="BACK",fg="Red", font=("Arial","24"),command=DIGITAL).grid(row = 4, column = 2)
    refresh= Button(miframeDI6, text="REFRESH", fg="blue",font=("Arial","24"),command=DI6).grid(row = 5, column = 2)

def DI7():
    limpiar()
    root.title("ENTRADA DIGITAL 7")
    miframeDI7.pack()
    GPIO.output(16,False)
    GPIO.output(20,False)
    GPIO.output(21,True)
    if GPIO.input(12) == 1:    #evalúa si en la entrada 4 hay voltaje
```

```
        DI7var="24"
        DI7var2="ON"
        hora = datetime.now().time()
        parameters = {'valor':DI7var,'hora':hora,'estado':DI7var2}
        #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradadigital7.php', data=parameters)

    else:                   #si no, indica que está apagado
        DI7var= "0"
        DI7var2="OFF"
        hora = datetime.now().time()
        parameters = {'valor':DI7var,'hora':hora,'estado':DI7var2}
        #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradadigital7.php', data=parameters)

    #escribe en la ventana el valor que se lee
    labdis =Label( miframeDI7, text= DI7var, font=("Arial","24")).grid(row=2, column =2)
    #botones para la navegación en la interfaz

    homebutton = Button(miframeDI7,text="HOME",fg="Red", font=("Arial","24"),command=main).grid(row = 3, column = 2)
    backbutton = Button(miframeDI7,text="BACK",fg="Red", font=("Arial","24"),command=DIGITAL).grid(row = 4, column = 2)
    refresh= Button(miframeDI7, text="REFRESH", fg="blue",font=("Arial","24"),command=DI7).grid(row = 5, column = 2)

def DI8():
    limpiar()
    root.title("ENTRADA DIGITAL 8")
    miframeDI8.pack()
    GPIO.output(16,False)
    GPIO.output(20,False)
    GPIO.output(21,False)
    if GPIO.input(12) == 1:     #evalúa si en la entrada 4 hay voltaje
        DI8var="24"
        DI8var2="ON"
        hora = datetime.now().time()
        parameters = {'valor':DI8var,'hora':hora,'estado':DI8var2}
        #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradadigital8.php', data=parameters)

    else:                   #si no, indica que está apagado
        DI8var= "0"
        DI8var2="OFF"
        hora = datetime.now().time()
        parameters = {'valor':DI8var,'hora':hora,'estado':DI8var2}
        #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradadigital8.php', data=parameters)

    #escribe en la ventana el valor que se lee
    labdis =Label( miframeDI8, text= DI8var, font=("Arial","24")).grid(row=2, column =2)
    #botones para la navegación en la interfaz

    homebutton = Button(miframeDI8,text="HOME",fg="Red", font=("Arial","24"),command=main).grid(row = 3, column = 2)
    backbutton = Button(miframeDI8,text="BACK",fg="Red", font=("Arial","24"),command=DIGITAL).grid(row = 4, column = 2)
    refresh= Button(miframeDI8, text="REFRESH", fg="blue",font=("Arial","24"),command=DI8).grid(row = 5, column = 2)

def DO1():                  #comando de salida digital 1
    limpiar()
    root.title("SALIDA DIGITAL 1")
    miframeDO1.pack()
    def DO1ON():
        GPIO.output(17,True)       #enciende salida digital 1
        DO1var2= 24
        DO1var=" ON "              #activa etiqueta de encendido
        labdo1 = Label( miframeDO1, bg="black", fg="red", text=DO1var, font=("Arial","24")).grid(row=3, column =2)
        parameters = {'valor':DO1var2,'hora':hora,'estado':DO1var}
        #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarsalidadigital1.php', data=parameters)
```

