TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ

Empresa Productiva Subsidiaria Transmisión
Gerencia Regional de Transmisión Sureste
Zona de Transmisión Tuxtla

REPORTE DE RESIDENCIA PROFESIONAL

INGENIERÍA ELECTRÓNICA

MEDICIÓN Y ANÁLISIS ESTADÍSTICOS DE VARIABLES EN SISTEMAS DE FUERZA DE 48 VCD Y BANCO DE BATERÍAS LIBRES DE MANTENIMIENTO

PRESENTA
Alejandra Muñiz Alvarado

ASESOR
M en C. Joaquín Eduardo Domínguez Zenteno

DICIEMBRE 2017

CFE
Comisión Federal de Electricidad

Índice

ÍNDICE DE FIGURAS

Introducción

La Empresa Productiva Subsidiaria Transmisión, Zona de Transmisión Tuxtla, utiliza sistemas de fuerza de 48VCD para alimentar equipos de comunicaciones de clientes internos y externos, una de las principales funciones de dichos sistemas es el monitoreo de alarmas de alto y bajo voltaje de los equipos. Se encuentran operando con un inversor SET-3000, el cual es una fuente de energía de CD-CA, utiliza baterías como fuente de CD de entrada y entrega una onda senoidal a la salida, de esta forma el voltaje y frecuencia de salida son muy estables y pueden trabajar continuamente y prevenir problemas de caída de energía, voltaje inestable, ruido y picos de voltaje.

Los sistemas de fuerza cuentan con su propio software en el cual es posible observar el comportamiento del sistema completo, sin embargo no proporciona datos específicos acerca del banco de baterías. El siguiente proyecto consiste en la implementación de un sistema que permite monitorear el banco de baterías. El monitoreo consiste en medir la corriente, temperatura y voltaje de cada una de las baterías, esto con la finalidad de determinar el porqué de los daños que se presentan frecuentemente en las baterías. Para ello, los datos adquiridos durante el tiempo de medición definido, serán enviados a un servidor denominado como ZABBIX, el cual es utilizado por EPS como una base de datos.

La implementación de este sistema de monitoreo está pensado para que los ingenieros se encuentren informados del estado del banco de baterías durante el transcurso del día.

Capítulo I EPS Transmisión

1.1 ¿Qué es EPS Transmisión?

Es una empresa productiva del Estado de propiedad exclusiva del Gobierno Federal, con personalidad jurídica y patrimonio propios y gozará de autonomía técnica, operativa y de gestión, conforme a lo dispuesto en la nueva Ley de la Comisión Federal de Electricidad, y que tiene por objeto prestar, en términos de la legislación aplicable, el servicio público de transmisión de energía eléctrica.

El compromiso de la empresa es ofrecer servicios de excelencia, garantizando altos índices de calidad en todos sus procesos, al nivel de las mejores empresas eléctricas en el mundo.

1.2 Política de calidad de la empresa

Prestar el servicio público de transmisión de energía eléctrica y otros servicios relacionados en el área de las Telecomunicaciones, que generen valor económico y rentabilidad, procurando el mejoramiento de la productividad con sustentabilidad para el desarrollo nacional. Considerando los aspectos ambientales y de seguridad, y a una mejora continua de la eficacia del sistema integral de Gestión, con el compromiso de:

- Formar y desarrollar el capital humano
- Gestión eficiente de los riesgos
- Prevenir la contaminación y aprovechar de manera responsable los recursos naturales
- Cumplir con la legislación, reglamentación y otros requisitos

1.3 Objetivo EPS Transmisión

Asegurar el acceso a la Red Nacional de Transmisión de la CFE, mediante su operación, mantenimiento, expansión y modernización, además de suministrar otros productos y servicios asociados para crear valor económico al Estado.

1.4 Visión

Ser una empresa transportista de energía eléctrica respetuosa del medio ambiente que tiene un desempeño acorde con lo establecido por el ente regulador, que cuenta con autonomía de gestión y separación legal del resto de los procesos sustantivos de la CFE, que cumple con la rentabilidad definida por el Estado y genera ingresos adicionales prestando servicios que permiten obtener un máximo beneficio de su talento e infraestructura física.

1.5 Misión

Asegurar el acceso a la Red Nacional de Transmisión (RNT) de la CFE en condiciones de disponibilidad, eficiencia, calidad, confiabilidad, y continuidad, mediante su operación, mantenimiento, expansión y modernización además de suministrar otros productos y servicios asociados para crear valor económico al Estado Mexicano, de manera sustentable.

1.6 Valores Institucionales

- Integridad
- Productividad
- Responsabilidad

1.7 Retos

- Implementar la Gestión de Activos mediante el uso de la Red Eléctrica Inteligente, con el propósito de optimizar la vida útil de los activos de la Red Nacional de Transmisión.
- Cero fallas en Red Nacional de Transmisión
- Desarrollo del talento de los trabajadores de Transmisión, para el cumplimiento de las nuevas funciones.

1.8 Área relacionada con el proyecto

La Empresa Productiva Subsidiaria EPS Transmisión, es columna vertebral del Sistema Eléctrico Nacional, ya que apoya a cumplir el compromiso de mantener, modernizar y rehabilitar las instalaciones como son las Subestaciones Eléctricas de Potencia que operan en los voltajes de 400, 230, 115 kV o voltajes iguales o superiores a 69 kV, ubicadas a lo largo y ancho de los casi 2 millones de kilómetros cuadrados que ocupa el territorio nacional; para lo cual se encuentra organizado en nueve Gerencias Regionales de Transmisión.

La Gerencia Regional de Transmisión Sureste (GRTSE), forma parte de la EPS Transmisión y tiene injerencia en los estados de Chiapas, Tabasco, Oaxaca y parte de Veracruz a través de sus cinco Zonas de Transmisión y una Zona de Operación (ZOTSE) que la conforman; asegurando en todo momento el servicio a sus clientes regionales y nacionales aproximadamente de 20 subestaciones eléctricas de potencia en voltajes de 400, 230 y 115 kV, mismas que están interconectadas y enlazadas al Sistema Eléctrico Nacional; y de acuerdo a la Ley de la Industria Eléctrica, publicado en el Diario Oficial de la Federación, tiene el activo y control físico de las Subestaciones con tensión mayor o igual a 69 kV.

La Gerencia Regional de Transmisión Sureste (GRTSE) forma parte integral de la Red Troncal de Subestaciones de 400 kV, 230 kV y 115 kV del Sistema Eléctrico Nacional (SEN), en donde las subestaciones deben estar supervisadas y tele controladas desde los centros de control del Centro Nacional de Control de

Energía (CENACE), con el fin de coordinar la operación del SEN. El proceso de transmisión requiere asegurar la confiabilidad física y operativa de sus instalaciones, orientada a mantener una alta seguridad de la red eléctrica.

La Zona de Transmisión Tuxtla, perteneciente a la Gerencia Regional de Transmisión Sureste, se encuentra en la ciudad de Tuxtla Gutiérrez, Chiapas. Atiende actualmente las Subestaciones Eléctricas Manuel Moreno Torres "Chicoasén", Angostura y El Sabino, las cuales se encuentran interconectadas al Sistema Eléctrico Nacional mediante enlaces de 400 kV y a la red de 115 kV, con la finalidad de suministrar la demanda de energía de las principales ciudades del Estado de Chiapas. Cuenta con 3 Secciones Sindicales del SUTERM: Sección 155 Chicoasén, Sección 130 Angostura y Sección 47 Tuxtla. Las oficinas Sedes están ubicadas en Carretera Panamericana km. 1077 No. 5675 Interior 300 m., Col. Plan de Ayala en la Ciudad de Tuxtla Gutiérrez.

Está conformada por las especialidades o Departamentos de Comunicaciones, Protección y Medición, Control e Informática, Administración, así como de Subestaciones y Líneas. Siendo cada una de estas, importantes para el logro de los objetivos y metas programadas por parte de la EPS Transmisión Zona Tuxtla.

La Zona de Transmisión Tuxtla, además del Sistema Eléctrico de Potencia, tuvo la necesidad y el compromiso de proporcionar servicios de Telecomunicaciones a clientes internos y externos, dentro de una Red de Fibra Óptica troncal y de última milla en la ciudad de Tuxtla Gutiérrez, Cintalapa, Copainalá y Chiapa de Corzo; misma que se encuentra instalada en la Red Eléctrica de Distribución en postería de 13.8 kV; con la finalidad de proporcionar servicios a los diferentes usuarios como son: SCT, CJF, BANOBRAS, SAGARPA, SCJN, PEMEX, EPS Distribución y EPS Generación. El objetivo principal es mantener en operación los servicios que se proporcionan a dichos clientes mediante los enlaces de fibra óptica, para cumplir con los requisitos de los contratos, satisfacción del cliente, disponibilidad y la confiabilidad de los servicios.

1.9 Estructura de EPS Transmisión Tuxtla

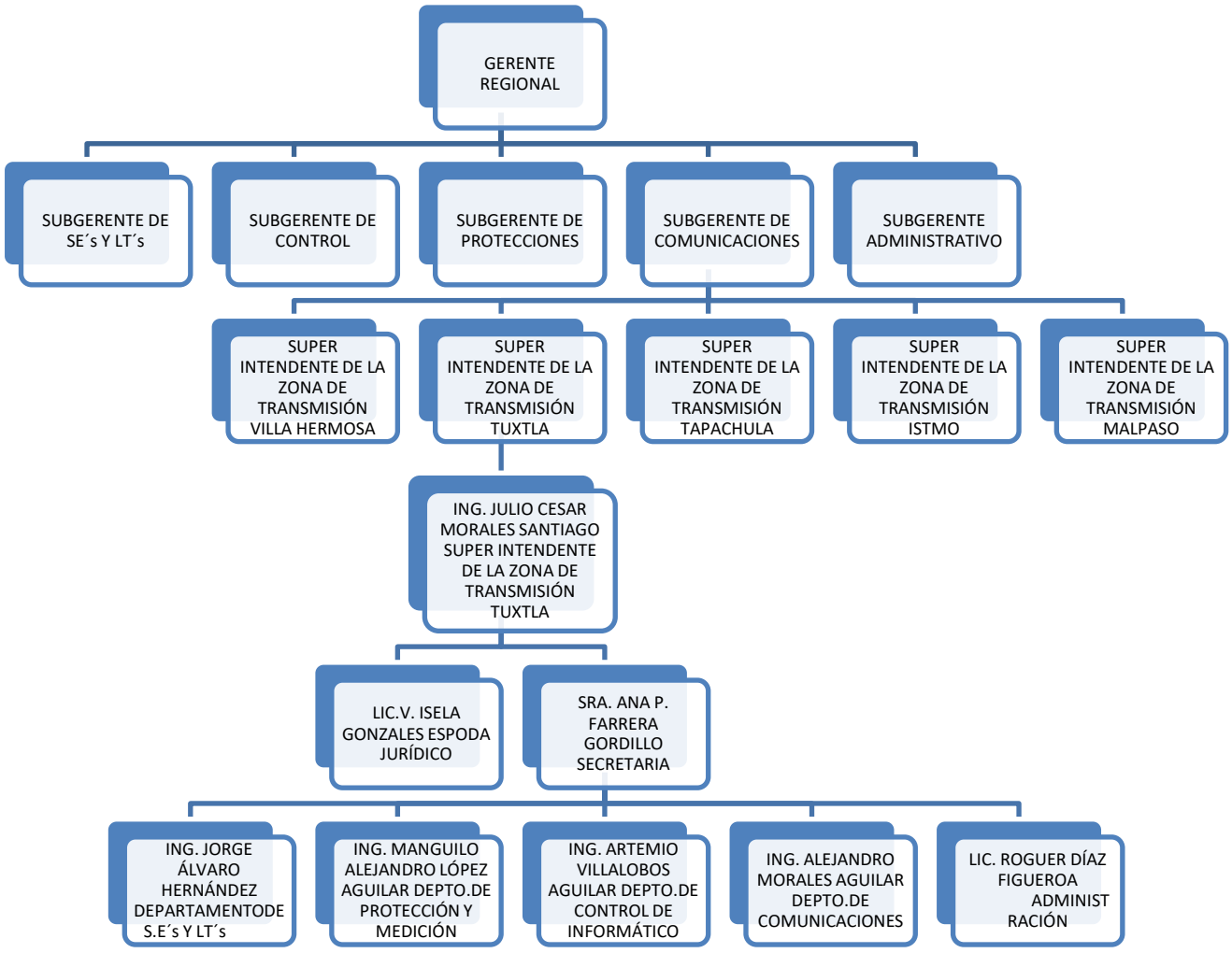

Ilustración 1. Organigrama de EPS Transmisión Tuxtla.

Capitulo II Generalidades

2.1 Antecedentes

Implementación del sistema de monitoreo de temperatura del cuarto de servidores 3A. [1]

El proyecto consistió en la elaboración de un sistema de monitoreo de temperatura en la empresa Sykes Latin America, S.A. que permitiera al personal de mantenimiento de sistemas de información, contar con un dispositivo que alertara por medio de correo electrónico y mensajes tipo beeper si la temperatura, dentro de un cuarto de servidores sobrepasaba uno o varios valores predeterminados.

