INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

ESCUELA DE INGENIERIA Y CIENCIAS
PROGRAMA DE GRADUADOS

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE NOTIFICACIÓN DE ALARMAS PARA PROTECCIONES DE SUBESTACIONES ELÉCTRICAS

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER
EL GRADO ACADÉMICO DE:

MAESTRO EN CIENCIAS
CON ESPECIALIDAD EN INGENIERIA ENERGÉTICA

POR:

ING. MARCO ANTONIO RINCÓN GONZÁLEZ

MONTERREY, N.L. MAYO DE 2017

PÁGINA
INTENCIONALMENTE EN BLANCO

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Escuela de Ingeniería y Ciencias

Programa de Graduados

Los miembros del Comité de Tesis recomendamos que la presente tesis del Ingeniero Marco Antonio Rincón González sea aceptada como requisito parcial para obtener el grado académico de **Maestro en Ciencias,** con especialidad en:

Ingeniería Energética

Comité de Tesis

Dr. Armando Rafael Llamas Terrés
Tecnológico de Monterrey
Asesor Principal

M.C. Jesús Antonio Baez Moreno
Tecnológico de Monterrey
Sinodal

M.C. Luis Enrique Camargo Reyes
Tecnológico de Monterrey
Sinodal

Dr. Rubén Morales Menéndez
Director Nacional de Posgrado
Escuela de Ingeniería y Ciencias
Mayo 2017

PÁGINA
INTENCIONALMENTE EN BLANCO

Declaración de autoría

Yo, Marco Antonio Rincón González, declaro que esta disertación titulada, Diseño e Implementación de un Sistema de Notificación de Alarmas para Protecciones de Subestaciones Eléctricas, y el trabajo que se presenta en ella es de mi autoría. Adicionalmente, confirmo que:

- Realice este trabajo en su totalidad durante mi candidatura al grado de doctor en esta universidad.
- He dado crédito a cualquier parte de esta disertación que haya sido previamente sometida para obtener un grado académico o cualquier otro tipo de titulación en esta o cualquier otra universidad.
- He dado crédito a cualquier trabajo previamente publicado que se haya consultado en esta disertación.
- He citado el trabajo consultado de otros autores, y la fuente de donde los obtuve.
- He dado crédito a todas las fuentes de ayuda utilizadas.
- He dado crédito a las contribuciones de mis coautores, cuando los resultados corresponden a un trabajo colaborativo.
- Esta disertación es enteramente mía, con excepción de las citas indicadas.

_____

Marco Antonio Rincón González

Monterrey Nuevo León, 15 de Mayo de 2017

PÁGINA
INTENCIONALMENTE EN BLANCO

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE NOTIFICACIÓN DE ALARMAS PARA PROTECCIONES DE SUBESTACIONES ELÉCTRICAS

POR:

ING. MARCO ANTONIO RINCÓN GONZÁLEZ

TESIS

Presentada al Programa de Graduados de la Escuela de Ingeniería y Ciencias

Este trabajo es requisito parcial para obtener el grado académico de Maestro en Ciencias con especialidad en Ingeniería Energética

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

Mayo 2017

Agradecimientos

A mis padres M.C. Marco Antonio Rincón Jasso y la Sra. Maricela González Mancillas por creer en mí, brindarme su apoyo incondicional durante toda la vida y alentarme a ser siempre mejor; a ellos les debo mis valores.

A mis hermanas Maricela y Mariana por estar conmigo y a mis más cercanas amistades por no permitir que me rindiera en los momentos difíciles.

A todos ellos mis más sinceros agradecimientos.

Reconocimientos

A mis asesores de tesis M.C. Jesús Antonio Baez Moreno y al Dr. Armando Rafael Llamas Terrés por sus consejos, guía, asesorías y correcciones que brindaron para el desarrollo de esta Tesis; y sobre todo por el interés y confianza en apoyarme a desarrollar este tema que es de mucho valor personal.

A los sinodales por el apoyo y tiempo dedicado a estudiar esta tesis y sus invaluables retroalimentaciones para mejorar la calidad de la misma.

A Power Testing de México SA de CV por brindar los recursos financieros y tecnológicos para el desarrollo de la etapa de diseño, implementación y pruebas de integración del tema de tesis en campo.

A mis profesores de la maestría porque además de impartir incondicionalmente sus conocimientos, experiencias y anécdotas siempre me motivaron a dar lo mejor en todo momento.

Y finalmente a todas aquellas personas y amigos que directa e indirectamente brindaron su apoyo para la elaboración de este proyecto.

Índice de Contenido

Índice de Figuras

Índice de Tablas

Índice de Apéndices

Acrónimos

AC	Alternating Current
ADC	Analogue to Digital Converter
AES	Advanced Encryption Standard
API	Application Programming Interface
AVR	Alf Vegard RISC Processor
BOM	Bill of Materials
CFE	Comisión Federal de Electricidad
DC	Direct Current
DI	Digital Input
DO	Digital Output
E/S	Entrada / Salida
EEPROM	Electrically Erasable Programmable Read Only Memory
EMS	Energy Management System
EPRI	Electric Power Research Institute
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
HB	Heartbeat
HMI	Human Machine Interface
ICSP	In-circuit Serial Programming
IDE	Integrated Development Environment
IEC	International Electro technical Commission
IED	Intelligent Electronic Devices
IEEE	Institute of Electrical and Electronics Engineers
IMEI	International Mobile Station Equipment Identity
IOT	Internet of Things
ISP	Internet Service Provider
JSON	JavaScript Object Notation
LAN	Local Area Network
MSN	Manufacturer Serial Number
NERC	North America Electric Reliability Council
NIST	National Institute of Standards and Technology
NTP	Network Time Protocol
OAuth	Open standard for Authorization
P2P	Peer to Peer
PCB	Printed Circuit Board
POP	Post Office Protocol
PSTN	Public Switched Telephone Network
PSU	Power Supply Unit
PWM	Pulse Width Modulation
RISC	Reduced Instructions Set Computing
RTU	Remote Terminal Unit
SCADA	Supervisory Control And Data Acquisition
SD	Secure Digital

SEL	Schweitzer Engineering Laboratories
SF6	Sulfur Hexafluoride / Hexafloruro de Azufre
SIM	Subscriber Identity Module
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SSL	Secure Sockets Layer
TCP/IP	Transmission Control Protocol / Internet Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
URL	Uniform Resource Locator
USB	Universal Serial Bus
VPN	Virtual Private Network
WAP	Wide Area Protection
XML	eXtensible Markup Language

Resumen

La notificación temprana de eventos y disturbios generados por fallas eléctricas o un problema en una subestación eléctrica es de suma importancia para garantizar el estado óptimo de los equipos de la caseta de control y que todos los elementos de protección e interruptores se encuentren en condiciones de operación y servicio. Se sabe además que el mantenimiento y servicio oportuno a equipos de control y protección de una subestación aminora costos de reparación y garantizan la continuidad en el servicio eléctrico a nivel industrial y social teniendo un impacto positivo.

En este trabajo de tesis para la Maestría en Ciencias con especialidad en Ingeniería Energética se propone el desarrollo de un equipo de fácil integración al circuito de control y protección de una subestación eléctrica de cualquier tamaño permitiendo que la misma pueda transmitir y notificar el estado de sus alarmas en tiempo real notificando a personal competente para atender de forma oportuna el evento o falla en la subestación.

Muchas subestaciones eléctricas no cuentan con un método de notificación que permita la toma de acción oportuna ante un problema por lo que en este trabajo se establece que el dispositivo en diseño sea al menor costo sin escatimar eficiencia. Se cree además, que al trabajar el dispositivo en una red utilizando recursos de Cloud Computing puede ayudar a reducir los costos de operación sin perjudicar el tren de comunicación y con ello permitir que el servicio eléctrico se mantenga de una forma eficiente, confiable y seguro.

1 Introducción

Los eventos eléctricos y las interrupciones en el servicio existen desde el día en que se instalaron las primeras redes, ante la problemática y afectación económica que trae este tipo de situaciones se empezaron a diseñar e instalar elementos de protección con el propósito de realizar la desconexión del elemento que presente una falla y con ello mantener la estabilidad en la red eléctrica afectando al menor número de usuarios.

De acuerdo al EPRI en sus estudios por estimar el impacto económico que conlleva una mala calidad o una interrupción en el servicio eléctrico se determinó que solamente en Estados Unidos el costo ante este tipo de eventos se encuentra entre $104 y $164 billones de dólares por año debido a interrupciones más un adicional de $15 a $24 billones de dólares imputables a problemas de calidad de energía eléctrica [1]. Ante esta situación se puede determinar el impacto y la importancia de mantener en óptimas condiciones todos aquellos elementos que forman parte de la cadena de suministro del servicio eléctrico, es decir, plantas generadoras, subestaciones, líneas de transmisión y sistemas de distribución.

La continuidad en el servicio eléctrico se encuentra bajo la sombra de los relevadores de protección los cuales han evolucionado en tecnología desde su primera implementación en 1934 siendo los primeros de carácter electromecánico cuyo principio de funcionamiento es en inducción que estimula el movimiento de un mecanismo que realiza la operación del elemento y desconexión de la falla [2]. En la actualidad, estos equipos son de carácter electrónico y basados en tecnologías de microcontrolador; aunque todavía es posible encontrar relevadores de protección electromecánicos, estos se han reemplazados por equipos más modernos que tienen la capacidad de guardar información del evento eléctrico y además tienen funciones de autodiagnóstico de tal forma que si se detecta un problema interno en el equipo de protección, este puede anunciarlo apropiadamente [3].

De acuerdo con Altuve y Schweitzer, la mayoría de las fallas eléctricas se encuentran a nivel de distribución debido a la ausencia de redundancia en equipos de protección en las subestaciones [4] por lo que se debe de prestar especial

atención en ese punto de la cadena de suministro; dentro del esquema de protecciones de una subestación eléctrica se debe de garantizar que se cumpla la detección oportuna de una falla o anomalía eléctrica. Posteriormente, realizar sin atraso alguno la desconexión del circuito o sección que presente la falla y por último anunciar e indicar la presencia de una falla eléctrica [5]. No obstante se debe de considerar que en muchas ocasiones los equipos de protección se encuentran confinados dentro del perímetro de la subestación donde muy poco personal tiene acceso a ella y no se visita con frecuencia; por lo que queda evidente que ante la presencia de una falla en un equipo o un evento que sea detectado por los relevadores de protección, si estos no llegan a interrumpir el suministro eléctrico por completo, se carece de un medio de notificación temprana que de manera oportuna informe a personal competente sobre la situación, y se brinde atención inmediata para evaluar y restaurar la subestación a condiciones óptimas de operación, antes de que un pequeño problema evolucione por falta de atención a un situación más grave, generando altos costos de mantenimiento, reparación y/o restauración de servicio.

Este proyecto de tesis busca integrar los sistemas de protección a un equipo que sea capaz de brindar de una forma económica y automatizada la transmisión de mensajes que entregue información de manera oportuna de la presencia de eventos eléctricos y que ésta sea canalizada directamente al personal competente para brindar atención inmediata al área. Aunque esta función es existente, la implementación de la misma representa una inversión de alto costo en infraestructura que desalienta a los usuarios con instalaciones privadas a incluirla. No obstante, su ausencia, compromete la integridad de la red a una mayor escala.

La premisa de este trabajo radica en la implementación de un dispositivo confiable y de bajo costo que realice la labor de adquisición de datos de las señales del panel de alarma y del circuito de control de una subestación y las digitalice para crear un mensaje transmitido al personal competente a través de una plataforma de recursos informáticos compartidos o *Cloud Computing,* que ofrece a los usuarios una alternativa económica para el monitoreo remoto de las instalaciones eléctricas de potencia.

2 Definición del Problema

Mantener las subestaciones eléctricas privadas en condiciones óptimas de operación es una obligación del usuario y garantizan que siempre se encuentran aptas para transformar y distribuir la energía de una forma eficiente y segura y puedan además, realizar apropiadamente la desconexión de la red eléctrica cuando se detecte una falla en la red del usuario con el objetivo de evitar que elementos de protección que protegen un área más amplia entren en operación y saquen de servicio a toda una zona o región. Es común que en la industria se de mantenimiento a las subestaciones para detectar y corregir un problema y asegurar que los equipos se encuentren en buenas condiciones. Sin embargo, esto no siempre pasa o no se hacen revisiones rutinarias que pudieran detectar el inicio de algún problema.

Las funciones con las que debe de cumplir un relevador de protección de cualquier clase son, de acuerdo con Altuve, y no se encuentran limitadas a:

i. Detectar oportunamente la presencia de una falla o anomalía eléctrica.
ii. Realizar la desconexión de la falla eléctrica.
iii. Anunciar e indicar la presencia de una falla eléctrica [5].

Sin embargo, existe la necesidad de agregar un cuarto punto donde la notificación temprana de una falla eléctrica o una anomalía en la subestación sea informada de manera oportuna a personal capacitado o experto en el tema para que se pueda brindar atención de forma inmediata. Queda claro que la tecnología de comunicación juega un rol muy importante en el manejo y operación de los equipos de protección de una subestación; sobre todo para la corrección temprana de problemas, que si no se atienden en tiempo y en forma pueda desencadenar en una falla en la que los equipos principales de una subestación sean dañados y el costo de reparación sea consecuentemente mayor.

Skendzic asegura que con el crecimiento de la red eléctrica se requiere de una mejor eficiencia, confiabilidad y robustez en la operación de los equipos de potencia para poder tener una respuesta más rápida ante un evento eléctrico [6], por lo que tener un mecanismo que permita un diagnostico generalizado del estado de operación o nivel de salud de una subestación eléctrica a distancia impacta en la confiabilidad y persistencia de la robustez en el sistema de protección de la misma.

En la actualidad existen numerosos métodos y plataformas que permiten resolver la cuestión de la comunicación en diferentes niveles de resolución hacia los equipos de la subestación como lo son enlaces dedicados, servidores VPN y conexiones punto a punto por mencionar algunos. No obstante, en la industria privada no se tiene la infraestructura que permita el monitoreo constante de las mismas. A pesar que los equipos de protección actuales son capaces de anunciar problemas o fallas internas en la operación del mismo, la implementación de una infraestructura que permita la notificación de una avería, problema o evento eléctrico de manera remota es costosa por lo que frecuentemente se prefiere no instalar, confiando ciegamente en los elementos de protección de la subestación eléctrica.

No obstante, si se logra la implementación de un dispositivo que de forma no invasiva a un circuito de control y protección logre leer y transmitir las señales del panel de alarmas de una subestación mediante una conexión tradicional a la red a través de servicios de internet de *Cloud Computing* (el cual permite la utilización de recursos informáticos de manera remota y bajo demanda [7]), disminuiría la cantidad de recursos informáticos y tecnológicos dentro de una subestación, con la posibilidad de eliminar la adquisición de un RTU; al disminuir los recursos de hardware requeridos en el dispositivo, este disminuye su costo, ya que todo el procesamiento se realiza de forma externa.

La implementación de este sistema dentro de los esquemas de control de la subestación permite al personal de mantenimiento detectar el origen de una anomalía y que esta pueda ser corregida a tiempo, antes que tenga una afectación mayor dentro o fuera del perímetro de la subestación.

3 Objetivos

El objetivo de este trabajo es implementación de un dispositivo no invasivo y de menor costo que la oferta comercial, que permita la notificación del estado de alarmas de una subestación eléctrica mediante los canales de comunicación apropiados y utilizando *Cloud Computing* como uno de sus recursos, realizando una integración transparente con los circuitos de control existentes y sin perjudicar el desempeño de una subestación que cuyo diseño original no tenga contemplado la integración de un equipo de comunicación. Una vez que se logre la integración, el sistema o dispositivo permitirá que de forma periódica se esté transmitiendo el estado de la subestación y notificar de forma inmediata alguna señal de alarma o problema que ponga en juego la integridad así como la efectividad de la operación de una subestación eléctrica.

De forma particular, se enlistan los siguientes objetivos:

i. Proponer un esquema de comunicación con el cual se pueda notificar de manera oportuna y de forma periódica la operación, la falla o *health level* de una subestación eléctrica.
ii. Implementar de forma razonable un esquema que utilice *Cloud Computing* como un método de ahorro en recursos informáticos.
iii. Desarrollar a manera de prototipo funcional una interfaz de potencia que interactúe con señales de control y monitoreo local de una subestación.
iv. Desarrollar el firmware de monitoreo para el dispositivo, con la capacidad de monitorear la operación de las alarmas en una subestación y transmitirla por un canal de comunicación a personal competente.
v. Definir un canal de comunicación de respaldo con el propósito de ofrecer una forma alterna de transmisión en caso de una falla afecte las comunicaciones dentro de la subestación.
vi. Realizar un análisis costo/beneficio del sistema propuesto contra algunos equivalentes en el mercado, como lo son: enlaces dedicados, servidores VPN, conexión P2P, etc.

4 Hipótesis

La hipótesis es demostrar la funcionalidad de la alternativa de comunicación en sistemas de potencia y con ello mejorar la confiabilidad en el suministro eléctrico por medio de la detección oportuna de anomalías eléctricas que pongan en juego la continuidad del servicio. Además se tiene la certeza que la detección y notificación temprana de una anomalía en una subestación eléctrica impacta de manera positiva en el tiempo de vida de una subestación.

i. ¿Cuál es el método más efectivo para realizar una interconexión del sistema propuesto con el cableado de control de una subestación eléctrica?
ii. ¿De qué forma se puede diseñar una interfaz de potencia que obtenga señales de sistemas críticos de la operación en una subestación?
iii. ¿Qué alarmas o señales son críticas para identificar la presencia de un evento eléctrico o una anomalía en la operación de los equipos de una subestación?
iv. ¿Qué mecanismo de comunicación es el más apropiado para mandar estas señales de una forma efectiva, económica y funcional? ¿Es el servicio de *Cloud Computing* la solución más viable?

5 Fundamentos

Actualmente, las comunicaciones en las subestaciones eléctricas juegan un rol importante. Un ejemplo del uso de las comunicaciones se puede ver en la red eléctrica mexicana donde las subestaciones eléctricas de las líneas de subtransmisión por lo regular están protegidas en esquemas tradicionales con protecciones a distancia y de sobre corriente. Actualmente, ese esquema no es una buena solución para ese tipo de configuración y desde el año 2000 se implementaron los primeros sistemas de comunicación entre subestaciones basados en radio [8], donde los relevadores de protección intercambian información sobre el estatus de sus zonas de protección.

Estos esquemas de comunicación permiten aumentar la velocidad con la que se realiza la desconexión de una falla y es más veloz cuando se compara con un esquema tradicional. Este es un ejemplo de cómo las comunicaciones ofrecen una ventaja en la operación de una subestación solamente si se ve por el lado de los esquemas de protección. Sin embargo, para este proyecto se desea utilizar la comunicación para brindar información oportuna y adelantada de la presencia un problema en la subestación.

Por otro lado se debe de buscar que la alternativa propuesta utilice sistemas de comunicación y de recursos informáticos compartidos mediante el uso de *Cloud Computing* lo cual limita la infraestructura de una subestación a que cuente con un nodo con acceso a la Web.

Este dispositivo debe de tomar en cuenta además que la mayoría de la señales con las que va a interactuar son señales de corriente directa de 125VDC por lo que se debe de tomar en cuenta el desarrollo de la interfaz de potencia apropiada.

5.1 Comunicaciones en un ambiente de trabajo de eléctrico

Los sistemas de comunicación modernos suelen ser muy grandes y complejos. Por lo que proteger, controlar y monitorear los sistemas de potencia requieren del intercambio de información, algunas de las aplicaciones de acuerdo a Skendzic y Altuve que se pueden lograr con la comunicación de equipos de potencia son las siguientes:

i. Protección piloto
ii. Automatización de subestaciones y sistemas de distribución de energía
iii. Monitoreo, control y WAP
iv. Sistemas de Administración de Energía (EMS)
v. SCADA
vi. Seguridad
vii. Accesos de Ingeniería y Mantenimiento [5, 6]

Cada una de estas aplicaciones cumplen con un propósito particular, y demuestran el impacto tecnológico que se puede lograr con la implementación de sistemas de comunicación en las redes eléctricas; los dispositivos actuales de protección, ofrecen la integración de muchas de estas funciones para lograr la comunicación entre equipos y la operación efectiva de los mismos sin perder coordinación en la operación de protecciones ante un evento eléctrico. Se debe de recordar que el propósito de este trabajo es utilizar la tecnología existente para integrar de forma efectiva, económica y funcional la información crítica necesaria para transmitir un mensaje de alarma canalizado directamente al personal competente.

Por otro lado, es importante aclarar la función de cada una de estas aplicaciones para reflexionar sobre el rol crítico que cumple cada una de ellas y hacer énfasis en la importancia de estar enlazados a un centro de monitoreo o bien, notificar a personal competente. Por ejemplo, en el caso de la protección remota o piloto, común en los sistemas de protección de líneas de transmisión, donde se usa un canal de comunicación para comparar información sobre cada una de las terminales de la línea de transmisión y poder tener una protección efectiva a lo largo de toda la línea [9].

Por otro lado, la automatización de una subestación se refiere a la implementación de funciones de la operación de la misma y aplicaciones como lo son: el control supervisorio y adquisición de datos (SCADA), el procesamiento de alarmas o el control integrado de Volt/VAr de tal forma que se pueda optimizar la

administración de los recursos y mejorar la operación y mantenimiento de la subestación con la mínima intervención humana [10]. McDonald, muestra en su artículo *Substation Automation* los cinco niveles de automatización identificados en el área, una representación gráfica se muestra en la Figura 5-1 a continuación.

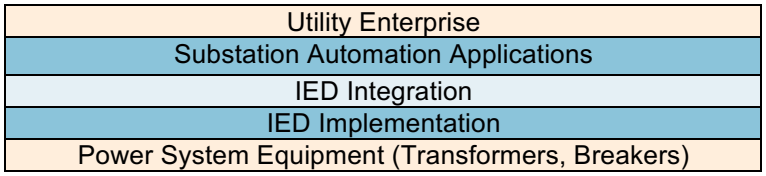


Figura 5-1. Cinco Niveles de Automatización e Integración en una subestación

No obstante, a consecuencia de las interrupciones en el servicio eléctrico a gran escala, conocido como apagones o *blackouts* se han desarrollado esquemas de protección de área amplia (WAP o *Wide Area Protection*). En muchos casos, una operación en falso o una falla en los relevadores de protección pueden llevar a un sistema de inestable. Los estudios de disturbios en la red eléctrica realizados por la NERC han demostrado que la operación en falso de un respaldo de protección puede conllevar a un efecto cascada que desencadena la interrupción en el servicio [11]. Los esquemas de protección amplia o WAPs involucran un canal de comunicación que permite la comunicación entre sistemas de operación para identificar una falla de manera eficaz y aislarla de una forma eficiente evitando efectos cascada.

Otra particularidad de las comunicaciones en un ambiente eléctrico es el de la implementación de los sistemas de administración de energía (*EMS Energy Management Systems)* esto es un paquete de software utilizado en una gran variedad de aplicaciones para monitorear y controlar sistemas de generación, distribución o de consumo de energía. Conforme la demanda energética crece, la importancia de implementar EMS en los sectores, industrial, comercial y residencial, también crece [12].

Por otro lado, también permite la implementación de los sistemas de Control Supervisorio y Adquisición de Datos (SCADA) ya que son sistemas altamente distribuidos usados para el control y monitoreo de recursos que se encuentran localizados en áreas geográficamente grandes. Donde la presencia de centros de

adquisición de datos y de control centralizado son críticos para la operación de una aplicación o sistema. Por lo que basados en la información recibida de estaciones remotas, comandos automatizados o indicados por un operador pueden ser enviados hacia los dispositivos esclavos o remotos [13].

Sin embargo, siempre se debe de tomar en cuenta la seguridad en la red de comunicación ya que basado en los comentarios del Dr. Edmund Schweitzer III, fundador de SEL *"Thou shall never connect critical infrastructure to the internet"* [14, 5] y efectivamente, la infraestructura de una subestación eléctrica debe de tener una red independiente de forma local cuyas funciones estén limitadas cuando se dé acceso a la red del exterior, ya que el uso no autorizado de la misma puede desencadenar un riesgo de seguridad. Para ello se sugiere el uso de firewalls y el uso administrado de contraseñas seguras.

Tomando la seguridad en cuenta, es posible que se implementen accesos de ingeniería los cuales permiten hacer ajustes y actualizaciones a los dispositivos de control y protección de una forma remota y segura. También, permite el acceso a datos de un evento eléctrico para ser analizado, así como el acceso a otras funciones administrativas de cada uno de los dispositivos dentro de la red. Por lo regular, el acceso de ingeniería está restringido a un grupo muy selecto de ingenieros altamente entrenados y competentes para llevar a cabo la tarea con responsabilidad y certeza.

5.2 Red de comunicación típica para equipos de protección y control de potencia

La Figura 5-2 muestra un ejemplo típico de las diferentes tecnologías y protocolos que se utilizan para intercomunicar los dispositivos de protección y control de una subestación. Observe que la información puede fluir entre diferentes plataformas a través de medios diferentes como lo es: cobre, radio, fibra, etc.

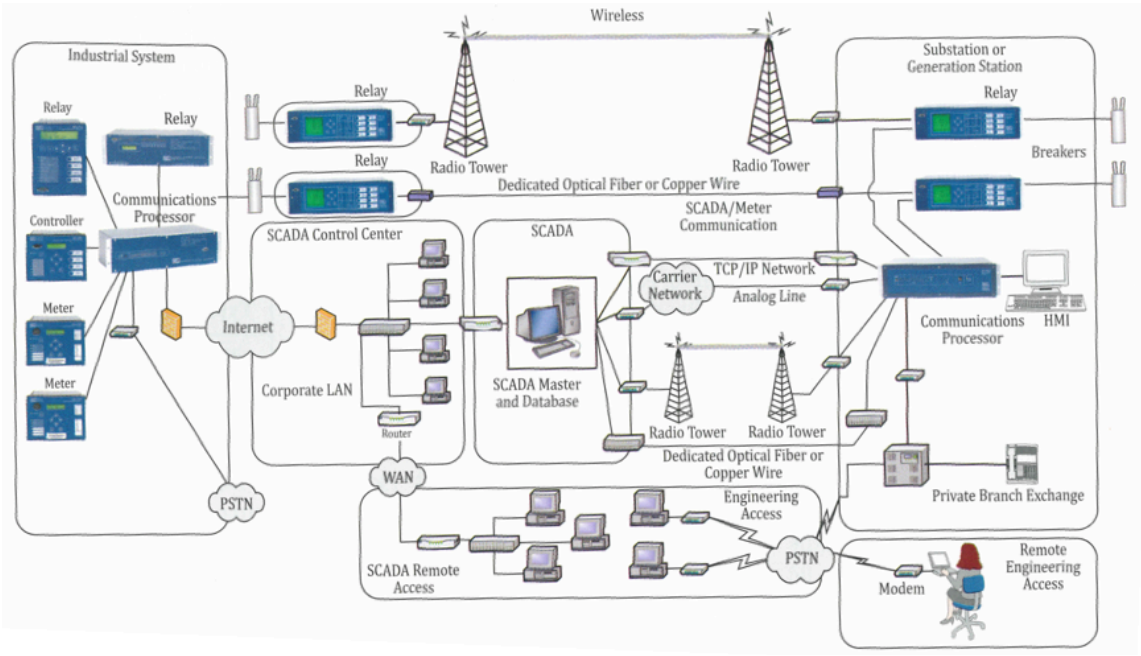


Figura 5-2. Red de comunicación típica en subestaciones eléctricas

5.3 Servicios de Informática en la nube

En la actualidad, la demanda por recursos y capacidad de procesamiento de datos ha incrementado y se ha visto que la alquilación de un servicio externo que ofrezca este procesamiento resulta ser mucho más económico que la implementación de una infraestructura de computo 100% privada. Los servicios de cómputo en la nube o *Cloud Computing* es una nueva tecnología que provee los recursos informáticos necesarios para realizar una tarea. De acuerdo con Idris y Shelly, *Cloud Computing* es una innovación en los servicios de internet que permite el uso de la tecnología de la información bajo demanda por lo que los beneficios que se obtienen de ello es la baja inversión en capital para poder obtener estos servicios [15].

De acuerdo con el autor Jadeja [16], *Cloud Computing* ofrece diversas ventajas en comparación con la inversión de una infraestructura privada. Entre ellas se encuentran:

i. Reducción en los trabajos administrativos de la plataforma

Debido a que la infraestructura, es decir, los servidores, *firewalls*, *routers*, sistemas de almacenamiento en línea y demás equipo de cómputo se encuentra fuera del sitio, los trabajos administrativos y de mantenimiento disminuyen.

i. Reducción de Costos

De la misma forma, al tener los recursos informáticos fuera del sitio en modalidad *outsourcing* los costos disminuyen ya que no se está adquiriendo el capital en sí, solamente se realizan los pagos de una tarifa que da derecho al uso del recurso. Además, los costos relacionados con el mantenimiento de la plataforma o del personal requerido para manejarla son totalmente eliminados del plan de trabajo.

ii. Servicio sin interrupciones

Al ser una infraestructura compartida y administrada por compañías especialistas en el área, los servicios de *Cloud Computing* tienen sistemas que permiten el funcionamiento de manera continua siendo muy remota la posibilidad de una falla en el servicio por parte del proveedor.

iii. Manejo y control de desastres

En el caso de una aplicación de almacenamiento en línea de datos o servicios de correo electrónico, con el simple hecho de tener la información fuera del sitio de trabajo se obtiene una ventaja en la protección de la información. Contando, además, que los mismos proveedores de servicios de *Cloud Computing* tienen una poderosa infraestructura que maneja apropiadamente la protección de los datos de los clientes ante un siniestro.

iv. Beneficios ambientales

Al utilizar una infraestructura de computo compartida, la cantidad de equipos que se tienen que desechar al fin del ciclo de vida de los mismos disminuyen drásticamente. También, se puede tomar en cuenta que el consumo eléctrico para el usuario por la ausencia de estos equipos dentro de sus instalaciones es de consideración y puede ser eliminado.

El uso de *Cloud Computing* para llevar a cabo las tareas de comunicación en una subestación eléctrica es una tecnología prometedora para aquellos usuarios que no deseen una infraestructura privada que realice estas tareas. No obstante, se debe de tomar en cuenta la importancia de limitar físicamente el sistema a que solamente realice operaciones de lectura y transmisión de datos y por ningún motivo, dado a cuestiones de seguridad e integridad en la operación de la subestación, se permita el control remoto de los elementos de la misma.

5.4 Interfaces de potencia y acondicionamiento de señales

El acondicionamiento de señal es el elemento más importante de un sistema de adquisición de datos ya que si no se optimiza la calidad de las señales provenientes de campo para el sistema que va a procesar y utilizar la información del sensor, no se pudiera entonces confiar en la lectura de la medición [17]. Algunos sensores, requieren de voltaje de excitación o de cierta preparación para que estos puedan funcionar apropiadamente; esto forma parte del sistema de acondicionamiento de datos. Antes de que la señal de un dispositivo pueda ser “digitalizada” esta debe de pasar por el acondicionamiento que modifica la señal para que se encuentre en las óptimas condiciones para que puedan ser procesadas por el sistema de adquisición. De acuerdo con Ashlock y Warren [17], algunas formas de acondicionamiento de señal se muestran a continuación y su uso depende de la aplicación y la señal especifica que se quiera tratar.

i. Amplificación

Los amplificadores incrementan el voltaje para estar dentro del rango de un convertidor análogo-digital, una consecuencia inmediata de la amplificación de la señal es que la resolución y sensibilidad de la misma se incrementa.

ii. Atenuación

La atenuación es lo opuesto a la amplificación, es necesario cuando los voltajes que se busca digitalizar están más arriba que el voltaje de operación del convertidor

análogo-digital y la ausencia del acondicionador puede dañar los componentes electrónicos del sistema de adquisición de datos.

iii. Filtros

La función de los filtros es la de rechazar el ruido no deseado dentro de un rango de frecuencia. Los filtros son usados también para atenuar todas aquellas señales por arriba de la frecuencia Nyquist junto con un filtro *anti-aliasing.*

iv. Aislamiento

Las señales de voltaje que estén por fuera del rango de operación del convertidor análogo digital o del sistema de adquisición de datos pueden dañar todos los componentes electrónicos del mismo e inclusive dañar al operador, por ello, el acondicionamiento basado en aislamiento es una opción que permite proteger a los componentes y al personal de voltajes peligrosos o picos de voltaje. Este tipo de acondicionamiento funciona en conjunto con la atenuación de la señal.

v. Excitación

Muchos sensores en campo requieren de voltaje de operación para su funcionamiento; algunos ejemplos: galgas, acelerómetros, termistores, RTDs, etc.

vi. Linealización

La linealización es necesaria cuando los sensores producen voltajes que no son lineales al elemento físico que está midiendo. Es, además, el proceso en el cual se interpreta la señal desde el sensor con la ayuda de un software.