```
  def DO1OFF():
    GPIO.output(17,False)      #apaga salida digital 1
    DO1var2= 0
    DO1var="OFF"               #activa etiqueta de apagado
    labdo1 = Label( miframeDO1, bg="black",fg="blue", text=DO1var, font=("Arial","24")).grid(row=3, column =2)
    parameters = {'valor':DO1var2,'hora':hora,'estado':DO1var}
    request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarsalidadigital1.php', data=parameters)
  #botones de encendido y apagado con comando cada uno
  BOTONDO1ON = Button(miframeDO1,text="ON",fg="Red", bg="WHITE",font=("Arial","24"),command=DO1ON).grid(row = 2,
column = 1)
  BOTONDO1OFF = Button(miframeDO1,text="OFF",fg="blue",bg="white" ,font=("Arial","24"),command=DO1OFF).grid(row = 2,
column = 3)
#labdo1title = Label( miframeDO1, bg="white", text=textbut, font=("Arial","24")).grid(row=3, column =2)
  #botones para navegación en la interfaz
  homebutton = Button(miframeDO1,text="HOME",fg="Red", font=("Arial","24"),command=main).grid(row = 4, column = 2)
  backbutton = Button(miframeDO1,text="BACK",fg="Red", font=("Arial","24"),command=DIGITAL).grid(row = 5, column = 2)
#refreshbutton=Button(miframeDO1,text="Refresh",fg="Red", font=("Arial","24"),command=DO1).grid(row = 6, column = 2)

def DO2():                     #comando de salida digital 2
  limpiar()
  root.title("SALIDA DIGITAL 2")
  miframeDO2.pack()
  def DO2ON():
    GPIO.output(27,True)       #enciende salida digital 2
    DO2var2= 24
    DO2var="ON"                #activa etiqueta de encendido
    parameters = {'valor':DO2var2,'hora':hora,'estado':DO2var}
    #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
    request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarsalidadigital1.php', data=parameters)
  def DO2OFF():
    GPIO.output(27,False)      #apaga salida digital 3
    DO2var2= 0
    DO2var="OFF"               #activa etiqueta de apagado
    parameters = {'valor':DO2var2,'hora':hora,'estado':DO2var}
    request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarsalidadigital2.php', data=parameters)
  BOTONDO1ON = Button(miframeDO1,text="ON",fg="Red", bg="WHITE",font=("Arial","24"),command=DO2ON).grid(row = 2,
column = 1)
  BOTONDO1OFF = Button(miframeDO1,text="OFF",fg="blue",bg="white" ,font=("Arial","24"),command=DO2OFF).grid(row = 2,
column = 3)
  #botones de navegación en la interfaz
  homebutton = Button(miframeDO2,text="HOME",fg="Red", font=("Arial","24"),command=main).grid(row = 4, column = 2)
  backbutton = Button(miframeDO2,text="BACK",fg="Red", font=("Arial","24"),command=DIGITAL).grid(row = 5, column = 2)


def DO3():                     #comando de salida digital 3
  limpiar()
  root.title("SALIDA DIGITAL 3")
  miframeDO3.pack()
  def DO3ON():
    GPIO.output(22,True)       #enciende salida digital 3
    DO3var2= 24
    DO3var="ON"                #activa etiqueta de encendido
    parameters = {'valor':DO3var2,'hora':hora,'estado':DO3var}
    #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
    request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarsalidadigital3.php', data=parameters)
  def DO3OFF():
    GPIO.output(22,False)      #apaga salida digital 3
    DO3var2= 0
    DO3var="OFF"               #activa etiqueta de apagado
    parameters = {'valor':DO3var2,'hora':hora,'estado':DO3var}
    request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarsalidadigital3.php', data=parameters)
  #botones de encendido y apagado con comando cada uno
  BOTONDO1ON = Button(miframeDO3,text="ON",fg="Red", bg="WHITE",font=("Arial","24"),command=DO3ON).grid(row = 2,
column = 1)
  BOTONDO1OFF = Button(miframeDO3,text="OFF",fg="blue",bg="white" ,font=("Arial","24"),command=DO3OFF).grid(row = 2,
column = 3)
```

```
    #botones de navegación en la interfaz
    homebutton = Button(miframeDO3,text="HOME",fg="Red", font=("Arial","14"),command=main).grid(row = 4, column = 2)
    backbutton = Button(miframeDO3,text="BACK",fg="Red", font=("Arial","14"),command=DIGITAL).grid(row = 5, column = 2)