Este sistema micro-controlado emplea un PIC16F877 de la empresa Microchip, obtiene la temperatura de los sensores digitales por medio de polling. A su vez el dispositivo despliega la temperatura localmente en una pantalla de cristal líquido y la envía a través de un puerto RS-232 a una computadora personal.

El sistema permite que los usuarios programen en una computadora personal que corra en la plataforma Windows, las diferentes temperaturas a las cuales las alarmas deben de ser activadas. Dicha aplicación fue desarrollada en Visual Basic.

2.2 Planteamiento del Problema

Actualmente para proporcionar servicios a clientes internos y externos se ocupan convertidores de medios, nodos SDH, equipos de microondas y nodos MPLS/TP, los cuales son energizados por voltajes de corriente directa de 48v, mismos que proporciona el sistema de fuerza. Estos equipos no siempre se encuentran ubicados dentro de las instalaciones de la Zona Transmisión Tuxtla, también se encuentran en dependencias externas, lo que dificulta realizar una inspección de los equipos de forma periódica. Derivado a lo anterior, es importante tener el monitoreo de los equipos de forma remota, con la finalidad de detectar fallas o anomalías que pudieran afectar a los equipos y por lo consiguiente la interrupción de los servicios del cliente.

Estos sistemas cuentan con un controlador el cual indica el estado de funcionamiento del sistema en general, pero para poder consultar dicha información se debe ir al sitio en donde se encuentra ubicado el equipo o se debe consultar utilizando el software PowCom, sin embargo la información que se

proporciona es generalizada y no proporciona información acerca del banco de baterías. Los problemas de funcionamiento en estos sistemas generalmente se presentan en las baterías debido a que cuando una de estas baterías comienza a trabajar de forma inusual afecta el funcionamiento de las demás, incluso se presentan situaciones en las que las baterías explotan debido a un sobrecalentamiento, ocasionando asi que el sistema de fuerza presente problemas y por lo tanto deje de brindar energía a los equipos correspondientes.

2.3 Objetivos del proyecto

2.3.1 Objetivo general:

Diseñar e implementar un sistema que permita monitorear de forma remota el funcionamiento de sistemas de fuerza y banco de baterías, de manera que se puedan prevenir fallas en dichos sistemas.

2.3.2 Objetivos específicos:

Implementar un sistema que permita la medición de temperatura, corriente y voltaje de un banco de baterías de 48 vcd e integrarlo al servidor ZABBIX.

Implementar el envió de notificaciones en condiciones anormales del banco de baterías que conforma el sistema de Fuerza.

2.4 Justificación

EPS Zona de Transmisión Tuxtla cuenta con equipos especializados en el área de telecomunicaciones, radiocomunicación y control, como son los sistemas de fuerza, los cuales están diseñados para suministrar energía de forma constante y en cualquier circunstancia. A su vez, dichos sistemas de fuerza se encuentran alimentados por baterías selladas libres de mantenimiento, las cuales en el caso de una interrupción o falla en el sistema de alimentación, son capaces de respaldar cargas por un tiempo limitado. Estos equipos cuentan con un

controlador, cuyo objetivo principal es monitorear y gestionar los Sistemas de Fuerza. Sin embargo el monitoreo no toma en cuenta el estado de funcionamiento de las baterías y de esta manera se vuelve indispensable para el sistema de fuerza, que el banco de baterías funcione sin interrupciones o anomalías. El monitoreo del banco de baterías se convierte en una herramienta para dar seguimiento estadístico de su comportamiento, y así prevenir fallos catastróficos. Permitiendo que el servicio que proporciona EPS a sus clientes sea seguro y eficiente.

2.5 Alcances y limitaciones del proyecto

2.5.1 Alcances

Llevar a cabo el monitoreo del estado de un banco de baterías mediante el censado de variables como voltaje, temperatura, corriente, ultima presencia detectada o puerta abierta.

Avisar en tiempo real de anomalías que se presenten a través de alertas enviadas por correo electrónico mediante el servidor ZABBIX y así realizar las acciones correctivas necesarias.

2.5.2 Limitaciones

Un sistema de fuerza tiene la capacidad de trabajar con cuatro camas de baterías en el banco. El sistema de monitoreo que se plantea está pensado para el monitoreo de un banco de baterías con un máximo de dos camas por lo que en caso de contar con una o dos camas más, se tendría que utilizar otro arduino debido a que las entradas analógicas de uno solo serían insuficientes.

2.6 Metodología para el desarrollo del proyecto

2.6.1 Visita al sitio

EPS zona de transmisión Tuxtla cuenta con un sistema de fuerza ubicado en el laboratorio de Comunicaciones por lo que no fue necesario trasladarse a otro lugar si no que se estuvo trabajando dentro de la zona.

2.6.2 Selección de equipos y herramientas a utilizar

De acuerdo a las necesidades del proyecto se determinó que tarjeta para adquisición de datos debía utilizarse, los tipos de sensores que debían utilizarse de acuerdo a las especificaciones del sistema de fuerza y banco de baterías.

Debido a que los sensores que se eligieron son de salidas analógicas y la programación de los mismos no requería de mayor elaboración, se optó por el uso de un arduino mega como el controlador de todo el sistema.

2.6.3 Programación y puesta en servicio de quipos

Una vez que se determinaron los sensores a utilizar, se comenzó a construir el programa que permitiría medir temperatura, voltaje y corriente.

2.6.4 Instalación de sensores

Durante la instalación de sensores, los sensores de temperatura debían colocarse sobre las baterías de manera que la medición fuera lo más acertada posible, los sensores de voltaje se conectaron alrededor de los tornillos que conectan a las baterías en serie y por los cuales circula el voltaje generado por cada batería.

2.6.5 Comunicación con plataforma ZABBIX

Una vez que se elaboró el programa que permitiría obtener las mediciones, se continuó con el programa para establecer la comunicación con la plataforma ZABBIX y así enviar los datos adquiridos.

Capitulo III Fundamento Teórico

3.1 Sistema Modular de Fuerza

Este es un sistema de fuerza modular de CA/CD/CA ininterrumpible (UPS) de potencia de 3.0KVA, el cual ofrece una fuente de energía constante hacia cargas críticas que no toleran variaciones de voltaje, frecuencia o interruptores en la alimentación. El voltaje de entrada del sistema es de 110/220VCA,1-2F,60Hz y ofrece una salida de 120VCA,1F,60Hz. [2]

El Sistema está asociado a un banco de baterías selladas libres de mantenimiento (FRONT TERMINAL) de 48VCD y capacidad total de 360AH., capaz de soportar una carga de 73.0Amps. durante 3.00 hrs. El sistema cuenta con una etapa de corte por bajo voltaje para proteger al banco de baterías de una descarga profunda.

La etapa de fuerza de C.D. cuenta con un controlador (ACC) diseñado para realizar el control y monitoreo del sistema en un amplio rango de aplicaciones de acuerdo a las necesidades del usuario, cuenta con una interface USB o serie (RS232) para el control remoto mediante una PC y el Software Pow Com.

3.1.1 Módulo de Entrada, Salida y Alarmas.

Este módulo contiene el interruptor general de entrada de C.A. y también está localizado el interruptor de entrada de C.A. (Auxiliar). Cuenta con las terminales de salida de C.A. del sistema y terminales de entrada y salida de alarmas, también se encuentra alojada la barra de tierra y el supresor de transitorios (s.t.). [2]

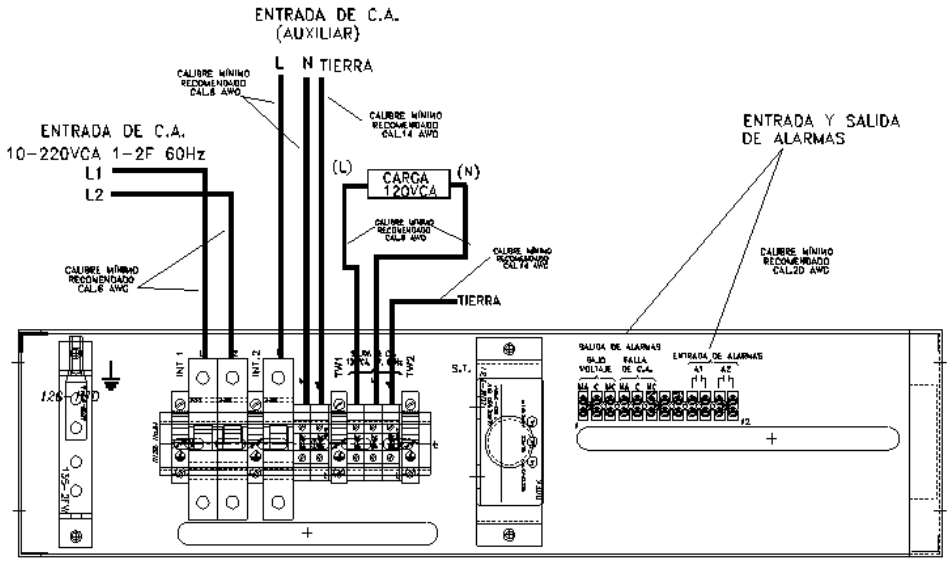

Ilustración 2. Módulo de entradas, salidas y alarmas.

3.1.2 Módulo de Distribución.

En éste módulo se encuentran alojados los interruptores de Distribución de C.D. el interruptor del inversor y el de Batería así como las terminales de conexión para la carga y batería. [2]

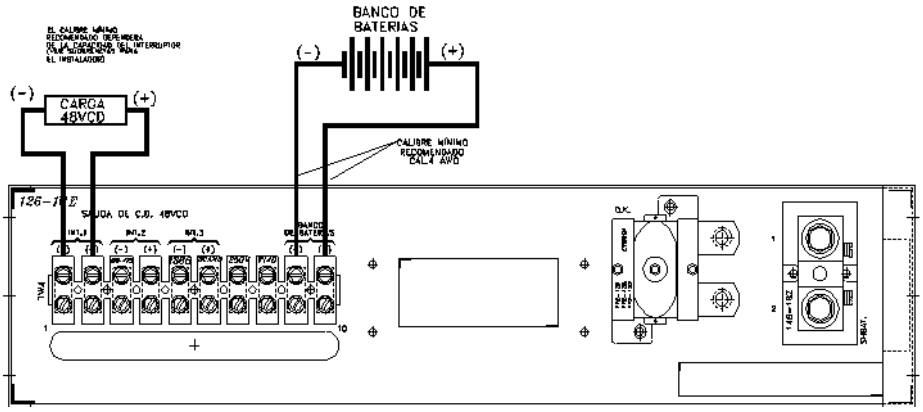

Ilustración 3.Modulo de Distribución C.D. y Baterías.

3.1.3 Repisa de Rectificadores

Esta repisa puede alojar cuatro rectificadores (modelo Guardián FMP25-48 marca Power-One-MEI de 50A-48VCD de alta frecuencia. Cuando alguno de estos

rectificadores está dañado o se encuentra fuera de su repisa, el sistema de fuerza emitirá una señal de alarma, misma que será desplegada en la pantalla del controlador, y activará los contactos de alarma correspondiente. [2]

La repisa de rectificadores sirve para interconectar los módulos de rectificadores tipo Plug-in modelo FMP25-48 que trabajen en paralelo. Cada repisa de rectificadores aloja hasta 4 rectificadores numerados de izquierda a derecha vistos desde la parte frontal.

Cada repisa de rectificadores está equipada de acuerdo a los requerimientos de potencia y posteriormente ser completada con módulos FMP adicionales.

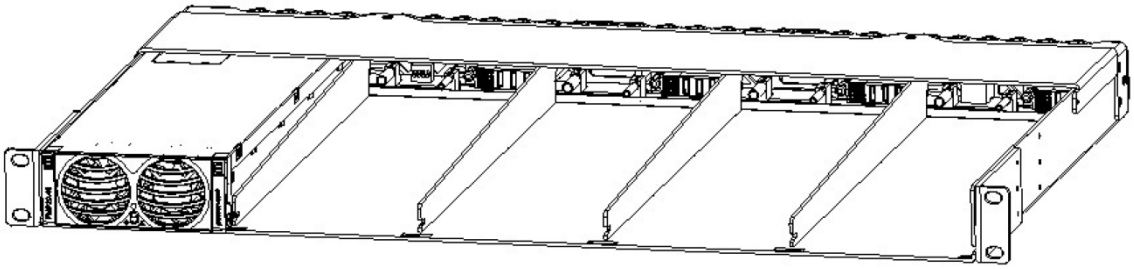

Ilustración 4. Repisa de Rectificadores.