Para el correcto desempeño del dispositivo de transmisión de alarmas en una subestación, es importante que se considere de forma apropiada el acondicionamiento de la señal. Ya que con ello se asegurará que las señales que se estén considerando sean analizadas adecuadamente sin provocar algún efecto en el cual exista una atenuación de la misma o provoque un daño físico en el acondicionador o en el circuito de control en el lado de campo.

6 Especificaciones Generales del Dispositivo

Esta sección tiene el propósito de especificar los requerimientos generales del sistema, su forma de operación y objetivos específicos que se deben de alcanzar. Se encuentra dividida en dos secciones, una de ellas exclusivas para requerimientos del firmware y otra dedicada a las especificaciones de hardware.

6.1 Requerimientos de firmware

El sistema de monitoreo para la subestación deberá de ser tal que pueda transmitir el estado de las alarmas en la subestación de forma periódica, en caso de encontrar una alarma o un problema en el equipo este deberá de poder transmitir un mensaje por el canal principal de comunicación y en su ausencia por una vía secundaria.

De forma general la manera en la que el firmware del dispositivo funciona es mostrado en la Figura 6-1 en forma de diagrama de flujo, las tareas más elementales son mostradas en forma de procesos y se centran en verificar el estado de conexión a la red y en verificar el estado general de la subestación a través del panel de alarmas.

El firmware debe de ser capaz de obtener la hora exacta mediante servidores NTP para poder llevar un registro del momento en el que los eventos en una subestación ocurren. Deberá también de ejecutar tareas que permitan además de la transmisión de un mensaje el cual idealmente es un correo electrónico.

Por otro lado, se deberá de llevar un registro en un documento de red del historial de eventos de la subestación y especificar una manera en la cual el sistema bajo condición de alarma pueda retornar a un modo de operación de monitoreo el cual imite la función de reset / reponer existente en anunciadores de alarmas en subestaciones.

Si el firmware detecta un problema de comunicación debido a la ausencia de conexión a la red o una falla en cualquier punto que se genere entre el equipo y el servidor de Cloud Computing este deberá de ser capaz de cambiar al canal de

comunicación de respaldo y transmitir las condiciones de la subestación; una vez que se restaure la conexión se deberá de notificar y regresar al estado normal.

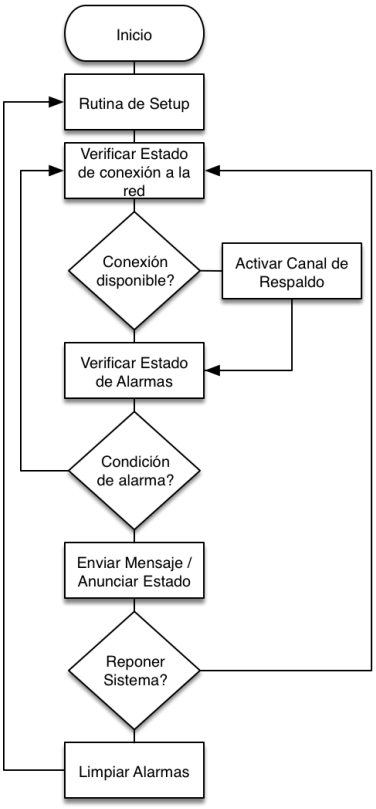


Figura 6-1. Diagrama general de procesos del dispositivo

Es de mencionar que el firmware conlleva más rutinas de verificación y acondicionamiento, así como de selección y construcción de mensajes más complejas y que será especificado en este mismo documento en la sección correspondiente.

6.2 Requerimientos de Comunicación

El dispositivo, bajo condición de alarma enviará un correo electrónico a una serie de contactos que puede ser especificado bajo una lista de distribución o programados directamente en el dispositivo. El mensaje deberá de proveer información detallada del evento en el que se especifique el detalle y estampa de tiempo sobre la alarma o alarmas en particular que se activaron en la subestación. Una representación gráfica del esquema de comunicación del dispositivo se muestra en la Figura 6-2.

El canal primario de comunicación es realizado mediante una conexión a Internet a través de una red ethernet cobre en el formato 10/100 BASE-T con un conector RJ-45.

El canal de respaldo que es activado solamente bajo la ausencia del canal primario o de forma manual es a través de mensajes de texto en una red GSM. La lista de contactos a los que se enviará mensaje deberá de ser programada en el dispositivo. Idealmente se busca que el equipo pueda responder a notificaciones push enviadas a través de esta vía, en otras palabras, que sea capaz de responder un mensaje de texto dando estatus de la subestación.

El envío de mensajes de texto como canal de comunicación de respaldo ofrece las siguientes ventajas:

i. Es una forma económica de comunicación
ii. Área de cobertura en lugares donde no hay conexión a Internet
iii. Dependiendo de las especificaciones del hardware no es necesaria una conexión fuerte a la red celular lo que evita la necesidad de colocar una antena externa en el dispositivo.
iv. La programación y conexión de la tarjeta celular es un canal independiente de comunicación directo al personal competente por lo que no se requiere de servidores externos para entregar el mensaje de alarma.
v. La comunicación con SMS demanda poca potencia por lo que fuentes de poder o acondicionadores especiales exclusivos para la tarjeta celular no son necesarios.

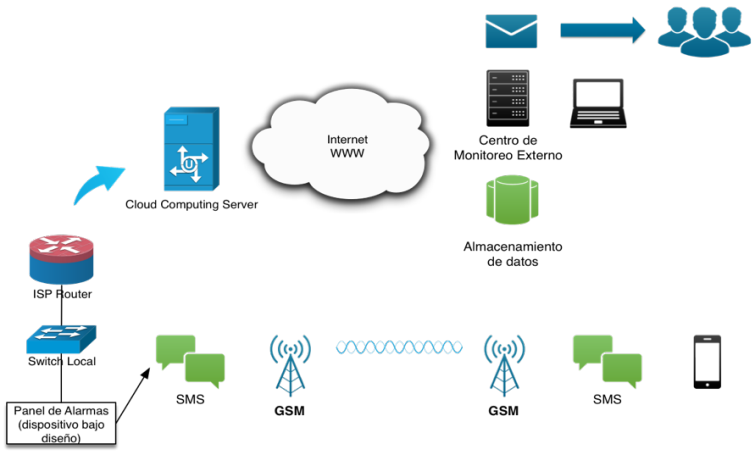


Figura 6-2. Diagrama de Comunicación Generalizado

6.3 Requerimientos de Hardware

Los requerimientos de hardware para el dispositivo deben de ser lo apropiados para poder interactuar con el circuito de control de una subestación eléctrica el cual, por lo regular opera a 125 VCD.

El diseño del dispositivo tiene que ser tal que no interfiera con el cuadro de alarmas existente en el cuarto de control, si es que está instalado ya que pudiera darse el caso que al generarse un evento y la señal de campo se active solamente un dispositivo de los dos (panel de alarmas y el dispositivo bajo diseño) pueda detectar la alarma, y lo que se busca es que ambos puedan lograrlo. De forma general, en la Figura 6-3 se muestra un diagrama de interacción entre los diferentes dispositivos que se deben de considerar para el diseño del dispositivo.

De manera particular se debe de considerar lo que el hardware cumpla con los siguientes puntos:

i. Las señales a monitorear deben de ser acondicionadas de 125VCD a lo que la unidad de control del dispositivo requiera para la adquisición de datos.
ii. Los acondicionadores de señal deberán de ser de alta impedancia de tal forma que no interfieran con otros dispositivos de control o protección que utilicen esa señal para sus funciones.
iii. Los acondicionadores tienen que ser aptos para detectar señales sostenidas y pulsos.
iv. El dispositivo deberá de contar con al menos la capacidad de monitorear 32 señales digitales referentes a alarmas de una subestación eléctrica
v. Al diseñar el dispositivo se tiene que considerar la integración de una protección de bloqueo por hardware o *watchdog* el cual permita reestablecer el dispositivo ante una falla en software.
vi. Es importante recalcar que en campo las señales pueden o no tener lógica negada por lo que se deberá de precisar el diseño por hardware o software para que contemple esa situación.

vii. El dispositivo bajo diseño tiene que considerar su montaje dentro de la subestación. Una manera de lograrlo es tomando en cuenta el estándar de montaje en rack de 19”.

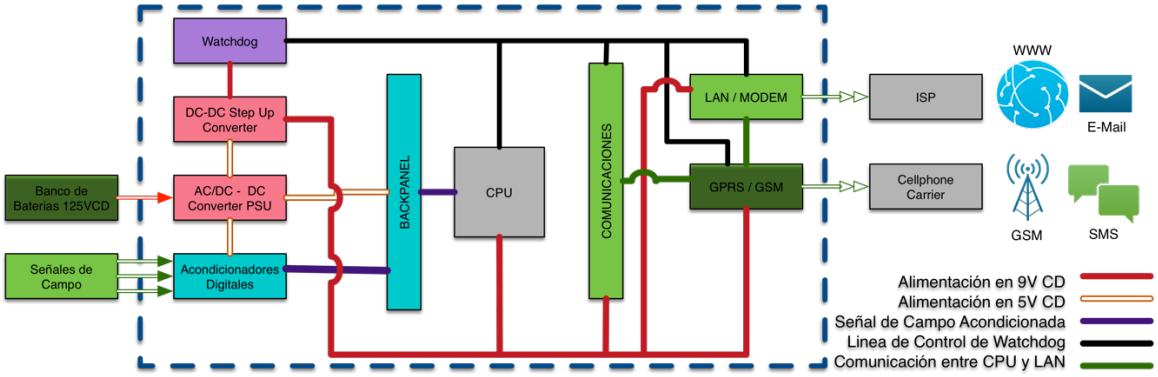


Figura 6-3. Diagrama de Interacción de Subsistemas del Dispositivo

En la sección correspondiente de este documento se precisarán los elementos de hardware seleccionados para el diseño del dispositivo que cumplen con los objetivos anteriormente listados.

7 Componentes

Esta sección tiene por objetivo dar una introducción a los componentes seleccionados para la integración del dispositivo de monitoreo para subestaciones, se presentará la diversidad de componentes y se justificará la razón por la cual se ha seleccionado como apto para el proyecto. Se omite para fines de este documento la inclusión de elementos pasivos.

7.1 Microcontrolador ATmega 2560

El microcontrolador seleccionado para la realización del prototipo es el ATmega 2560 por su facilidad de programación, bajo costo y confianza en operación. El controlador puede ser adquirido utilizando la tarjeta Arduino Mega Rev. 3. La Figura 7-1 muestra una imagen de la tarjeta y en la Figura 7-2 se muestra un diagrama del controlador [18]. El esquemático con conexiones detalladas se puede apreciar en el Apéndice A y en la Figura 7-3 dentro de esta sección. La tarjeta cuenta con las siguientes características de operación:

Microcontrolador:	ATmega 2560
Voltaje de operación:	5-12 VCD
Arquitectura:	AVR
Memoria Flash:	256KB
SRAM:	8KB
Velocidad de Reloj:	16MHz
E/S Analógicas:	16
EEPROM:	4KB
E/S Digitales:	54
Salidas PWM:	15

El Arduino Mega 2560 cuenta con tres diferentes formas de ser alimentado, la primera de ellas puede ser a través del puerto USB integrado; de forma externa a través de un convertidor AC-DC y un conector de 2.1mm, o directamente de una fuente regulada de 6 a 12 VCD [18]. Esta última es la que se recomienda para el

proyecto ya que la alimentación es directamente a través de un pin de la tarjeta y permitiría una conexión más apropiada y robusta ante muy pequeñas variaciones en el voltaje.

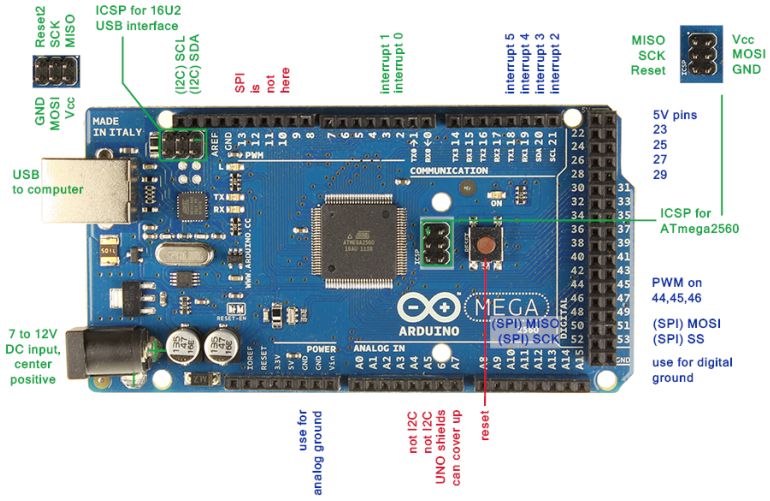


Figura 7-1. Arduino Mega 2560 Rev. 3

El Arduino Mega 2560 es apto para el proyecto porque además de ser versátil en la manera de alimentarse cuenta con los canales suficientes para monitorear la mayoría de las señales de alarma en una subestación eléctrica, sin embargo, estas solamente pueden ser de 5Vcd por lo que tienen que pasar por un acondicionamiento apropiado para su correcto funcionamiento y no poner en riesgo la integridad de las instalaciones o el equipo.

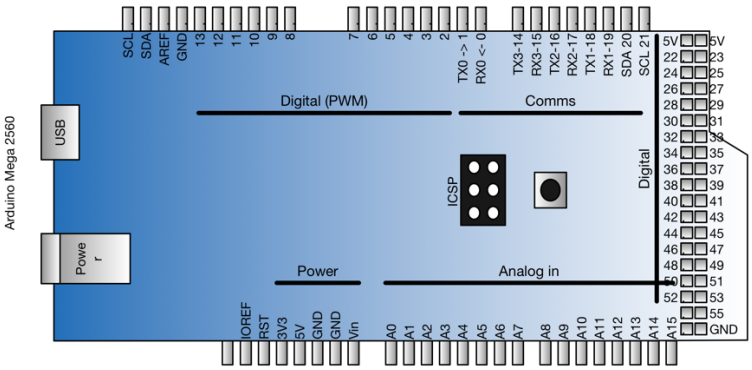


Figura 7-2. Pinout Arduino Mega 2560 Rev. 3

Por otra parte, la facilidad del entorno de programación y la manera en cómo integrar accesorios permiten que el proyecto sea flexible para poder interactuar en múltiples esquemas de comunicación como lo son a través de redes inalámbricas,

bluetooth, ethernet cobre, tecnología celular, ente otros; a través de tarjetas auxiliares que se ensamblan en la parte superior

No está demás que la tarjeta Arduino se puede programar a través de software *open source* que es distribuido a través del fabricante y la tarjeta viene con un *bootloader* que permite su programación sin requerir hardware adicional al microcontrolador directamente a través de comunicación serial por USB utilizando protocolo STK500. También la tarjeta permite la programación directa al microcontrolador mediante ICSP.

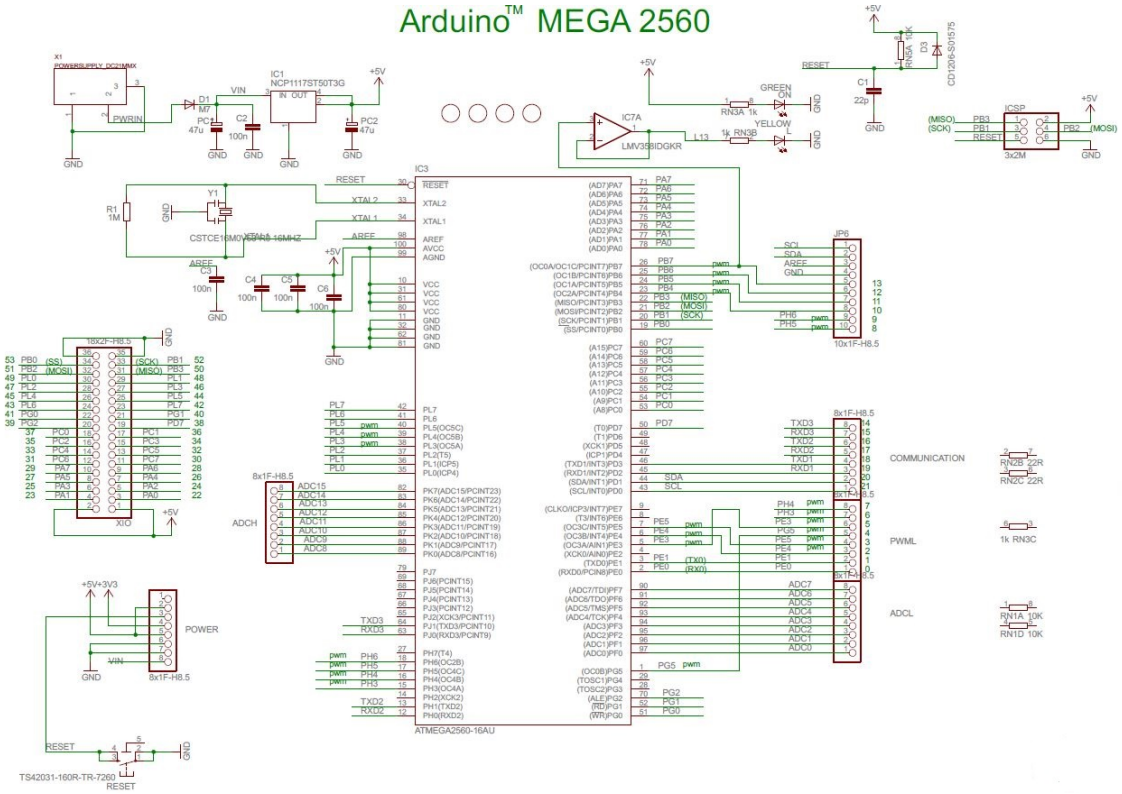


Figura 7-3. Diagrama de Conexión Arduino Mega 2560 Rev. 3

7.2 Microcontrolador Ethernet Wiznet W5100

Este dispositivo permite que el controlador se conecte a internet a través de un conector RJ-45. El dispositivo más apto para cumplir esta función tomando en cuenta que el Microcontrolador es un Atmel 2560 en un Arduino Mega es el *shield* del mismo fabricante con el micro Wiznet W5100; este controlador soporta protocolo TCP y UDP hasta en cuatro conexiones simultaneas. El controlador principal tiene la capacidad de comunicarse con la tarjeta ethernet a través del bus SPI [19]. En la

Figura 7-4 se muestra una imagen con el aspecto físico de la tarjeta [19] y en la Figura 7-5 se muestra el diagrama de bloques de la forma de operar del W5100 [20]. El apéndice A muestra el esquemático detallado de la tarjeta ethernet [19].

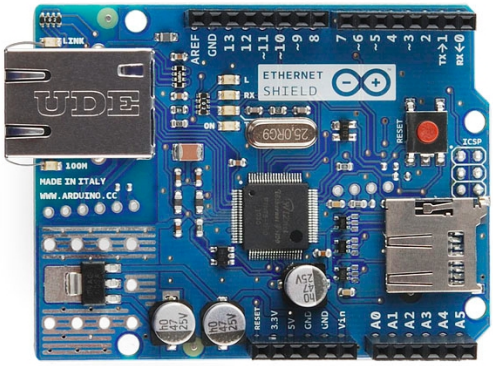


Figura 7-4. Controlador Arduino Ethernet Shield Rev. 3

De forma particular este elemento cuenta con las siguientes características:

Alimentación:	5Vcd (alimentados de la tarjeta principal)
Controlador:	Wiznet W5100
Buffer:	16Kb
Velocidad:	10/100 Mb
Comunicación:	SPI
Protocolos:	TCP, UDP, ICMP, IPv4 ARP, IGMP, PPPoE, Ethernet
Negociación:	Auto negociable Full y Half Dúplex

Además, contiene un slot para lectura y escritura en una memoria micro SD.

El Arduino Ethernet Shield es apto para el proyecto que aquí se documenta como comunicación primaria ya que la conexión por 10/100 BASE-T es muy confiable y estable al ser comparada con las redes inalámbricas que son sujetas a interferencias y son además vulnerables a ataques externos.

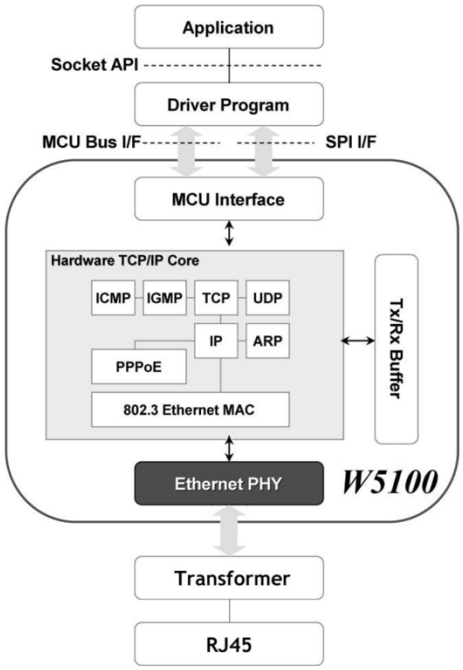


Figura 7-5. Diagrama de Bloques del Wiznet W5100

Esta tarjeta establece comunicación con el controlador principal a través del bus SPI el cual representan los pins 50, 51 y 52 por lo que se deberá de reservar el uso de los mismos y no utilizarlos para lectura de señales de campo.

No obstante, se debe destacar que para esta tarjeta en particular no se pueden ejecutar instrucciones al W5100 y al slot SD de forma simultanea ya que ambas utilizan SPI, la selección sobre si se desea mandar instrucciones al slot SD o el microcontrolador ethernet se realiza mediante los pins 10 y 4.

7.3 Modem Quectel M10 de Red Celular / GSM

Este componente tiene la intensión de proveer al proyecto un canal de respaldo de comunicación en caso de que el canal principal presente una falla. Se busca que pueda ser capaz de mandar mensajes de texto a personas elementales y también funcione como medio de comunicación para dar retroalimentación del estado actual en una modalidad bajo demanda. Es decir, que si en un momento determinado se desea conocer el estado actual de cuadro de alarmas el equipo tenga la capacidad de responder a un mensaje de texto con la información correspondiente.

Para ello con la arquitectura que se está proponiendo la opción más apropiada es utilizar la tarjeta GSM de Arduino, ésta en particular utiliza un modem del fabricante Quectel modelo M10 de cuatro bandas GSM/GPRS 850Mhz, 900MHz, 1800MHz y 1900MHz. Además, tiene la posibilidad de enviar y recibir SMS, realizar una conexión a la red a través de GSM y aunque no es proyectado que se utilice esta función también tiene la capacidad de contestar y realizar llamadas telefónicas. Una imagen de la tarjeta celular es mostrada en la Figura 5-1 [21] El esquemático de conexiones detalladas se encuentra en el apéndice A de este documento.

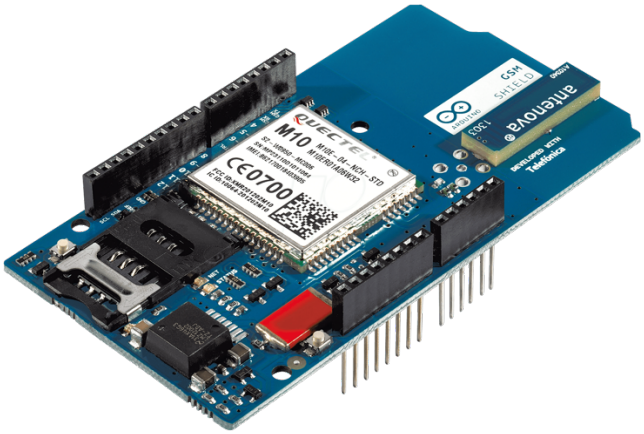


Figura 7-6. Arduino GSM Shield

Se debe de considerar que para su correcto funcionamiento el modem GSM debe de contar con una tarjeta SIM activada por algún proveedor de servicios autorizado. Un diagrama de bloques sobre la arquitectura del M10 se muestra en la Figura 7-7 [22].

De manera general, el GSM Shield cuenta con las siguientes características:

Modem:	Quectel M10
Bandas:	850/900/1800/1900 MHz
Ancho de banda:	85.6 kbps en GPRS
Corriente:	700mA a 1000mA (2A Max)
Alimentación:	3.3V a 4.6V

Esta tarjeta utiliza los pins 2 y 3 del puerto serial del controlador principal (ATMega 2560) para mandar instrucciones al M10.

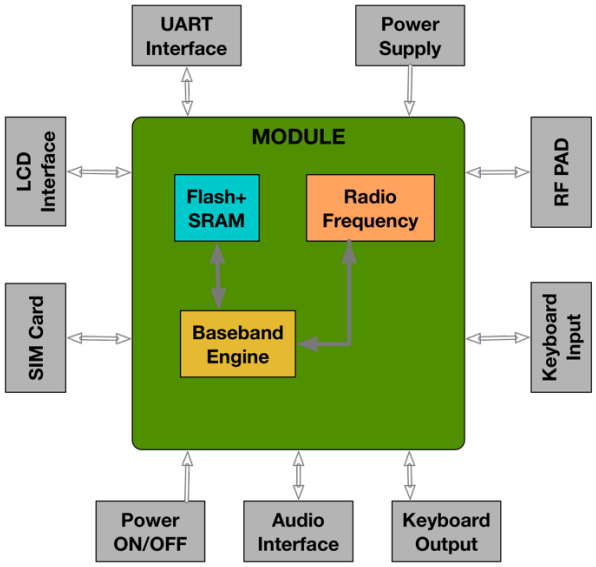


Figura 7-7. Diagrama de bloques del Quectel M10

7.4 Microcontrolador Pro Micro ATmega32U4

El microcontrolador ATmega32U4 embebido en la tarjeta Pro Micro del fabricante SparkFun es al momento el mejor dispositivo al menor costo para que cumpla como *Watchdog* del proyecto. Esta tarjeta ofrece una plataforma completamente independiente al Arduino Mega 2560 con el objetivo de detectar un *heartbeat* proveniente del controlador principal del dispositivo de monitoreo y la pérdida del mismo da indicación de un problema de software. Se busca que con esta tarjeta al detectar la perdida de la señal del *heartbeat* reinicie por hardware la tarjeta y recuperar las funciones del dispositivo de forma automatizada.

Este controlador es programado en el mismo entorno que el ATMega 2560 y es de tamaño reducido por lo que se puede localizar fácilmente dentro del gabinete del dispositivo de monitoreo.

La Figura 7-8 muestra una imagen del aspecto físico de la tarjeta con el controlador ATmega32U4 [23]. Por otra parte, el fabricante Atmel nos provee en su website el diagrama funcional de bloques de este controlador particular mostrado

en la Figura 7-9 [24]. El diagrama esquemático de conexiones de la tarjeta en cuestión puede ser localizada en el Apéndice A de este documento.

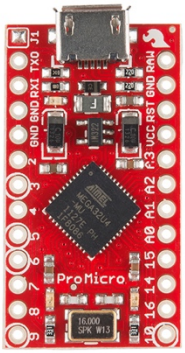


Figura 7-8. Tarjeta Pro Micro 5V/16MHz ATmega32U4

De forma particular esta tarjeta y controlador cuentan con las siguientes características por parte del proveedor:

Microcontrolador:	ATmega32U4
Arquitectura:	AVR
Alimentación:	5Vcd
Velocidad de Reloj:	16MHz
Programación:	Micro USB 2.0
Memoria Flash:	32kB
EEPROM:	1kB
SRAM:	2.5kB
ADC Pins:	4x 10 bits
E/S Digitales:	12x (5x PWM)
Puertos Seriales:	1x

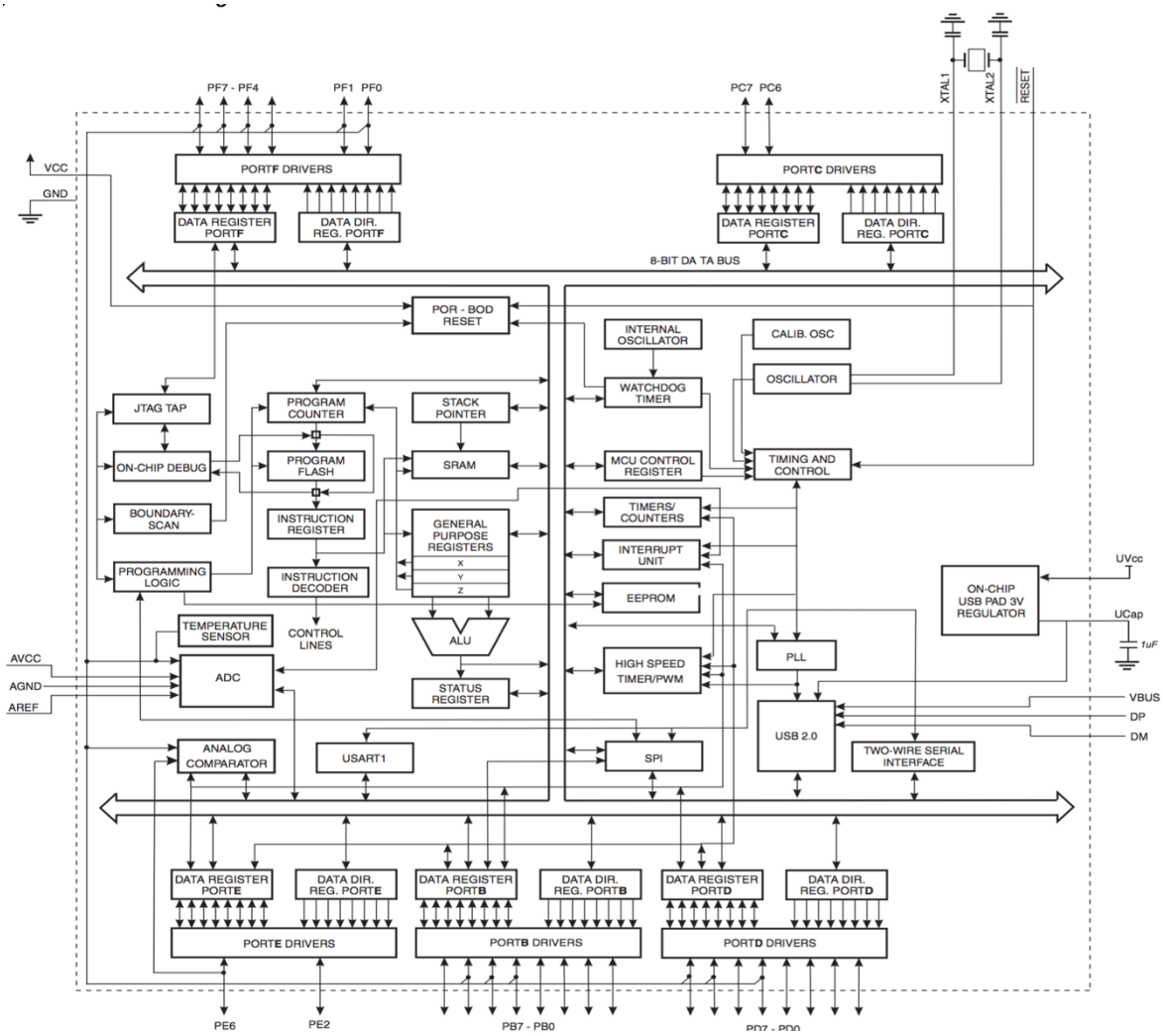


Figura 7-9. Diagrama funcional de bloques de ATmega32U4

7.5 Convertidor DC-DC SEL-9321

El convertidor SEL-9321 mostrado en la Figura 7-10 es una fuente de alimentación diseñada para acondicionar alto voltaje de DC proveniente de los bancos de baterías en subestaciones eléctricas y alimentar apropiadamente equipo electrónico o dispositivos de comunicaciones. Este elemento ha sido seleccionado como apropiado para el proyecto ya que puede proveer de energía a los microcontroladores tomando como fuente de alimentación 125 Vcd.