def DO4():                  #comando de salida digital 4
    limpiar()
    root.title("SALIDA DIGITAL 4")
    miframeDO4.pack()
    def DO4ON():
        GPIO.output(5,True)       #enciende salida digital 4
        DO4var2= 24
        DO4var="ON"               #activa etiqueta de encendido
        parameters = {'valor':DO4var2,'hora':hora,'estado':DO4var}
        #envía parámetros a página que se conecta a la base de datos e inserta a la tabla
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarsalidadigital4.php', data=parameters)
    def DO4OFF():
        GPIO.output(5,False)      #apaga salida digital 4
        DO4var2= 0
        DO4var="OFF"              #activa etiqueta de apagado
        parameters = {'valor':DO4var2,'hora':hora,'estado':DO4var}
        request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarsalidadigital4.php', data=parameters)
    #botones de encendido y apagado con comando cada uno
    BOTONDO1ON = Button(miframeDO3,text="ON",fg="Red", bg="WHITE",font=("Arial","24"),command=DO4ON).grid(row = 2,
column = 1)
    BOTONDO1OFF = Button(miframeDO3,text="OFF",fg="blue",bg="white" ,font=("Arial","24"),command=DO4OFF).grid(row = 2,
column = 3)
    #botones de navegación en la interfaz
    homebutton = Button(miframeDO4,text="HOME",fg="Red", font=("Arial","14"),command=main).grid(row = 4, column = 2)
    backbutton = Button(miframeDO4,text="BACK",fg="Red", font=("Arial","14"),command=DIGITAL).grid(row = 5, column = 2)

def DO5():
    limpiar()
    root.title("SALIDA DIGITAL 5")
    miframeDO5.pack()
    homebutton = Button(miframeDO5,text="HOME",fg="Red", font=("Arial","14"),command=main).grid(row = 2, column = 2)
    backbutton = Button(miframeDO5,text="BACK",fg="Red", font=("Arial","14"),command=DIGITAL).grid(row = 3, column = 2)

def DO6():
    limpiar()
    root.title("SALIDA DIGITAL 6")
    miframeDO6.pack()
    homebutton = Button(miframeDO6,text="HOME",fg="Red", font=("Arial","14"),command=main).grid(row = 2, column = 2)
    backbutton = Button(miframeDO6,text="BACK",fg="Red", font=("Arial","14"),command=DIGITAL).grid(row = 3, column = 2)

def DO7():
    limpiar()
    root.title("SALIDA DIGITAL 7")
    miframeDO7.pack()
    homebutton = Button(miframeDO7,text="HOME",fg="Red", font=("Arial","14"),command=main).grid(row = 2, column = 2)
    backbutton = Button(miframeDO7,text="BACK",fg="Red", font=("Arial","14"),command=DIGITAL).grid(row = 3, column = 2)

def DO8():
    limpiar()
    root.title("SALIDA DIGITAL 8")
    miframeDO8.pack()
    homebutton = Button(miframeDO8,text="HOME",fg="Red", font=("Arial","14"),command=main).grid(row = 2, column = 2)
    backbutton = Button(miframeDO8,text="BACK",fg="Red", font=("Arial","14"),command=DIGITAL).grid(row = 3, column = 2)

def INHART():
    limpiar()               #llama al comando para limpiar la ventana
    global banderainhart
    banderainhart=1
    vrefin=2.042
    root.title("ENTRADA HART")
    root["bg"]="WHITE"
    miframeINHART.pack()
```

```
    miframeINHART["bg"]="WHITE"
    GPIO.output(CS, True)
    GPIO.output(CLK, True)
    GPIO.output(CS, False)
    binData=0
    i1 = 14
    while (i1 >= 0):
            GPIO.output(CLK, False)
            bitDOUT = GPIO.input(DOUT)
            GPIO.output(CLK, True)
            bitDOUT = bitDOUT << i1
            binData |= bitDOUT
            i1 -= 1
    GPIO.output(CS, True)
    binData &= 0xFFF
    res = vrefin * binData/4096.0
    res2= res*10.2
    if res2 > 20.5:
       displayINHART='in>20mA'
       print ('Corriente mayor a 20mA')
    elif res2 < 3.9:
       displayINHART='in<4mA'
       print('corriente baja')
    else:
       displayINHART=str(res2)+'mA'
       print('Input Voltage = ' + str(res) + 'V')
       print('Input Current = ' + str(res2)+ 'mA')

    hora = datetime.now().time()
    parameters = {'valor':displayINHART,'hora':hora}  #establece parámetros para enviar
    #envía mediante un POST a la página donde guarda los datos en la tabla dentro de la base de datos
    request = requests.post('http://monitoreoremotoindustrial.com/tesis/guardarentradahart.php', data=parameters)
    labadc1disp =Label( miframeINHART, bg="white",text= displayINHART, font=("Arial","24")).grid(row=1, column =2)