3.1.4 Controlador ACC

Esté sistema contiene un controlador (ACC C/Display) el cual supervisa y controla el funcionamiento del sistema en general y genera señalizaciones en forma “local” por medio de un display localizado al frente del controlador o “remota” a través de contactos secos tipo “C” a través de una tarjeta de alarmas. Este control digital monitorea el total de condiciones del sistema tales como Voltaje de C.D., Corriente y Temperatura de los rectificadores, capacidad del sistema, parámetros de la batería y estado de los interruptores de distribución y batería. Este controlador cuenta con un puerto ETHERNET y protocolos SNMP V2, TCP/IP con lo cual se puede supervisar vía remota o mediante un Software de Administración recibir las alarmas y parámetros de este. [2]

3.1.5 Igualación Automática

La planta cuenta con un sistema de igualación automática (43.0 A 57.0VCD) el cual controla el voltaje de flotación ó igualación de acuerdo a las condiciones de la batería, esta función es realizada por el controlador “ACC”. [2]

3.1.6 Inversor CD/CA

El sistema cuenta con un inversor con tecnología de alta frecuencia marca ENATEL modelo SET3000 de 3KVA, 48VCD/120VCA. El cual ofrece una salida de onda senoidal de 120VCA,1F,60Hz., para la alimentación de la carga. [3]

Es un inversor completo, modular, de funcionamiento independiente, ideal para suministrar corriente a aparatos sensibles que necesiten de confiabilidad en el suministro de C.A. La máxima confiabilidad del Inversor se basa en que se entrega plenamente probado y preparado para trabajar de inmediato y su colocación y montaje es muy sencilla y práctica.

El inversor se alimenta con un voltaje de corriente continua de 48V.C.D. (42 a 56VCD) y entrega una tensión alterna senoidal de 120 V.C.A. ± 1.5% 1 fase, 60Hz; con una potencia de salida máxima de 3KVA. Cuenta con terminales de entrada y salida debidamente identificadas que facilita su conexión.

3.1.7 Barra Multicontactos

El sistema de Fuerza Modular de CA/CD/CA, ofrece también una barra multicontacto que se encuentra instalada en el gabinete y que contiene doce contactos polonizados (NEMA5-15P) con fusible de protección de 30Amp's.

3.1.8 Banco de Baterías

El sistema cuenta con un banco de baterías selladas libres de mantenimiento (48VCD, 360AH) capaz de respaldar una carga de 73.0Amps durante 3.00 hrs. [4]

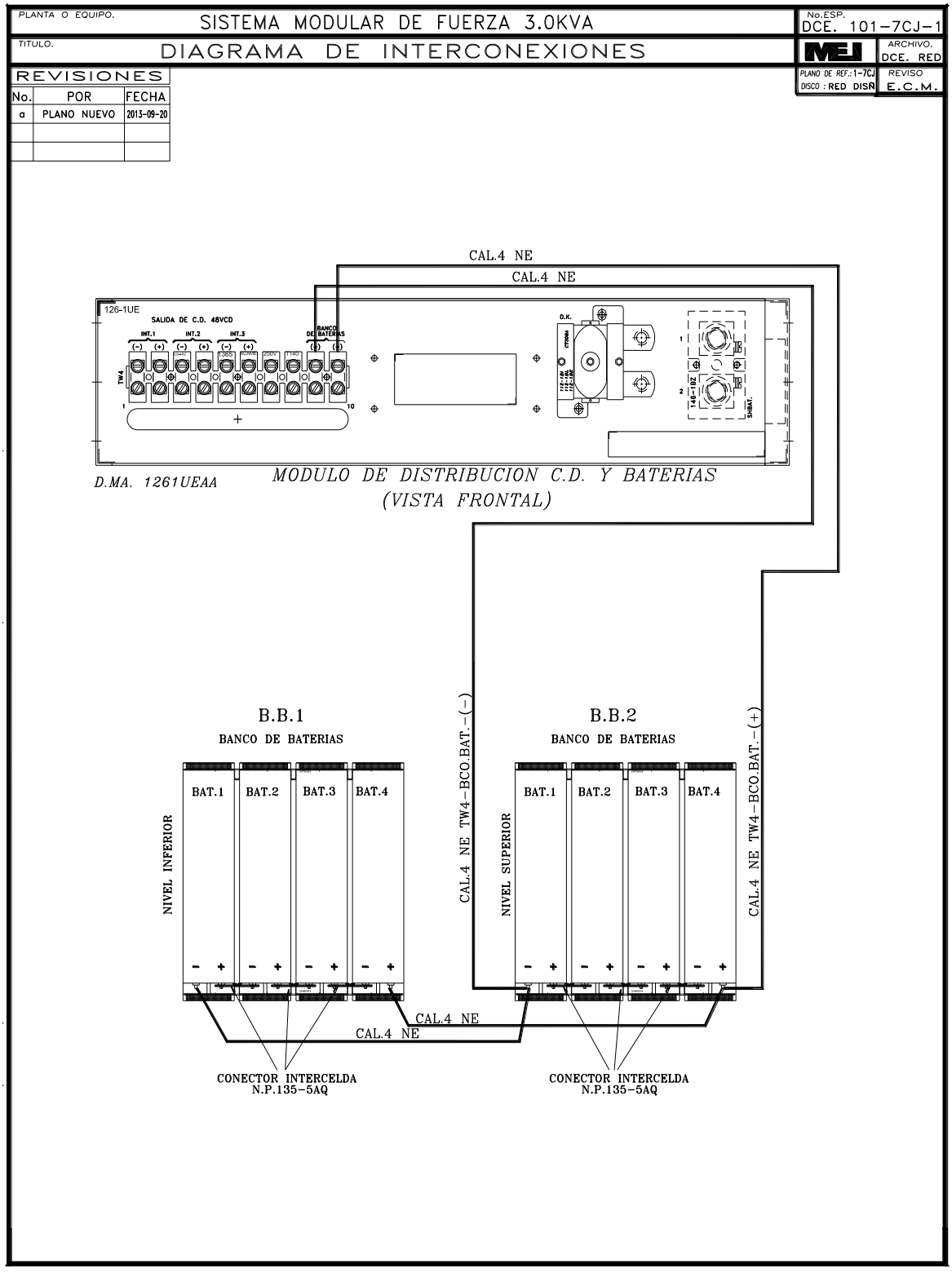

Ilustración 5. Diagrama de interconexiones del sistema de fuerza.

3.1.9 Gabinete

Todo el equipo que forma el sistema está contenido en un gabinete metálico de Acero al Carbón para montaje y aseguramiento de los equipos, del sistema.

El sistema está instalado en la parte inferior y ocupa una altura máxima de 23 unidades de un rack de 23 pulgadas. El gabinete tiene un acabado con pintura electrostática anticorrosiva, resistente a la corrosión, color negro. Cuenta con ventilas para permitir el correcto flujo del aire, el techo está preparado para instalar cuatro ventiladores de 41/2 pulgadas, el acceso de cableado en la tapa superior e inferior y acceso frontal y posterior por medio de puertas con cerraduras de seguridad.

El acceso y conexiones del Banco de Baterías será por la parte posterior del gabinete; para cambiar ó reemplazar alguna batería se deberá quitar el tope frontal. [2]

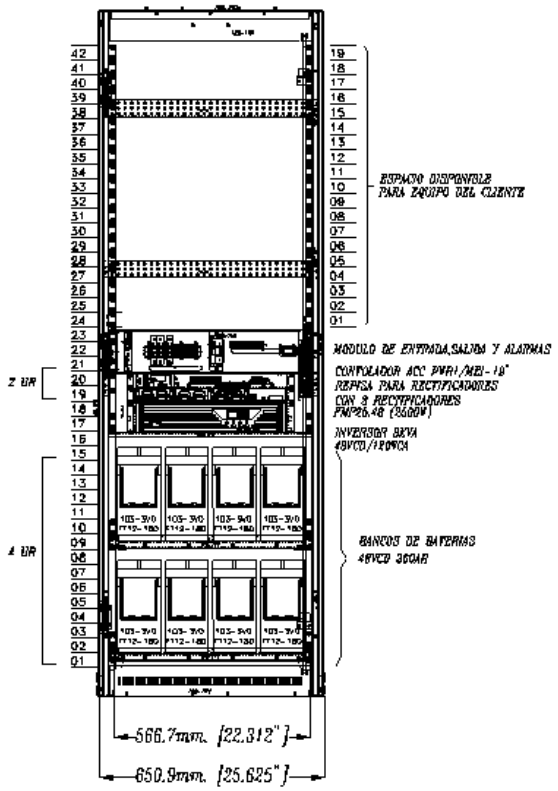

Ilustración 6. Vista Frontal del Gabinete.

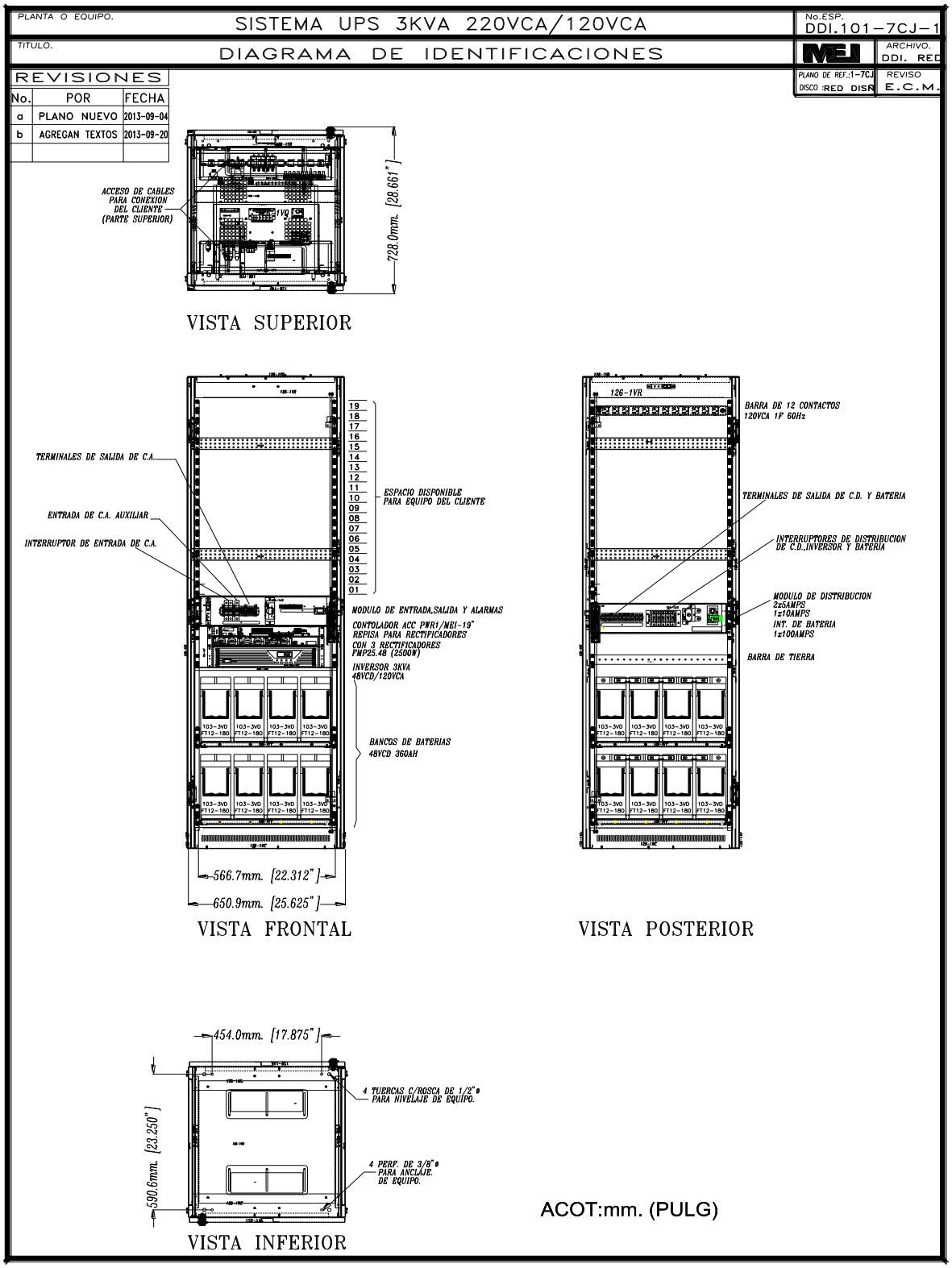

Ilustración 7. Diagrama de Identificaciones del sistema de fuerza.

3.2 Pow Com

El software PowCom es un programa basado en Windows™ que comunica, controla y supervisa a los rectificadores MEI. El software permite al usuario acceder a las plantas de fuerza MEI desde una localización remota usando una interfase gráfica de usuario para desempeñar las mismas funciones que están disponibles en los menús del PCU/PCS. [2]

Algunas de estas funciones incluyen:

- Ver los datos de Prueba de Batería.
- Mostrar las curvas de descarga.
- Función de base de datos para almacenar la información de la prueba de baterías.
- Ayuda del contexto. Al presionar la tecla F1 se abre un menú de ayuda sobre la ventana activa.
- Información de instalación del sitio.

3.3 Zabbix

Es un sistema de monitoreo de redes para supervisar y registrar el estado de múltiples servicios de red, servidores, y hardware de red, utiliza un flexible mecanismo de notificación que permite a los usuarios configurar e-mail basado en alertas prácticamente cualquier evento.

ZABBIX soporta tanto sondeos como captura. Todos los informes y estadísticas de ZABBIX, así como los parámetros de configuración, se acceden a través de una interfaz web. Un Interfaz basado en web asegura que el estado de su red y la salud de sus servidores se pueden evaluar desde cualquier ubicación. Configurado correctamente, ZABBIX puede desempeñar un papel importante en el monitoreo de la infraestructura de TI. Esto es igualmente cierto para las pequeñas organizaciones con algunos servidores y para las grandes empresas con una multitud de servidores.