Figura 7-10. Convertidor DC-DC SEL-9321

Cabe destacar que este equipo puede proveer 5Vdc y 10Vdc tanto del banco de baterías como una fuente de alimentación en AC. De acuerdo con el fabricante, este dispositivo cumple con los estándares de la IEEE C37.90, IEC 60255 e IEEE 1613 [25] por lo que lo hace un dispositivo robusto y confiable para la alimentación eléctrica del proyecto de monitoreo. El diagrama de conexión se muestra en la Figura 7-11. De manera particular el equipo cuenta con las siguientes características:

Alimentación:	24Vdc, 48/125 Vdc o 125Vac, 125/250 Vdc o Vac
Burden:	<11W
Output:	+5Vdc (4.75-5.25 V, 100mA a 1.0 A)
	±10Vdc (8.5-11,5 V, 10 mA a 100mA)
Temperatura de Operación:	-40º a +85º C
Humedad:	5 a 95% No condensada

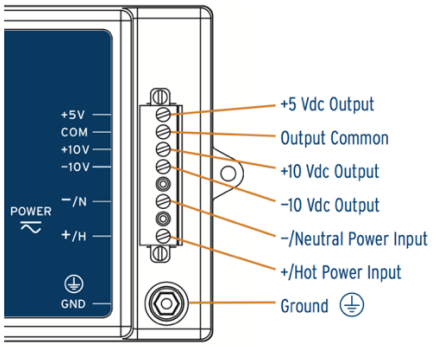


Figura 7-11. Diagrama de conexión SEL-9321

Se debe de considerar que para este proyecto las tarjetas de los microcontroladores enfocados a los servicios GSM, ethernet, watchdog y el micro principal deben de ser alimentados con en un rango de 6-12Vdc para garantizar que aun con las caídas de voltaje internas propias de los circuitos de cada una de ellas el microcontrolador obtenga 5Vdc; ya que la corriente que los equipos utilizan sobrepasa los 100mA, utilizar la fuente de ±10Vdc no es una opción viable. Esto significa que se deberá de colocar un convertidor DC-DC que nuevamente eleve el voltaje de 5Vcd a 9Vcd y poder aprovechar al máximo la capacidad en corriente de la fuente de poder SEL-9321.

7.6 Convertidor DC-DC Step-Up XL6009

Dado a que la fuente de 10 Vdc SEL-9321 se encuentra limitada en corriente, se ha optado por colocar un segundo convertidor DC-DC adicional a la fuente SEL-9321 con el objetivo de levantar el voltaje de 5Vdc a 9Vcd, pero ahora con un rango de corriente ampliado hasta 1A. El convertidor seleccionado para cumplir con esta tarea es el XL6009 del fabricante Kylinchip Electronic Co, Ltd. El cual es un convertidor de rango ampliado capaz de generar voltaje positivo o negativo; de acuerdo a la hoja de datos del dispositivo tiene la flexibilidad de trabajar como convertidor tipo *boost, flyback,* o *inverso*.

Para el caso del proyecto de monitoreo de subestaciones se utilizará el tipo *boost*. Un diagrama de bloques sobre el funcionamiento del XL6009 se muestra en la Figura 7-12 [26].

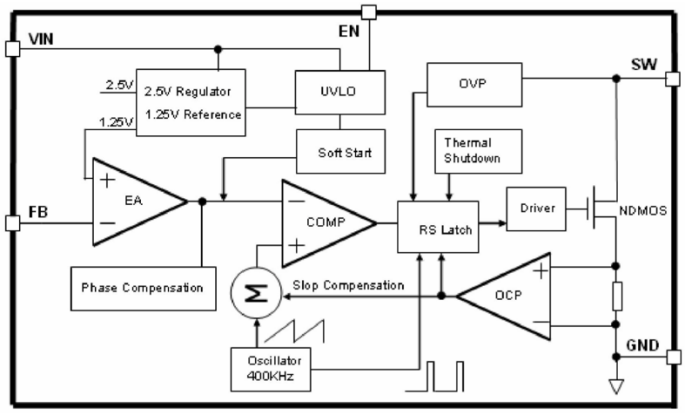


Figura 7-12. Diagrama de bloques del XL6009

La hoja de datos también informa que se cuentan con las siguientes características:

i. Rango de voltaje de 5 a 32Vcd.
ii. Voltaje de salida positivo o negativo.
iii. Frecuencia de conmutación a 400kHz.
iv. Protección de sobre voltaje.
v. Función para limitar corriente incluida.
vi. Protección térmica.
vii. Eficiencia del 94%.

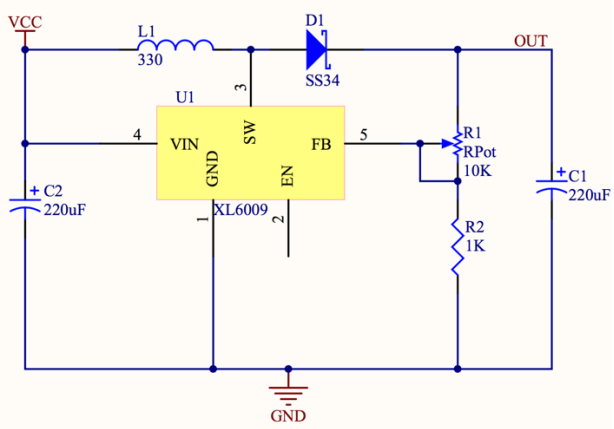


Figura 7-13. Circuito de conexión del convertidor DC-DC confugurado en boost

Por lo regular es sencillo encontrar este convertidor en el mercado en la configuración de *boost* con el circuito que se muestra en la Figura 7-13 y que además es la forma de conexión sugerida por el fabricante. La forma en la que comercialmente se encuentra el convertidor DC-DC *Step Up / boost* de esta tarjeta se puede apreciar en la Figura 7-14 [27].

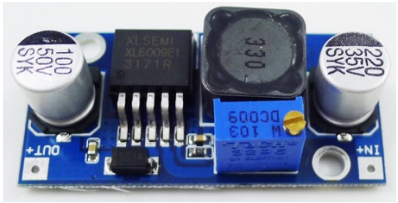


Figura 7-14. Convertidor DC-DC boost basado en el XL6009

7.7 Módulo de Aislamiento y Acondicionamiento Digital

Para poder convertir una señal proveniente de campo que trabaja en 125Vcd es necesario realizar un acondicionamiento eléctrico a 5Vcd y que también aísle y proteja el equipo de comunicación y control. Para ello se ha optado por seleccionar los módulos de aislamiento y acondicionamiento digital de la compañía Dataforth Inc. quien ofrece una gama de interfaces de potencia y acondicionamiento confiable ya que proveen una barrera de aislamiento efectiva a 4kV. Para el proyecto se han seleccionado los acondicionadores de la serie SCMD por su reducido tamaño, estos dispositivos están diseñados para recibir señales de encendido y apagado a una computadora a través de módulos de estado sólido [28].

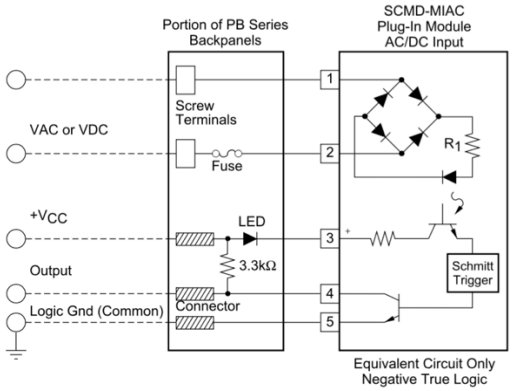


Figura 7-15. Diagrama funcional de bloques del SCMD-MIAC5

Cabe destacar que estos módulos son de alta impedancia por lo que al colocarlos paralelo a otro sistema de monitoreo no afecta y tampoco interviene en las funciones del otro. Un diagrama de bloques donde se detalla la arquitectura del dispositivo es otorgado por el fabricante y mostrado en la Figura 7-15. De manera adicional una fotografía del módulo es presentada en la Figura 7-16. Entre la variedad de modelos de acondicionadores en el catálogo del fabricante se optó por seleccionar el SCMD-MIAC5.

Figura 7-16. Módulo de acondicionamiento digital SCMD-MIAC5

Entre las características de este componente se encuentran:

Alimentación:	5Vcd
Salida de Voltaje:	5Vcd (OFF) 0Vcd (ON)
Aislamiento Óptico:	4000Vrms
Entrada de Voltaje:	80 a 140 Vcd o Vac
Temperatura de Operación:	-30 a +80 ºC
Pinout Estándar Industrial	

7.8 Back panel para Módulos de Acondicionamiento Digital

Para instalar todos los módulos de acondicionamiento de forma apropiada se sugiere la adquisición de un *back panel* que permita la fácil conexión y desconexión de los dispositivos que acondicionan la señal de campo. La tarjeta SCMD-PB16TSMD ofrece la posibilidad de conectar módulos de la serie SCMD en 16 canales de comunicación independientes. En la Figura 7-17 se muestran las dimensiones y aspecto del accesorio y en la Figura 7-18 el diagrama esquemático del mismo [29].

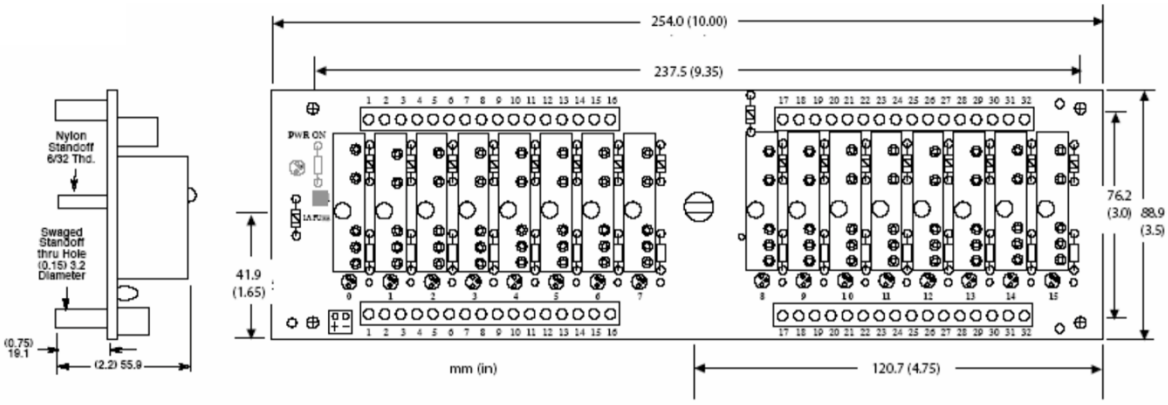


Figura 7-17. Back panel SCMD-PB16TSMD

Entre las características principales del accesorio se encuentran las siguientes:

i. Contiene Resistencia *pull up* en el lado de la señal acondicionada de 3kΩ
ii. Fusible de protección de 5 Amp en cada canal reemplazable por el usuario
iii. LEDs para indicar el estatus de operación del canal
iv. Voltaje de operación de 5 o 24 Vdc

v. Módulos de acondicionamiento de entrada y salida pueden coexistir en un mismo back panel.

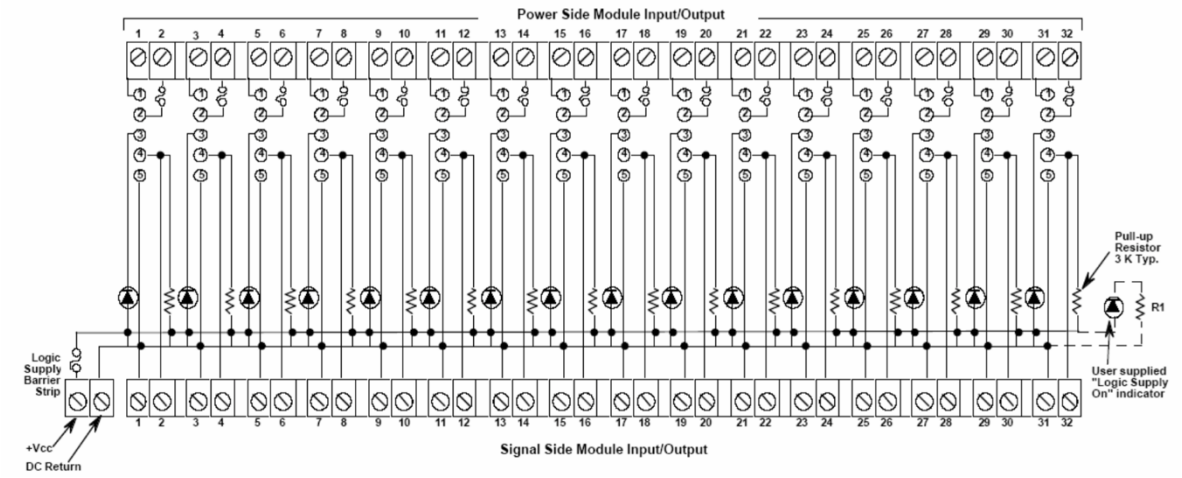


Figura 7-18. Esquemático de Back panel SCMD-PB16TSMD

8 Canal de Comunicación

El propósito de esta sección es mostrar al lector los servicios de Internet que se deben de mantener activos para que el dispositivo de monitoreo de subestaciones satisfactoriamente pueda entregar mensajes de alarma. Se describirá cada uno de estos servicios y las funciones en particular que cumplen. Recordemos que dentro de los objetivos del proyecto se encuentra crear un historial de eventos, así como transmitir el mensaje de alarma a personal competente.

El canal de comunicación propuesto para lograr estos objetivos se encuentra representado en la Figura 8-1 note que se utilizan los servicios de Temboo para ejecutar rutinas que permiten llevar a cabo actividades como lo son registro de datos en la Nube, creación de documentos, envío de correo electrónico, entre otros.

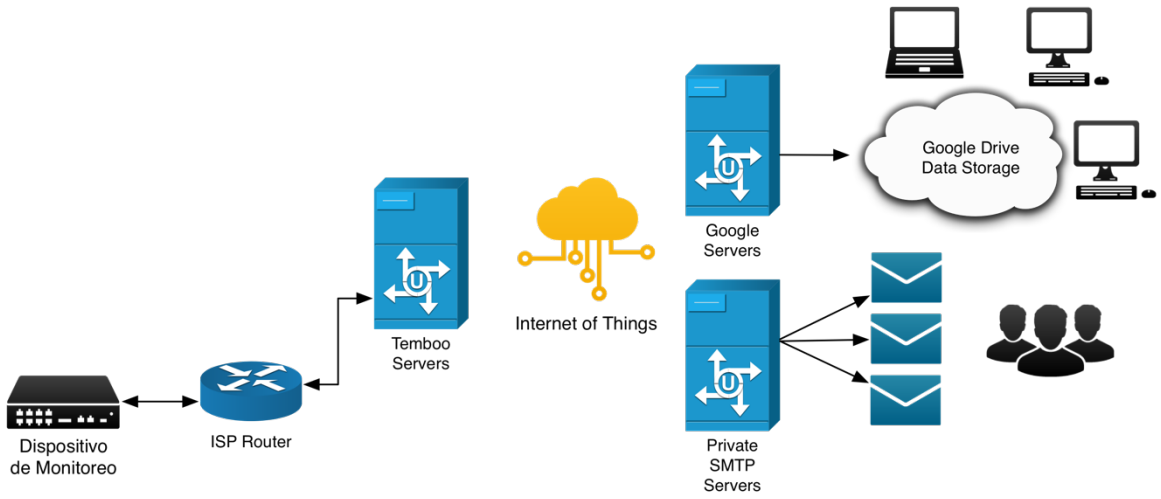


Figura 8-1. Esquema de Comunicación Primario usando Cloud Computing

El dispositivo de monitoreo para subestaciones tiene rutinas para diagnosticar el estado de la red en toda la ruta desde el origen hasta los servidores del proveedor de Cloud Computing. En el momento en que una falla en la conexión es detectada inmediatamente se establece una comunicación provisional de respaldo utilizando tecnología celular para enviar reportes de alarmas vía mensajes de texto, la representación de esta condición se puede apreciar en la Figura 8-2 a continuación.

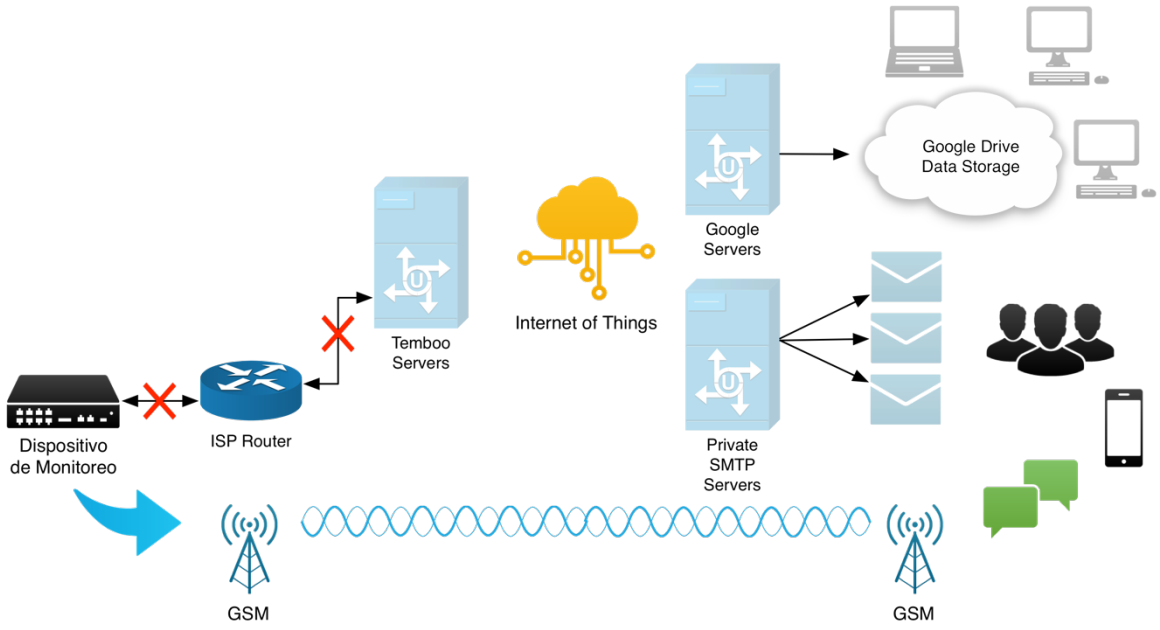


Figura 8-2. Esquema de Comunicación de Respaldo usando SMS

8.1 Servicio de Cloud Computing – Temboo

Temboo es una compañía dedicada a la ejecución de rutinas y funciones de software en línea, llamadas por ellos *choreos*, el nombre viene de coreografía donde son rutinas ya establecidas que pueden llamar a ejecutarse un número indefinido de veces por distintos usuarios en la red. Cuando un usuario desea que los servidores de Temboo ejecuten una tarea, el cliente solamente debe de invocar mediante una rutina de software la función que se deba de ejecutar y declarar todas las variables requeridas para que esto se pueda llevar a cabo.

Al ejecutarse el *choreo* en el lado del cliente, el servidor procede a revisar que la función que se solicita exista y se tengan todos los datos para poder ejecutarla, de lo contrario, se envía un código de error; al ejecutarse la tarea de forma satisfactoria se envía un código de no-error. De esta manera se puede simplificar el código de firmware del dispositivo de monitoreo para subestaciones y reducir la cantidad de hardware y capacidad de computo requerida para cumplir con las funciones requeridas.

Entre las capacidades de Temboo se pueden encontrar rutinas que se conectan con otros proveedores de servicios como WolframAlpha, Google, Dropbox,

Microsoft, Yahoo y solicitar a ellos a través de APIs que se ejecuten rutinas adicionales como realizar una búsqueda en internet, modificar un archivo que se encuentre localizado en un servidor de almacenamiento de datos, enviar correos electrónicos, mensajes de texto a celular, entre otras.

El proveedor de servicios Temboo también ofrece librerías con rutinas tradicionales como lo es validación de datos, obtención de hora global, envío y recepción de correos electrónicos a través de servidores SMTP y POP, generación de números aleatorios, entre otros.

Con referencia a los mecanismos de seguridad de Temboo, la compañía cuenta con las siguientes medidas implementadas para garantizar información referente al cliente que ejecuta las rutinas:

i. Todos los datos transmitidos desde el cliente al servidor son encriptados mediante TLS, el cual es un protocolo criptográfico que proporcionan comunicaciones seguras por una red mediante el uso de certificados y llaves que permite autentificar ambas partes que se estén comunicando [30].
ii. Todos los datos almacenados en Temboo, entre ellos los relacionados a las credencial e información sensible de las aplicaciones son encriptados a través de AES el cual es un esquema de cifrado por bloques y es el estándar más popular de cifrado y establecido por el NIST en el 2001 [31].
iii. Aislamiento de procesos o *sandboxing,* el cual es un mecanismo de seguridad para separar programas o rutinas que corren dentro de un sistema aislados de tal forma que su ejecución no perjudica ni pone en riesgo el equipo donde la rutina se ejecuta o su sistema operativo [32]

El utilizar o implementar rutinas de Temboo dentro del firmware del dispositivo de monitoreo para subestaciones ofrece las siguientes ventajas [33] lo cual lo hacen atractivo:

i. Reducción en componentes de hardware y capacidad de procesamiento al requerir solamente de un microcontrolador u *open-source hardware* que permita ser programado a través de un IDE utilizando software abierto u *open-source software* [34] con una conexión a la red.

ii. Protección de datos sensibles al utilizar encriptación TLS y AES, aislamiento de datos en servidores dedicados de Temboo y aislamiento de procesos por parte del proveedor de servicios.
iii. Utilizar Temboo ofrece estabilidad ya que es una arquitectura de programación modular que constantemente está en un proceso de mejora continua disminuyendo los tiempos muertos o *downtime* del servidor mediante el uso sistemas redundantes y monitoreo activo.
iv. Algunas de las rutinas de Temboo pueden ser modificadas en línea de tal forma que no se tiene que reprogramar el dispositivo en campo, a esto se le conoce como *Cloud Reprogramming.*
v. Es un proveedor de servicios que ofrece *Future Proof* ya que, aunque los APIs de los servicios de almacenamiento en nube (Google Drive, Dropbox), utilidades de envío de correos (Gmail, Hotmail, SMTP) cambien; las rutinas de Temboo son actualizadas automáticamente en sus servidores sin la necesidad de reprogramar los dispositivos en campo.
vi. Ofrece la posibilidad de programar rutinas o *choreos* personalizadas en línea a través de una herramienta de Temboo, lo que permite hacer aplicaciones específicas a las necesidades del proyecto.

Entre las rutinas o *choreos* que ofrece Temboo, las que se utilizaron para la elaboración de este proyecto se muestran a continuación con su descripción:

8.1.1 Prueba de comunicación a servidor

Esta rutina permite ejecutar una prueba de comunicación directamente al servidor de Temboo. Si el equipo cuenta con conexión a internet que se encuentre activa podrá ejecutar satisfactoriamente el *choreo,* cuando la rutina es ejecutada regresa un código definido por el usuario en el *dashboard* principal de Temboo, en caso de que el dispositivo detecte un código diferente al definido o simplemente no lo reciba se considera que la prueba de comunicación ha fallado y se tomaran las medidas de software correspondientes para hacer frente a la situación.

La Tabla 8-1 muestra los parámetros de entrada y salida de la rutina que requieren ser declarados en el firmware del dispositivo para que esta pueda ejecutarla satisfactoriamente.

Nombre de la rutina			Utilities.Test.HelloWorld
Variable	E/S	Tipo	Descripción
Value	Entrada	String	Un valor de prueba opcional que arroja la rutina al ser ejecutada satisfactoriamente
Result	Salida	String	Contiene una salida predefinida por Value al ejecutar la rutina, en caso de no ser especificado este valor la rutina regresa “Hello, world!”

Tabla 8-1. Formato de Rutina de Prueba de Comunicación

Esta rutina es ejecutada de forma periódica por el dispositivo de monitoreo de subestaciones para detectar algún problema de comunicación.

8.1.2 Envío de correo electrónico SMTP

Esta rutina permite que se envíen correos especificando el servidor SMTP, para el caso del dispositivo de monitoreo de subestaciones, esta rutina es de las más elementales para llevar a cabo la notificación de problemas o alarmas en la subestación hacia el personal competente de la misma. La rutina es ejecutada de forma periódica cada 24 horas para notificar que la subestación se encuentra en orden y ante cada evento que se registre en el dispositivo, ya sea una alarma dado un problema eléctrico o se reestablezca el panel de alarmas a través de la función RESET.

La Tabla 8-2 muestra los parámetros de entrada y salida de la rutina que requieren ser declarados en el firmware del dispositivo para que esta pueda ejecutada satisfactoriamente, también indica aquellos parámetros que son considerados opcionales

Nombre de la rutina			Utilities.Email.SendEmail
Variable	E/S	Tipo	Descripción
Password	Entrada	pwd	Contraseña de la cuenta de usuario para enviar correos.
Username	Entrada	String	El nombre de usuario de la cuenta para enviar correos.

Nombre de la rutina			Utilities.Email.SendEmail
Variable	E/S	Tipo	Descripción
FromAddress	Entrada	String	La dirección de correo electrónico de donde se envían los correos
MessageBody	Entrada	String	El contenido del mensaje del correo electrónico
Port	Entrada	Integer	Se especifica el número de puerto de la conexión en el cual se va a escuchar el protocolo, por lo regular 25 o 465
Server	Entrada	String	El nombre o dirección IP del servidor de correos
Subject	Entrada	String	El asunto del correo electrónico
ToAddress	Entrada	String	Las direcciones de correo electrónico a las cuales se les enviará el mensaje
Attachment	Entrada	String	[Opcional] Contenido codificado Base64 del archivo a adjuntar en el correo electrónico.
AttachmentName	Entrada	String	[Opcional] El nombre del archivo adjunto al correo electrónico
AttachmentURL	Entrada	String	[Opcional] Dirección URL de un archivo hospedado en algún servidor para adjuntar al correo electrónico
BCC	Entrada	String	[Opcional] Direcciones de correo electrónico a las que se les envía el mensaje en copia oculta
CC	Entrada	String	[Opcional] Direcciones de correo electrónico a las que se les envía el mensaje en copia.
UseSSL	Entrada	Boolean	[Opcional] Definición sobre si se desea realizar una conexión SSL al servidor.
Success	Salida	Boolean	Indica el resultado de la operación SMTP donde arrojara un valor True en caso de ejecutarse de manera satisfactoria

Tabla 8-2. Formato de Rutina de envío de correo SMTP

8.1.3 Actualizar renglón en hoja de cálculo

Esta rutina utiliza el API de Google para ver y actualizar documentos y en específico hojas de cálculo en el servicio de almacenamiento en línea *Google Drive.* La mecánica en la que funciona este procedimiento consiste en proveer el nombre de la hoja de datos, el renglón en particular que se debe de trabajar y los datos que se desean colocar separados por comas y al correr la rutina debe de dar un código de ejecución satisfactoria o errónea en caso de existir problema alguno.

Esta rutina es utilizada periódicamente por el dispositivo de monitoreo de subestaciones para actualizar el estado de las señales que monitorea y poder saber si en un momento presente existe o no la presencia de alguna alarma; también es una forma de diagnosticar la conexión a la red del sistema ya que entre los datos que actualiza también se incluye una estampa de tiempo. En caso de pérdida de

comunicación la unidad deja de actualizar la hoja de cálculo dejando un rastro del último momento en que lo hizo.

Se debe de recalcar que esta rutina hace uso de servicios que son proveídos por un tercero diferente a los servidores Temboo o al cliente en el dispositivo de monitoreo, en este caso, Google por lo que cada vez que esta rutina se ejecuta se debe de llevar un proceso de autenticación conocido como OAuth el cual protege la información que se transmite e intercambia entre Temboo y Google para su almacenamiento en línea. OAuth, es un estándar de autorización abierto y es una manera sencilla en la que se pueden identificar usuarios mediante el uso de aplicaciones con servicios de terceros sin tener que exponer sus credenciales [35]

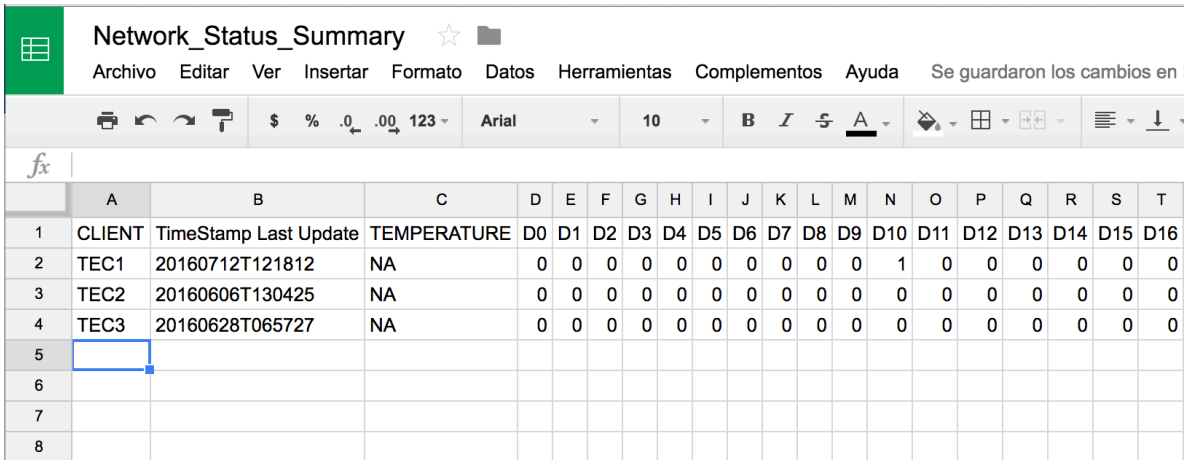
La Tabla 8-3 que se muestra a continuación indica los parámetros que se requieren especificar en la rutina de Temboo para que pueda ser ejecutada satisfactoriamente y se actualice la información de un documento ubicado en Google a partir de los datos generados por el cliente (dispositivo de monitoreo).

Nombre de la rutina			Google.Spreadsheets.UpdateRow
Variable	E/S	Tipo	Descripción
ClientID	Entrada	String	Credencial o Token que Google genera para identificar la aplicación del usuario en los servicios proveídos por Google.
ClientSecret	Entrada	String	Credencial o Token que Google genera para identificar el proceso del usuario en los servicios proveídos por Google
RefreshToken	Entrada	String	Un Token OAuth que se genera para tramitar un nuevo Token de Acceso general una vez que los originales expiran,
Row	Entrada	Integer	El número de renglón a actualizar, para actualizar el renglón este debe de existir previamente. El renglón 1 es considerado como encabezado y no puede actualizarse.
RowData	Entrada	String	Un string de datos separados por comas para actualizar en cada una de las celdas del renglón Row.
SpreadsheetKey	Entrada	String	ID único que asocia la hoja de cálculo en Google Drive con el renglón que se desea actualizar.
WorksheetID	Entrada	String	ID único que asocia el renglón con la hoja dentro del libro de cálculo de Google Drive
AccessToken	Entrada	String	[Opcional] Un Token OAuth sin expiración que identifica al usuario, la aplicación y el proceso. En caso de declararse, no es necesario proveer ClientID, ClientSecret ni RefreshToken
ResponseFormat	Entrada	String	[Opcional] El formato de la respuesta de Google al ejecutar la rutina. Seleccionar entre XML y JSON.

Nombre de la rutina			Google.Spreadsheets.UpdateRow
Variable	E/S	Tipo	Descripción
SpreadSheetName	Entrada	String	[Opcional] El nombre del libro de cálculo a actualizar. En caso de declararse, no es necesario proveer SpreadsheetID.
WorksheetName	Entrada	String	[Opcional] El nombre de la hoja del libro de cálculo a actualizar. En caso de declararse no es necesario proveer WorksheetID
NewAccessToken	Salida	String	Nuevo token de acceso al proveer el RefreshToken
Response	Salida	String	Respuesta de Google con código de ejecución satisfactoria o error

Tabla 8-3. Formato de Rutina de actualización de documentos en Google Drive

En la Figura 8-3 se muestra la hoja de cálculo donde la rutina UpdateRow de Temboo actualiza los datos cada vez que el dispositivo de monitoreo lo solicita. Cabe mencionar que múltiples clientes o dispositivos de monitoreo pueden estar trabajando en el mismo documento al mismo tiempo.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	CLIENT	TimeStamp Last Update	TEMPERATURE	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16
2	TEC1	20160712T121812	NA	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
3	TEC2	20160606T130425	NA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	TEC3	20160628T065727	NA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5																				
6																				
7																				
8																				

Figura 8-3. Resumen de datos en línea de dispositivos de monitoreo de subestaciones

8.1.4 Anexar renglón en hoja de cálculo

Esta rutina utiliza el mismo API de Google para la manipulación de documentos en línea, a diferencia del anterior este *choreo* en particular anexa una nueva fila de datos en la hoja de cálculo predefinida en las variables de entrada de la función. Al igual que en otras funciones se utiliza el método de OAuth para la autenticación de la aplicación que accede a los datos almacenados en línea.