    #botones para la navegación en la interfaz
    homebutton = Button(miframeINHART,text="HOME",fg="Red", font=("Arial","24"),command=main).grid(row = 3, column = 2)
    backbutton = Button(miframeINHART,text="BACK",fg="Red", font=("Arial","24"),command=HART).grid(row = 4, column = 2)
    refresh= Button(miframeINHART, text="REFRESH", fg="blue",font=("Arial","24"),command=INHART).grid(row = 5, column = 2)


def OUTHART():
    limpiar()
    root.title("SALIDA HART")
    root["bg"]="WHITE"
    miframeOUTHART.pack()
    miframeOUTHART["bg"]="WHITE"


def limpiar():                  #limpia las ventanas olvidando las anteriores
    miframe.pack_forget()
    miframe.grid_forget()
    miframeINHART.pack_forget()
    miframeINHART.grid_forget()
    miframeOUTHART.pack_forget()
    miframeOUTHART.grid_forget()
    miframeADC1.pack_forget()
    miframeADC1.grid_forget()
    miframeADC2.pack_forget()
    miframeADC2.grid_forget()
    miframeADC3.pack_forget()
    miframeADC3.grid_forget()
    miframeADC4.pack_forget()
    miframeADC4.grid_forget()
    miframeDAC.pack_forget()
    miframeDAC.grid_forget()
```

```
    miframeDAC2.pack_forget()
    miframeDAC2.grid_forget()
    miframeDAC3.pack_forget()
    miframeDAC3.grid_forget()
    miframeDAC4.pack_forget()
    miframeDAC4.grid_forget()
    miframeDIGITALES.pack_forget()
    miframeDIGITALES.grid_forget()
    miframeANALOG.pack_forget()
    miframeANALOG.grid_forget()
    miframeHART.pack_forget()
    miframeHART.grid_forget()
    miframeDO1.pack_forget()
    miframeDO2.pack_forget()
    miframeDO3.pack_forget()
    miframeDO4.pack_forget()
    miframeDO5.pack_forget()
    miframeDO6.pack_forget()
    miframeDO7.pack_forget()
    miframeDO8.pack_forget()
    miframeDI1.pack_forget()
    miframeDI2.pack_forget()
    miframeDI3.pack_forget()
    miframeDI4.pack_forget()
    miframeDI5.pack_forget()
    miframeDI6.pack_forget()
    miframeDI7.pack_forget()
    miframeDI8.pack_forget()
    miframeDO1.grid_forget()
    miframeDO2.grid_forget()
    miframeDO3.grid_forget()
    miframeDO4.grid_forget()
    miframeDO5.grid_forget()
    miframeDO6.grid_forget()
    miframeDO7.grid_forget()
    miframeDO8.grid_forget()
    miframeDI1.grid_forget()
    miframeDI2.grid_forget()
    miframeDI3.grid_forget()
    miframeDI4.grid_forget()
    miframeDI5.grid_forget()
    miframeDI6.grid_forget()
    miframeDI7.grid_forget()
    miframeDI8.grid_forget()

def DIGITAL():            #comando para salidas y entradas digitales
    limpiar()
    root.title("DIGITALES") #título en la ventana
    root["bg"]="cyan"
    miframeDIGITALES.pack()
    miframeDIGITALES["bg"]="cyan"   #establece el color dentro de la ventana
    #título de las entradas y salidas digitales
    lab01 = Label( miframeDIGITALES, bg="cyan", text="ENTRADAS", font=("Arial","20")).grid(row=0, column =2)
    lab02 = Label( miframeDIGITALES, bg="cyan", text="Y SALIDAS", font=("Arial","20")).grid(row=0, column =3)
    lab03 = Label( miframeDIGITALES, bg="cyan", text="DIGITALES", font=("Arial","20")).grid(row=0, column =4)
    #botones para las salidas digitales con comandos de ventana individual para cada salida
    DO1button = Button(miframeDIGITALES,text="DIGITAL OUT 1",fg="Black", bg="Blue",font=("Arial","18"),command=DO1).grid(row
= 2, column = 2)
    DO2button = Button(miframeDIGITALES,text="DIGITAL OUT 2",fg="Black", bg="Blue",font=("Arial","18"),command=DO2).grid(row
= 3, column = 2)
    DO3button = Button(miframeDIGITALES,text="DIGITAL OUT 3",fg="Black", bg="Blue",font=("Arial","18"),command=DO2).grid(row
= 4, column = 2)
    DO4button = Button(miframeDIGITALES,text="DIGITAL OUT 4",fg="Black", bg="Blue",font=("Arial","18"),command=DO4).grid(row
= 5, column = 2)
    DO5button = Button(miframeDIGITALES,text="DIGITAL OUT 5",fg="Black", bg="Blue",font=("Arial","18"),command=DO5).grid(row
= 6, column = 2)
```