Es libre de costo. ZABBIX está escrito y distribuido bajo la licencia GPL (General Public License) versión 2. Esto significa que su código fuente es distribuido libremente y disponibles para el público en general. [5]

Características principales de ZABBIX

- Monitoreo centralizado con administración web.
- Interfaz de usuario basada en la web.
- Notificación de eventos a través de E-mail, SMS, Jabber, etc .
- Autenticación de los usuarios.
- Representación gráfica de la información.
- Red de mapas.
- Pantallas personalizadas.
- Protocolo **SNMP**, JMX, IPMI.

3.4 Servidor

El servidor realiza el sondeo y captura de datos, calcula disparadores, envía notificaciones a los usuarios. Es el componente central al que los agentes y proxies de ZABBIX informan datos sobre disponibilidad e integridad de sistemas. El servidor puede revisar remotamente los servicios en red (como servidores web y servidores de correo) mediante comprobaciones de servicio simples.

El servidor es el repositorio central en el que se almacenan todos los datos de configuración, estadísticos y operativos, además funge como entidad que alertará activamente a los administradores cuando surjan problemas en cualquiera de los sistemas supervisados. [5]

3.5 Proteus Design Suite

Proteus Design Suite es software de automatización de diseño electrónico, desarrollado por Labcenter Electronics Ltd., que consta de los dos programas principales: Ares e Isis, y los módulos VSM y Electra.

ISIS:

El Programa ISIS, Intelligent Schematic Input System (Sistema de Enrutado de Esquemas Inteligente) permite diseñar el plano eléctrico del circuito que se desea realizar con componentes muy variados, desde simples resistencias, hasta alguno que otro microprocesador o microcontrolador, incluyendo fuentes de alimentación, generadores de señales y muchos otros componentes con prestaciones diferentes.

Módulo VSM:

Una de las prestaciones de Proteus, integrada con ISIS, es VSM, el Virtual System Modeling (Sistema Virtual de Modelado), una extensión integrada con ISIS, con la cual se puede simular, en tiempo real, con posibilidad de más rapidez; todas las características de varias familias de microcontroladores, introduciendo el programa que controlará el microcontrolador y cada una de sus salidas, y a la vez, simulando las tareas que queramos que lleve a cabo con el programa. Se pueden simular circuitos con microcontroladores conectados a distintos dispositivos, como motores eléctricos, pantallas de cristal líquido (LCD), teclados en matriz, etc. Incluye, entre otras, las familias de microcontrolador PIC, tal como PIC10, PIC12, PIC16, PIC18, PIC24 y dsPIC33.

3.6 VirtualBox

VirtualBox es una herramienta de virtualización de código abierto multiplataforma disponible para Windows, Linux y Mac OS X u otros sistemas operativos, que permite crear unidades de disco virtuales donde se puede instalar un sistema

operativo invitado dentro del que se utiliza normalmente en nuestro equipo y así poder usarlo del mismo modo que si hubiera sido instalado realmente.

La máquina virtual sobre la que corre el sistema virtualizado es completamente personalizable, permitiendo modificar el hardware virtual de acuerdo a las necesidades del usuario, ya sea el tipo de procesador, la memoria RAM dedicada o el espacio de almacenamiento al que podrá recurrir. Se debe tener en cuenta que todos esos recursos son una parte de las especificaciones reales del equipo, por lo que es necesario tener un equipo lo suficientemente potente para poder correr e interactuar con los sistemas operativos guest y host. [6]

3.7 Tarjeta de adquisición de datos

El Arduino Mega es una tarjeta de desarrollo open-source construida con un microcontrolador modelo Atmega2560 que posee pines de entradas y salidas (E/S), analógicas y digitales. Esta tarjeta es programada en un entorno de desarrollo que implementa el lenguaje Processing/Wiring.

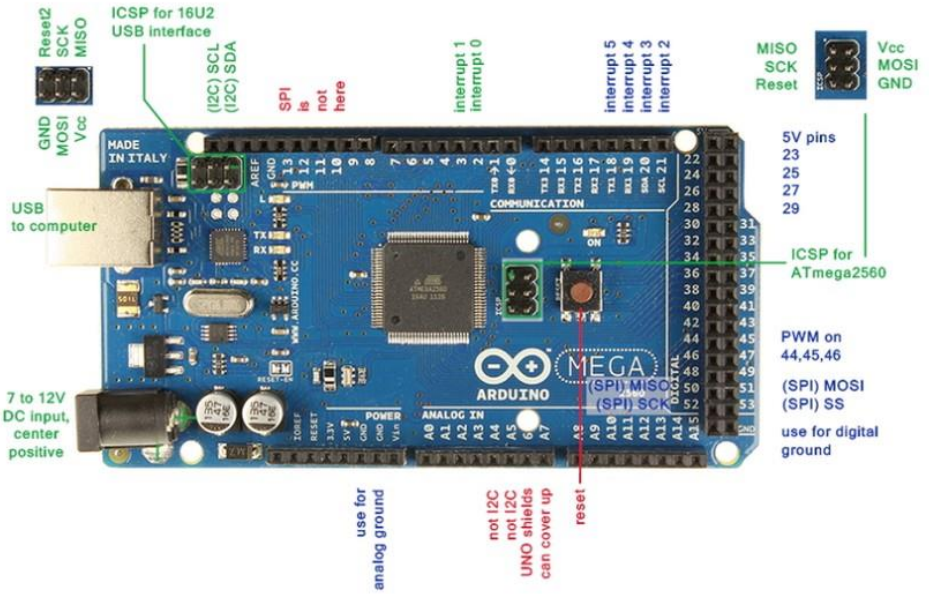

Ilustración 8. Arduino MEGA 2560.

Dispone de 54 entradas/salidas digitales, 14 de las cuales se pueden utilizar como salidas PWM (modulación de anchura de pulso). Además dispone de 16 entradas analógicas, 4 UARTs (puertas series), un oscilador de 16MHz, una conexión USB, un conector de alimentación, un conector ICSP y un pulsador para el reset. El Arduino Mega puede ser alimentado vía la conexión USB o con una fuente de alimentación externa.

Para empezar a utilizar la placa sólo es necesario conectarla al ordenador a través de un cable USB, también puede alimentarse con una batería, o alimentarla con un adaptador de corriente AC/DC.

El Arduino MEGA2560 es compatible con la mayoría de los shield o tarjetas de aplicación/ampliación disponibles para las tarjetas Arduino UNO original.

Las características principales son:

- Microprocesador ATmega2560
- Tensión de alimentación (recomendado) 7-12V
- Integra regulación y estabilización de +5Vcc
- 54 líneas de Entradas/Salidas Digitales (14 de ellas se pueden utiliza como salidas PWM)
- 16 Entradas Analógicas
- Máxima corriente continua para las entradas: 40 mA
- Salida de alimentación a 3.3V con 50 mA
- Memoria de programa de 256Kb (el bootloader ocupa 8Kb)
- Memoria SRAM de 8Kb para datos y variables del programa
- Memoria EEPROM para datos y variables no volátiles

- Velocidad del reloj de trabajo de 16MHz
- Reducidas dimensiones de 100 x 50 mm

3.8 Hardware Ethernet

10, 27, 22, 63 IP

El Arduino ethernet shield nos da la capacidad de conectar un Arduino a una red ethernet. Es la parte física que implementa la pila de protocolos TCP/IP.

Está basada en el chip ethernet Wiznet W5100. El Wiznet W5100 provee de una pila de red IP capaz de soportar TCP y UDP. Soporta hasta cuatro conexiones de sockets simultáneas. Usa la librería Ethernet para leer y escribir los flujos de datos que pasan por el puerto ethernet. Permite escribir sketches que se conecten a internet usando esta shield.

Arduino usa los pines digitales 10, 11, 12, y 13 (SPI) para comunicarse con el W5100 en la ethernet shield. Estos pines no pueden ser usados para e/s genéricas.

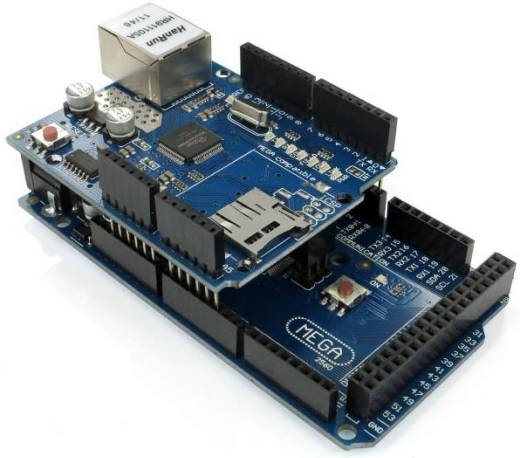

Ilustración 9. Tarjeta Ethernet Shield.

Características:

- Opera a 5V suministrados desde la placa de Arduino

- El controlador ethernet es el W5100 con 16K de buffer interno. No consume memoria.
- El shield se comunica con el microcontrolador por el bus SPI.
- Soporta hasta 4 conexiones simultáneas.
- Usar la librería Ethernet para manejar el shield.
- El shield dispone de un lector de tarjetas micro-SD que puede ser usado para guardar ficheros y servirlos sobre la red. Para ello es necesaria la librería SD.

3.9 Sensor FZ0430

El FZ0430 es un módulo que permite medir tensiones de hasta 25V con un procesador como Arduino. Este módulo es un simple divisor de tensión con resistencia de 30kOhm y 7.5 kOhm, lo que supone que la tensión percibida tras el módulo sea dividida por un factor de 5 (7.5/(30+7.5)). Es una buena opción para comprobar la alimentación de circuitos de 12 – 24 V o monitorear el estado de una batería de un prototipo.

3.10 Sensor LM35

El LM35 es un sensor de temperatura digital. A diferencia de otros dispositivos como los termistores en los que la medición de temperatura se obtiene de la medición de su resistencia eléctrica, el LM35 es un integrado con su propio circuito de control, que proporciona una salida de voltaje proporcional a la temperatura.

La salida del LM35 es lineal con la temperatura, incrementando el valor a razón de 10mV por cada grado centígrado. El rango de medición es de -55ºC (-550mV) a 150ºC (1500 mV). Su precisión a temperatura ambiente es de 0,5ºC.

3.11 Sensor ACS712

El ACS712 es un sensor de corriente tanto alterna como continua, que permite medir la intensidad eléctrica que atraviesa un conductor. Podemos emplear el ACS712 junto con un procesador como Arduino para medir la intensidad o potencia consumida por una carga.

Internamente el ACS712 consiste en un sensor hall de precisión y bajo offset junto con un canal de conducción localizado cerca de la superficie del integrado. Cuando la corriente fluye por el canal de cobre genera un campo magnético que es detectado por el sensor Hall y es convertido en una tensión.

La salida del sensor es una tensión proporcional a la corriente, y altamente independiente de la temperatura. El sensor viene calibrado desde fábrica, aunque para una medición de precisión hará falta un ajuste de la calibración.

El camino conductor está aislado galvánicamente del IC garantizado un mínimo 2.1 kVRMS Su resistencia es muy baja, 1.2 mO, lo que se traduce en pequeñas pérdidas. [7]

3.12 Sensor DHT11

El DHT11 es un sensor de temperatura y humedad digital de bajo costo. Utiliza un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos (no hay pines de entrada analógica).

Características

- Alimentación: 3Vdc ≤ Vcc ≤ 5Vdc
- Rango de medición de temperatura: 0 a 50 °C
- Precisión de medición de temperatura: ±2.0 °C .

- Resolución Temperatura: 0.1°C
- Rango de medición de humedad: 20% a 90% RH.
- Precisión de medición de humedad: 4% RH.
- Resolución Humedad: 1% RH

Capitulo IV Desarrollo del Proyecto

4.1 Diseño del sistema de medición

A partir del reconocimiento del equipo de trabajo se comenzó con el diseño del sistema de medición, para el caso de los sistemas de fuerza que son utilizados por CFE, los sistemas cuentan con un espacio que permite la instalación de hasta 4 camas de baterías, teniendo 4 baterías por cada cama. Se calculó el número de sensores que serían necesarios y se propusieron las posibles ubicaciones de cada uno. Se establecieron los valores que se tomarían en cuenta para indicar anomalías en el funcionamiento del banco.

Una vez que se trabajó con los detalles de las variables principales, se consideraron otras variables posibles a sensar, tales como, la humedad en el ambiente, indicador de puerta abierta o ultima presencia registrada.

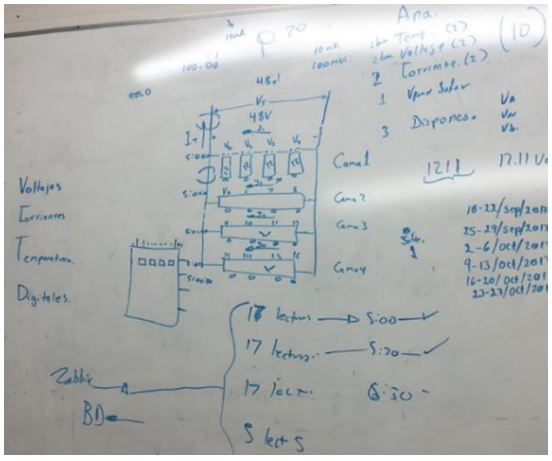

Ilustración 10. Diseño del sistema de medición.