En el caso del dispositivo de monitoreo de subestaciones esta función es elemental si se desea llevar un registro o historial de eventos que se desee guardar en línea y en una ubicación externa a la subestación. Esta función es ejecutada en los servidores de Temboo cada vez que se genera una alarma o el sistema es reestablecido manualmente por el cliente al hacer un RESET en el panel de alarmas.

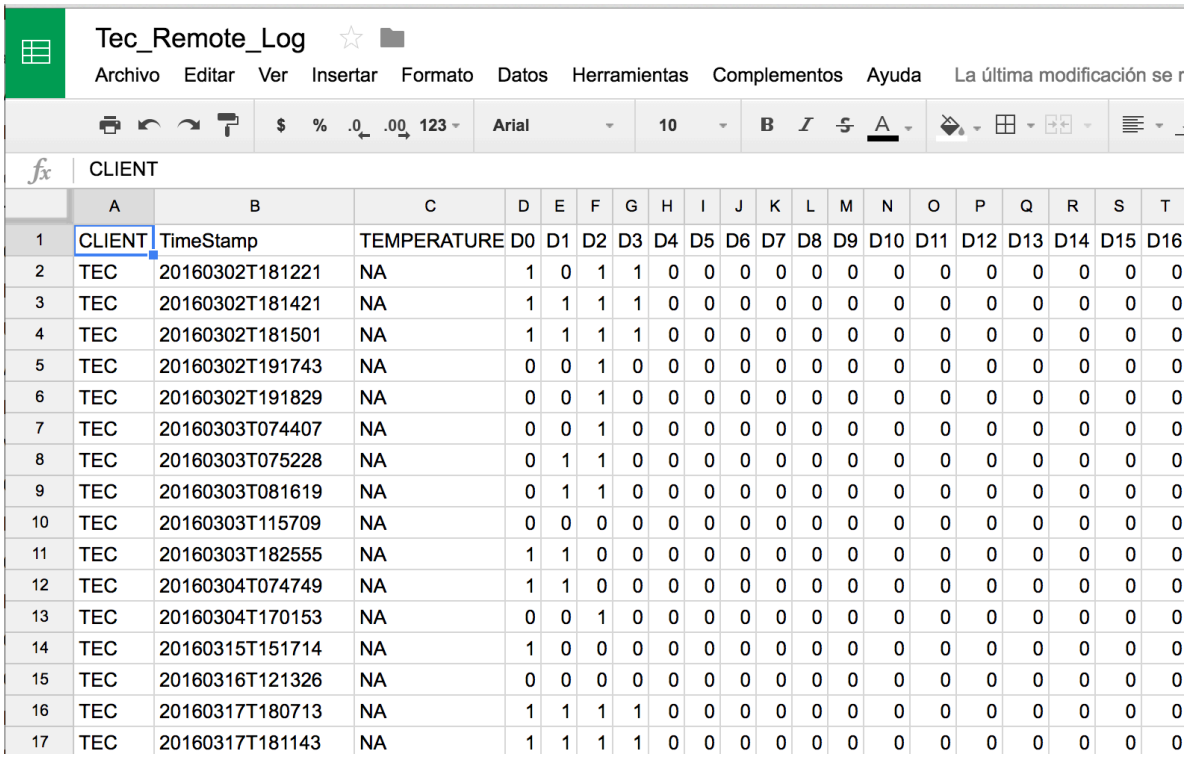
En la Tabla 8-4 se indican los parámetros de entrada a la rutina de Temboo que son necesarios para ejecutar la función y se pueda crear una línea con los datos de la alarma generada en un archivo que funcione como historial para todos los eventos que fueron detectados por el dispositivo de monitoreo de subestaciones.

Nombre de la rutina			Google.Spreadsheets.AppendRow
Variable	E/S	Tipo	Descripción
ClientID	Entrada	String	Credencial o Token que Google genera para identificar la aplicación del usuario en los servicios proveídos por Google.
ClientSecret	Entrada	String	Credencial o Token que Google genera para identificar el proceso del usuario en los servicios proveídos por Google
RefreshToken	Entrada	String	Un Token OAuth que se genera para tramitar un nuevo Token de Acceso general una vez que los originales expiran,
RowData	Entrada	String	Un string de datos separados por comas para crear el nuevo renglón del documento definido en SpreadsheetTitle.
SpreadsheetTitle	Entrada	String	Nombre de la hoja de cálculo que va a contener el renglón que se desea agregar.
AccessToken	Entrada	String	[Opcional] Un Token OAuth sin expiración que identifica al usuario, la aplicación y el proceso. En caso de declararse, no es necesario proveer ClientID, ClientSecret ni RefreshToken
SheetName	Entrada	String	[Opcional] El nombre de la hoja del libro de cálculo a actualizar. En caso de no declararse el renglón se agregara a la primera hoja del libro.
NewAccessToken	Salida	String	Nuevo token de acceso al proveer el RefreshToken
Response	Salida	String	Respuesta de Google con código de ejecución satisfactoria o error

Tabla 8-4. Formato de Rutina para agregar un renglón a hoja de cálculo

La Figura 8-4 es evidencia de la forma en la que se explota esta función y se implementa en Google Drive para llevar un registro de los eventos detectados por el dispositivo de monitoreo de subestaciones. Note que por cada evento se registra

la hora en la que se transmite el mensaje y el estado de todas las entradas del dispositivo en una hoja de cálculo exclusiva para este propósito; en el caso que se desee restablecer el panel de alarmas y la subestación primero se envía a registrar el estado previo al RESET para llevar un control de los eventos que se desean reponer.



Tec_Remote_Log

Archivo Editar Ver Insertar Formato Datos Herramientas Complementos Ayuda La última modificación se r

Arial 10

fx CLIENT

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	CLIENT	TimeStamp	TEMPERATURE	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16
2	TEC	20160302T181221	NA	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
3	TEC	20160302T181421	NA	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	TEC	20160302T181501	NA	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
5	TEC	20160302T191743	NA	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	TEC	20160302T191829	NA	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	TEC	20160303T074407	NA	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	TEC	20160303T075228	NA	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	TEC	20160303T081619	NA	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	TEC	20160303T115709	NA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	TEC	20160303T182555	NA	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	TEC	20160304T074749	NA	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	TEC	20160304T170153	NA	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	TEC	20160315T151714	NA	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	TEC	20160316T121326	NA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	TEC	20160317T180713	NA	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
17	TEC	20160317T181143	NA	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 8-4. Historial de eventos registrados por el dispositivo de monitoreo

8.1.5 Extraer información de celda de hoja de calculo

Al solicitar la ejecución de esta rutina dentro de los servidores de Temboo se puede obtener a través de ella el contenido de una celda en específico de una hoja de cálculo o documento que se encuentre dentro del servicio de almacenamiento en línea de Google Drive. Al igual que el resto de los *choreos* o rutinas de Cloud Computing que Temboo ofrece utilizan de la API desarrollada por Google para este objetivo y utilizan los mismos mecanismos de seguridad bajo el estándar y protocolo de OAuth.

En lo particular, esta rutina es utilizada como una capa de seguridad adicional desarrollada exclusivamente para tener control remoto sobre todos los Tokens que se utilizan para encriptar la información que se transmite del dispositivo de monitoreo hasta los servidores de Temboo consiste en programar el dispositivo para que periódicamente consulte en línea dentro de un documento protegido de Google Drive por el Token vigente y se trabaja con él durante un periodo de 24 horas. En caso de que el administrador del sistema detecte una anomalía puede solicitar generar un token nuevo en los servidores de Temboo y actualizar la hoja de cálculo en Google Drive con ese mismo Token para reanudar el servicio con una nueva llave de encriptación y autenticación; cuando el dispositivo ejecute esta rutina, la llave programada originalmente en el firmware se actualizará y el equipo reanudará su servicio encriptado bajo la nueva llave.

La Tabla 8-1 a continuación muestra los parámetros requeridos para invocar la rutina dentro de los servidores de Temboo.

Nombre de la rutina			Google.Spreadsheets.RetrieveCellValue
Variable	E/S	Tipo	Descripción
ClientID	Entrada	String	Credencial o Token que Google genera para identificar la aplicación del usuario en los servicios proveídos por Google.
ClientSecret	Entrada	String	Credencial o Token que Google genera para identificar el proceso del usuario en los servicios proveídos por Google
RefreshToken	Entrada	String	Un Token OAuth que se genera para tramitar un nuevo Token de Acceso general una vez que los originales expiran.
CellLocation	Entrada	String	Un string con el nombre de la celda en particular de la cual se desea conocer su valor
SpreadsheetKey	Entrada	String	El ID único que identifica la hoja de cálculo con la celda en la que se desea hacer la lectura.
WorksheetID	Entrada	String	El ID único de la hoja dentro del libro de calculo que relaciona la celda a la que se desea realizar la lectura
SpreadsheetName	Entrada	String	[Opcional] Nombre de la hoja de cálculo que va a contener la celda en la que se desea realizar la lectura.
AccessToken	Entrada	String	[Opcional] Un Token OAuth sin expiración que identifica al usuario, la aplicación y el proceso. En caso de declararse, no es necesario proveer ClientID, ClientSecret ni RefreshToken
WorksheetName	Entrada	String	[Opcional] El nombre de la hoja del libro de cálculo que contiene la celda en la que se realizará la lectura.
NewAccessToken	Salida	String	Nuevo token de acceso al proveer el RefreshToken
CellValue	Salida	String	Respuesta de Google con el valor contenido dentro de la celda a la que se realizó la lectura

Tabla 8-5. Formato de Rutina para realizar lectura de celda en hoja de cálculo en línea

La Figura 8-5 que se muestra a continuación representa de manera gráfica la forma en la que esta rutina es utilizada para modificar los tokens de seguridad sin la necesidad de reprogramar el dispositivo en campo, y es una forma sencilla en la que ante cualquier sospecha se puede suspender el monitoreo y proteger los datos hasta que el token de transmisión se renueve por el administrador del equipo.

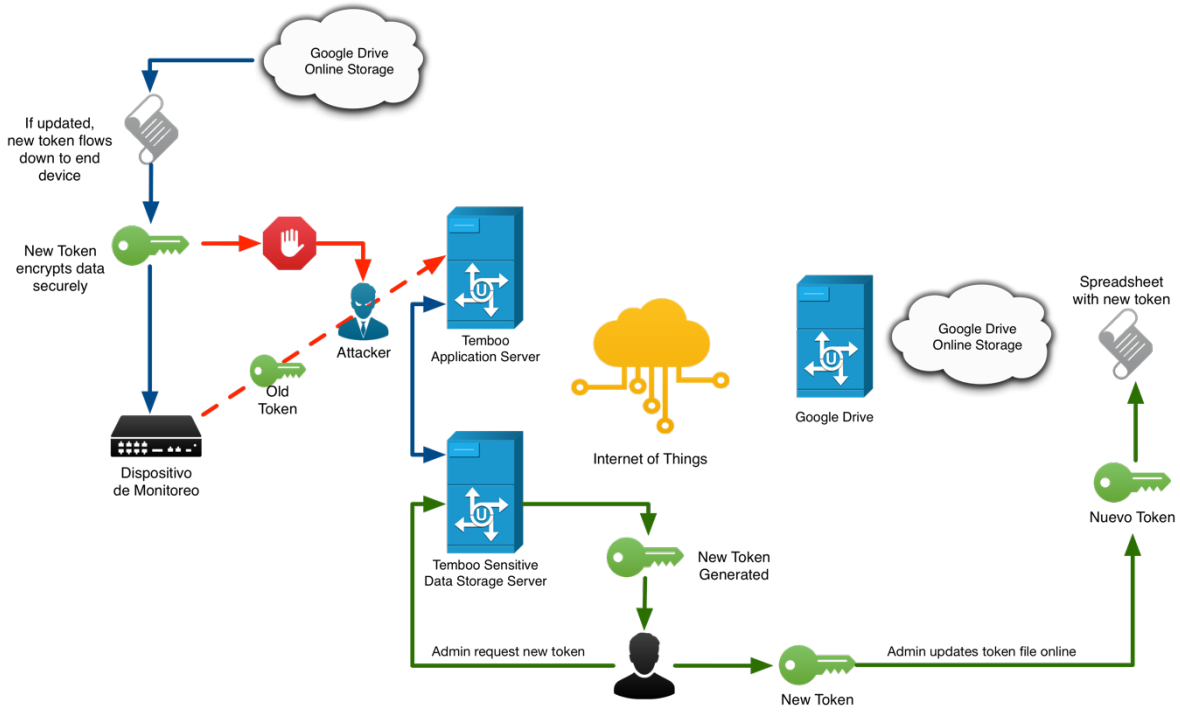


Figura 8-5. Metodología de protección de datos en dispositivo de monitoreo

8.2 Comunicación SMS a través de GSM

El modo de operación del dispositivo de monitoreo de subestaciones se define a partir del estado de la conexión a la red del equipo. Donde el enlace primario de comunicación es a través de Internet utilizando servicios de *Cloud Computing* para el relevo de actividades como lo son actualizar datos en servidores de almacenamiento y envío de mensajes de correo electrónico. En caso de que la comunicación se pierda el modo de operación en GSM se activa de forma automatizada para enviar a personal clave las condiciones de la subestación solo en caso de alarma.

Esto permite que el equipo tenga siempre un medio de respaldo para notificar la aparición de un evento en la subestación. Cabe destacar que dentro de las capacidades del dispositivo que se desarrolló se tiene la modalidad de responder a mensajes SMS en modo bajo demanda con el estado actual de la subestación mediante una función en específico en la cual el dispositivo lee si tiene mensajes SMS dentro de la bandeja de entrada del modem GSM y si encuentra alguno con el comando apropiado, la unidad responde con el código de alarma presente.

La Figura 8-6 muestra el diagrama de estado del dispositivo en referencia al modo de manejar las comunicaciones, todo parte en función de las pruebas que periódicamente realiza la unidad para determinar si tiene o no una conexión fiable a Internet. La tarjeta GSM aunque permanece encendida el 100% del tiempo el modem solamente se activa cuando se le solicita haciendo más eficiente su uso.

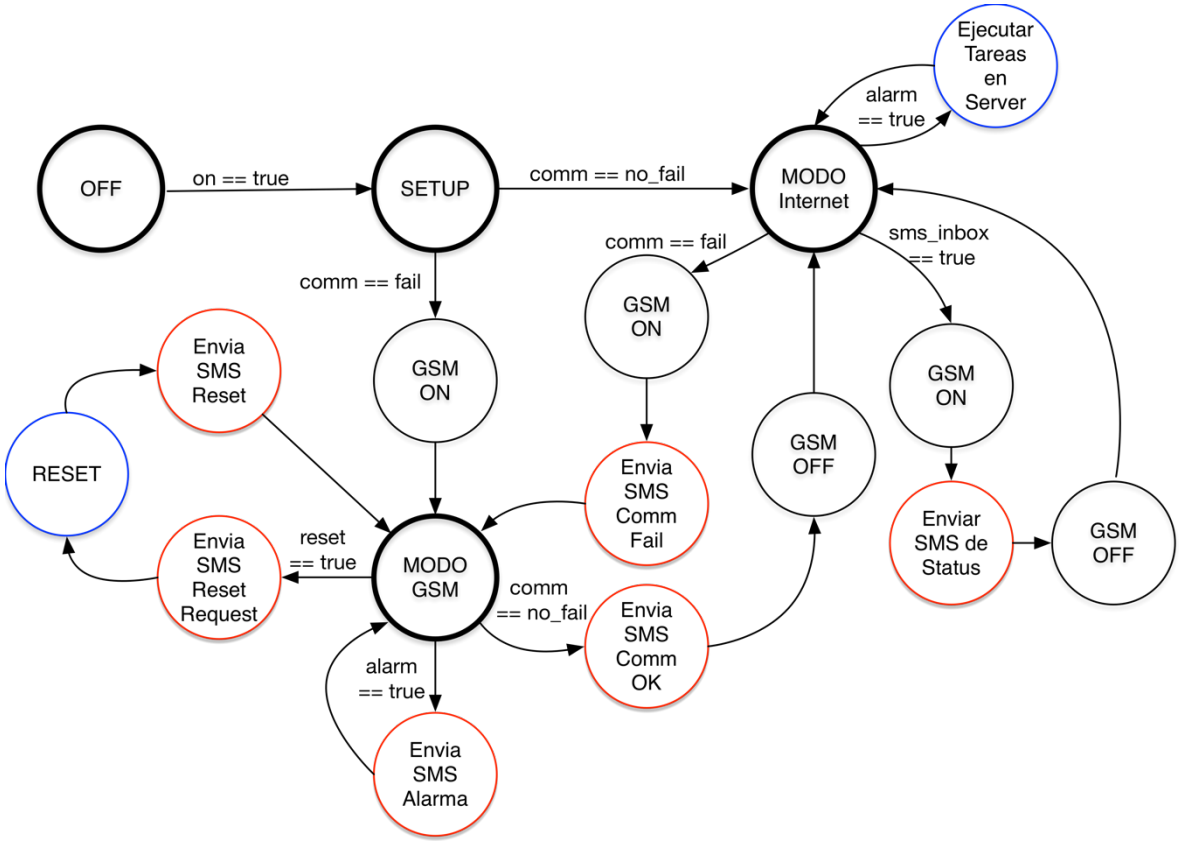


Figura 8-6. Máquina de Estados del dispositivo

El dispositivo de monitoreo maneja cuatro rutinas de envío de mensajes: mensaje de texto bajo condición de alarma; mensaje de texto por perdida o recuperación de comunicación; mensaje de texto para notificar reposición de panel de alarma de la subestación cuando no hay comunicación y mensaje de texto de dos vías para dar un estatus de las condiciones en la subestación solamente cuando la unidad recibe un mensaje de texto solicitando explícitamente el mensaje.

La Figura 8-7 muestra un ejemplo de un mensaje de texto generado por el dispositivo de monitoreo y la Tabla 8-6 guía al usuario a interpretar el código de alarma presente, donde la posición de cada uno de los dígitos corresponde a la entrada digital del dispositivo siendo 0 para no alarma y 1 para condición de alarma

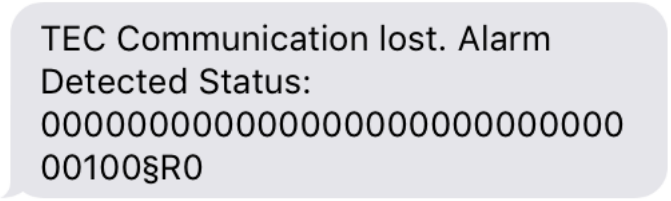


Figura 8-7. SMS Típico de alarma en subestación

DI32	DI31	DI30	DI29	DI28	DI27	DI26	DI25	DI24	DI23	…	DI n=0	R
0	0	0	0	0	0	0	1	0	0	…	0	§R0
No Alarma	No Alarma	No Alarma	No Alarma	No Alarma	No Alarma	No Alarma	Entrada Digital 25 Alarmada	No Alarma	No Alarma	No Alarma	No Alarma	RESET No Solicitado

Tabla 8-6. Interpretación de SMS

8.2.1 Mensaje de texto por perdida o recuperación de comunicación

La unidad al fallar pruebas de comunicación o al recuperar la comunicación envía mensajes de texto notificando a usuarios y administradores sobre la situación; el criterio que se sigue para determinar si la unidad pierde comunicación es mediante la ejecución no satisfactoria de las rutinas de prueba en los servidores de Temboo. Una muestra del mensaje que se envía con la perdida y la recuperación de la conexión a la red se muestra en la Figura 8-8.

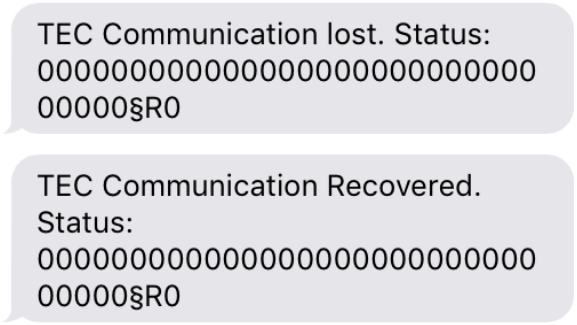


Figura 8-8. SMS Típico para perdida o restauración de comunicación

8.2.2 Mensaje de texto por detección de alarma

En cualquier momento en que no se tenga conexión a la red o exista algún problema en el canal de comunicación primario hacia los servidores y se detecte una alarma o un evento dentro del perímetro de la subestación automáticamente se envía un mensaje de texto similar a la de la Figura 8-7.

El número de mensajes que se envían es por las alarmas detectadas en el momento y se envían automáticamente a la hora del evento. Se debe de tener presente que la comunicación SMS es un método de respaldo utilizado en condiciones extraordinarias. Ante la detección de un problema en la comunicación, aunque no grave, se debe de prestar atención y resolverse a la brevedad.

8.2.3 Mensaje de texto por reposición de panel de alarma

Este mensaje de texto es enviado solamente cuando bajo condiciones de no comunicación se le solicita al dispositivo realizar la reposición del panel de alarma y borrar mediante la función RESET. Para ello, se realiza la transmisión del SMS en tres etapas.

La primera de ellas notificando que el RESET ha sido solicitado y se transmite con todas las alarmas previas a la reposición; posteriormente la segunda etapa consiste en reponer el panel de alarmas y restaurar el dispositivo de monitoreo y finalmente la tercera etapa consiste en notificar con otro mensaje SMS que el RESET ha sido completado.

La Figura 8-9 muestra un ejemplo de mensajes de texto que trabajaron bajo este modo de operación. Note que el código del mensaje termina en R1 en lugar de R0

para denotar que la función RESET ha sido invocada por parte del usuario directamente en el dispositivo de monitoreo.

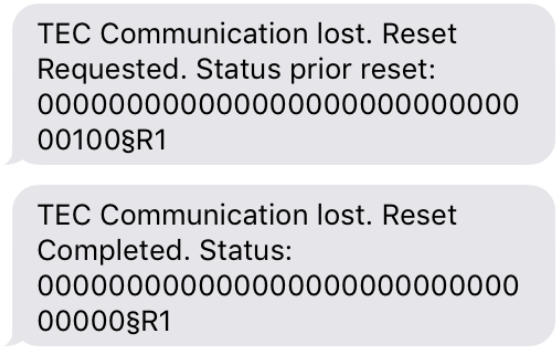


Figura 8-9. SMS Típico para reponer panel de alarma

8.2.4 Mensaje de texto en dos vías para reportar estatus

Esta función puede ser ejecutada en cualquier momento por cualquier usuario que conozca el número telefónico asociado a la tarjeta SIM del modem GSM y además indique dentro del mensaje de texto el comando apropiado para que la unidad responda con un código del estado de sus entradas digitales.

La rutina que el firmware tiene programada para escuchar los mensajes de texto se ejecuta cada quince minutos, por lo que al enviarse un mensaje de texto con el comando puede tardar hasta quince minutos en responder de regreso.

Esta función es útil también para determinar si el dispositivo de monitoreo se encuentra funcional. La Figura 8-10 muestra la interacción entre un usuario y el dispositivo de monitoreo con mensajes de texto.

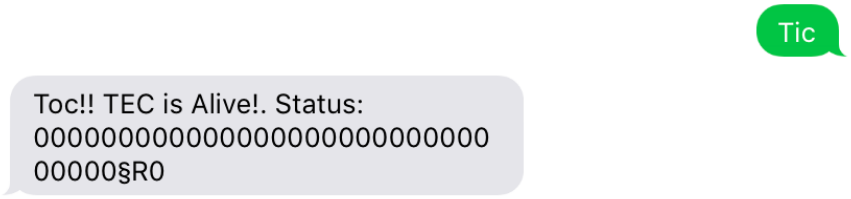


Figura 8-10. SMS Típico de dos vías en el dispositivo de monitoreo

9 Diseño de Firmware

Esta sección contiene todos los artefactos de software utilizados para la creación del firmware del dispositivo de monitoreo de subestaciones, la sección se encuentra dividida en dos firmwares diferentes uno que es utilizado por el microcontrolador que realiza la función de *watchdog* y el otro utilizado por el microcontrolador principal que lleva a cabo las tareas de monitoreo; cada sección contiene los diagramas de flujo, descripciones y validaciones realizadas en primer plano así como las variables más representativas de cada función utilizada. Se descartan por fines de extensión, variables de apoyo a estructura de software y funciones provenientes de librerías de dominio público.

En la sección de apéndices se encuentra el seudocódigo mnemotécnico del firmware del dispositivo como referencia al proyecto y que pueda ser exportado a cualquier lenguaje de programación de alto nivel.

La Tabla 9-1 contiene los detalles sobre el IDE y el lenguaje de programación utilizado para programas los microcontroladores del dispositivo, también se documentan las librerías adicionales utilizadas para la elaboración del software.

IDE	Arduino Software	Arduino Software
Versión IDE	1.6.7	1.6.7
Nombre proyecto	Watchdog	Monitoreo_Tec
Versión proyecto	2	20
Funciones utilizadas	2	22
Funciones documentadas	2	20
Funciones abiertas	0	2
Librerías	Ninguna	SPI.h DHCP.h DNS.h Ethernet.h EthernetClient.h Temboo.h TimeLib.h EthernetUDP.h Bridge.h GSM.h
Tarjeta	Sparkfun Pro Micro	Arduino Mega 2560
Procesador	ATmega32U4 5V 16Mhz	ATmega2560
OS del IDE	Mac OSX 10.11.5	Mac OSX 10.11.5

Tabla 9-1. Encabezado de los artefactos de software

9.1 Firmware en Microcontrolador de Watchdog

El propósito de este proyecto de software es vigilar la correcta operación del dispositivo principal mediante el monitoreo de un *heartbeat* generado por el microcontrolador, se busca que se genere una señal eléctrica de carácter periódica que sea monitoreada y su ausencia es indicativo que el microcontrolador principal se ha inhibido y requiere ser intervenido para reiniciar la rutina ciclando el suministro eléctrico en todas las tarjetas de control.

9.1.1 Encabezados y Variables Globales

A continuación en la Tabla 9-2 se presentan las variables globales más representativas y su descripción. También se enlistan aquellas funciones que son utilizadas dentro del algoritmo.

Nombre de Variable/ O función	Tipo	Valor	Descripción
Timeout	Const long	600000	Tiempo en milisegundos en que se debe de perder el *heartbeat* antes de reiniciar el microcontrolador principal
Millis()	función	-	Devuelve el tiempo en milisegundos en el que el microcontrolador ha permanecido encendido
previousMillis	long	0	Variable a contener el tiempo de ejecución del microcontrolador en el periodo n-1
currentMillis	long	0	Variable a contener el tiempo de ejecución del microcontrolador en el periodo n

Tabla 9-2. Variables globales y funciones nativas del proyecto Watchdog

9.1.2 Función Watchdog (Loop)

En la Figura 9-1 se muestra el diagrama de flujo del watchdog, básicamente se declaran las variables globales que funcionan de control de tiempo y se define el pin por el que se espera recibir la señal de *heartbeat* del controlador principal y el pin por el que se espera comandar el ciclado eléctrico en las tarjetas.

Primero se realiza la lectura de la señal determinando si esta se encuentra ausente o no ausente y se mide el tiempo en que ha permanecido en esa condición. Si el resultado es mayor a la tolerancia definida con la variable timeout se considera que hay un problema con el microcontrolador y se manda la señal de reiniciar, en caso contrario se sigue esperando a que la señal del *heartbeat* cambie de estado y

los temporizadores se reinician. Este programa corre de manera cíclica de forma indefinida, al no existir variables que acumulen datos no existe riesgo que la tarjeta que funciona como *watchdog* se inhiba.

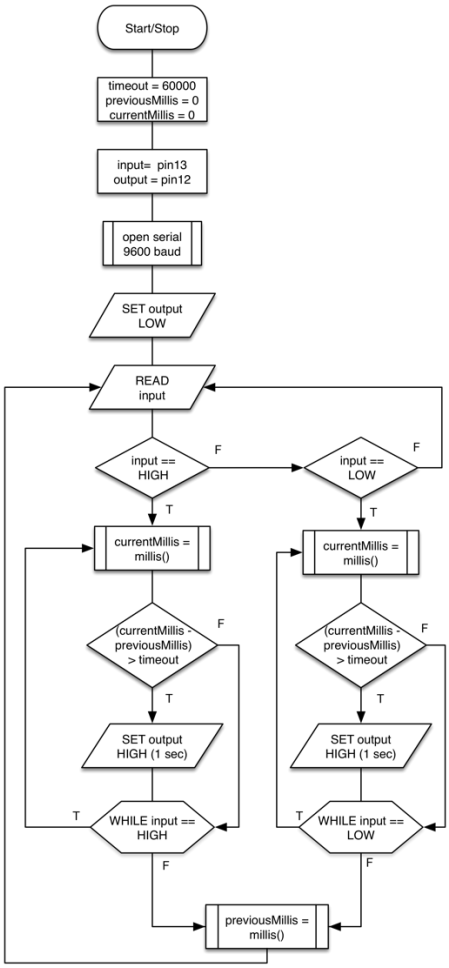


Figura 9-1. Diagrama de Flujo del Watchdog

9.2 Firmware en Microcontrolador Principal

El propósito de este artefacto de software es de llevar el control de monitoreo de todas las entradas digitales de la tarjeta Arduino Mega 2560 rev 3; de tomar las decisiones sobre el método que se utilizara para notificar, construir correos electrónicos, mensajes de texto a celular y de también realizar la conexión a los servidores de Cloud Computing de Temboo para poder llevar a cabo las tareas de entrega de correos y almacenamiento de datos. Dentro del encabezado de este

proyecto deberán de especificarse las características de cada alarma en particular como lo es: lógica de operación, si su presencia o ausencia desencadena el envío inmediato de correo electrónico o diferido, los usuarios que recibirán notificaciones, el mapeo de memoria de la alarma, entre otras configuraciones.

9.2.1 Encabezados y Variables Globales

En la Tabla 9-3 se muestra las variables globales más representativas de este proyecto de software y su descripción. También se hace mención de aquellas funciones que son utilizadas a lo largo del proyecto y son nativas de las librerías del IDE.

Nombre de Variable/ O función	Tipo	Valor	Descripción
MaskUnmaskPins	Integer Array	0/1	Arreglo de enteros que pueden tener el valor 0 o 1 para habilitar o deshabilitar por software una entrada digital desde D0 a D31
MaskUnmaskEmail	Integer Array	0/1-	Arreglo de enteros que pueden tener el valor de 0 o 1 para indicar cuál de todas las entradas digitales puede desencadenar el envío de una alerta de correo
MaskUnmaskLOGIC	Integer-Array	0/1	Arreglo de enteros que pueden tener el valor de 0 o 1 para indicar si la lógica de una alarma es negada o lo contrario. 1 = Lógica No Negada; 0 = Lógica Negada
Alarm_Code_Name	String Array	***	Arreglo de cadenas de texto que relaciona la entrada digital del dispositivo al nombre de una alarma en particular
Interval	Const Long Int	900000	Tiempo de timeout, cada cuanto tiempo se va a transmitir de forma automatizada el estado del equipo a los servidores de almacenamiento de datos en línea. También es el periodo de tiempo en el que se analiza el estado de la conexión
BatchInterval	Const Long Int	120000	Tiempo de espera antes de que se transmita una alarma
LOSTCOMInterval	Const Long Int	120000	En condiciones de perdida de comunicación es la frecuencia con la que se volverá a hacer intentos de conexión a la red.
Healthlevel	Int	4	Nivel de salud del dispositivo, es un indicador que decrementa su valor dependiendo de las condiciones de falla en la operación de monitoreo, si hay una falla al transmitir un dato esta variable decrementa al punto en el que se considera que hay una pérdida total de comunicación.
GSM_Enable	Int	0/1	Bandera para indicar si la función GSM se encuentra habilitada

Tabla 9-3. Variables globales y funciones nativas del proyecto Monitoreo

9.2.2 Función Setup del dispositivo

Esta función dentro del dispositivo de monitoreo de subestaciones es ejecutada cada vez que la unidad se enciende y tiene el objetivo de inicializar el pinout de la tarjeta, hacer una verificación de la conexión a la red local, verificar la conexión a la red GSM y hacer pruebas de comunicación con los servidores de Cloud Computing antes de pasar a la etapa de monitoreo.