```
   DO6button = Button(miframeDIGITALES,text="DIGITAL OUT 6",fg="Black", bg="Blue",font=("Arial","18"),command=DO6).grid(row
= 7, column = 2)
   DO7button = Button(miframeDIGITALES,text="DIGITAL OUT 7",fg="Black", bg="Blue",font=("Arial","18"),command=DO7).grid(row
= 7, column = 2)
   DO8button = Button(miframeDIGITALES,text="DIGITAL OUT 8",fg="Black", bg="Blue",font=("Arial","18"),command=DO8).grid(row
= 7, column = 2)
   #botones para las entradas digitales con comandos de ventana individual para cada entrada
   DI1button = Button(miframeDIGITALES,text="DIGITAL IN 1",fg="Blue", bg="Black",font=("Arial","18"),command=DI1).grid(row = 2,
column = 4)
   DI2button = Button(miframeDIGITALES,text="DIGITAL IN 2",fg="Blue", bg="Black",font=("Arial","18"),command=DI2).grid(row = 3,
column = 4)
   DI3button = Button(miframeDIGITALES,text="DIGITAL IN 3",fg="Blue", bg="Black",font=("Arial","18"),command=DI2).grid(row = 4,
column = 4)
   DI4button = Button(miframeDIGITALES,text="DIGITAL IN 4",fg="Blue", bg="Black",font=("Arial","18"),command=DI4).grid(row = 5,
column = 4)
   DI5button = Button(miframeDIGITALES,text="DIGITAL IN 5",fg="Blue", bg="Black",font=("Arial","18"),command=DI5).grid(row = 6,
column = 4)
   DI6button = Button(miframeDIGITALES,text="DIGITAL IN 6",fg="Blue", bg="Black",font=("Arial","18"),command=DI6).grid(row = 7,
column = 4)
   DI7button = Button(miframeDIGITALES,text="DIGITAL IN 7",fg="Blue", bg="Black",font=("Arial","18"),command=DI7).grid(row = 7,
column = 4)
   DI8button = Button(miframeDIGITALES,text="DIGITAL IN 8",fg="Blue", bg="Black",font=("Arial","18"),command=DI8).grid(row = 7,
column = 4)
   #boton para regresar a ventana principal
   homebutton = Button(miframeDIGITALES,text="HOME",fg="brown", bg="yellow", font=("Arial","20"),command=main).grid(row =
10, column = 3)


def ANALOG():                #comando de señales analogicas
   limpiar()               #limpia ventana
   root.title("ANALOGICAS")   #coloca título a ventana
   root["bg"]="yellow"
   miframeANALOG.pack()
   miframeANALOG["bg"]="yellow"    #establece el color a ventana
   #coloca el título de encabezado en etiquetas
   lab01 = Label( miframeANALOG, bg="yellow",text="ENTRADAS", font=("Arial","25")).grid(row=0, column =2)
   lab02 = Label( miframeANALOG, bg="yellow",text="Y SALIDAS", font=("Arial","25")).grid(row=0, column =3)
   lab03 = Label( miframeANALOG, bg="yellow",text="ANALOGICAS", font=("Arial","25")).grid(row=0, column =4)
   #botones para entradas analogicas, cada uno con comando para ventana individual
   adc1button = Button(miframeANALOG,text="ADC 1",bg="Red",fg="Blue", font=("Arial","35"),command=ADC1).grid(row = 2,
column = 2)
   adc2button = Button(miframeANALOG,text="ADC 2",bg="Red",fg="Blue", font=("Arial","35"),command=ADC2).grid(row = 3,
column = 2)
   adc3button = Button(miframeANALOG,text="ADC 3",bg="Red",fg="Blue", font=("Arial","35"),command=ADC3).grid(row = 4,
column = 2)
   adc4button = Button(miframeANALOG,text="ADC 4",bg="Red",fg="Blue", font=("Arial","35"),command=ADC4).grid(row = 5,
column = 2)
   #botones para entradas analogicas, cada uno con comando para ventana individual
   dacbutton = Button(miframeANALOG,text="DAC1",bg="Blue",fg="Red", font=("Arial","35"),command=DAC).grid(row = 2, column
= 4)
   dac2button = Button(miframeANALOG,text="DAC2",bg="Blue",fg="Red", font=("Arial","35"),command=DAC2).grid(row = 3,
column = 4)
   dac3button = Button(miframeANALOG,text="DAC3",bg="Blue",fg="Red", font=("Arial","35"),command=DAC3).grid(row = 4,
column = 4)
   dac4button = Button(miframeANALOG,text="DAC4",bg="Blue",fg="Red", font=("Arial","35"),command=DAC4).grid(row = 5,
column = 4)
   #boton de navegación para regresar a menú principal
   homebutton = Button(miframeANALOG,bg="black",text="HOME",fg="yellow", font=("Arial","20"),command=main).grid(row = 6,
column = 3)