Para la selección de sensores se consideraron las dimensiones con las que cuenta el banco de baterías, debido a que los sensores, especialmente los de temperatura, debían colocarse de manera estratégica, ya que las baterías por el uso constante tienden a presentar calor sin embargo no siempre significa un mal funcionamiento. A su vez el sobrecalentamiento de una batería, no debía afectar la medición de la otra u otras en el caso de las que se encontraban en medio. Otro punto importante a considerarse fueron los niveles de voltaje y corriente máximos a los que se someterían los sensores.

Por otro lado el sistema de medición no debía interferir con el consumo de energía del banco o con los voltajes generados por el mismo.

Ilustración 11. Sistema de Fuerza de 48 VCD.

Se decidió que lo más conveniente era utilizar un sensor de voltaje y un sensor de temperatura por cada batería, debido a que al utilizar uno por cada dos baterías,

habría que ir descartándolas una por una de acuerdo a las condiciones establecidas, convirtiendo la medición tardada e imprecisa.

Ilustración 12. Banco de Baterías.

Debido a que el sistema de fuerza ubicado en el laboratorio de comunicaciones es utilizado para pruebas y comprobaciones de los estados de baterías que traen de otras dependencias, una de las primeras pruebas que se llevaron a cabo fue la de carga y descarga de las baterías con la intención de identificar si alguna de ellas presentaba un bajo voltaje y de acuerdo a los resultados obtenidos, efectivamente una de las baterías se encontraba trabajando con un voltaje de ll.12, por lo que fue removida y sustituida por otra.

Cuando se habían determinado los sensores a utilizar y la cantidad que serían necesarios se comenzó a desarrollar el código, específicamente para adquirir los resultados de los sensores, para posteriormente realizar pruebas de funcionamiento mediante simulación.

Finalmente el modo de funcionamiento del sistema debía quedar de la siguiente forma.

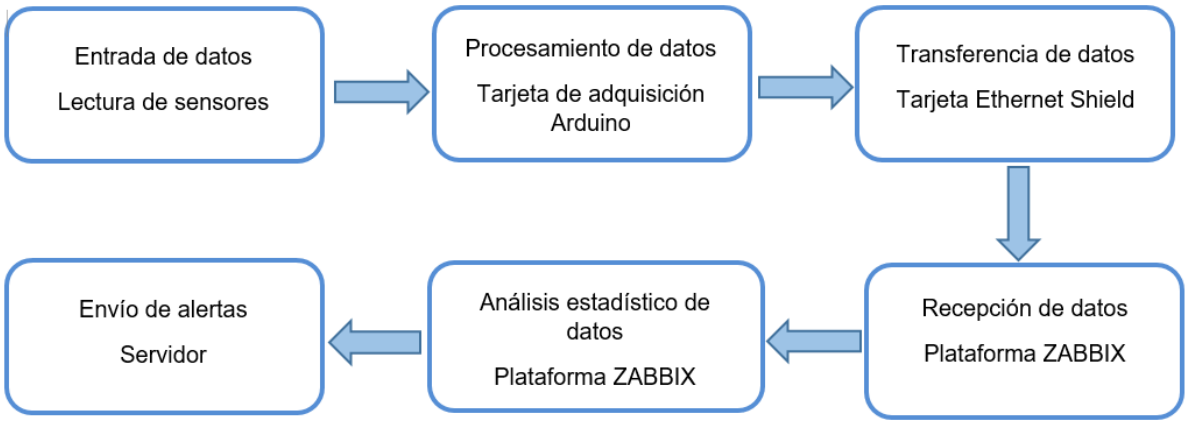

Ilustración 13. Diagrama a bloques de Software

Para la entrada de datos se tuvo que determinar un rango de medición el cual indicara los niveles altos en caso de sobrecarga o calentamiento, o bajos en el caso de una perdida de energía. Los rangos establecidos fueron los siguientes:

Temperatura alta ≤ 35°C: para una temperatura que supere los 35°C deberá enviarse una alerta notificando el incremento, si la medición se encuentra por debajo de este rango la temperatura será considerada como estable.

Temperatura normal ≈ 30°C: esta es la temperatura ideal con la que debería trabajar una batería, sin embargo, lecturas de 28°C o 34°C todavía serian consideradas como temperaturas estables por lo que no se necesitaría el envío de una alerta o notificación.

Temperatura baja ≥ 25°C: cuando la temperatura baje a menos de 25°C se notificara el descenso de esta, es decir si la última medición tuviera un valor de 24.9°C la temperatura se considerara como baja.

Voltaje alto ≤14 V: En el caso del voltaje el rango de variaciones permitidas no podían extenderse a más de 2 V, ya que esto generaría una sobrecarga en la batería y por consiguiente un sobrecalentamiento de la misma.

Voltaje normal ≈12 V: El voltaje que deben generar las baterías es de 12 V, sin embargo su voltaje en funcionamiento normal puede encontrarse entre 11 V y 13V.

Voltaje bajo ≥ 10 V: Un voltaje por debajo de los 10 V automáticamente debería enviar una notificación, ya que esto indica que la batería se encuentra en un proceso de descarga.

Como el sistema se encuentra alimentado de forma constante, es normal que se presenten picos de carga por lo que los rangos que se establecieron fueron los siguientes corriente alta ≤ 30 A, corriente normal 20 A y corriente baja ≥ 10 A.

En el caso de la última presencia detectada, ZABBIX sería el encargado de llevar el registro de los últimos resultados del sensor. De esta manera el arduino solo sería el encargado de decir si hubo o no alguna presencia reciente.

4.2 Desarrollo del código en Arduino

Las condiciones de temperatura alta o baja así como de voltaje alto o bajo, no se tomarían en cuenta durante esta etapa del desarrollo, ya que dichas condiciones serian establecidas en la plataforma ZABBIX, por lo que no era necesario incluirlas en el código.

Debido a los tipos de sensores con los que se trabajaron se utilizaron puras entradas analógicas, por lo que para leer los sensores se utilizó la función analogRead la cual permite obtener un valor entre o y 1023. A partir de esto era necesario obtener una fórmula matemática que permitiera calcular los valores deseados de temperatura, voltaje o corriente respectivamente.

```
const int TEMPB1 = A0;//Pin analógico A0 del Arduino donde conectaremos el pin de datos del sensor LM35
const int TEMPB2 = A1;
const int TEMPB3 = A2;
const int TEMPB4 = A3;
const int voltajeB1 = A7; //Pin donde conectaremos el sensor Fz0430
const int voltajeB2 = A6;
const int voltajeB3 = A5;
const int voltajeB4 = A4;
const int corriente1 = A8;//Pin donde conectaremos el sensor ACS712
const int corriente2 = A9;
```

Ilustración 14.Entradas en el código de Arduino.

Para la obtención de temperatura se utilizó la siguiente fórmula:

$$Temperatura = \frac{valor \times 5 \times 100}{1023}$$

Esta fórmula se genera debido a que los 5V proporcionados por el arduino representa un valor de 1023, por lo tanto el voltaje obtenido en el sensor debe ser multiplicado por su precisión y posteriormente transformar el resultado en grados.

En el caso de la obtención de corriente se tuvieron que aplicar dos fórmulas:

$$Voltaje = \frac{valor \times 5}{1023}$$

$$Corriente = \frac{Voltaje - 2.5}{Sensibilidad}$$

La primera fórmula permitía únicamente obtener el voltaje del sensor. Finalmente la segunda fórmula se aplicaba para la obtención de la corriente la cual se derivaba de la primera debido a que el tipo de sensor entrega un valor de 2.5V para una corriente de 0A y a partir de ese punto incrementa proporcionalmente de acuerdo a la sensibilidad, por lo que se considera una relación lineal entre el voltaje de salida del sensor y la corriente. [8]

Puede notarse que en el caso de la corriente primeramente se obtuvo el voltaje medido por el sensor, por lo que la fórmula que se utilizaría para la obtención del

voltaje seria la misma, con la única diferencia de que habría que declarar al principio del código el valor de las resistencias utilizadas, ya que el funcionamiento del sensor está basado en un divisor de tensión. Como el voltaje generado por cada batería es ≈ 12V las resistencias que se utilizaron fueron de un valor de 30kΩ y 7.5kΩ.

Se aplicaron los arreglos correspondientes para la lectura de cada sensor, de manera que el ruido en las lecturas disminuyera y fuera lo más acertada posible.

En un principio se había acordado con los ingenieros que para una mejor medición de temperatura, lo más adecuado era utilizar un sensor en forma de sonda, se había considerado el sensor DS18B20 [9] pero debido al presupuesto que se estableció finalmente se optó por el sensor LM35. Debido a esto, el sensor en forma de sonda fue agregado en el código final para que en el momento en que el departamento de comunicaciones contara con el presupuesto para futuras mejoras, se llevara a cabo un cambio de los sensores de temperatura.

Una vez finalizada la parte de lecturas por parte de los sensores, se procedió con las declaraciones para el envió de datos a ZABBIX. Durante este paso se tuvo que asignar un nombre a parte a cada resultado que se obtuviera, este nuevo nombre sería la variable que se crearía en ZABBIX, y en la cual se guardaría el valor de la lectura del sensor.

4.3 Simulación con ISIS Proteus

La comprobación con simulación se llevó a cabo en el programa ISIS Proteus, para lo que se utilizó un arduino mega, sensores de corriente ACS712 y sensores de temperatura LM35. Como el programa no cuenta con un emulador del sensor de voltaje FZ0430, este se simulo mediante un divisor de voltaje [10] ya que en este principio se encuentra basado el funcionamiento de dicho sensor. Se utilizaron resistencias de 30kΩ y 7.5kΩ, con una alimentación de 12v, el cual era el voltaje con el que debía trabajar idealmente una batería.

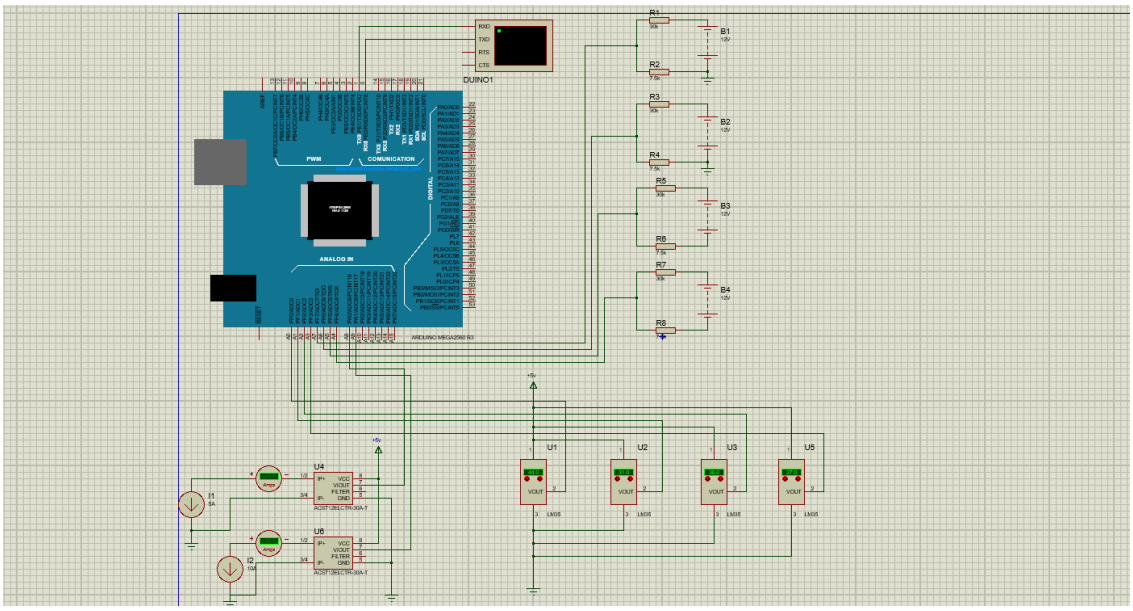

Ilustración 15. Armado de Simulación en Isis Proteus.

Durante las pruebas en simulación se hicieron variaciones que representaran los altos y bajos de voltaje, temperatura y corriente que pueden presentarse en una batería, de manera que fuera posible observar si el código cargado en el arduino funcionaba correctamente y era capaz de detectar dichas variaciones, así como también cuánto tiempo le tomaba detectarlas.

En la etapa de diseño del sistema de medición los valores que se establecieron para condiciones ideales de voltaje fueron de 12v, para temperatura era de 30°C y para corriente se estableció un valor de 20 A.

Como se puede observar en la ilustración 15 en la medición de voltaje las ultimas 2 baterías indicaban un valor de 0.0, ya que el voltaje aplicado era menor al establecido, esto debido a que el divisor de voltaje había sido diseñado específicamente para una entrada de 12V. Por otro lado el sensor si sería capaz de mostrar el valor medido en la plataforma ZABBIX.

Al contrario de lo que ocurría durante la prueba para la medición de voltaje, para las pruebas de los sensores de temperatura y corriente, el simulador si cuenta con los sensores correspondientes por lo que no se presentó algún inconveniente en

observar las lecturas obtenidas, así fueran menores o mayores las variaciones a los rangos establecidos.