Una vez que se concluye el proceso, se envía un mensaje de correo electrónico de prueba, un mensaje de texto para indicar la conexión satisfactoria y se descarga el token de seguridad vigente para encriptar los datos durante la transmisión de la información.

En caso de que la unidad no cuente con conexión a internet durante el proceso de setup y tampoco cuente con conexión a la red celular, permanecerá en este estado sin generar *heartbeat* de tal forma que se reinicie completamente a una frecuencia comandada por el *watchdog* hasta que recupere la conexión a la red. No corresponde a esta sección, pero se adelanta que si se pierde la comunicación dentro de la etapa de monitoreo la unidad no se reinicia, solamente cambia al canal de respaldo con el propósito de no interrumpir el monitoreo; al recuperar la conexión, sin necesidad de reiniciar se regresa al canal primario de comunicación.

Dentro del dispositivo, se cuenta con un puerto serial USB en donde se transmite información del estado actual del dispositivo con fines de hacer *troubleshooting* durante la etapa de desarrollo de software.

La Figura 9-2 muestra el diagrama de flujo donde se muestra la lógica de operación de esta sección del firmware programado en el dispositivo. Note que este diagrama solamente se ejecuta una vez durante el periodo en el que el programa se encuentra corriendo en el microcontrolador, al terminar su ejecución se procede a cambiar a la función Loop que se ejecuta indefinidamente.

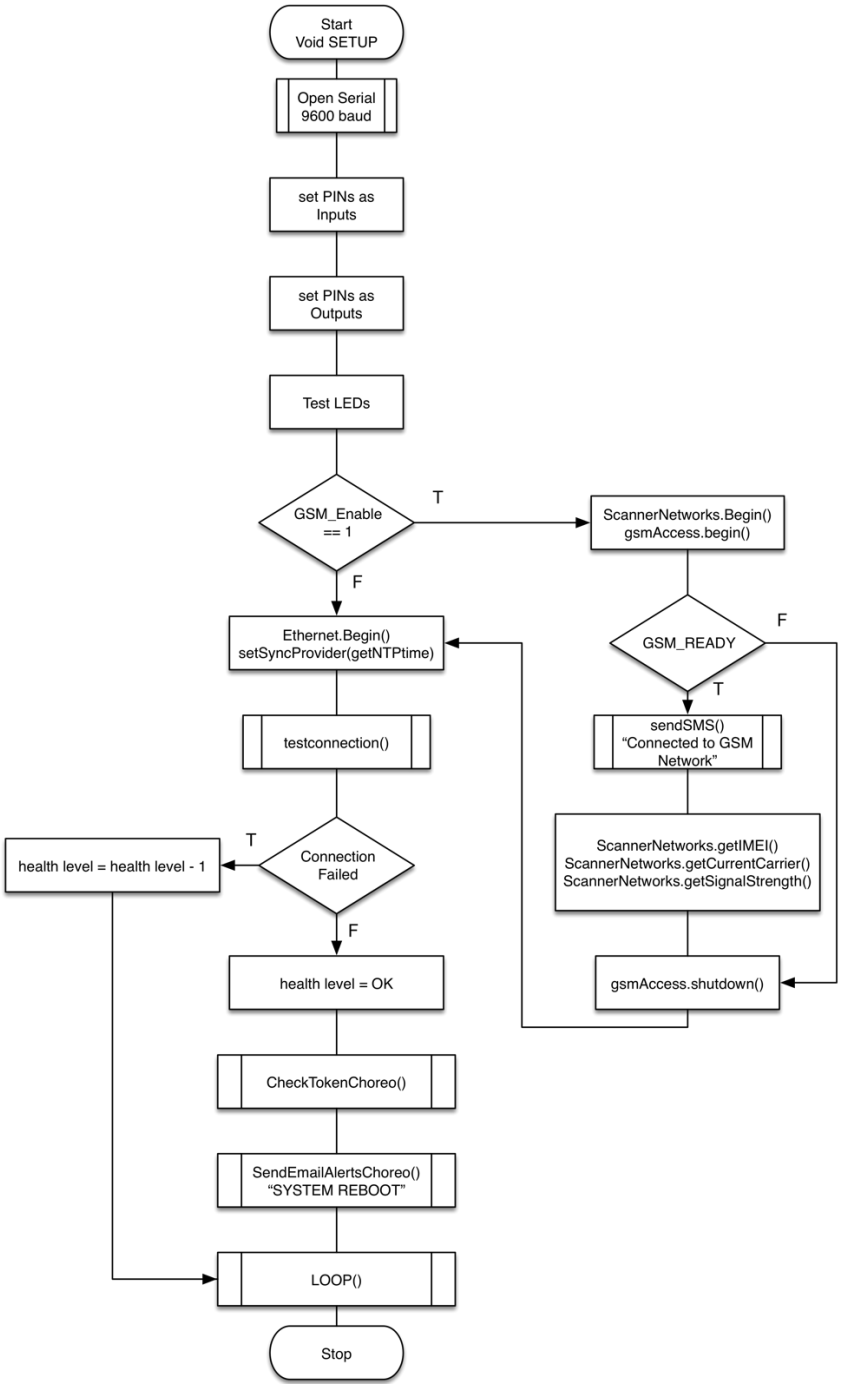


Figura 9-2. Diagrama de flujo de la rutina Setup

9.2.3 Función Loop del dispositivo

La función loop del dispositivo de monitoreo es aquella que se está ejecutando de manera indefinida con el mejor tiempo de muestreo posible para el microprocesador de la tarjeta programable. Esta función manda a llamar otras

funciones de manera periódica como lo es el revisar el estado físico de cada una de las entradas digitales de la tarjeta, revisar la calidad de la conexión, controlar el *heartbeat*, monitoreo de la función restablecer así como otras rutinas de verificación de alarmas, actualización periódica de estados de entradas digitales en servidor, envío de correos electrónicos rutinarios, entre otros.

Esta función contiene rutinas temporizadas que se ejecutan a un periodo predefinido para realizar enlistadas a continuación:

i. Reportar estado de todas las entradas digitales en periodos de 15 minutos a partir de la ejecución del firmware, verificar la bandeja de entrada de mensajes de texto SMS en el modem GSM, verificar la calidad de la conexión a la red de Internet.
ii. En caso de activarse una alarma dar una ventana de monitoreo activo de 2 minutos para dar oportunidad a enviar todos los datos en un solo bloque de transmisión.
iii. Envío de correo electrónico matutino con el ultimo estado de alarmas, verificación de cambio de token de seguridad una vez al día, a la hora y minuto especificado
iv. En caso de que la comunicación a la red no sea satisfactoria en cuatro ocasiones consecutivas el *health level* se decrementa al punto de activar el modo GSM en el dispositivo y se verifica la conexión a la red cada 2 minutos hasta que se recupere y el *health level* se restablece
v. Generar señal periódica de *heartbeat* cada 500ms.

Cabe destacar que la función principal del firmware es la que se documenta en esta sección y el resto son sub-funciones que son mandadas a llamar cada vez que se ejecuta una tarea. La Figura 9-3 describe el diagrama de flujo donde se especifica la manera de trabajar de esta función y aquellas funciones que se mandan a llamar para cumplir con todos los objetivos establecidos en el dispositivo de monitoreo de subestaciones eléctricas.

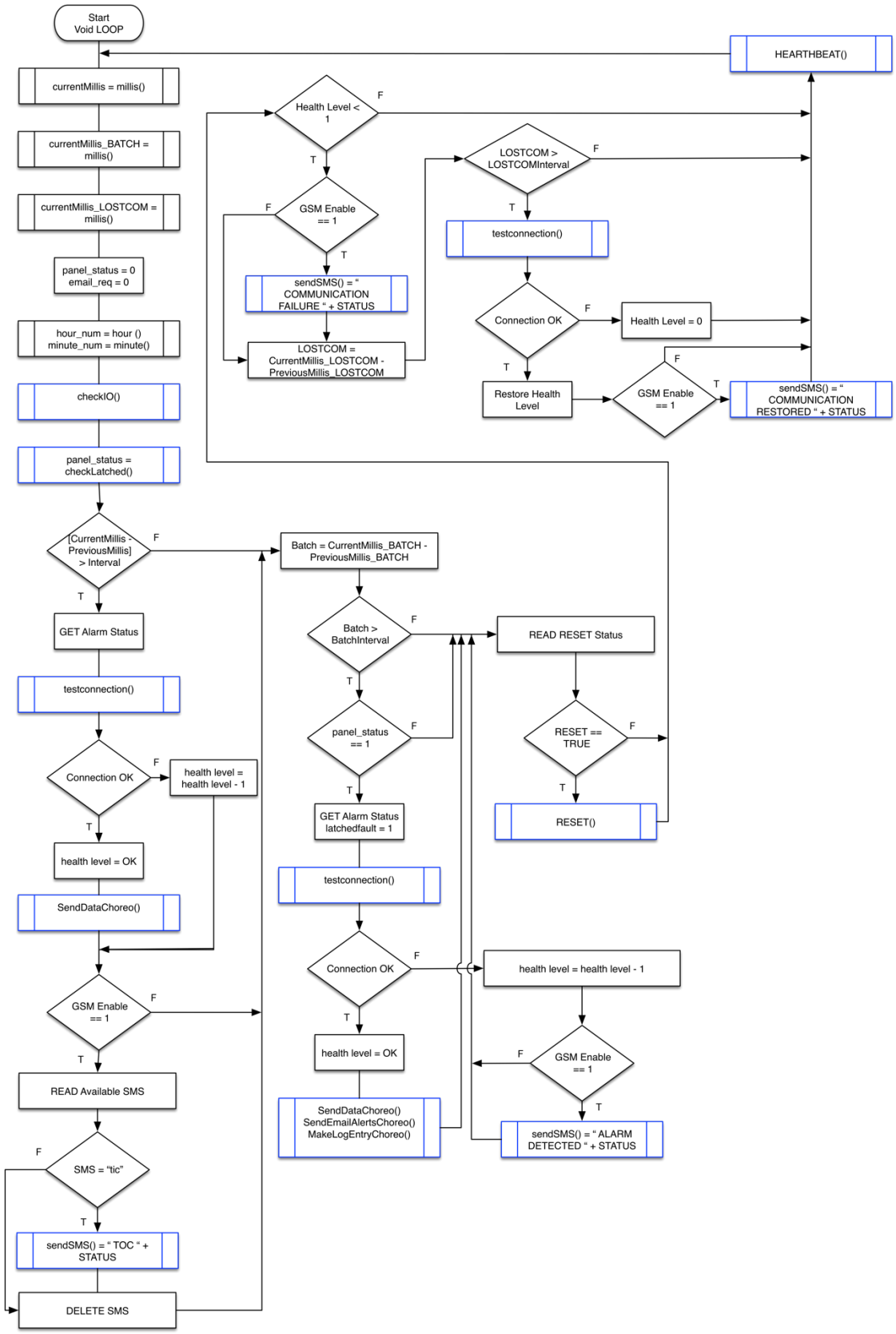


Figura 9-3. Diagrama de Flujo de la rutina Loop

9.2.4 Función Heartbeat

La función *heartbeat* consiste en una rutina que activa o desactiva una DO en el controlador Arduino MEGA 2560, esta salida está ligada a una luz indicadora que tiene por propósito alertar de forma visual que el dispositivo se encuentra en operación y además sirve como señal de monitoreo para el *watchdog* del dispositivo. La rutina consiste en verificar si han pasado más de 500 ms desde la última vez que se ejecutó la función *heartbeat,* en cuyo caso se activa o desactiva la salida en cuestión según corresponda.

El efecto de esta función se representa de tal forma que si el programa principal se inhibe ya sea por algún problema en el microcontrolador o algún sector de memoria corrupto, la función *heartbeat* deja de ejecutarse. Esto interfiere directamente con el *watchdog* del dispositivo de monitoreo de subestaciones quien al no detectar la señal del *heartbeat* este acciona un relevador que reinicia el dispositivo de manera general. La Figura 9-4 muestra el diagrama de flujo de la función *heartbeat.*

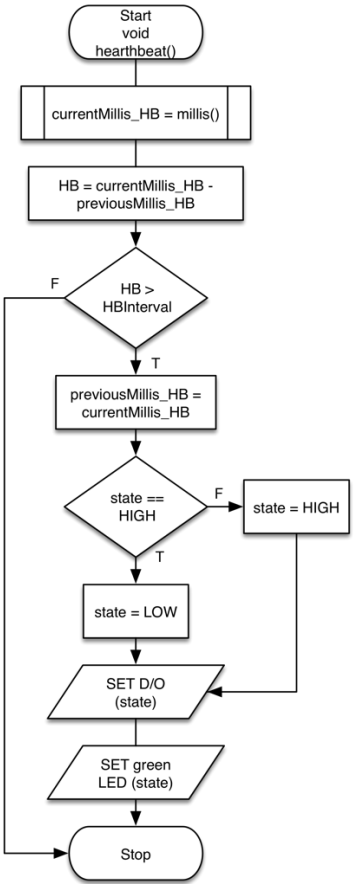


Figura 9-4. Diagrama de Flujo de la función Heartbeat

9.2.5 Función Reset

La función reset entra en acción cada vez que la función loop detecta que se ha presionado el interruptor de restablecer y la DI correspondiente detecta un valor alto. Esta rutina tiene por objetivo limpiar todas las variables de alarma, probar la comunicación y restablecer cualquier alarma memorizada dentro del dispositivo. El procedimiento consiste en reportar a los servidores de almacenamiento de datos en la nube el ultimo estado de las alarmas (que se desean reponer) y que además se está solicitando restablecerlas, posteriormente se envía un correo notificando a todo el personal involucrado.

En caso de existir perdida de comunicación al momento de solicitar el reset, este se ejecuta sustituyendo todas las actividades que se realizan en la red de Internet por mensajes de texto a aquellos usuarios que se encuentren suscritos en el dispositivo de monitoreo de subestaciones para este servicio. En ambos casos, la última acción de esta rutina es en si la función reset, posterior a haber debidamente ejecutado todas las actividades de notificación previas.

La Figura 9-5 muestra el diagrama de flujo de la operación de esta rutina, note que la última actividad antes de terminar el proceso es el limpiar las variables y colocarlas en un estado de no alarma y finalmente actualizar los datos en servidor en caso de que la conexión se encuentre en óptimas condiciones o en su defecto notificar por SMS que la operación ha sido completada.

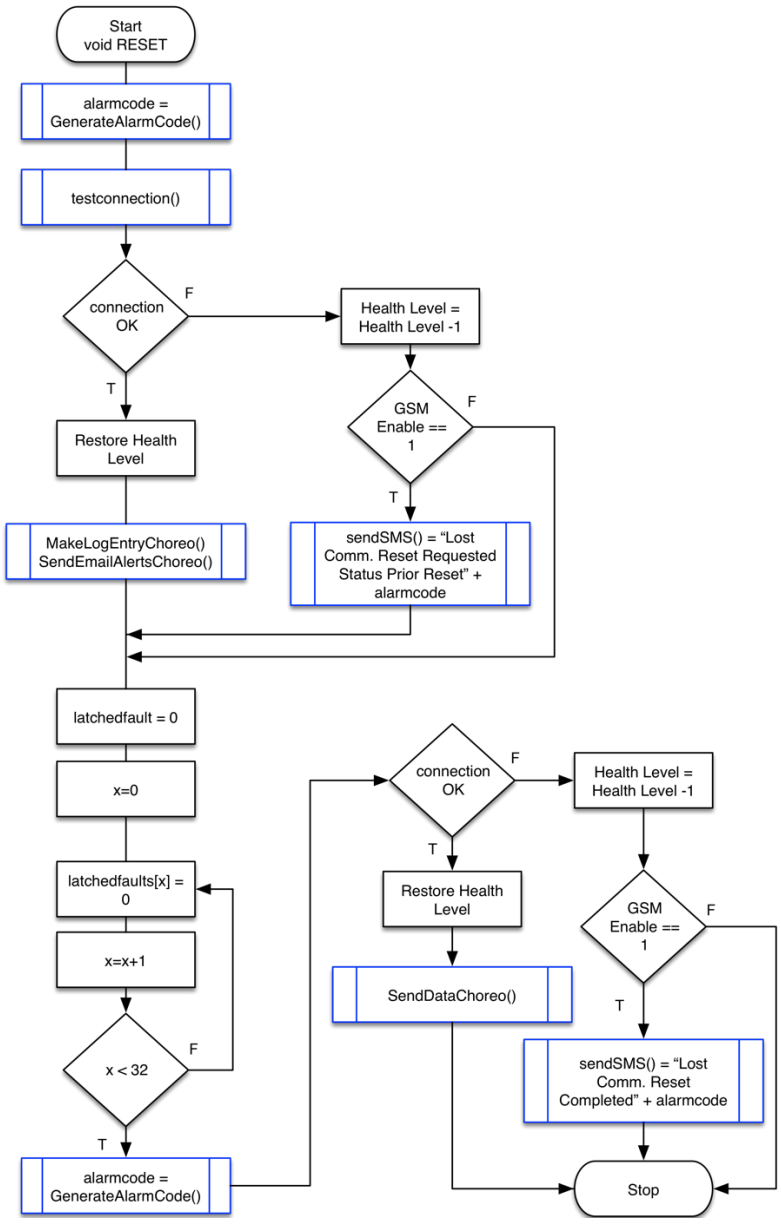


Figura 9-5. Diagrama de flujo de la función Reset

9.2.6 Función CheckIO

Esta función tiene por objetivo realizar un barrido de lecturas a todas aquellas entradas digitales declarados en los encabezados del código. La lectura se realiza haciendo referencia cruzada con dos parámetros que definen la manera en la que se debe de tratar la señal. La primera es revisar la lógica de la alarma, es decir si es lógica negada (la señal se encuentra presente, y al existir un evento se pierde la señal) o no negada (no existe señal hasta que se genera el evento) esta operación

se hace mediante la variable MaskUnmaskLOGIC[32] que puede adoptar valores enteros en 1 y 0 para diferenciar la lógica que se usará para ese canal en particular.

El segundo parámetro que se revisa consiste en comprobar si la alarma se encuentra habilitada por código, en los encabezados declarada se encuentra la variable MaskUnmaskPINS[32] que es un arreglo unidimensional y permite al código discriminar uno o más canales dentro de la rutina de monitoreo.

Al detectar una alarma en cualquier canal tomando en consideración la lógica de la misma se corre un proceso de memorización, de tal forma que si una alarma proveniente de campo es un pulso no sostenido esta pueda mantenerse dentro de la memoria del programa para que otras funciones corran el proceso de notificación y actualización de datos; las alarmas memorizadas se manejan a través de un arreglo unidimensional llamado LatchedFaults[32].

Este algoritmo puede ser visualizado en la Figura 9-6.

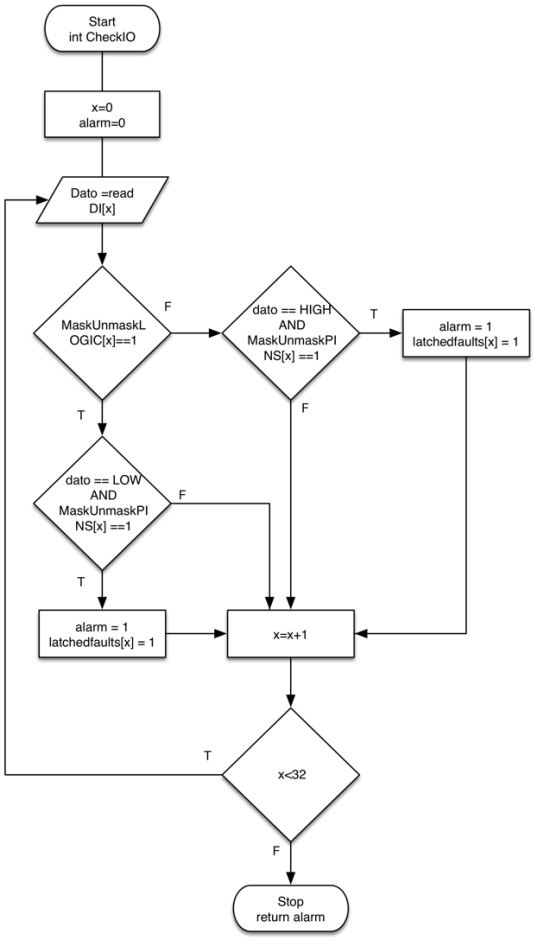


Figura 9-6. Diagrama de flujo de la rutina CheckIO

9.2.7 Función CheckLatched

Esta rutina es similar a la función CheckIO pero en lugar de revisar las señales de campo busca por las alarmas dentro de la memoria del programa. Cabe destacar que la función CheckIO es la única en la que se ha programado la lógica de disparo de alarma; por otra parte, la función CheckLatched trabaja bajo una lógica no negada. Esta función es ejecutada directamente por la rutina Loop cada vez que se desea verificar si existe una alarma presente en el dispositivo de monitoreo de subestaciones.

Consiste en un barrido del arreglo latchedfaults[32] y en caso de detectar que existen alarmas memorizadas, entonces la función regresa un 1 indicando presencia de alarmas o un 0 para el estado no alarmado. La Figura 9-7 conceptualiza el diagrama de flujo de la función en cuestión.

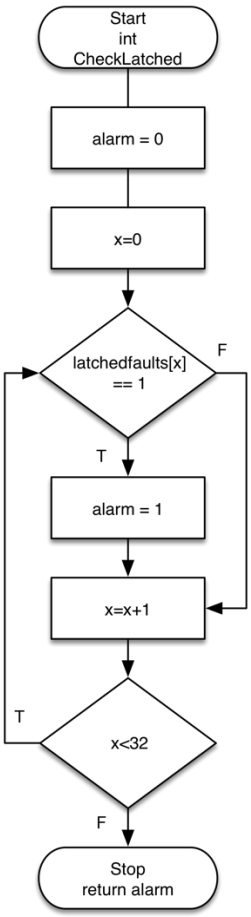


Figura 9-7. Diagrama de flujo de la función CheckLatched

9.2.8 Función GenerateAlarmCode

El objetivo de esta función es generar una cadena de caracteres de 0 y 1 en formato string que represente el estado de todas las alarmas presentes en la variable latchedfaults[32], cada vez que se manda a llamar regresa la cadena. Esto es utilizado como un método práctico para enviar el estado de todas las alarmas en un SMS donde la cantidad de caracteres por mensaje es limitada. También es utilizada para mostrar el estado de las alarmas a través del puerto USB de la tarjeta con comunicación serial, sin embargo esto solamente es utilizado para fines de depuración del proyecto.

La Figura 9-8 muestra la conceptualización gráfica de esta rutina.

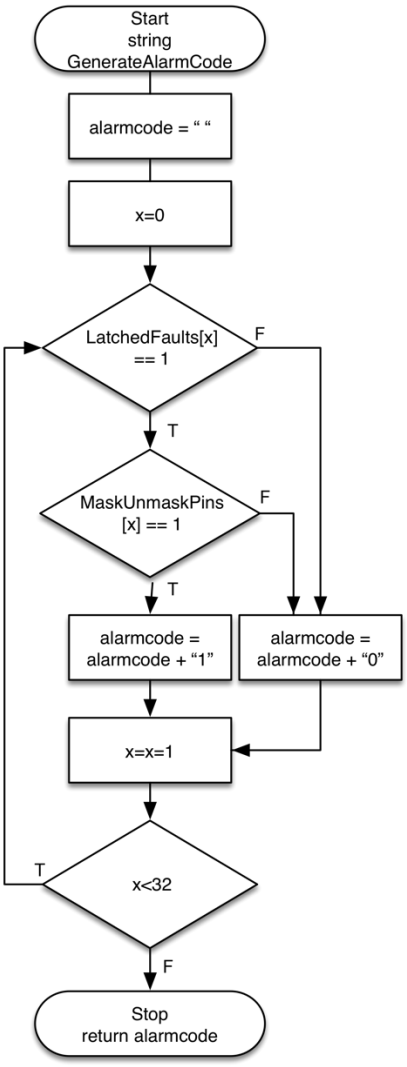


Figura 9-8. Diagrama de Flujo de la rutina GenerateAlarmCode

9.2.9 Función Generate_TimeStamp

Como su nombre lo dice esta función tiene por objetivo generar una estampa de tiempo cada vez que es ejecutada, es utilizada para identificar la fecha y hora del evento a transmitir en correos electrónicos y para generar el registro de eventos cuando se almacenan datos a través de cloud computing a los servidores de Google Drive. Al ser invocada se devuelve un string con el formato indicado en la Tabla 9-4 a continuación.

Formato de Time Stamp	Año AAAA	Mes MM	Día DD	T	Hora HH	Minuto MM	Segundo SS
Ejemplo de Time Stamp	2016	08	03	T	12	53	05
Cadena de Caracteres de Time Stamp	20160803T125305						

Tabla 9-4. Formato de Estampa de Tiempo

Esta función obtiene los datos de la fecha y hora actual a través de una función llamada GetNTPTime que en conjunto con la librería Ethernet se conecta servidores NTP para obtener y sincronizarse con la hora de internet. El servidor seleccionado para obtener la fecha y hora exacta pertenece al NIST y se encuentra localizado en Boulder Colorado en Estados Unidos y responde al siguiente URL e IP: time-a.timefreq.bldroc.gov 132.163.4.101

9.2.10 Función Generate_Datagram

Esta función de tipo string concatena tres variables: la estampa de tiempo, una variable que identifica al cliente y el código de alarma. Esta función requiere como variable de entrada el código de alarma generado en GenerateAlarmCode y regrese un string cuyo formato se especifica en la Tabla 9-5. Esta cadena de texto es utilizada además como parte del mensaje SMS cuando el dispositivo entra en el modo de comunicación de respaldo.

Formato de Datagrama	Time Stamp	_	Client ID	_	Alarm Code
Ejemplo de Datagrama	20160803T154728	_	TEC	_	00000100011000001000100000000000
Cadena de Caracteres	20160803T154728_TEC_00000100011000001000100000000000				

Tabla 9-5. Formato de Datagrama

9.2.11 Función LogEntry_Datagram

Al igual que la función Generate_Datagram esta concatena las mismas tres variables que involucran la estampa de tiempo, la variable que identifica al cliente y el código de alarma generado en GenerateAlarmCode. Esta función requiere como variable de entrada el código de alarma y a diferencia de la función Generate_Datagram esta regresa una cadena de strings que separa por comas cada alarma en particular, ya que esta función trabaja en conjunto con el Choreo de Temboo para crear un renglón en una hoja de cálculo ubicada en la nube y se requiere que el datagrama cumpla con un formato en el que las columnas se separen por el carácter identificado.

El formato del datagrama para trabajar en conjunto con el Choreo *Append Row* y *Update Row* se encuentra especificado en la Tabla 9-6.

Formato de Datagrama	Client ID	,	Time Stamp	,	Alarm Code (Separado por Coma)
Ejemplo de Datagrama	TEC	,	20160803T154728	,	0,0,0,0,0,1,0,0,0,1,1,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0
Cadena de Caracteres	TEC,20160803T154728,0,0,0,0,0,1,0,0,0,1,1,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0				

Tabla 9-6. Formato de Dato para insertar texto en hoja de cálculo

9.2.12 Función CheckIf_EMAIL_required

Esta función es mandada a llamar por la rutina Loop cada vez que se detecta una alarma nueva, su objetivo es determinar si el evento en cuestión desencadena el envío de mensajes de correo; esto quiere decir que se puede seleccionar que eventos en el panel de alarmas generan correos electrónicos, sin embargo el estado de la alarma es transmitido al servidor de almacenamiento en línea de forma periódica independientemente de la configuración que se le asigne a cada canal en particular.

Esto se logra a través del código de alarma generado en la función GenerateAlarmCode el cual es dato de entrada para la rutina que en esta sección se documenta y el arreglo MaskUnmaskEMAIL[32] el cual es un vector donde cada posición representa un canal en el dispositivo de monitoreo y el valor de 1 define

que esa alarma en particular envía un correo y 0 que no lo envía. La función en cuestión regresa un 1 en caso de que el envío de correo electrónico sea obligado para al menos una de las alarmas que se encuentren activadas en el momento en que se ejecuta la rutina o bien retorna 0 en caso de que todas las alarmas activadas en el momento no requieran el envío obligado de correos. La Figura 9-9 muestra el algoritmo al momento de realizar el monitoreo para determinar si la alarma en cuestión desencadena el envío de correos electrónicos.

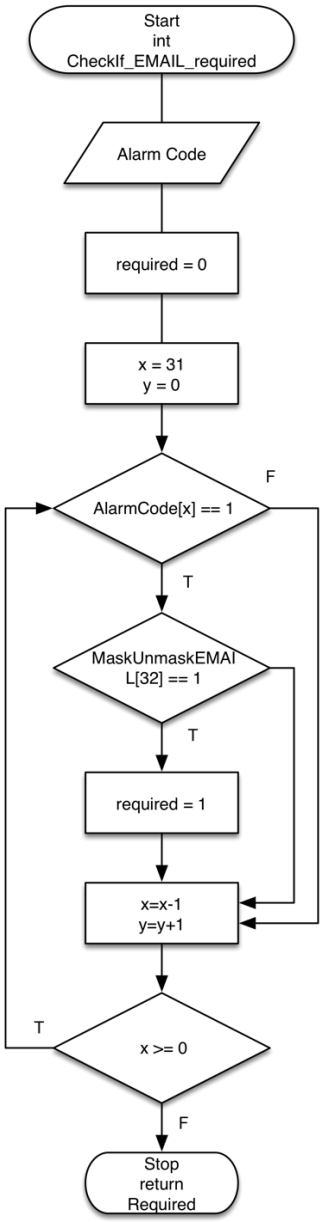


Figura 9-9. Diagrama de flujo de la rutina CheckIf_EMAIL_required

9.2.13 Función Compose_EMAIL

La función Compose_EMAIL tiene por objetivo construir el cuerpo del mensaje que se transmite por correo electrónico dependiendo del código de alarma generado en GenerateAlarmCode y condiciones generales de estado del dispositivo. Los correos electrónicos se envían bajo cualquiera de las siguientes condiciones:

i. Al encender o reiniciar el dispositivo de monitoreo y confirmar una conexión válida a la red (Envía correo de reinicio satisfactorio).
ii. Al momento de activarse una alarma que tenga configurada la opción de envío de correo electrónico (Envía correo de alarma detectada con el detalle de las alarmas disparadas en el panel).
iii. Al menos una vez al día a la hora programada (Envía recordatorio de alarmas o aviso de operación normal).
iv. Al reponer el panel de alarmas activando la función reset (Envía correo de reposición de sistema).
v. Al recuperar comunicación (Envía correo con alarmas presentes u operación normal del sistema).

La función Compose_EMAIL es mandada a llamar por el Choreo de Temboo encargado de realizar el envío del mensaje de correo. El resultado de ejecutar esta función se aprecia en la Figura 9-10, Figura 9-11, Figura 9-12 y Figura 9-13.