def HART():                  #comando comunicación HART
   limpiar()
   root.title(" HART ")       #coloca título de ventana
   root["bg"]="orange"
   miframeHART.pack()
   miframeHART["bg"]="ORANGE"  #establece color a la ventana
```

```
   #coloca encabezado a la ventana
   lab01 = Label( miframeHART, bg="orange", text="ENTRADAS", font=("Arial","20")).grid(row=0, column =2)
   lab02 = Label( miframeHART,  bg="orange", text="Y SALIDAS", font=("Arial","20")).grid(row=0, column =3)
   lab03 = Label( miframeHART,  bg="orange", text=" HART  ", font=("Arial","20")).grid(row=0, column =4)
   #coloca boton de entrada y de salida de comunicación HART
   inbutton = Button(miframeHART,text="ENTRADA",bg="black",fg="orange", font=("Arial","20"),command=INHART).grid(row = 3,
column = 2)
   outbutton = Button(miframeHART,text="SALIDA",bg="black",fg="orange", font=("Arial","20"),command=OUTHART).grid(row = 3,
column = 4)
   #coloca boton de regreso a menú principal
   homebutton = Button(miframeHART,text="HOME",fg="yellow",bg="black", font=("Arial","20"),command=main).grid(row = 5,
column = 3)

def main():              #ventana principal
   limpiar()            #limpia la ventana
   root.title("PRINCIPAL") #coloca título en la ventana
   root["bg"]="black"     #establece color en el fondo
   miframe.pack()
   miframe["bg"]="black"   #establece color en la ventana
   #coloca el título principal en una etiqueta
   lab01 = Label( miframe, fg="white",bg="black", text="MONITOREO REMOTO INDUSTRIAL", font=("Arial","25")).grid(row=0, column
=3)
   #coloca botones de los 3 módulos principales de señales y el de temperatura
   digitbutton=Button(miframe,text= "DIGITALES ",fg="Blue", bg="yellow",font=("Arial","40"),command=DIGITAL).grid(row = 2,
column = 3)
   analogbutton=Button(miframe,text="ANALÓGICAS",fg="yellow", bg="blue", font=("Arial","40"),command=ANALOG).grid(row = 5,
column = 3)
   hartbutton=Button(miframe,text="  HART  ",fg="blue", bg="yellow", font=("Arial","40"),command=HART).grid(row = 8, column =
3)
   offbutton=Button(miframe,text="  OFF  ",fg="yellow", bg="blue", font=("Arial","40"),command=OFF).grid(row = 9, column = 4)


t1 = threading.Thread(target = MANDARDB)
t1.shutdown = False
t1.start()


def OFF():
   print "off"
   global finished
   t1.shutdown = True
   t1.join()
   root.destroy()

#ESPACIO EN BLANCO
#lab0 = Label(miframe , text="       ", font=("Arial","11")).grid(row=5, column =1)
#ALARMAS
   #lab1 = Label( miframe, text="  Alarma1  ", relief=RAISED, bg="black", fg="white", font=("Arial","15")).grid(row=12, column =1)
   #lab2 = Label( miframe, text="   Alarma2   ", relief=RAISED, bg="black", fg="white", font=("Arial","15")).grid(row=12, column =2)
   #lab3 = Label( miframe, text="   Alarma3   ", relief=RAISED, bg="black", fg="white", font=("Arial","15")).grid(row=12, column =3)
   #lab4 = Label( miframe, text="   Alarma4   ", relief=RAISED, bg="black", fg="white", font=("Arial","15")).grid(row=12, column =4)
   #lab5 = Label( miframe, text="   Alarma5   ", relief=RAISED, bg="black", fg="white", font=("Arial","15")).grid(row=12, column =5)


main()
root.mainloop() #mantiene en un ciclo el principal
```