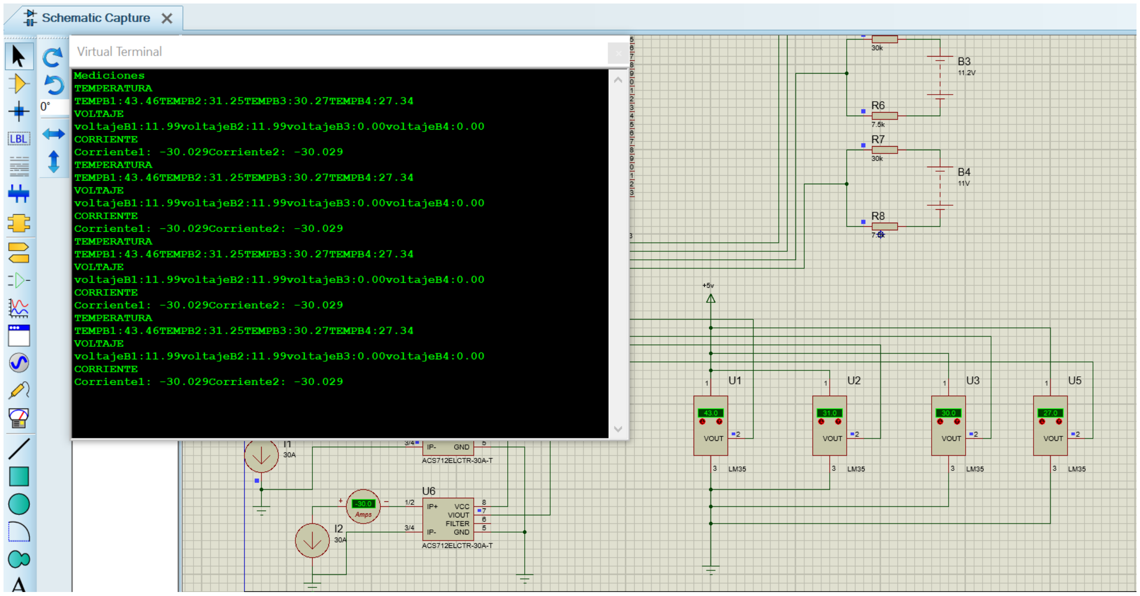

Ilustración 16. Pruebas en simulación.

Cuando se comprobó que el código llevaba a cabo las lecturas de manera exacta y se habían hecho las modificaciones indicadas por los ingenieros, se dio lugar a la implementación del código que enviaría las mediciones realizadas por los sensores a la plataforma ZABBIX.

4.4 Comunicación con plataforma ZABBIX

Para comenzar con la comunicación con ZABBIX, primeramente se tuvo que instalar el programa virtual box, y posteriormente iniciar sesión con una máquina virtual llamada Centos ya que esta es la que permite el ingreso a la plataforma.

Con el propósito de no alterar o modificar algunos de los diversos datos monitoreados por ZABBIX, se creó una cuenta para realizar las pruebas del monitoreo.

Para el ingreso a la plataforma primeramente se obtenía la dirección IP de la red de CFE y posteriormente se ingresaba la dirección obtenida en nuestro

navegador, lo cual nos llevaba a la página que había sido creada para las pruebas, agregábamos el link de ZABBIX y esto nos llevaba a la página de inicio de la plataforma, como se observa en la Ilustración 16.

Ilustración 17. Acceso a la plataforma ZABBIX.

Al iniciar sesión, la plataforma muestra los últimos eventos que ocurrieron recientemente, muestra los sistemas o variables que se encuentra monitoreando, cuando ocurrió la última medición, el valor que presento dicha variable y el cambio ocurrido con el valor anterior, en nuestro caso debía aparecer sin sucesos hasta que las variables fueran ingresadas.

En el apartado de monitoreo, que era donde se observaban las últimas mediciones adquiridas por ZABBIX, también puede encontrarse una pestaña llamada gráficas, la cual muestra una gráfica de los resultados de una sola variable o bien una sola grafica en la que se muestran los estados de todas las variables monitoreadas por la plataforma.

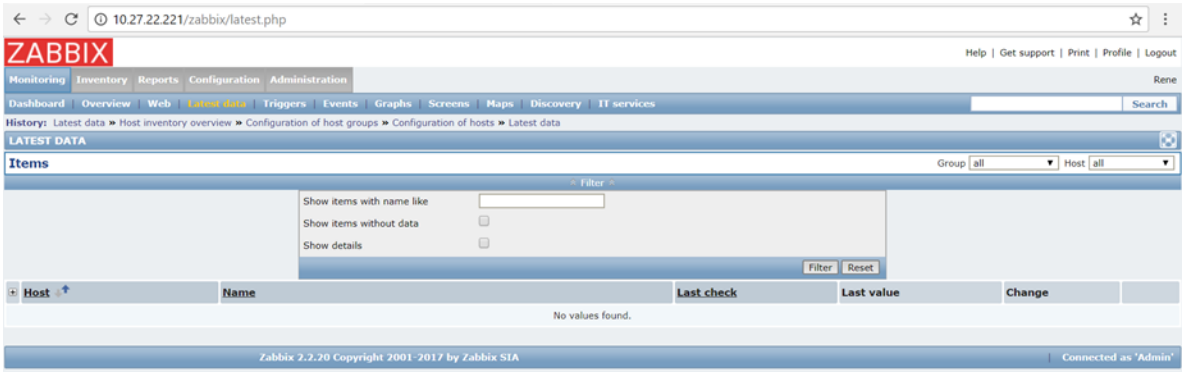

Ilustración 18. Página de Inicio de la plataforma ZABBIX.

Para cargar nuestro programa a ZABBIX se utilizó la tarjeta Ethernet shield, debido a que ZABBIX es un servidor en línea. De manera que había que indicar la dirección IP en el código de arduino y utilizando la librería Ethernet el programa seria cargado mediante la tarjeta Ethernet a ZABBIX.

Podíamos verificar que la comunicación con ZABBIX había sido exitosa, entrando en el apartado de configuración y seleccionando la pestaña de invitados, en la cual debían observarse el arduino conectado y el servidor ZABBIX, lo cual indicaba que el arduino se encontraba enviando información al servidor. Como puede observarse en la Figura.18 aparece la dirección IP de dónde provenía la información.

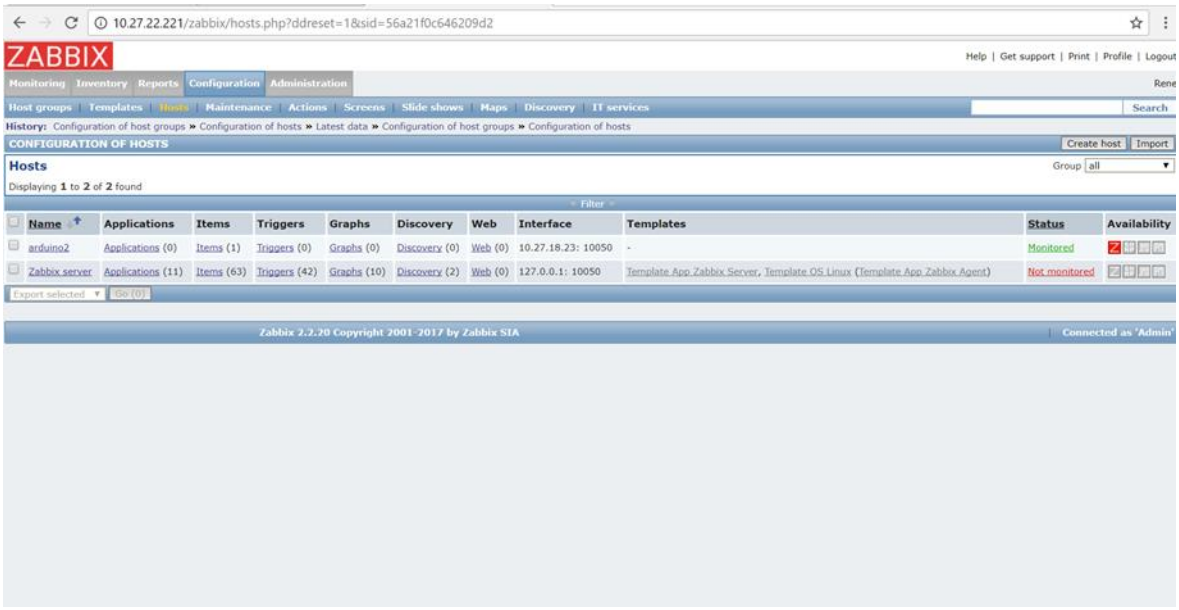

Ilustración 19. Sistemas monitoreados por ZABBIX.

Para ingresar nuestras variables a ZABBIX, pasamos al apartado de configuración, seleccionamos la pestaña de hosts y nos aparecía un espacio de llenado en el cual debía ingresarse los datos de la variable a crear. Este proceso debía realizarse por cada variable que se deseaba monitorear.

Creábamos y asignábamos el nombre que nosotros como usuarios deseáramos a nuestro host y de igual manera se asignaba un nombre el cual sería visualizado en la página de inicio. Previamente a este paso, durante la construcción del código de arduino ya se había asignado un nombre a cada resultado que se obtuviera con los sensores. De esta manera asignábamos el mismo nombre ya a nuestra variable en ZABBIX, ya que de otra forma la plataforma no reconocería un nombre diferente al asignado desde un principio y no realizaría la lectura de la misma.

Los nombres que fueron asignados a los resultados para el servidor en el código de arduino, se asignaron con vocales para mayor facilidad. Por ultimo en el caso de requerirse la modificación del nombre, había que modificarlo también en el código.

Posteriormente se creó un grupo con el nombre de “SISTEMA DE FZA LAB. COMUNICACIONES”, el cual sería el nombre para todo el sistema de monitoreo.

Una vez nombrado el sistema de monitoreo había que seleccionar al grupo al cual pertenecería de manera general, en nuestro caso se asignó al grupo de pruebas, ya que como se mencionó en un principio, la cuenta había sido creada única y exclusivamente para llevar a cabo las pruebas.

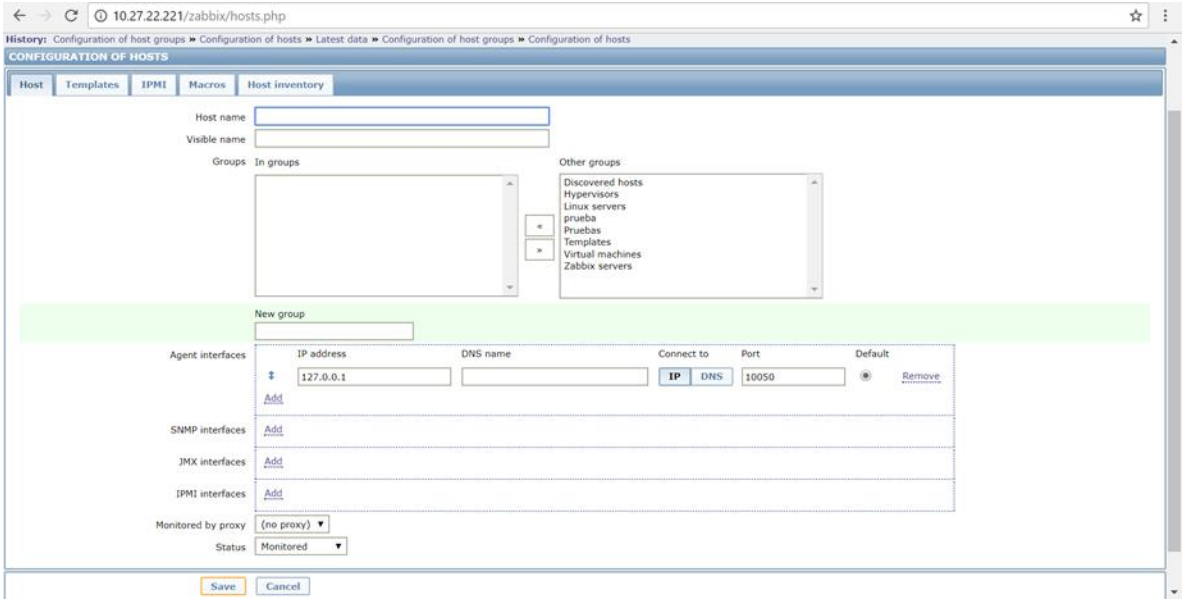

Ilustración 20. Ventana para dar de alta variables a monitorear.

Una vez que habían sido guardados los datos, se procedía con la configuración de acciones. Esta opción es en donde se habilitan los mensajes de alerta, de acuerdo a los niveles altos o bajos que se establecían. Para el caso del sistema de monitoreo se establecieron los siguientes valores:

Temperatura alta: ≤ 35°C

Temperatura normal: ≈ 30°C

Temperatura baja: ≥ 25°C

Voltaje alto: ≤14 V

Voltaje normal: ≈12 V

Voltaje bajo: ≥ 10 V

Corriente alta: ≤ 30 A

Corriente normal: 20 A

Corriente baja: ≥ 10 A

Al habilitar las alertas también se habilitaban los mensajes de alerta que serían enviados en caso de que se presentaran casos de niveles altos o bajos.