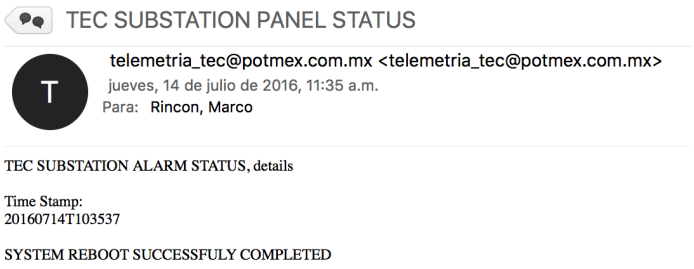


Figura 9-10. Muestra de correo electrónico de reinicio de sistema

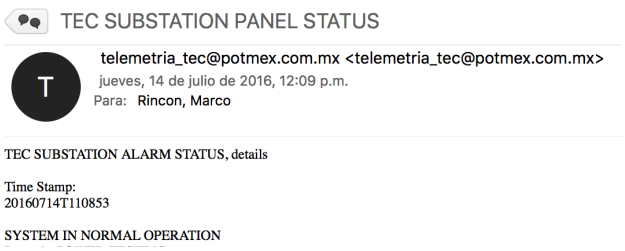


Figura 9-11. Muestra de correo electrónico de sistema en operación normal

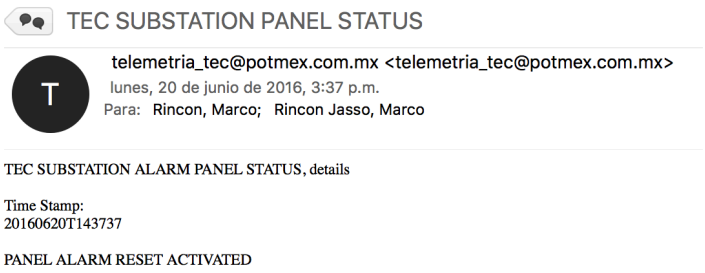


Figura 9-12. Muestra de correo electrónico de reposición de sistema

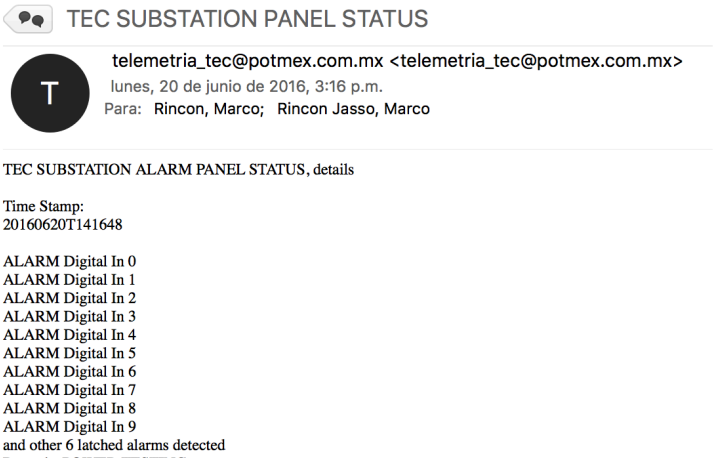


Figura 9-13. Muestra de correo electrónico con detalle de alarmas

La Figura 9-14 se identifica el diagrama de flujo con la lógica para construir el cuerpo del mensaje de correo electrónico.

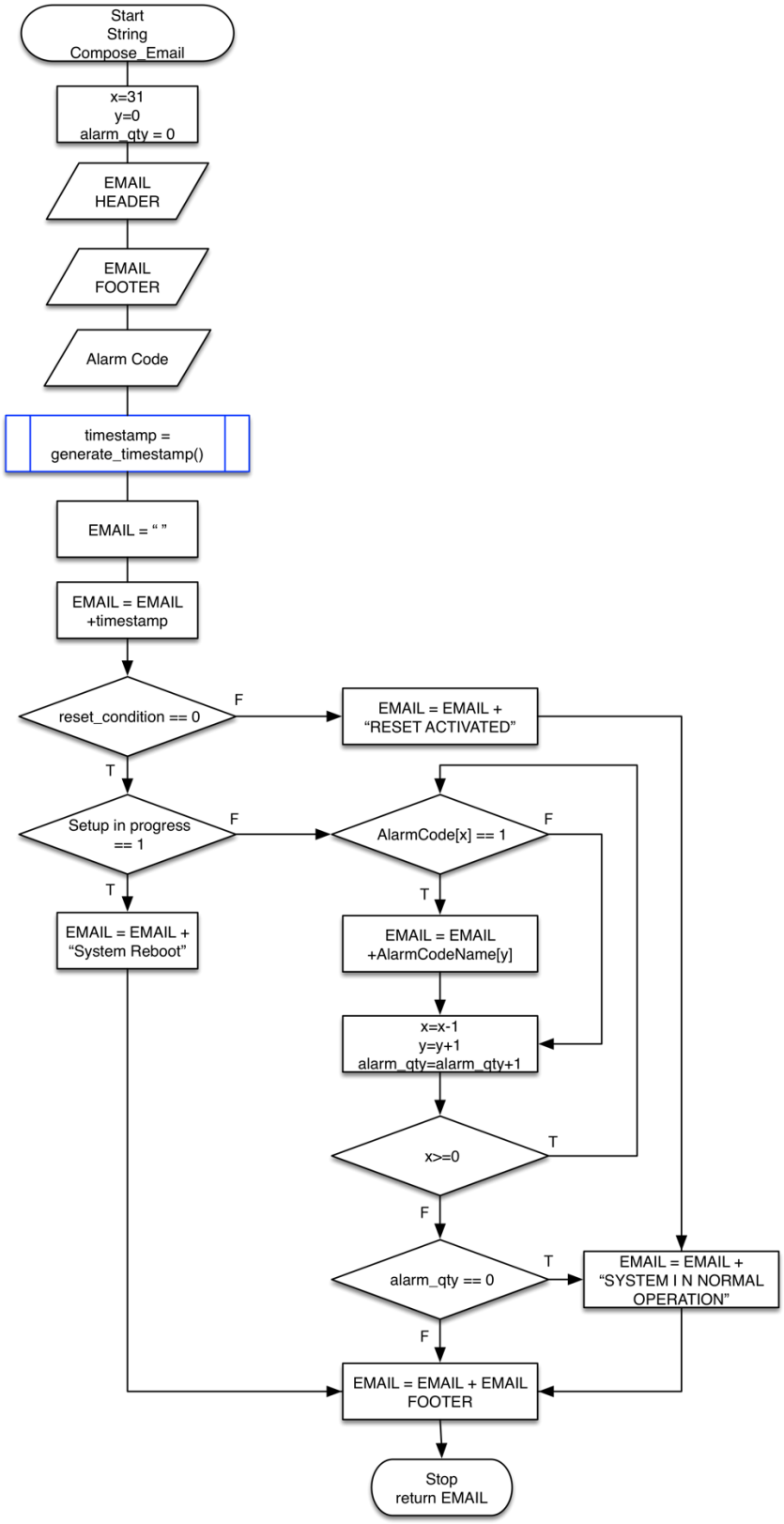


Figura 9-14. Diagrama de flujo de la función Compose_EMAIL

9.2.14 Funciones auxiliares del firmware y de Cloud Computing

El código del firmware desarrollado incluye funciones adicionales para acondicionamiento de datos, validación, sincronización y para invocar el desarrollo de tareas en los servidores de Temboo, dichas funciones se encuentran descritas en la Tabla 9-7 que a continuación se presenta.

Nombre de la función	Tipo	Parámetro de entrada	Descripción
digitalClockDisplay	Void	NA	Imprime por el puerto serial en pantalla la fecha y hora exacta una vez actualizado la hora del microcontrolador usando los servidores NTP
printDigits	Void	Int digits	Antecede el número cero a los días, meses, horas, minutos, segundos que sean menores al número 10. El resultado que se obtiene es que devuelve 08 en lugar de 8.
SendEmailAlertsChoreo	Void	String Alarm Code	Esta rutina invoca la ejecución de una tarea en Temboo que consiste enviar un correo electrónico a la dirección especificada en el encabezado del código con los datos del servidor SMTP también declarados. Esta función además requiere que se le provea el código de alarma ya que manda a llamar a la función Compose_EMAIL el cual requiere de este dato para dar detalle de las alarmas disparadas en el dispositivo de monitoreo.
MakeLogEntryChoreo	Void	String Alarm Code	Esta rutina invoca la ejecución de una tarea en Temboo que consiste crear un renglón en una hoja de cálculo en línea manejada a través de Google Drive, esta función requiere de los datos del documento y los tokens de autorización de Google declarados en el encabezado del código para que pueda entrar en función. Esta rutina manda a llamar a otras para dar formato a los datos que se requieren para hacer la entrada en el registro.
SendDataChoreo	Void	String Alarm Code	Esta rutina invoca la ejecución de una tarea en Temboo que consiste actualizar un renglón definido en una hoja de cálculo en línea manejada a través de Google Drive, esta función requiere de los datos del documento y los tokens de autorización de Google declarados en el encabezado del código para que pueda entrar en función. Esta rutina manda a llamar a otras para dar formato a los datos que se requieren para hacer la entrada en el registro

Nombre de la función	Tipo	Parámetro de entrada	Descripción
CheckLicenseChoreo	Void	NA	Esta rutina se ejecuta al reiniciar el equipo y cada día a la hora definida en el encabezado del firmware y consiste en invocar una tarea en los servidores de Temboo para consultar una celda especifica en una hoja de cálculo ubicada en los servidores de almacenamiento de Google y esta celda contiene el token de seguridad requerido para poder ejecutar las rutinas de transmisión de datos invocadas en las funciones SendDataChoreo y MakeLogEntryChoreo.
sendSMS	Void	char txtMsg char Number	Esta rutina acepta un mensaje de texto de hasta 200 caracteres y el número telefónico al que se desea contactar y se utiliza la librería GSM del Arduino MEGA para invocar a la tarjeta celular a enviar el mensaje SMS.
testconnection	Int	NA	Esta rutina invoca una tarea básica de prueba en los servidores de Cloud Computing con Temboo en el que se regresa un valor de prueba. En caso de que la prueba sea exitosa regresa un valor de 1. En caso de que la prueba falle por cualquier circunstancia regresa un valor de 0. Esta rutina es ejecutada periódicamente por la función loop y antes de querer transmitir un mensaje de alarma para verificar la calidad de la conexión y manipular el *Health Level* en caso de falla en comunicación.
getNtpTime	Time_t	NA	Esta función recibe un paquete UDP de un servidor NTP con la hora exacta proveniente del NIST. Cada vez que se ejecuta la rutina se envía un paquete de datos solicitando la respuesta del servidor con la hora y se queda a la espera de la respuesta para actualizar los datos internos del reloj.
sendNTPpacket	Void	IPAddress	Envía un mensaje de solicitud de respuesta a un servidor NTP especificado por la dirección declarada en IPAddress. Esta función es ejecutada por getNtpTime

Tabla 9-7. Resumen de funciones auxiliares del firmware

10 Integración de Dispositivo de Monitoreo

En esta sección se pretende documentar la cantidad y materiales requeridos para la construcción mediante un *Bill of Materials (BOM)* y ensamble del dispositivo de monitoreo, así como el procedimiento generalizado de construcción del dispositivo y dar a conocer la metodología que se siguió para la realización de pruebas de calificación y poder asegurar la correcta integración del dispositivo de monitoreo en una subestación eléctrica sin interferir en el esquema de control existente.

10.1 BOM

La Tabla 10-1 comprende el detalle en cantidad, número de parte además del proveedor de cada elemento que conforman el ensamble del dispositivo de monitoreo de subestaciones, para la construcción del prototipo.

Elemento	Proveedor	Descripción	NP	Cantidad
WATCHDOG	SparkFun	SparkFun MegaShield Kit	DEV-09346	1
WATCHDOG	SparkFun	Pro Micro - 5V/16MHz	DEV-12640	1
WATCHDOG	STEREN	3 pole 2.54mm 0.1" PCB Universal Screw Terminal Block Connector	TRT-03	1
WATCHDOG	stisfyelectronics eBay	40pin 2.54 breakaway male header	NA	1
WATCHDOG	STEREN	Sunhold Relay 1 pole NO NC 5VDC THD-0501L	THD-0501L	1
WATCHDOG	STEREN	LED 5mm Rojo Difuso	E5 ROJ D	1
WATCHDOG	STEREN	LED 5mm Ambar Difuso	E5 AMB D	1
WATCHDOG	STEREN	LED 5mm Verde Difuso	E5 VER D	1
WATCHDOG	atomindustries eBay	LED 5mm Azul Difuso	NA	1
WATCHDOG	STEREN	Microswitch push 4 terminales	AU-101	1
WATCHDOG	STEREN	Carbon resistor 1/2W 220ohm 5%	PR 1/2 220	4
WATCHDOG	STEREN	Carbon resistor 1/2W 1kOhm 5%	PR 1/2 1k	1
WATCHDOG	lovesell2013 eBay	1x40 pin 2.54mm 0.1" 19mm long single row breakaway male header	NA	1
DC-DC Converter	timbo-0012 eBay	XL6009 DC-DC Adjustable Step-up converter module	XL6009	1
DC-DC Converter		Férula Weidmuller Celeste (AWG24)	OLM7037662	6

Elemento	Proveedor	Descripción	NP	Cantidad
DC-DC Converter		Férula Weidmuller Naranja (AWG20)	9004440000	2
GSM SHIELD	Amazon kjdElectronics	Arduino GSM Shield 2 A000105	A000105	1
Ethernet SHIELD	Sparkfun	Arduino Ethernet Shield	DEV-11166	1
Main Board	Sparkfun	Arduino Mega 2560 R3	A000067	1
Main Board Case	Amazon SunFounderUS	Arduino Mega 2560 Acrylic Case Enclosure	NA	1
Main Harness	eBay wonderco_buy	DuPont Connector Housing Female 2.54 mm 2X18P	140897385643	1
Main Harness	Amazon TOP HUNTER	DuPont socket connector	NA	32
Main Harness		Férula Weidmuller Celeste (AWG24)	OLM7037662	124
Main Harness	Amazon GikFun	2 row Male-Male breakaway header 2x40pin 2.54mm	NA	1
Main Harness	Amazon LLC	Remington 24AWG Stranded Cable Yellow 300V	NA	1
Secondary Harness	Amazon NYING	DuPont Connector Housing Female 2.54mm 1X8P	NA	1
Secondary Harness	Amazon TOP HUNTER	DuPont socket connector	NA	8
Secondary Harness		Férula Weidmuller Celeste (AWG24)	OLM7037662	4
Power Supply	SEL	SEL-9321 Low-Voltage DC Power Supply	SEL-9321	1
DAQ	Dataforth	16Ch DI/O Back Panel with terminal block and din rail mount	SCMD-PB16TSMD	2
DAQ	Dataforth	Digital Input Conditioner 140V ACDC / 5VCD MIAC5	SCMD-MIAC5	32
Enclosure	METCASE OKWUSA	Rackmount 4U Vented Case BLACK	M6219469	1
Enclosure	METCASE OKWUSA	Internal Mating Plate	M6200365	1
Enclosure	OnlineComponents	Marathon Kulka Screw Feedthoug Solder Snug	699-GP-210410	4
Enclosure	Amazon LLC	ASI RJ45 Panel Mount Feedthrough connector	ASICPICRJ45S	1
Accesory	Amazon LLC	Remington 24AWG Stranded Cable Red 600V	NA	1
Accesory		Férula Weidmuller Celeste (AWG24)	OLM7037662	128
Accesory		DIN RAIL 0.5m		1
Accesory	Amazon Pilots HQ	Torque Seal	NA	1

Tabla 10-1. Bill of Materials del Dispositivo de monitoreo de subestaciones

10.2 Ensamble de componentes del dispositivo

El ensamble del dispositivo de monitoreo de subestaciones obedece al diagrama de conexión de subsistemas representado en la Figura 6-3. A continuación, se presenta evidencia fotográfica que documenta la construcción del dispositivo.

El equipo de monitoreo y sus componentes se ensamblan sobre una placa del proveedor MetCase OKW USA, las especificaciones sobre esa placa se detallan en la Figura 10-1, todo el ensamble del dispositivo sucede con este número de parte y es colocado en un gabinete para montaje en rack de 19” bajo las especificaciones detalladas de la Figura 10-2. El detalle de estos componentes del proveedor MetCase OKW USA se encuentran localizados en el apéndice A de este documento.

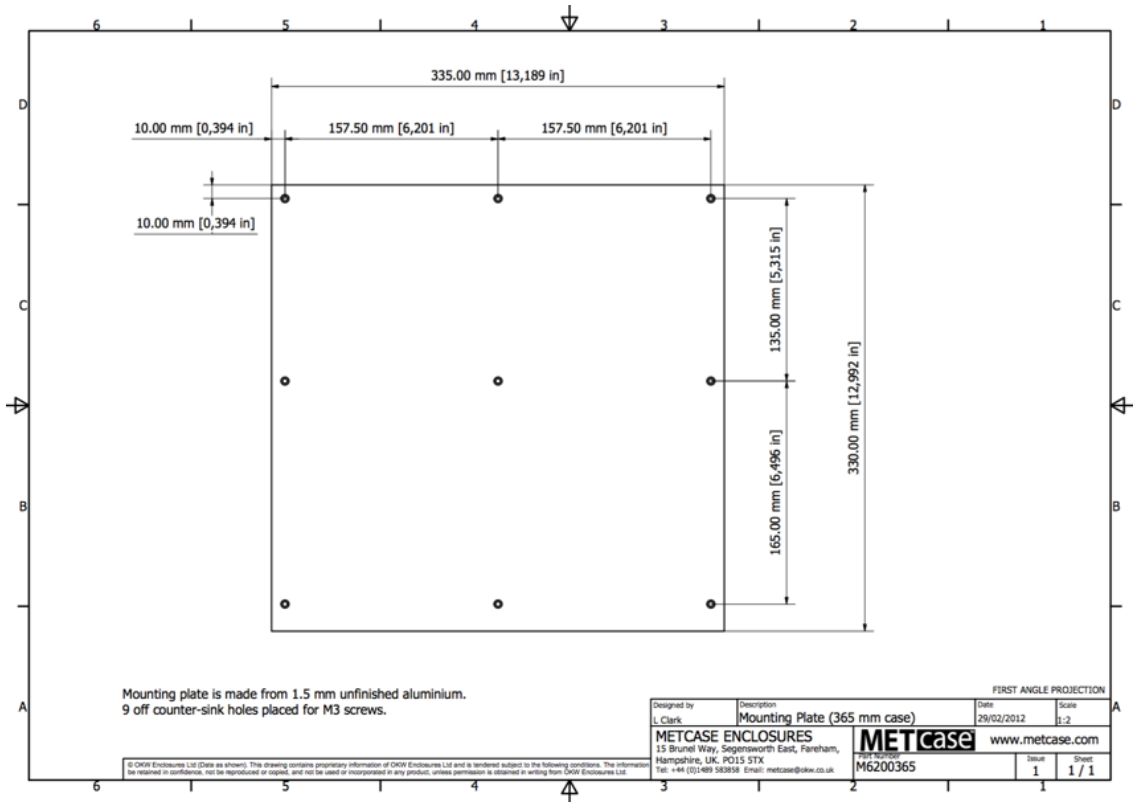


Figura 10-1. Placa de Montaje de PCB

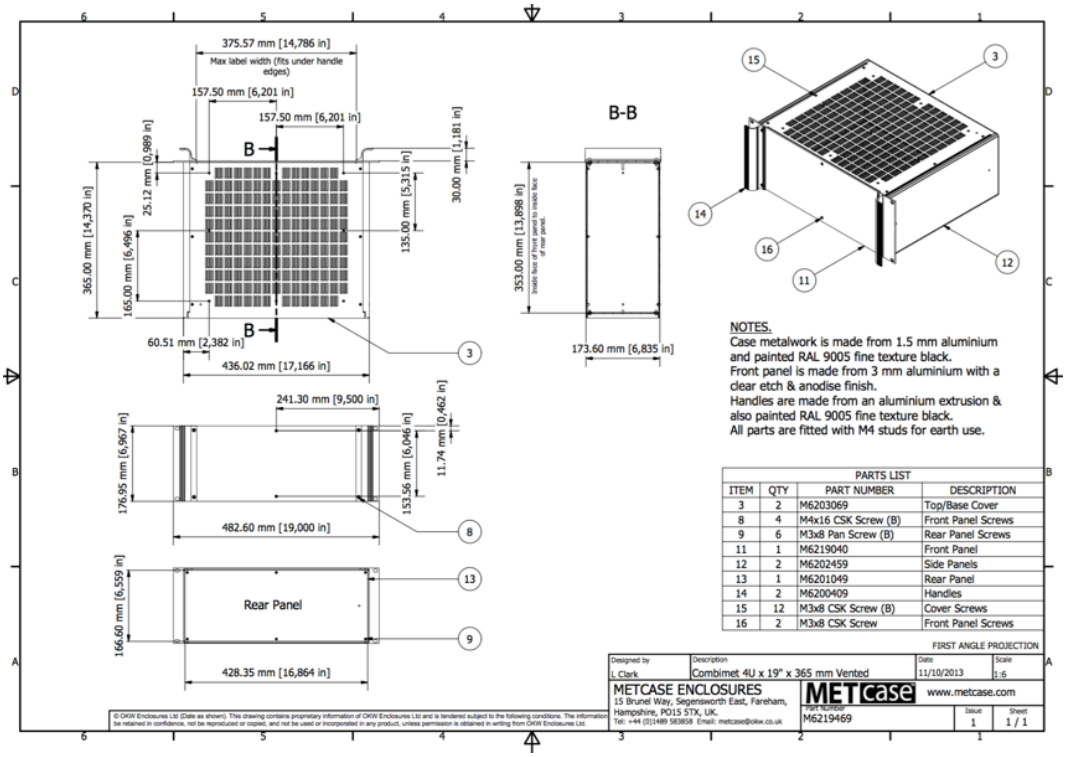


Figura 10-2. Gabinete para montaje en rack

La Figura 10-3 y Figura 10-4 muestran el ensamble final del dispositivo con todos los componentes conectados y colocados en posición.

Figura 10-3. Ensamble del dispositivo de monitoreo

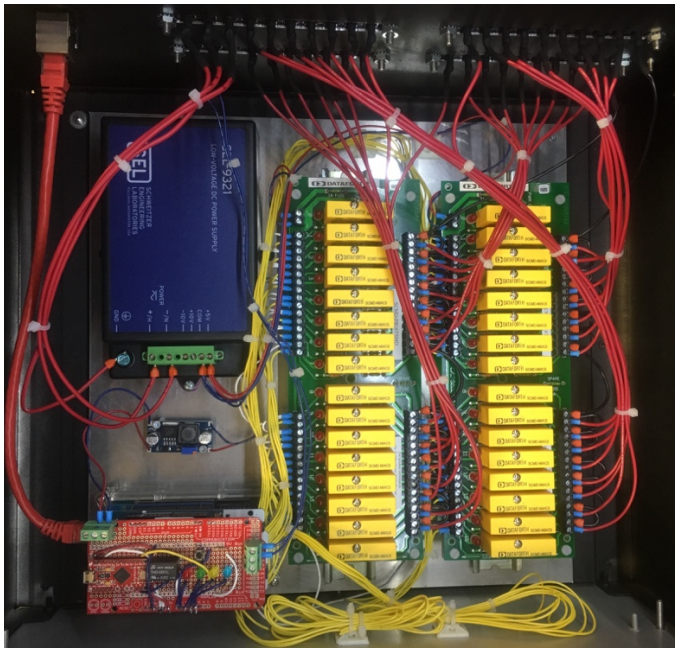


Figura 10-4. Ensamble del dispositivo de monitoreo

La Figura 10-5 muestra el detalle de la construcción de la tarjeta que funciona como *watchdog* del dispositivo, esta se ensambla sobre el Arduino MEGA 2560 como cualquier otro accesorio, observe que la tarjeta del *watchdog* tiene en si montado el Arduino Pro Micro al cual se le carga el firmware que tiene por objetivo proteger contra inhibición de software al dispositivo de monitoreo. La Figura 10-6 muestra el esquemático de conexión de esta tarjeta en particular y los puntos de contacto con el controlador principal.

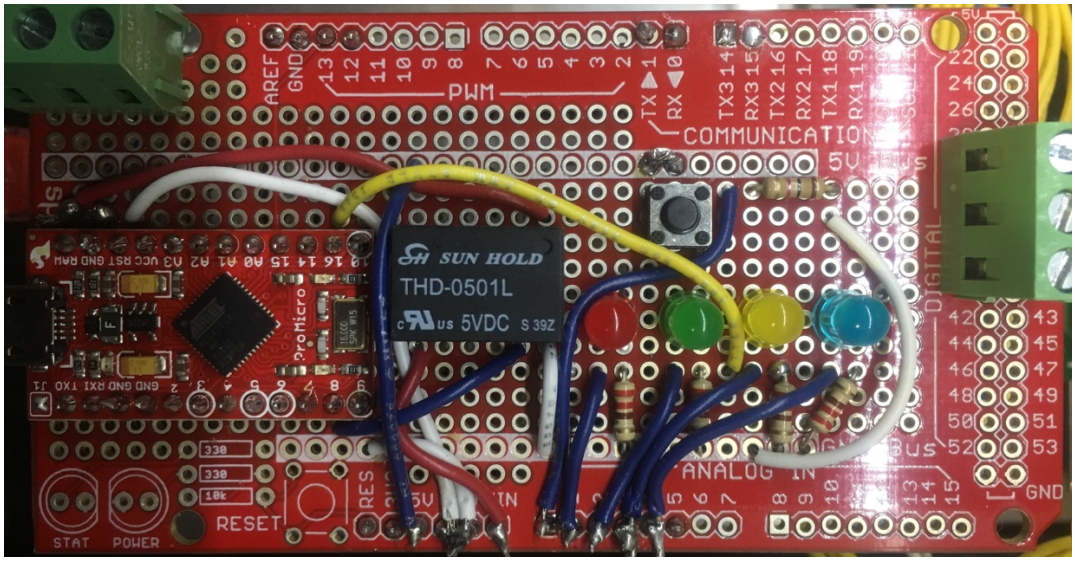


Figura 10-5. Detalle de ensamble del Watchdog del dispositivo

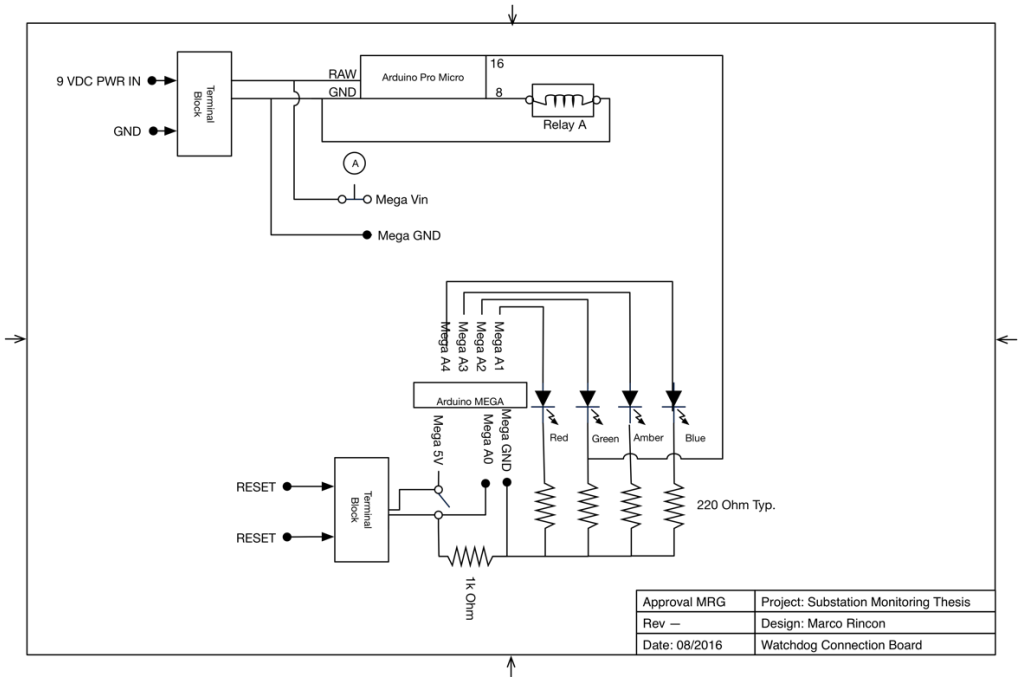


Figura 10-6. Diagrama de conexión del watchdog

El detalle del ensamble de tarjetas en vertical al Arduino principal se detalla en la Figura 10-7 y consiste en colocar la tarjeta Ethernet, la tarjeta GSM y el *watchdog* en posición con el MEGA 2560 para puedan interactuar entre sí.

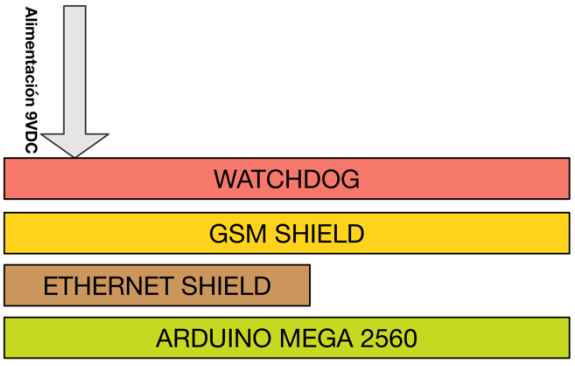


Figura 10-7. Ensamble de Arduino con Shields

Por otro lado la conexión del controlador con los módulos de acondicionamiento de señal se realiza mediante dos arneses cuyo diseño viene detallado en el Apéndice A y una referencia grafica de la conexión se detallan en la Figura 10-8, Figura 10-9 y Figura 10-10.

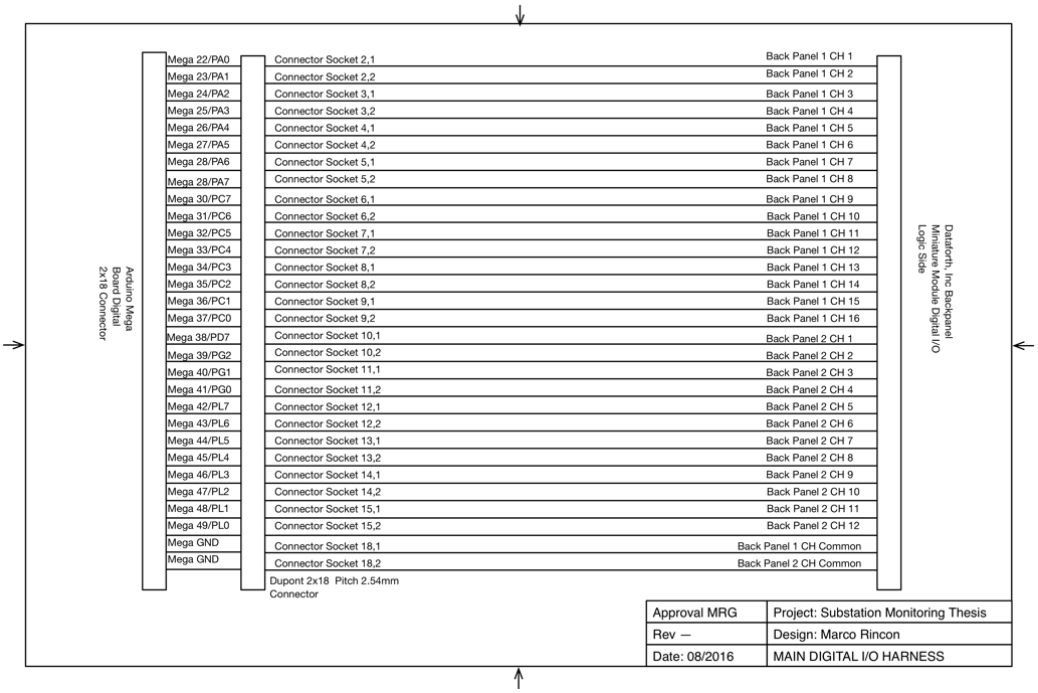


Figura 10-8. Diseño de arnés principal

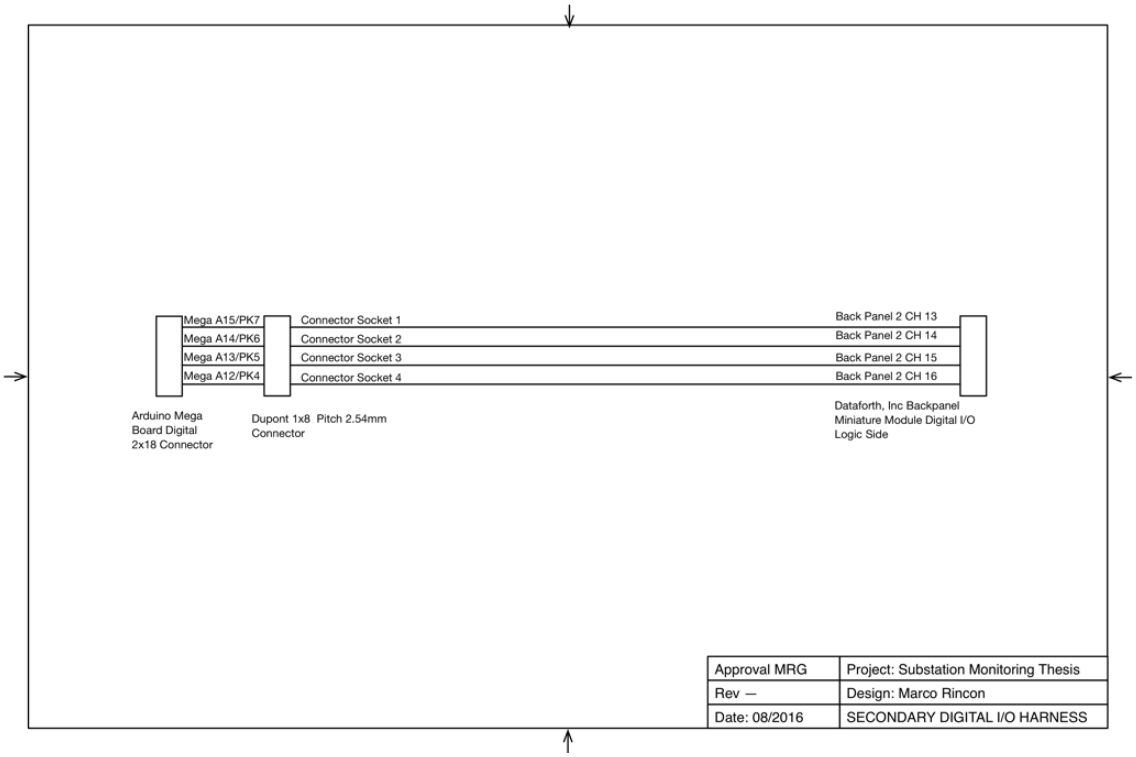


Figura 10-9. Diseño de arnés secundario

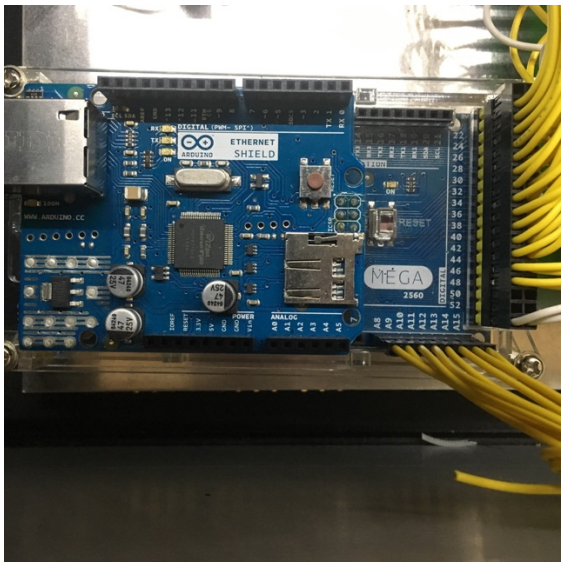


Figura 10-10. Detalle de la conexión de los arneses con el controlador

Los diferentes componentes del dispositivo de monitoreo de subestaciones obedecen el diagrama de conexiones que se presenta en la Figura 10-11 y en el Apéndice A del documento. Por otra parte, la única fuente de alimentación al equipo proviene del banco de baterías y este componente (SEL9321) reduce el voltaje a 5VDC que posteriormente es acondicionado a través de un convertidor DC-DC a 9VDC para el controlador principal del equipo. En lo que corresponde a las tablillas de acondicionadores estas son alimentados directamente de la fuente de 5 VDC.

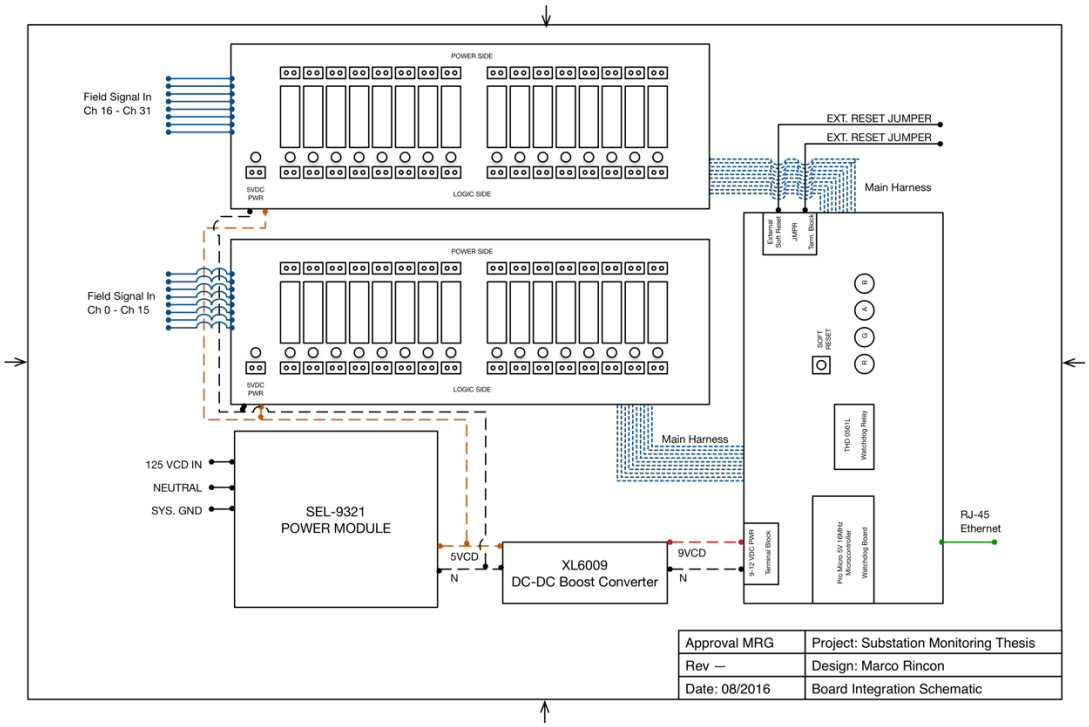


Figura 10-11. Diagrama de conexiones internas del dispositivo

Note que en el diagrama se especifican los arneses de conexión entre las tarjetas de acondicionamiento y el controlador principal. Finalmente, la Figura 10-12 muestra el esquemático de conexión de la tapa posterior del gabinete.

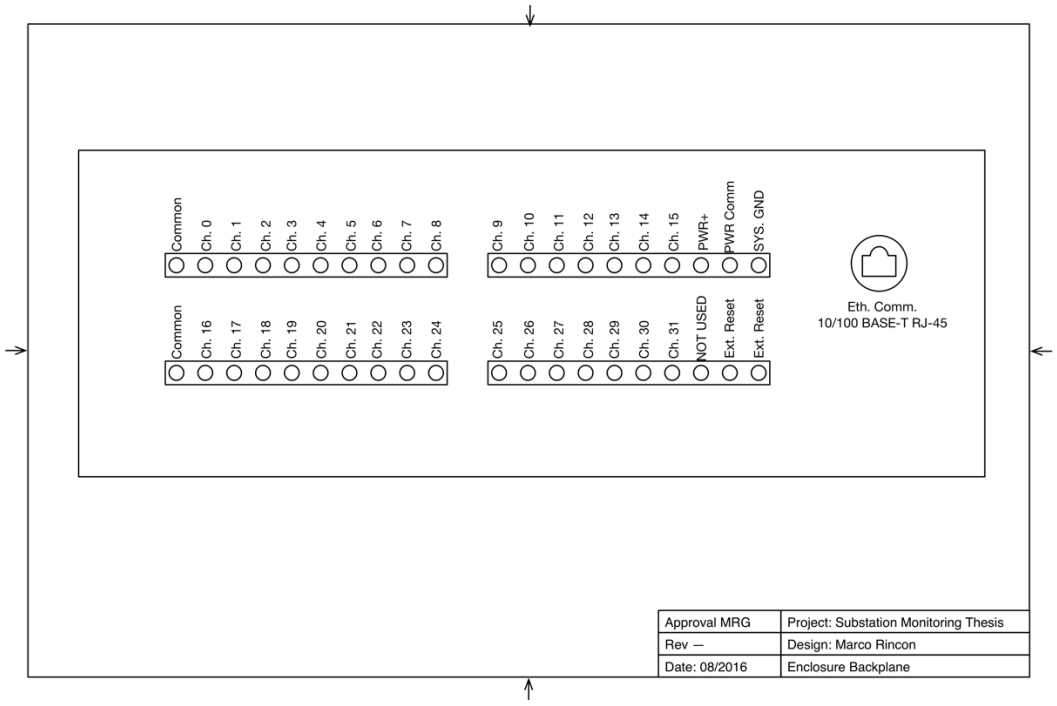


Figura 10-12. Diseño posterior de dispositivo de monitoreo

10.3 Pruebas de Integración del dispositivo

El dispositivo de monitoreo de subestaciones fue sometido a una campaña de pruebas que consistieron en tres etapas: pruebas a nivel prototipo, pruebas *stand alone* y pruebas de integración del sistema; todas las pruebas se llevaron a cabo en el transcurso de un periodo de 4 a 6 meses.

La primera etapa a nivel prototipo consiste en realizar las primeras versiones del firmware y madurarlo con un proceso de depuración hasta que cualquier error de programación sea corregido y se pueda obtener el comportamiento que se desea que obtenga la unidad estando en campo, también es objetivo de esta etapa probar el canal de comunicación, así como el concepto de *cloud computing.* Durante el desarrollo de esta prueba no fue necesaria la integración de todos los componentes de hardware puesto que es una prueba la cual únicamente se concentra en la validación del firmware. La Figura 10-13 muestra el prototipo utilizado para probar el firmware, el cual consistió en utilizar únicamente un controlador Arduino Mega con un *protoboard* para realizar simulaciones de alarmas.

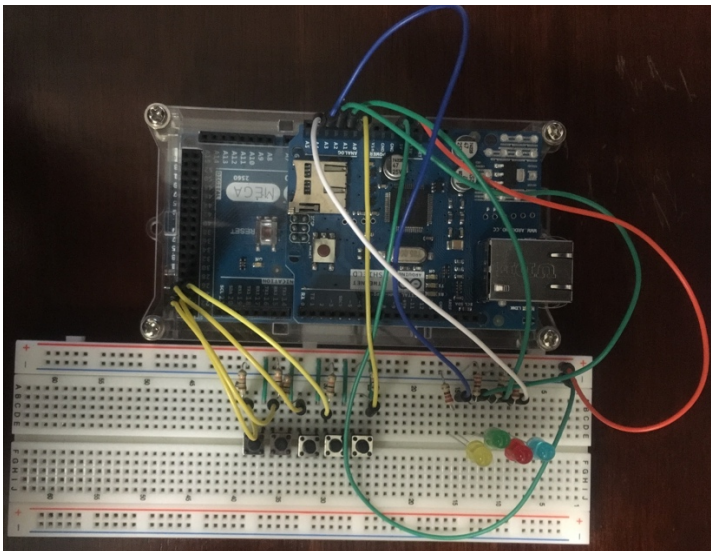


Figura 10-13. Prototipo de prueba

El firmware pasó por sus primeras etapas de maduración utilizando ese prototipo hasta la obtención de la primera versión suficientemente para garantizar que pudiera mantenerse por un periodo prolongado de tiempo sin intervención alguna; ya que, en campo, los equipos deben de operar por largos periodos de tiempo en los que la intervención por parte del usuario debe de ser nula. Durante esta etapa, el firmware

estuvo en pruebas por un periodo de un mes, realizándose actualizaciones hasta el punto en que todo el firmware fuera debidamente depurado.

La segunda etapa de pruebas desarrollada se llevó a cabo con el dispositivo de monitoreo completado al cien por ciento; es objetivo de esta prueba que la unidad se mantenga en estado operativo sin intervención alguna, monitoreando esporádicamente alarmas simuladas en laboratorio. El firmware cargado en la unidad no cambió a la misma frecuencia que en la etapa de pruebas anteriores ya que se quieren conocer los efectos del firmware cargado a largo plazo por lo que esta prueba se desarrolló a lo largo de 3 meses. Al final del periodo de pruebas se detectó que la tarjeta GSM se inhibe si esta se enciende y coincide que no encuentra señal del operador del proveedor de servicios; esto se resolvía simplemente encendiendo y apagando la tarjeta GSM por lo que se optó por el desarrollo del *watchdog* del dispositivo que hasta ese entonces no se había tomado en cuenta.

Durante el desarrollo de la segunda etapa de pruebas se estuvo realizando hasta con tres controladores simultáneos lo que permitía ver el efecto de largo plazo en uno de ellos y al mismo tiempo analizar con el resto de las unidades posibles soluciones a cualquier problema que se presentara con el equipo bajo prueba de referencia. La Tabla 10-2 muestra un resumen de las características de los tres controladores utilizados que al final del periodo de prueba sumaron 15 meses de maduración de software.

Características	MSN 001	MSN 002	MSN 003
LAN	Si	Si	Si
GSM	Si	No	No
Alarmas Habilitadas	32/32	4/32	4/32
Correo Matutino	Si	Si	Si
Correo por Evento	Si	Si	Si
Detección de Perdida de comunicación	Si	Si	Si
Tiempo de muestreo para transmisión rutinaria	15 minutos	Cada hora	Cada hora
Tiempo de muestreo DAQ	1ms (máx.)	1ms (máx.)	1ms(máx.)
Ventana de transmisión de alarma	2 minutos	2 minutos	2 minutos
Versión de Firmware al inicio de la campaña	Versión 11	Versión 11	Versión 11

Características	MSN 001	MSN 002	MSN 003
Versión de Firmware al final de la campaña	Versión 18	Versión 18	Versión 18

Tabla 10-2. Resumen de campaña de pruebas

Finalmente, se tuvo que considerar realizar una prueba que involucrara el circuito de control de la subestación y esto consiste en la tercera etapa de la campaña de pruebas en la que se buscó realizar una integración en un ambiente controlado para verificar que el dispositivo de monitoreo de subestaciones no interfiere con los cuadros de alarma existentes en el mercado. Para ello se realizó el emparalelamiento del dispositivo con un anunciador de alarma de la marca SEL y las alarmas estuvieron simuladas con un equipo de prueba para relevadores de protección de la marca MEGGER.

Esta prueba busca demostrar los siguientes objetivos en particular:

i. Replicar el voltaje de operación en campo al que se va a someter la unidad ya que todas las pruebas hasta ese entonces se habían hecho a 127VCA y no a 125VCD, por lo que la especificación del aislamiento se validaría con esta prueba.
ii. Comprobar que al conectar en paralelo el dispositivo de monitoreo de subestaciones a un canal del panel de alarmas SEL ninguno de los equipos detecte alarma en condiciones de no alarma.
iii. Comprobar que al conectar en paralelo el dispositivo de monitoreo de subestaciones a un canal del panel de alarmas SEL ambos equipos detecten alarmas en condiciones de alarma.
iv. Comprobar que no hay interferencia en la operación entre ambos equipos.
v. Demostrar que el canal de comunicación es funcional y transmite datos de manera oportuna a todas aquellas personas involucradas en la lista de distribución.

La Tabla 10-3 muestra el resumen de la prueba final de integración y muestra que todos los objetivos fueron cumplidos, además la Figura 10-14, Figura 10-15, Figura 10-16 muestran evidencia fotográfica de la prueba realizada y en la Figura 10-17 se muestra un diagrama de conexión de la prueba de integración.

Prueba de Integración de Sistema	
Equipo de prueba	Megger MPRT 8430
Equipo de referencia	SEL-2522 PN 25220X1340XX
Dispositivo de Monitoreo	MSN001
Versión del Firmware	V19
Prueba de resistencia de aislamiento	Satisfactoria
Prueba de emparalelamiento en condiciones de no alarma	Satisfactorio
Prueba de emparalelamiento en condiciones de alarma	Satisfactorio
Prueba de no interferencia entre dispositivos	Satisfactorio
Prueba de canal de comunicación	Satisfactorio

Tabla 10-3. Resumen de prueba de integración de sistema

Figura 10-14. Equipo de prueba para validación de integración de dispositivo

Figura 10-15. Dispositivo de monitoreo en prueba de integración

Figura 10-16. Panel de Alarma SEL para referencia en prueba de integración

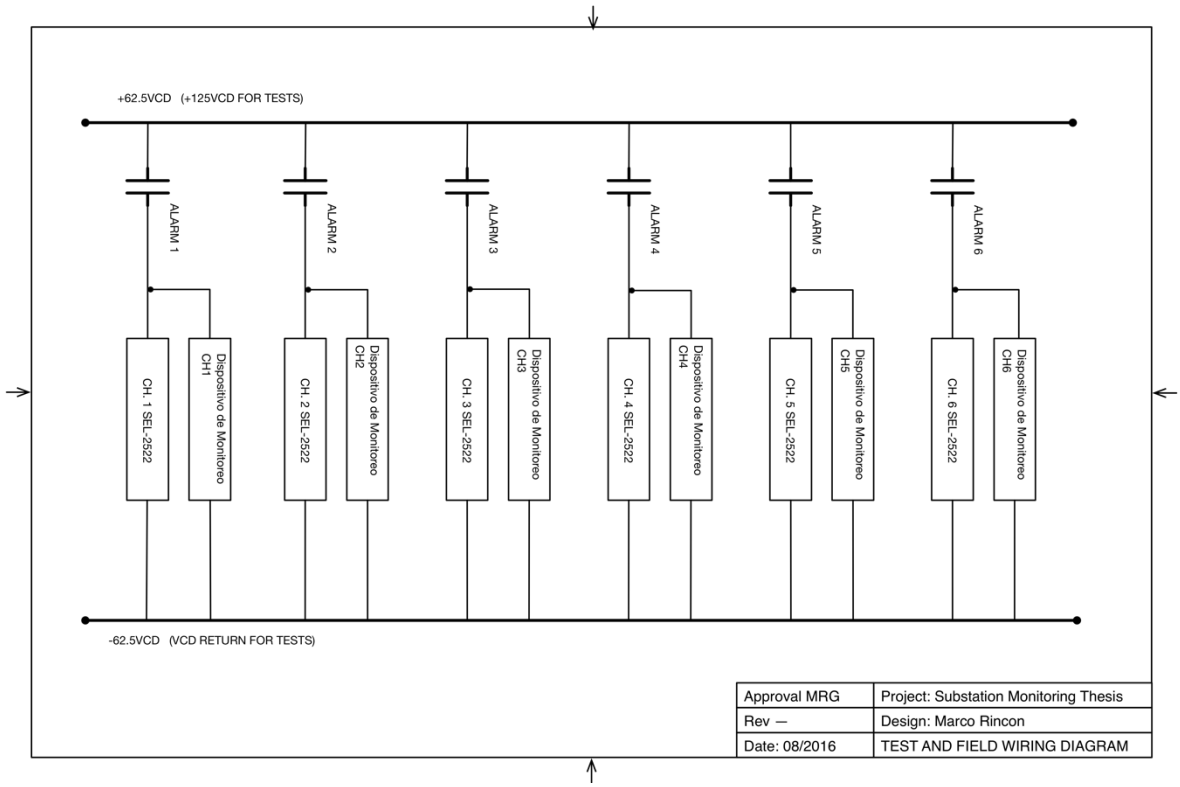


Figura 10-17. Diagrama de conexión del dispositivo

Es objetivo de toda la campaña de pruebas poder llegar a una plataforma que sea lo suficientemente madura como para poder realizar pruebas y ser llevado a una subestación eléctrica de alto potencia pública o privada para demostrar la efectividad del dispositivo de monitoreo.

11 Caso de aplicación

Parte de los objetivos de este documento de Tesis es demostrar la efectividad del dispositivo en una instalación eléctrica además de documentar aquellos efectos a largo plazo de estar exponiendo la unidad a un ambiente diferente al de un laboratorio de pruebas; por lo que esta sección documenta tal hecho. La búsqueda del lugar de instalación consistió en encontrar una subestación que tuviera para practicidad un nodo de red cercano y un cuadro o panel de alarmas que concentrara todas las señales en un solo punto. El lugar de instalación del dispositivo seleccionado fue en la ciudad de Nuevo Laredo, Tamaulipas en una subestación de 138kV a 13.8kV Y / 7.97kV de 11MVA con 31 alarmas diferentes para monitorear cada uno de los equipos de potencia que en ella se encuentran instalados.

11.1 Descripción de la instalación eléctrica

La ubicación de la subestación se encuentra en la ciudad de Nuevo Laredo, Tamaulipas en México en una instalación privada de 138kV a 13.6kV Y / 7.97kV con tres transformadores de 3.75 / 4.68 / 5.25 MVA mas un cuarto transformador de reserva. La instalación cuenta con un interruptor de línea de tanque muerto con SF6 con control neumático, apertura de cuchillas con mandos remotos e interruptores de potencia para distribución interna a subestaciones derivadas y otros servicios de la planta. En cuanto al área de protecciones, la subestación está cubierta con protección diferencial (87) para cada una de los transformadores y protección 50/51N para cada uno de los transformadores y para el interruptor principal.

Con referencia al cuadro de alarmas de la instalación se cuenta con uno de la marca SYS-UCAD el cual concentra y anuncia todas las anomalías detectadas dentro de su margen de cobertura en la instalación y es a este dispositivo al que se realizó un emparalelamiento con el dispositivo de monitoreo que en este documento se describe. Aunque la capacidad del panel de alarmas existente en la instalación puede monitorear hasta 64 señales solamente puede anunciar 32 y se encuentran listadas en la Tabla 11-1 que además muestra el canal del dispositivo al que es

conectado, si esta es vigilada con el dispositivo de este documento y el tipo de lógica que esa señal maneja.

Nombre de Alarma	Canal	Monitoreada por el dispositivo	Lógica de operación
BAJO NIVEL DE ACEITE EN TR1	01	SI	NO NEGADA
ALTA TEMPERATURA DE ACEITE EN TR1	02	SI	NO NEGADA
DISPARO POR RELEVADOR 86T1 EN TR1	03	SI	NEGADA
APERTURA VALVULA SOBREPRESION TR1	04	Si	NO NEGADA
BAJA PRESION DE NITROGRENO N2 TR1	05	Si	NO NEGADA
ALTA TEMPERATURA DEVANADO TR1	06	Si	NO NEGADA
BAJO NIVEL DE ACEITE TR2	07	SI	NO NEGADA
ALTA TEMPERATURA DE ACEITE EN TR2	08	SI	NO NEGADA
DISPARO POR RELEVADOR 86T2 TR2	09	SI	NEGADA
APERTURA VALVULA SOBREPRESION TR2	10	SI	NO NEGADA
BAJA PRESION DE NITROGENO N2 TR2	11	SI	NO NEGADA
ALTA TEMPERATURA DEVANADO TR2	12	SI	NO NEGADA
DISPARO POR RELEVADOR 86T3 TR3	13	SI	NO NEGADA
BAJO NIVEL DE ACEITE TR3	14	SI	NO NEGADA
ALTA TEMPERATURA DE ACEITE TR3	15	SI	NO NEGADA
APERTURA VALVULA SOBREPRESION TR3	16	SI	NO NEGADA
BAJA PRESION DE NITROGENO N3 TR3	17	SI	NO NEGADA
ALTA TEMPERATURA DEVANADO TR3	18	SI	NO NEGADA
BAJA PRESION DE SF6 INTERRUPTOR 52L	19	SI	NO NEGADA
BAJA PRESION DE AIRE INTERRUPTOR 152L	20	SI	NO NEGADA
DISPARO DE INTERRUPTOR 152L	21	SI	NO NEGADA
APERTURA DE CUCHILLA 189T1	22	SI	NO NEGADA
APERTURA DE CUCHILLA 189T2	23	SI	NO NEGADA
APERTURA DE CUCHILLA 189T3	24	SI	NO NEGADA
DISPARO DE INTERRUPTOR 52 T1	25	SI	NO NEGADA
DISPARO DE INTERRUPTOR 52 T2	26	SI	NO NEGADA
DISPARO DE INTERRUPTOR 52 T3	27	SI	NO NEGADA
BAJO VOLTAJE CA CARGADOR BATERIAS	28	SI	NO NEGADA
BAJO VOLTAJE CD CARGADOR BATERIAS	29	SI	NO NEGADA
FALLA A TIERRA EN SISTEMA DE CD	30	SI	NO NEGADA
ALARMA GENERAL TR4 RESERVA	31	SI	NO NEGADA
SPARE	32	NO	NO NEGADA

Tabla 11-1. Alarmas monitoreadas por el dispositivo

La Figura 11-1, Figura 11-2, Figura 11-3 y Figura 11-4 muestran detalles de la subestación eléctrica en la que el dispositivo de monitoreo se instaló para realizar pruebas.

Figura 11-1. Subestación principal T1, T2 y T3

Figura 11-2. Bus subestación principal

Figura 11-3. Interruptor de línea subestación principal

Figura 11-4. Cuarto de Control Subestación Principal

También se muestra el diagrama unifilar simplificado de la subestación en la Figura 11-5 donde se muestra lo anterior descrito en esta sección.

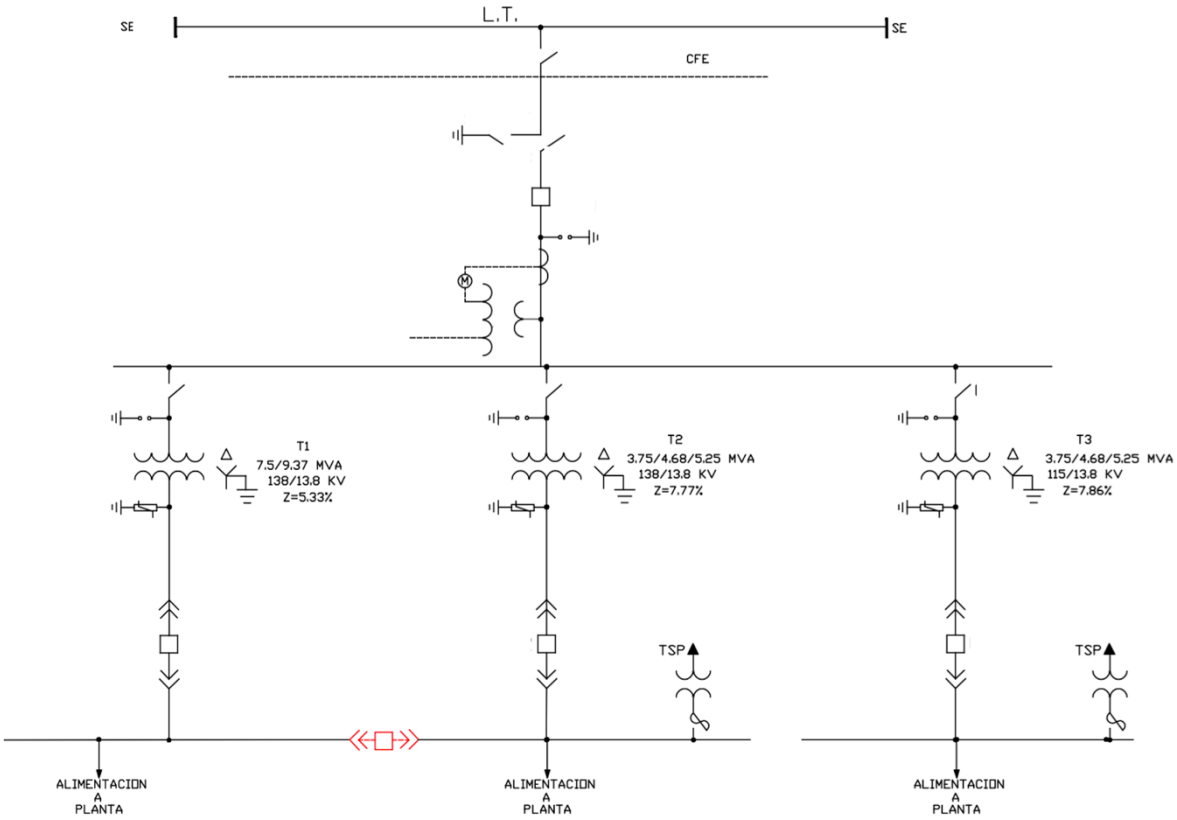


Figura 11-5. Diagrama unifilar simplificado de subestación principal

11.2 Diagrama de control del panel de alarma

El panel de alarmas existente en la ubicación de la subestación eléctrica es de la marca SYS-UCAD que tiene la capacidad de enviar correos electrónicos, se conectará en paralelo el dispositivo de monitoreo para medir la efectividad del mismo. La conexión se realizará de manera similar a la indicada en la Figura 10-17 variando únicamente la marca del cuadro de alarmas y la cantidad de alarmas. También se respetará la lógica de operación de las alarmas indicadas en la Tabla 11-1.

Para referencia, se encuentra en la Figura 11-6 el diagrama de conexión de un panel SYS-UCAD de una subestación eléctrica similar la cual sirve de guía para determinar los puntos de interconexión y el común lógico de las señales de alarma que se desean monitorear.

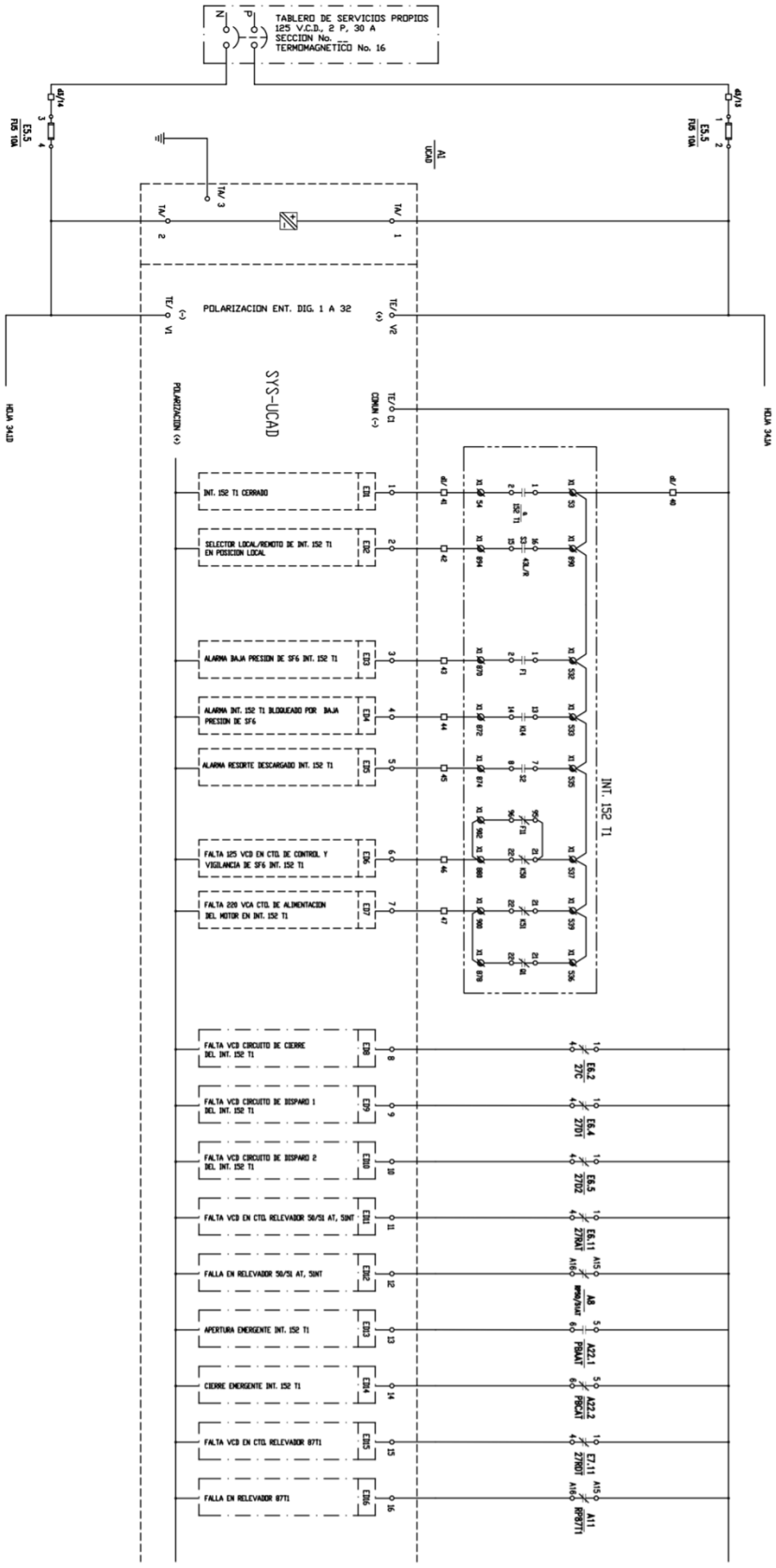


Figura 11-6. Diagrama de conexión del SYS-UCAD

11.3 Evidencia fotográfica de la instalación del dispositivo

El dispositivo de monitoreo de subestaciones eléctricas se instaló durante el día 27 de junio del 2016 en la ubicación anteriormente descrita, sin mayor complicación que realizar una actualización en campo de último minuto para que se pudiera discriminar por canal si la lógica de la alarma es negada o no lo es.