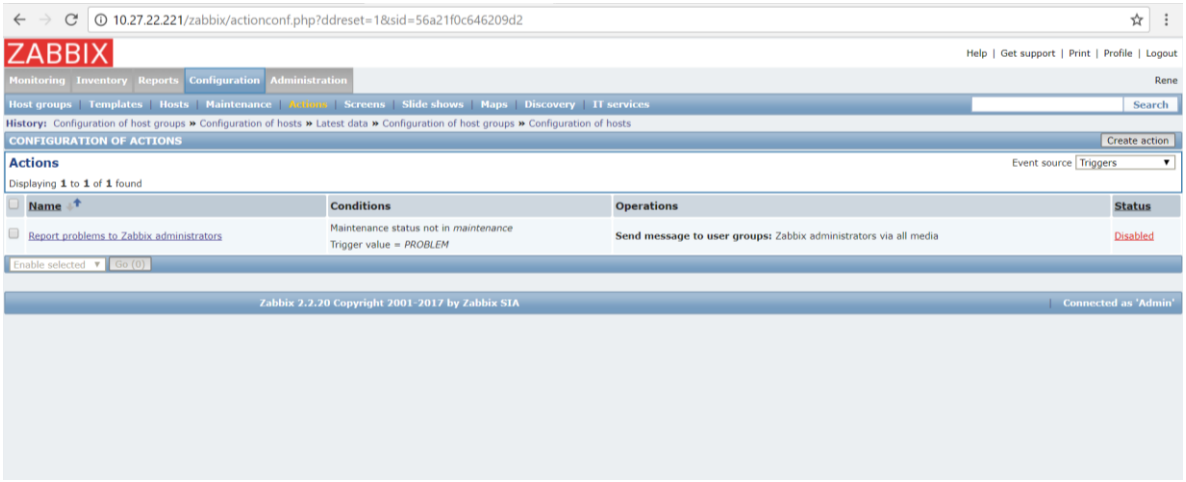

Ilustración 21. Configuración de alertas.

ZABBIX cuenta con su propia interpretación gráfica, la cual permite ver el comportamiento de la variable que se encuentre monitoreando, por lo que no es necesario el desarrollo de una interfaz gráfica externa como suele ser el caso.

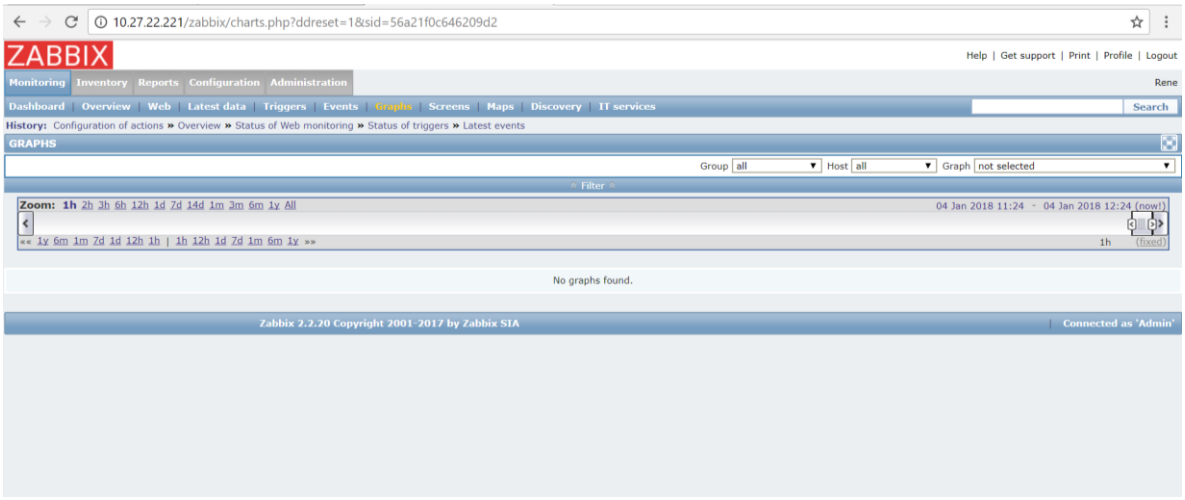

Ilustración 22. Visualización de gráficas.

La grafica muestra el valor de la última medición adquirida y los niveles máximos y mínimos de la variable en cuestión, también puede observarse en la Ilustración. 22 que la gráfica también da a conocer los tiempos en los que se llevaron a cabo las lecturas de los sensores.

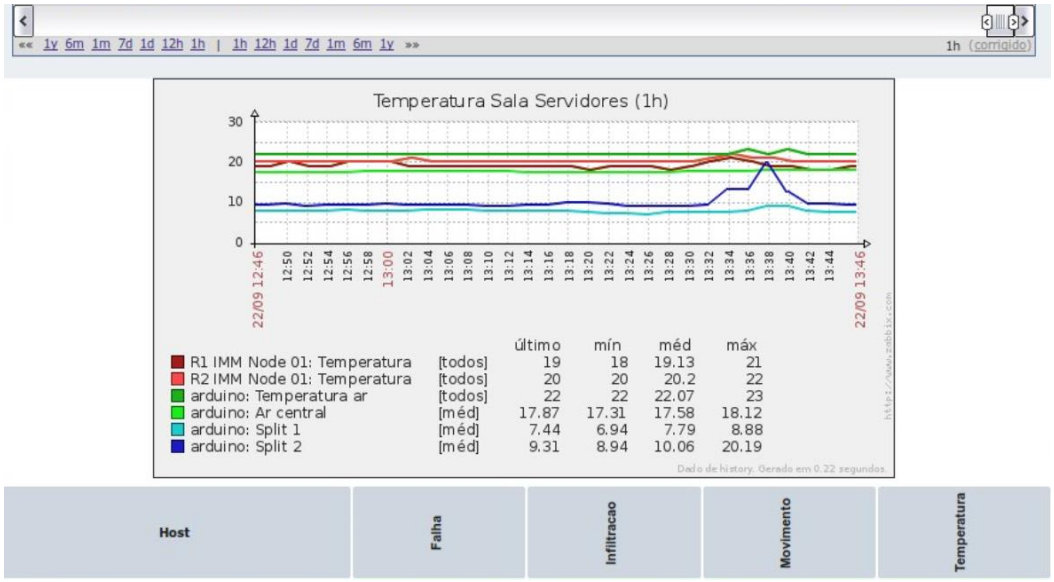

Ilustración 23. Grafica de las variables monitoreadas.

Al final podíamos observar que nuestro sistema de monitoreo ya aparecía en la plataforma ZABBIX de manera oficial con el nombre de pruebas. Esto indicaba

que nuestro sistema se encontraba en comunicación con el servidor, como puede apreciarse en la Ilustración 23.

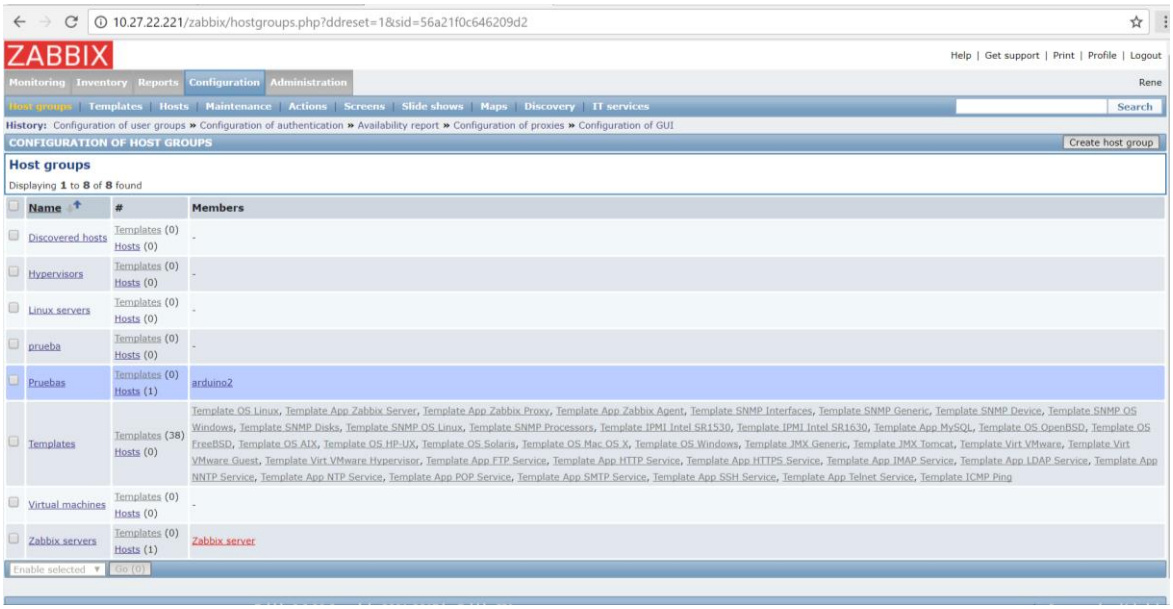

Ilustración 24. Visualización de sistemas monitoreados.

Conclusión

El departamento de comunicaciones de EPS Transmisión Tuxtla, lleva a cabo el monitoreo de diferentes sistemas, uno de ellos son los sistemas de fuerza, los cuales son encargados de energizar equipos de manera constante, por lo que el monitoreo de dichos sistemas se vuelve indispensable. Dichos sistemas de fuerza cuentan con su propio banco de baterías a los cuales no se les brinda un mantenimiento ya que las baterías que utiliza se encuentran totalmente selladas. De esta manera la medición y análisis de las lecturas registradas por el sistema de medición, permite que los ingenieros se encuentren informados del estado de los sistemas de fuerza que se encuentran instalados en distintas locaciones.

El proceso de medición por sí solo no presenta mayores dificultades, la parte crucial es él envió de los datos a la plataforma ZABBIX, ya que esta facilita él envió de mensajes de alertas y a su vez lleva un registro de las datos obtenidos con anterioridad, esto último es importante en el caso de la prevención de fallas del sistema, ya que si la gráfica presentan muchas variaciones lo más conveniente es trasladarse y hacer una revisión del banco de baterías, y así evitar las caídas de sistemas o equipos alimentados por un sistema de fuerza.

Observaciones y sugerencias

En un principio se habían elegido sensores más sofisticados para el sistema de monitoreo pero debido a que el presupuesto con el que se contaba para el desarrollo del proyecto era limitado, se optó por utilizar sensores sencillos.

Un caso en particular fue en el caso del sensor de temperatura, debido a que este tenía que ser el sensor más preciso que se utilizara, inicialmente se había seleccionado un sensor en forma de sonda, por lo que durante el desarrollo del código de Arduino también se implementó la lectura de este sensor, de manera que en el futuro puedan implementarse mejoras en el proyecto.

El proyecto requería de una interfaz gráfica en la que se mostraran los niveles de corriente, temperatura, voltaje y en el caso de ser necesario la humedad. Sin embargo, era necesario que los ingenieros pudieran consultarlo en cualquier momento y para ello debían contar con el software requerido. Debido a que CFE cuenta con su propio servidor en el cual monitorean varios sistemas, se decidió que era mucho más conveniente usar las aplicaciones que ofrecía este servidor y así no habría necesidad de la instalación de un software nuevo.

Referencias

[1] H. Z. Valverde, *Implementacion del sistema de monitoreo de temperatura del cuarto de servidores 3A,* Instituto Tecnológico de Costa Rica , 2002.

[2] «Sistema modular de Fuerza 3KVA,» de *Instructivo de Operaciones Sistema de fuerza*, MEI Multieléctrica Industrial, S.A. DE C.V., 2017.

[3] «Inversor MEI-ENATEL Modelo SET-3000-48VCD 3000 VA,» de *Instructivo de Operación Sistema de Fuerza*, MEI Multiélectrica Industrial S.A. DE C.V., 2017.

[4] *Curso Básico de Baterias,* MEI Multiélectrica Industrial, S.A. DE C.V. , 2017.

[5] J. A. S. Madrigal, *Unidad de supervision, control y adquisicion de datos para equipos de radiocomunicación en los repetidores de VHF,* Tuxtla Gutiérrez, Chiapas : Instituto Tecnologico de Tuxtla Gutierrez , 2017.

[6] Anónimo, «uptodown,» 2016. [En línea]. Available: https://virtualbox.uptodown.com/windows.

[7] Anónimo, «Naylamp mechatronics,» octubre 2016. [En línea]. Available: http://www.naylampmechatronics.com/blog/48_tutorial-sensor-de-corriente-acs712.html.

[8] V. García, «EPA Electrónica Práctica Aplicada,» 28 septiembre 2015. [En línea]. Available: https://www.diarioelectronicohoy.com/blog/sensor-de-corriente-acs712.

[9] L. Llamas, «Ingeniería, informática y diseño,» 2016. [En línea]. Available: https://www.luisllamas.es/temperatura-liquidos-arduino-ds18b20/.

[10] V. Garcia, «Hispavalia,» septiembre 2015. [En línea]. Available: https://www.hispavila.com/voltimetro-con-arduino/.

Anexos

Código de Arduino del proyecto Medición y Análisis estadísticos de variables en sistemas de fuerza de 48 VCD y banco de baterías libres de mantenimiento.

```
//*****************************************************************************
//* Zabbix Sensor Agent - Environmental Monitoring Solution *

#include <SPI.h>
#include <Ethernet.h>
//#include <dht.h>
#include <OneWire.h>
#include <string.h>

//----------------------------- Network settings ------------------------------
byte mac[] = { 0x90, 0xA2, 0xDA, 0x00, 0xE3, 0x1B };
//IPAddress ip(10, 1, 2, 235);
IPAddress ip(10, 27, 22, 63);
IPAddress gateway(10, 27, 22, 254);
IPAddress subnet(255, 255, 255, 0);

//------------ Los pines 10, 11, 12 e 13 son utilizados por la tarjeta ethernet shield ----------
#define MAX_CMD_LENGTH 25
#define LED_PIN 3              // LED pin con resistencia de 1k
#define DHT11_PIN 4            // DHT11 pin con resistencia de 10k
```

```
#define ONE_WIRE_PIN 5         // One wire pin con resistencia de 4.7k
#define PIR_PIN 6              // Presencia

//------------ Variables de entrada -----------------------------------------
const int TEMPB1 = A0;//Pin analógico A0 del Arduino donde conectaremos el pin de datos del sensor LM35
const int TEMPB2 = A1;
const int TEMPB3 = A2;
const int TEMPB4 = A3;
const int voltajeB1 = A7; //Pin donde conectaremos el sensor Fz0430
const int voltajeB2 = A6;
const int voltajeB3 = A5;
const int voltajeB4 = A4;
const int corriente1 = A8;//Pin donde conectaremos el sensor ACS712
const int corriente2 = A9;

EthernetServer server(10050);
EthernetClient client;
OneWire ds(ONE_WIRE_PIN);

//--------------------- variables-------------------------------------------------------------------
float gradosC1 = 0.0;
float gradosC2 = 0.0;
float gradosC3 = 0.0;
float gradosC4 = 0.0;
```

```
float vout1 = 0.0;
float vin1 = 0.0;
float vout2 = 0.0;
float vin2 = 0.0;
float vout3 = 0.0;
float vin3 = 0.0;
float vout4 = 0.0;
float vin4 = 0.0;
float R1 = 30000.0; //
float R2 = 7500.0; //
float sensibilidad = 0.066;  //Sensibilidad en Voltios/Amperio para sensor de 30A
float I1=0.0;
float I2=0.0;

boolean connected = false;
byte i;
byte present = 0;
byte type_s;
byte data[12];
byte addr[8];
float celsius;
float oneWire17 = 0;
float oneWireB6 = 0;
float oneWireD3 = 0;
```

```
double temp = 0;                // Temperatura
double umid = 0;                // Humedad
String cmd;                    //FOR ZABBIX COMMAND
String serialNum;
int counter = 1;                // para prueba
int chk;
int presence = 0;
int lastPresence = 0;
int limite = 1;                // Command size. Using 1 for better performance.
int waitTime = 15000;           // Default waiting time before reading sensor again.
int pirWaitTime = 200000;       // Default waiting time for PIR.
unsigned long dhtLastCheck = 0;
unsigned long pirLastCheck = 0;
unsigned long oneWireLastCheck = 0;

// Leer los sensores DS18B20 y guardar los resultados en variables
void readOneWire()
{
  if (millis() - oneWireLastCheck > waitTime) {
    if ( !ds.search(addr)) {
      //Serial.println("No more addresses.");
      ds.reset_search();
      //delay(250);
      return;
```

```
}
if (OneWire::crc8(addr, 7) != addr[7]) {
  Serial.println("CRC is not valid!");
  return;
}
switch (addr[0]) {
  case 0x10:
    type_s = 1;
    break;
  case 0x28:
    type_s = 0;
    break;
  case 0x22:
    type_s = 0;
    break;
  default:
    Serial.println("Device is not a DS18x20 family device.");
    return;
}
ds.reset();
ds.select(addr);
ds.write(0x44, 1);      // start conversion, with parasite power on at the end
//delay(1000);          // maybe 750ms is enough, maybe not
// we might do a ds.depower() here, but the reset will take care of it.
```

```
present = ds.reset();
ds.select(addr);
ds.write(0xBE);         // Read Scratchpad
for ( i = 0; i < 9; i++) {           // we need 9 bytes
  data[i] = ds.read();
}
int16_t raw = (data[1] << 8) | data[0];
if (type_s) {
  raw = raw << 3; // 9 bit resolution default
  if (data[7] == 0x10) {

    raw = (raw & 0xFFF0) + 12 - data[6];
  }
} else {
  byte cfg = (data[4] & 0x60);

  if (cfg == 0x00) raw = raw & ~7;      // 9 bit resolution, 93.75 ms
  else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
  else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
}
celsius = (float)raw / 16.0;
//Serial.print(celsius);
//Serial.println(" Celsius");
serialNum = String(addr[7], HEX);
```

```
    if (serialNum == "17") oneWire17 = celsius;
    else if (serialNum == "b6") oneWireB6 = celsius;
    else if (serialNum == "d3") oneWireD3 = celsius;
    // If Fahrenheit needed, (Fahrenheit = Celsius * 1.8) + 32;

    oneWireLastCheck = millis();
  }
}
// Leer sensores de temperatura y guardar los resultados en variables
void readTemperatura ()
{
  gradosC1 = (5.0 * analogRead(TEMPB1) * 100.0) / 1024;  //Esta es la funcion con la que obtenemos la medida del sensor en ºC

  gradosC2 = (5.0 * analogRead(TEMPB2) * 100.0) / 1024;  //Esta es la funcion con la que obtenemos la medida del sensor en ºCfloat gradosC;

  gradosC3 = (5.0 * analogRead(TEMPB3) * 100.0) / 1024;  //Esta es la funcion con la que obtenemos la medida del sensor en ºC

  gradosC4 = (5.0 * analogRead(TEMPB4) * 100.0) / 1024;  //Esta es la funcion con la que obtenemos la medida del sensor en ºC
}
// Leer sensores de voltaje y guardar los resultados en variables
void readVoltaje()
```

```
{
  float value1;
  value1 = analogRead(voltajeB1);// leer el valor en la entrada analogica
  vout1 = (value1 * 5.0) / 1024.0; // tomar en cuenta las declaraciones al inicio
  vin1 = vout1 / (R2 / (R1 + R2));

  float value2;
  value2 = analogRead(voltajeB2);// leer el valor en la entrada analogica
  vout2 = (value2 * 5.0) / 1024.0;
  vin2 = vout2 / (R2 / (R1 + R2));

  float value3;
  value3 = analogRead(voltajeB3);// leer el valor en la entrada analogica
  vout3 = (value3 * 5.0) / 1024.0;
  vin3 = vout3 / (R2 / (R1 + R2));

  float value4;
  value4 = analogRead(voltajeB4);// leer el valor en la entrada analogica
  vout4 = (value4 * 5.0) / 1024.0;
  vin4 = vout4 / (R2 / (R1 + R2));
 }
 // Leer sensores de corriente y guardar los resultados en variables
void readCorriente()
{
```

```
  float voltajeSensor1= analogRead(A8)*(5.0 / 1023.0); //lectura del sensor
        I1=(voltajeSensor1-2.5)/sensibilidad; //Ecuación  para obtener la corriente


  float voltajeSensor2= analogRead(A9)*(5.0 / 1023.0); //lectura del sensor
        I2=(voltajeSensor2-2.5)/sensibilidad; //Ecuación  para obtener la corriente
}
//Leer el sensor DHT11 y guardar los valores en variables
void readDHT11() {
 if (millis() - dhtLastCheck > waitTime) {
  /*
    chk = DHT.read11(DHT11_PIN);
    switch (chk) {
      case DHTLIB_OK:
        break;
      case DHTLIB_ERROR_CHECKSUM:
        Serial.print("Checksum error,\t");
        break;
      case DHTLIB_ERROR_TIMEOUT:
        Serial.print("Time out error,\t");
        break;
      default:
        Serial.print("Unknown error,\t");
        break;
    }
```

```
    */
    //temp = DHT.temperature;
    temp=millis()/100;
    umid= 1;

    //umid = DHT.humidity;
    dhtLastCheck = millis();
  }
}
// Leer comando recibido
void readTelnetCommand(char c) {
  if (cmd.length() == MAX_CMD_LENGTH) {
    cmd = "";
  }
  cmd += c;
  if (c == '\n' || cmd.length() == limite) {
    parseCommand();
  }
  else {
  }
}
// Leer PIR y si el valor es positivo guardar el resultado hasta la proxima lectura
void readPresence() {
  if (digitalRead(PIR_PIN)) {
```

```
    pirLastCheck = millis();
    lastPresence = 1;
  }
  if (digitalRead(PIR_PIN) && (lastPresence)) {
    presence = 1;
  }
  else {
    if (millis() - pirLastCheck > pirWaitTime) {
      presence = 0;
      lastPresence = 0;
    }
  }
}
//Commands received by agent on port 10050 parsing
void parseCommand() {
  if (cmd.equals("")) {  }
  else {
    counter = counter + 1;
    Serial.print(" Tempo: ");
    Serial.print(millis() / 1000);
    Serial.print("\t");
    Serial.print("Cmd: ");
    Serial.print(cmd);
    Serial.print("\t\t");
```

```
Serial.print("Resposta: ");
// AGENT ping
if (cmd.equals("p")) {
  server.println("1");
} // Agent version
else if (cmd.equals("l")) {
  //Serial.println("Version");
  server.println("Arduino Zabbix Agent 1.0");
  delay(100);
} //Agent temperatura
readTemperatura();
if (cmd.equals("a")){
Serial.println("TEMPERATURA");
Serial.print("TEMPB1:");
server.println(gradosC1,2);
Serial.print(gradosC1, 2);
}else if (cmd.equals("b")){
Serial.print("TEMPB2:");
server.println(gradosC2, 2);
Serial.print(gradosC2, 2);
}else if (cmd.equals("c")){
Serial.print("TEMPB3:");
server.println(gradosC3, 2);
Serial.print(gradosC3, 2);
```

```
 }else if (cmd.equals("d")){
 Serial.print ("TEMPB4:");
 server.println(gradosC4, 2);
 Serial.print (gradosC4, 2);
}//Agent voltaje
 readVoltaje();
 if (cmd.equals("e")){
 Serial.println("VOLTAJE");
 Serial.print("voltajeB1:");
 server.println(vin1, 2);
 Serial.print(vin1, 2);
}else if (cmd.equals("f")){
 Serial.print("voltajeB2:");
 server.println(vin2, 2);
 Serial.print(vin2, 2);
}else if (cmd.equals("g")){
 Serial.print("voltajeB3:");
 server.println(vin3, 2);
 Serial.print(vin3, 2);
}else if (cmd.equals("h")){
 Serial.print("voltajeB4:");
 server.println(vin4, 2);
 Serial.print(vin4, 2);
 }//Agent corriente
```

```
 if (cmd.equals("i")){
 readCorriente();
 Serial.println("CORRIENTE");
 Serial.print("Corriente1: ");
 server.println(I1,3);
 Serial.print(I1,3);
}else if (cmd.equals("j")){
 Serial.print("Corriente2: ");
 Serial.print(I2,3);
 server.println(I2,3);
 }// Agent air temperature
 if (cmd.equals("w")) {
   readDHT11();
   Serial.print(temp);
   server.println(temp);
   dhtLastCheck = millis() - waitTime;
 } // Agent air humidity
 else if (cmd.equals("z")) {
   readDHT11();
   server.println(umid);
   Serial.print(umid);
   } else if (cmd.equals("r")) {
   server.println(oneWire17);
   Serial.print(oneWire17);
```

```
    } else if (cmd.equals("f")) {
      server.println(oneWireB6);
      Serial.print(oneWireB6);
      oneWireLastCheck = oneWireLastCheck - (waitTime/3);
    } else if (cmd.equals("v")) {
      server.println(oneWireD3);
      Serial.print(oneWireD3);
      oneWireLastCheck = oneWireLastCheck - (waitTime/3);
    } else if (cmd.equals("t")) {
      server.println(presence);
      Serial.print(presence);
      oneWireLastCheck = oneWireLastCheck - (waitTime/3);
      pirLastCheck = millis() - pirWaitTime;
    } else { // Agent error
      //server.print("ZBXDZBX_NOTSUPPORTED");
      //server.print("Error");
    }
    cmd = "";
    Serial.println("");
    client.stop();
  }
}
// Loop que realizara el arduino despues de la funcion del setup ()
void loop()
```

```
{
  client = server.available();// Se verifica la conexión con el servidor
  if (client) {
    if (!connected) {
      Serial.println("Conection not available");
      client.flush();
      connected = true;
      client.stop();
    }
    if (client.available() > 0) {
      Serial.println("Client Available");
      Serial.println("Conection ok");
      int clientread = client.read();
      Serial.print(clientread);
      char charcr = clientread;
      digitalWrite(LED_PIN, HIGH);
      readTelnetCommand(clientread);
      digitalWrite(LED_PIN, LOW);
    }
  }
  if (millis()%2) {
    readPresence();
  }
  else {
```

```
    readOneWire();
  }
}
// Funcion que se llevara a cabo cuando se encienda el arduino o bien sea reseteado
void setup()
{
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
  Serial.println("Mediciones");
  delay(1200);
  //Serial.begin(115200);
  Ethernet.begin(mac, ip, gateway, subnet);
  server.begin();
  Serial.println("Setup");
  delay(1000);
}
```