A continuación en las siguientes figuras (Figura 11-7, Figura 11-8, Figura 11-9, Figura 11-10, Figura 11-11, Figura 11-12 y Figura 11-13) se presenta evidencia fotográfica que captura el momento de instalación del dispositivo dentro del cuarto de control de la subestación, así como el ajuste de los últimos detalles del firmware antes de liberar el dispositivo para que ejecute el monitoreo remoto.

Figura 11-7. Panel de Alarmas SYS-UCAD existente

Figura 11-8. Gabinete donde estará ubicado el dispositivo

Figura 11-9. Tablillas de interconexión del dispositivo con el panel existente

Figura 11-10. Actividades de instalación de dispositivo de monitoreo

Figura 11-11. Actividades de instalación de dispositivo de monitoreo

Figura 11-12. Dispositivo de monitoreo alambrado y listo para operar

Figura 11-13. Ajustes finos al dispositivo de monitoreo

11.4 Comparación económica contra sistema similar

A continuación, se muestra una comparativa entre la elaboración del dispositivo de monitoreo de subestaciones y el precio de venta de la solución comercial más cercana. Es de notar y muy importante mencionar que la solución en el mercado llega a tener una capacidad mucho mayor que la de simplemente realizar el monitoreo; sin embargo, es de aquellas soluciones con costos más viables entre las diferentes posibilidades que se pueden encontrar. No obstante, esto no garantiza que no se deba de adquirir equipo especializado como lo es una infraestructura de red bien definida, servidores, entre otros equipos para poder lograr exclusivamente lo que el dispositivo de monitoreo puede realizar.

La Tabla 11-2 muestra los costos por la compra de equipo especializado de la marca SEL en dólares americanos en las que se pudiera asegurar poder obtener una plataforma equivalente para realizar un monitoreo, pero dentro de una red local solamente.

Nombre del Producto	Cantidad	Costo	Costo Expandido
SEL 3530-4 Real Time Automation Controller Dispositivo concentrador de señales digitales, RTU y HMI	1	$ 5,040.00	$ 5,040.00
SEL-2730 Unmanaged Ethernet Switch Para comunicación entre panel de alarmas con otros dispositivos y RTAC	1	$ 1,869.00	$ 1,869.00
SEL-2523 Annunciator Panel with Communicatins Permite la digitalización del circuito de alarmas de una subestación.	1	$ 3,990.00	$ 3,990.00
TOTAL EN DOLARES AMERICANOS			$10,899.00

Tabla 11-2. Cotización de equipo para una solución alterna

Por otro lado, la construcción del dispositivo de monitoreo se llevó con los materiales que se describen en la Tabla 11-3. Estos materiales tienen un costo mucho menor al ser adquiridos en mercado.

Elemento	Descripción	NP	Cantidad
WATCHDOG	SparkFun MegaShield Kit	DEV-09346	1
WATCHDOG	Pro Micro - 5V/16MHz	DEV-12640	1
WATCHDOG	3 pole 2.54mm 0.1" PCB Universal Screw Terminal Block Connector	TRT-03	1
WATCHDOG	40pin 2.54 breakaway male header	NA	1
WATCHDOG	Sunhold Relay 1 pole NO NC 5VDC THD-0501L	THD-0501L	1
WATCHDOG	LED 5mm Rojo Difuso	E5 ROJ D	1
WATCHDOG	LED 5mm Ámbar Difuso	E5 AMB D	1
WATCHDOG	LED 5mm Verde Difuso	E5 VER D	1
WATCHDOG	LED 5mm Azul Difuso	NA	1
WATCHDOG	Microswitch push 4 terminales	AU-101	1
WATCHDOG	Carbon resistor 1/2W 220ohm 5%	PR 1/2 220	4
WATCHDOG	Carbon resistor 1/2W 1kOhm 5%	PR 1/2 1k	1
WATCHDOG	1x40 pin 2.54mm 0.1" 19mm long single row breakaway male header	NA	1
DC-DC Converter	XL6009 DC-DC Adjustable Step-up converter module	XL6009	1
DC-DC Converter	Férula Weidmuller Celeste (AWG24)	OLM7037662	6
DC-DC Converter	Férula Weidmuller Naranja (AWG20)	9004440000	2
GSM SHIELD	Arduino GSM Shield 2 A000105	A000105	1
Ethernet SHIELD	Arduino Ethernet Shield	DEV-11166	1
Main Board	Arduino Mega 2560 R3	A000067	1
Main Board Case	Arduino Mega 2560 Acrylic Case Enclosure	NA	1
Main Harness	Duppont Connector Housing Female 2.54 mm 2X18P	140897385643	1
Main Harness	Duppont socket connector	NA	32
Main Harness	Férula Weidmuller Celeste (AWG24)	OLM7037662	124
Main Harness	2 row Male-Male breakaway header 2x40pin 2.54mm	NA	1
Main Harness	Remington 24AWG Stranded Cable Yellow 300V	NA	1
Secondary Harness	Duppont Connector Housing Female 2.54mm 1X8P	NA	1
Secondary Harness	Duppont socket connector	NA	8
Secondary Harness	Férula Weidmuller Celeste (AWG24)	OLM7037662	4
Power Supply	SEL-9321 Low-Voltage DC Power Supply	SEL-9321	1
DAQ	16Ch DI/O Back Panel with terminal block and din rail mount	SCMD-PB16TSMD	2

Elemento	Descripción	NP	Cantidad
DAQ	Digital Input Conditioner 140V ACDC / 5VCD MIAC5	SCMD-MIAC5	32
Enclosure	Rackmount 4U Vented Case BLACK	M6219469	1
Enclosure	Internal Mating Plate	M6200365	1
Enclosure	Marathon Kulka Screw Feedthoug Solder Snug	699-GP-210410	4
Enclosure	ASI RJ45 Panel Mount Feedthrough connector	ASICPICRJ45S	1
Accesory	Remington 24AWG Stranded Cable Red 600V	NA	1
Accesory	Férula Weidmuller Celeste (AWG24)	OLM7037662	128
Accesory	DIN RAIL 0.5m		1
Accesory	Aircraft Tool Supply Torque Seal	NA	1

Tabla 11-3. Desglose de Materiales del dispositivo

Al analizar ambas tablas se deben de realizar las siguientes consideraciones, entre ellas se debe de tomar en cuenta que la Tabla 11-3 no toma en cuenta la mano de obra de fabricación, diseño e instalación del dispositivo mientras que la Tabla 11-2 muestra precios de lista del fabricante. Otra consideración es que los equipo de la Tabla 11-2 son especializados y tienen mayores capacidades que la del dispositivo de monitoreo; sin embargo, es de las soluciones comerciales más viables que pueden cumplir con la tarea de informar oportunamente de las situaciones en la subestación. Una ventaja que ofrece el dispositivo de monitoreo contra los equipos especializados es que este puede monitorear su estado y notificar por diferentes medios la presencia de un evento ante la falla en las comunicaciones.

En caso que se desee utilizar el dispositivo de monitoreo para llevar a cabo tareas de supervisión de la subestación con la capacidad de informar por GSM será necesario adquirir un contrato con algún proveedor de servicios de telefonía celular y tomar en consideración los costos de contratación por ese servicio. Para ambas alternativas, en caso de requerir mensajes de correo electrónico se deberá de considerar adquirir un proveedor de servicios de correo electrónico que permita la conexión al servidor a través de SMTP.

Hay que destacar que, a pesar de las limitaciones y consideraciones de cada una de las alternativas, el costo de fabricación del dispositivo de monitoreo que en esta tesis se presenta provee una solución de bajo costo, pero a la vez funcional.

11.5 Resultados

El dispositivo instalado ha permanecido en la subestación realizando monitoreo con las siguientes características: tarjeta GSM habilitada y con monitoreo activo de 31 alarmas diferentes descritas en la Tabla 11-1; cabe destacar que durante la instalación se tuvo que realizar un ajuste en el software para permitir que ciertas alarmas operaran con lógica negada.

Desde su instalación el equipo ha mandado un total de 4 alarmas: Baja presión de Nitrógeno en el transformador 2 en dos ocasiones con el que se diagnosticó una fuga y el resto por bajo voltaje de CA en el cargador de baterías de la caseta de control debido a una interrupción en el servicio desde CFE. El dispositivo ha enviado un estado diario de sus alarmas y responde apropiadamente a todas las solicitudes para reportar su estado a través de GSM, al igual que lleva registro de todos los eventos suscitados en la subestación en línea a través de Google Drive por medio de Temboo.

Por otro lado, también se ha dado seguimiento a los reinicios del dispositivo gracias al *watchdog* que se le habilitó a la unidad asegurando que el monitoreo se encuentre activo lo más posible de forma ininterrumpida; desde que la unidad se instaló en la subestación se han registrado un total de 3 reinicios en un periodo de 50 días, se debe recordar que el *watchdog* toma la decisión de reiniciar la unidad si el controlador principal le demora más de 10 minutos en completar una tarea y no necesariamente implica un problema o falla grave en el firmware o el equipo; simplemente por seguridad actúa.

No obstante, algo que no se vio durante el periodo de pruebas es que la calidad a la conexión a Internet dentro del laboratorio de pruebas es por mucho superior a la que se encuentra en la subestación en cuanto a confiabilidad y velocidad por lo que de forma muy recurrente se enviaban reportes de problemas de comunicación dando un total de 96 notificaciones relacionadas al tema.

Dado a los hechos registrados se tuvo que modificar el firmware un mes después de haber sido instalado en la subestación de prueba de tal forma que discriminara y fuera muy selectivo en la manera de detectar un fallo de comunicación antes de enviar una notificación al respecto. Esto se resolvió implementando el *Health Level* y se le asignó un nivel de 4, por lo que el dispositivo tendría que fallar cuatro intentos de comunicación o perder la conexión a la red por al menos una hora antes de enviar alertas. Desde el momento de la actualización al firmware el problema quedó resuelto y la unidad dejó de enviar notificaciones de perdida de comunicación por problemas de calidad en la conexión.

En general, el dispositivo ha permitido que se programe un mantenimiento para corregir la fuga en un transformador ya que los responsables del área no realizan inspecciones a la subestación de forma recurrente. Es de esperarse que la reacción del dispositivo para la notificación oportuna de los eventos abra una ventana que permita corregir situaciones de una forma temprana antes de que se agraven y tengan repercusiones económicas mayores. En el caso particular de la fuga de nitrógeno, ésta de manera temporal, se resuelve instalando cilindros de N2 para mantener una presión positiva adentro del transformador; en caso de que el dispositivo no se instalara y pasara mucho tiempo antes de que en una inspección se dieran cuenta de la anomalía, pudiera darse el caso en el cual al transformador le entrara humedad afectando en si al equipo y teniendo una repercusión económica mayor para repararlo o incluso reemplazarlo.

El hecho que el dispositivo envíe un correo matutino reportando el estado de las alarmas en la subestación permite y da confianza a que los usuarios sean notificados que sus instalaciones se encuentren en orden y que el dispositivo de monitoreo de subestaciones se encuentra operacional; además que las consultas de los eventos de la subestación y el estado en tiempo quince minutal es reportado en la nube utilizando *Cloud Computing*. Otra forma en que el dispositivo refuerza la confianza para conocer el estado de la subestación a distancia es mediante el envío de reportes por GSM el cual permite conocer la situación de la instalación eléctrica aún y cuando no se tiene acceso a una computadora.

No obstante, se debe de hacer énfasis en que el dispositivo no es un equipo que sustituya el realizar inspecciones periódicas a la instalación ya que son muchos los problemas que se pueden detectar antes de que un sensor conectado al panel de alarmas se active y este dispositivo solamente ofrece un método de notificación para que de manera oportuna se avise a personal competente y de manera inmediata se tome acción correctiva.

12 Lecciones aprendidas

Es importante mencionar que en muchas ocasiones por más que se estudie y analice una situación las cosas no salen como uno lo planea ya sea porque los equipos o componentes no tienen especificaciones completas, o porque son en realidad mal seleccionados; o en el caso del desarrollo de software el utilizar ciertas librerías puede llegar a limitar e incluso entrar en conflicto cuando se desean usar otras en conjunto. Por lo que se debe de buscar una solución alterna y aprender de la situación para buscar la mejora continua. Esta sección pretende documentar las lecciones aprendidas durante la elaboración de este proyecto. Se enlistan a continuación:

i. El Arduino MEGA 2560 tiene una capacidad bastante limitada para realizar operaciones con strings por lo que su uso debe de moderarse y optimizarse
ii. La tarjeta GSM Shield de Arduino no es compatible de primera mano con el controlador MEGA, se requiere hacer un corto entre dos pins, esto es un problema de interrupciones con el micro.
iii. La tarjeta GSM Shield de Arduino no es compatible de primera mano con el Ethernet shield ya que existe conflicto con el pinout, se debe de realizar una modificación a las librerías para cambiar el pin en conflicto.
iv. La tarjeta GSM Shield de Arduino, no puede ser alimentada desde un puerto USB ya que puede llegar a requerir hasta 2 amperes para realizar transmisión de datos en modo GSM, se debe de utilizar una fuente externa.
v. La librería ICMP Ping no es compatible con la librería Ethernet de Arduino, por lo tanto, no se puede utilizar la función ping como método de detección de perdida de comunicación.
vi. La fuente de poder que se requiere para la integración del dispositivo está limitada a 5VCD 1.0 Amp (su alimentación puede ser de 127 de AC o DC).

Se requiere de un convertidor DC-DC para poder alimentar el Arduino MEGA 2560 con 9VCD.

vii. La tarjeta GSM Shield de Arduino no permite el encendido del modem por comando de software, requiere que un usuario presione un interruptor en el circuito para que este encienda; para evitar esto, se debe de soldar un puente sobre el PCB de la tarjeta.

viii. Al ejecutar el comando de la librería GSM “begin()” existe la posibilidad que la red GSM del proveedor no se encuentre disponible, o la intensidad de la señal no sea la apropiada para la conexión. Esta situación hace que la tarjeta se inhiba y quede congelada, al momento no hay solución por software para evitarlo. Se sugiere involucrar el uso de un *watchdog* como medida de protección ante esta situación.

ix. En la tarjeta Arduino MEGA 2560 no se puede disponer de los pins digitales 50 al 52 ya que entran en conflicto con la tarjeta Ethernet Shield. Se debe de considerar utilizar pins análogos en sustitución.

x. El procesador Arduino MEGA 2560 omite la ejecución del código programado cuando la memoria dinámica utilizada supera al 55% de la capacidad de la misma (8.2kB).

xi. El modem de la tarjeta GSM Shield de Arduino requiere permanecer unos cuantos segundos encendido para permitir la recepción de mensajes SMS. Si se enciende, se corre el comando de verificar si hay SMS en cola y se apaga de forma continua no se le dará suficiente tiempo para que los mensajes entren a la memoria del modem y por lo tanto aparenta que no hay mensajes disponibles que procesar.

xii. El conector compatible con el Arduino MEGA 2560 (*mating connector*) es un Dupont 2X18 Hembra con paso de 2.54mm

xiii. La forma apropiada de realizar el conector es realizando un arnés cuyos conductores sean de calibre 24AWG 300V. Se debe de hacer un crimp con un socket DuPont e insertar en el conector 2X18. La conexión se realiza con un “header” macho-macho.

xiv. El consumo máximo de corriente con el software cargado en el Arduino a través del convertidor DC-DC es de 1.0 AMP y de 600mA nominales. El consumo de un solo acondicionador Dataforth es de 9mA (~320mA con todas las señales activadas).

Las lecciones anteriores deberán de tomarse en cuenta cada vez que se desee realizar un proyecto similar o se trabaje con los mismos componentes, muchas de las lecciones no se encuentran disponibles en foros de discusión y muestran las forma en las que un servidor hizo el *walk around solution*.

13 Recomendaciones para trabajos futuros

A continuación, se presentan mejoras, sugerencias, o nuevas opciones para una segunda versión a futuro del dispositivo de monitoreo de subestaciones. Estas recomendaciones están basadas en las lecciones aprendidas y los resultados a medio plazo de las pruebas realizadas en campo.

Se recomienda buscar formas de optimizar las llamadas al servidor de Cloud Computing (Temboo) para disminuir el uso de rutinas sin tener que sacrificar el tiempo de actualización de información. Esto puede lograrse mejorando la manera en la que se realizan las detecciones de fallos en comunicación.

Se deberá de considerar utilizar nuevas alternativas en controladores, ya que el Arduino MEGA 2560 tiene capacidades limitadas en memoria para manejo de variables tipo string y comportamientos erráticos cuando la memoria volátil alcanza niveles superiores al 55% por lo que para este proyecto se tuvo que implementar el código considerando esa limitación. Se cree que si se cambia la plataforma a una de mayor capacidad como Intel Edison se pueda explotar de una mejor forma el dispositivo de monitoreo de subestaciones. No obstante, un cambio de este tipo forzosamente implica rediseñar por completo el hardware del dispositivo a pesar de que el firmware de operación sea idéntico.

Se sugiere buscar alternativas en la tarjeta de comunicación celular a una más estable y posiblemente utilizar un Arduino MEGA 2560 exclusivamente para esa tarjeta mejore el rendimiento general de todo el dispositivo.

Por limitaciones de memoria no se presentó una interfaz de usuario gráfica, si se considera una nueva plataforma con un controlador diferente se sugiere considerar esta área de mejora, así como proveer de descripción de alarmas en los mensajes SMS en lugar de un código numérico de estado.

Es recomendable para una futura revisión incluir un medio de interpretación que ofrezca las posibles razones a las alarmas detectadas por el dispositivo limitando ofrecer soluciones para evitar exponer el personal a una situación de riesgo. También, con el controlador apropiado se pudiera inclusive incluir recordatorios para revisiones de rutina, mantenimiento, programación de análisis y preventivos.

14 Conclusiones

El dispositivo de monitoreo de subestaciones utilizando *Cloud Computing* como método de relevo de actividades de cómputo, para simplificar el diseño del equipo y disminuir su costo de fabricación fue sujeto a una serie de pruebas satisfactorias ejecutadas a nivel de software, a nivel local en laboratorio y en campo (en una subestación de 138kV -13.8kV con tres transformadores de potencia para sumar un total de capacidad de 11MVA). Esta alternativa de bajo costo para reducir el distanciamiento que se tiene entre las instalaciones y los programas de mantenimiento de las mismas permite que de forma oportuna se puedan atender contingencias antes de que exista un daño que perjudique de forma permanente la instalación.

No obstante, se debe de tomar en cuenta que el dispositivo no pretende ser un reemplazo de las inspecciones rutinarias a las instalaciones eléctricas, ya que son un número infinito de observaciones que pueden ser detectadas en un lugar que no están al alcance de un sensor o del mismo dispositivo que se presenta en este documento. Simplemente se busca una forma de brindar un método para notificar anomalías una vez que caen dentro del rango de detección de los equipos de protección y los sensores que se encuentren instalados en el área.

Por otra parte, la elaboración de este proyecto involucró integrar la selección de componentes, investigación de proveedores de *Cloud Computing*, y el desarrollo de un prototipo funcional que permitiera cumplir de forma satisfactoria los objetivos del proyecto siendo lo más laborioso el desarrollo del firmware del microcontrolador ya que su diseño y programación estuvo obstaculizada por problemas de compatibilidad entre librerías de todos los accesorios y limitaciones en memoria. De tal forma que el diseño del programa se tuvo que acotar dentro de esas limitaciones por lo que el nivel del reto se incrementó.

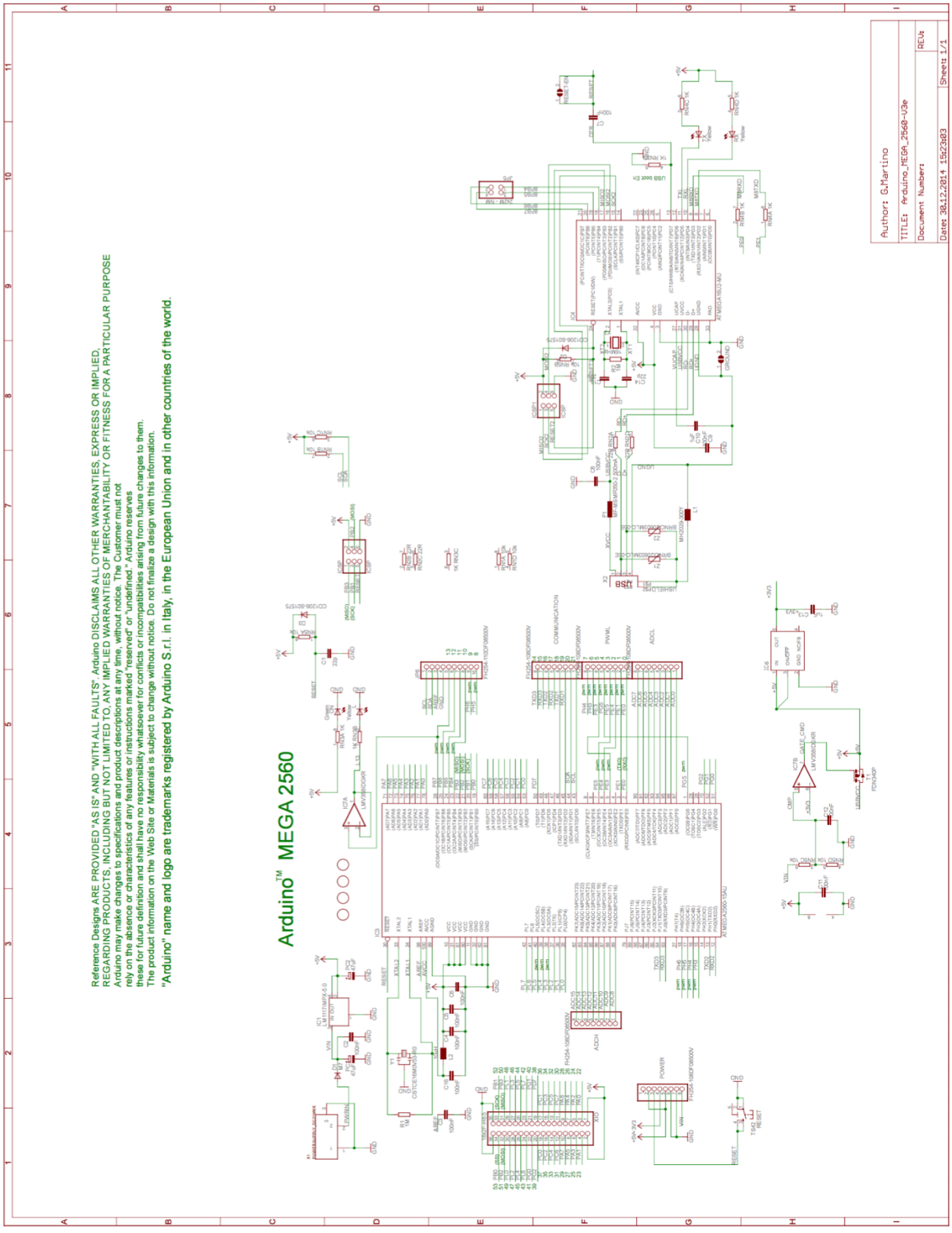
Además, se debe de tomar en cuenta que en este proyecto se desarrolló una alternativa de bajo costo en comparación con otros equipos más sofisticados y costosos que pueden tener la capacidad de realizar actividades similares y no es garantizado por parte del fabricante que específicamente sea capaz de enviar un mensaje de correo electrónico ante una alarma, pero brinda la posibilidad dentro de

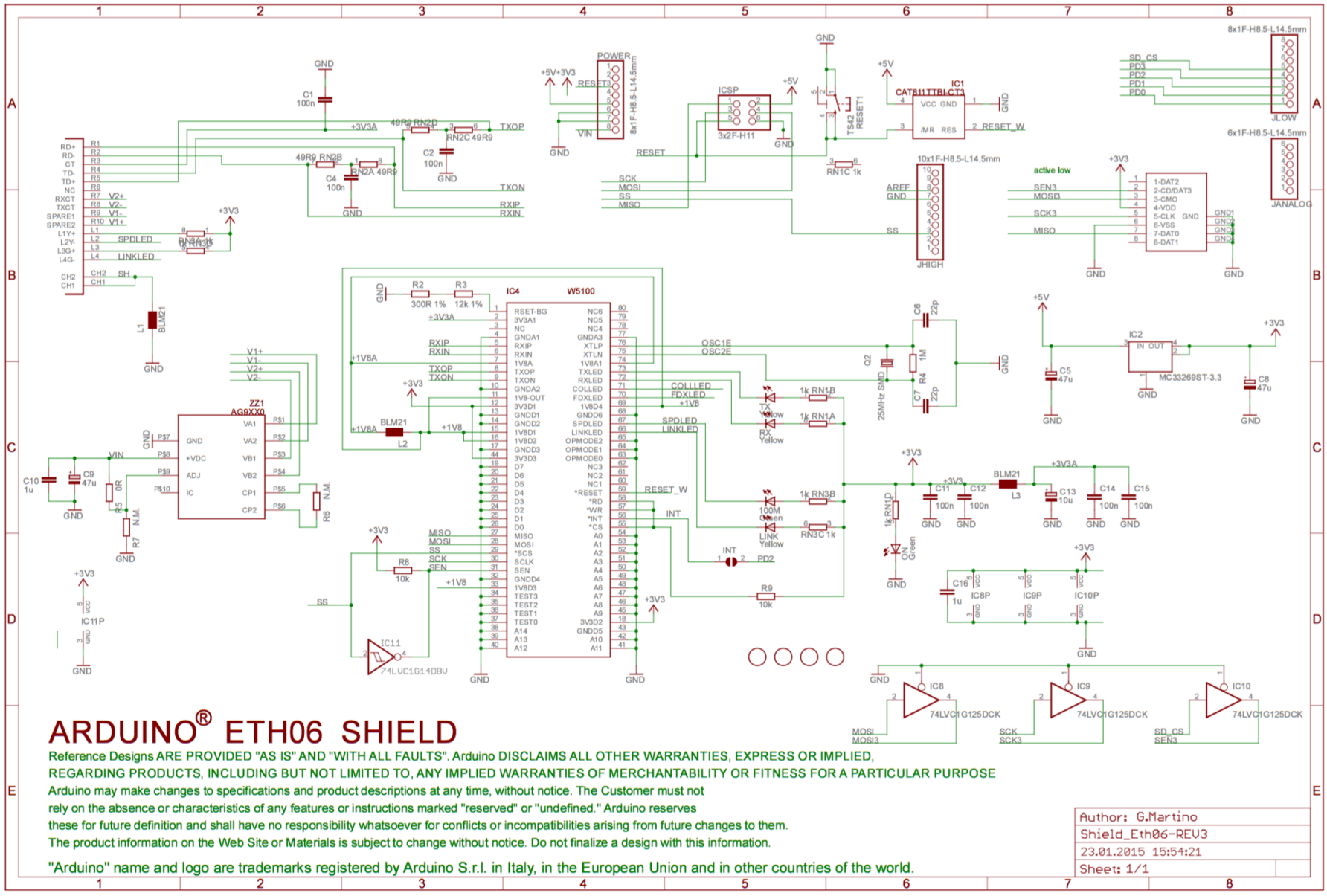
una red local, el crear una interfaz gráfica que indique el estado actual. Sin embargo, esto no cubre el objetivo de notificar con un canal de comunicación directo al personal competente para tomar acción en el área.

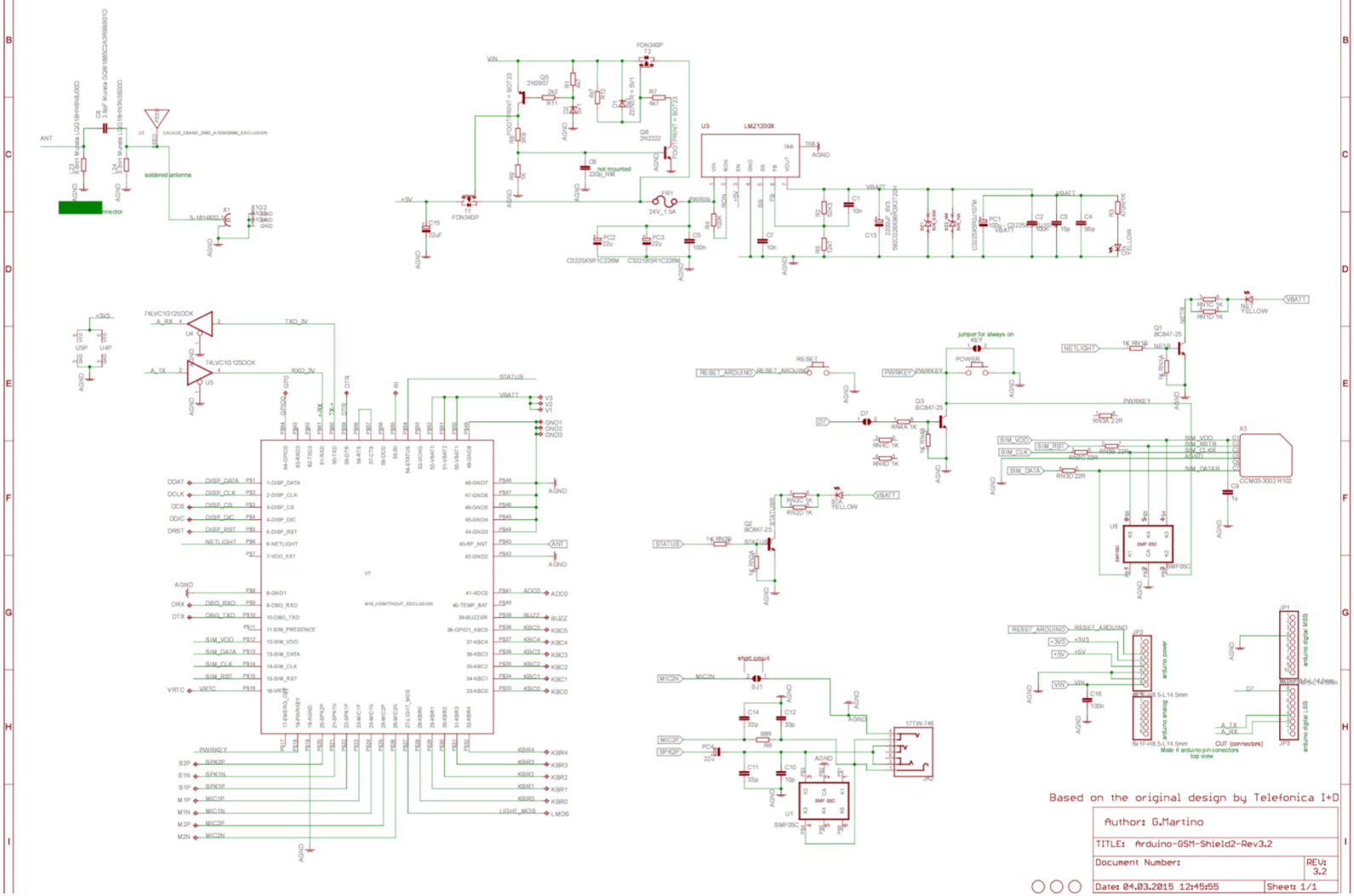
Finalmente, el hecho que durante la etapa de pruebas del dispositivo en campo se detectaran alarmas que originalmente no eran percibidas por los responsables de la subestación permitió realizar una acción correctiva en un transformador, propenso a daños internos por presencia de humedad. Además, realizó notificaciones de inestabilidad en conexión a la red y detectó en dos ocasiones un fallo en el suministro eléctrico por parte de CFE mediante el envío de alarmas de bajo voltaje de CA en el cargador de baterías. Lo anterior demuestra, la efectividad del dispositivo como herramienta de comunicación.

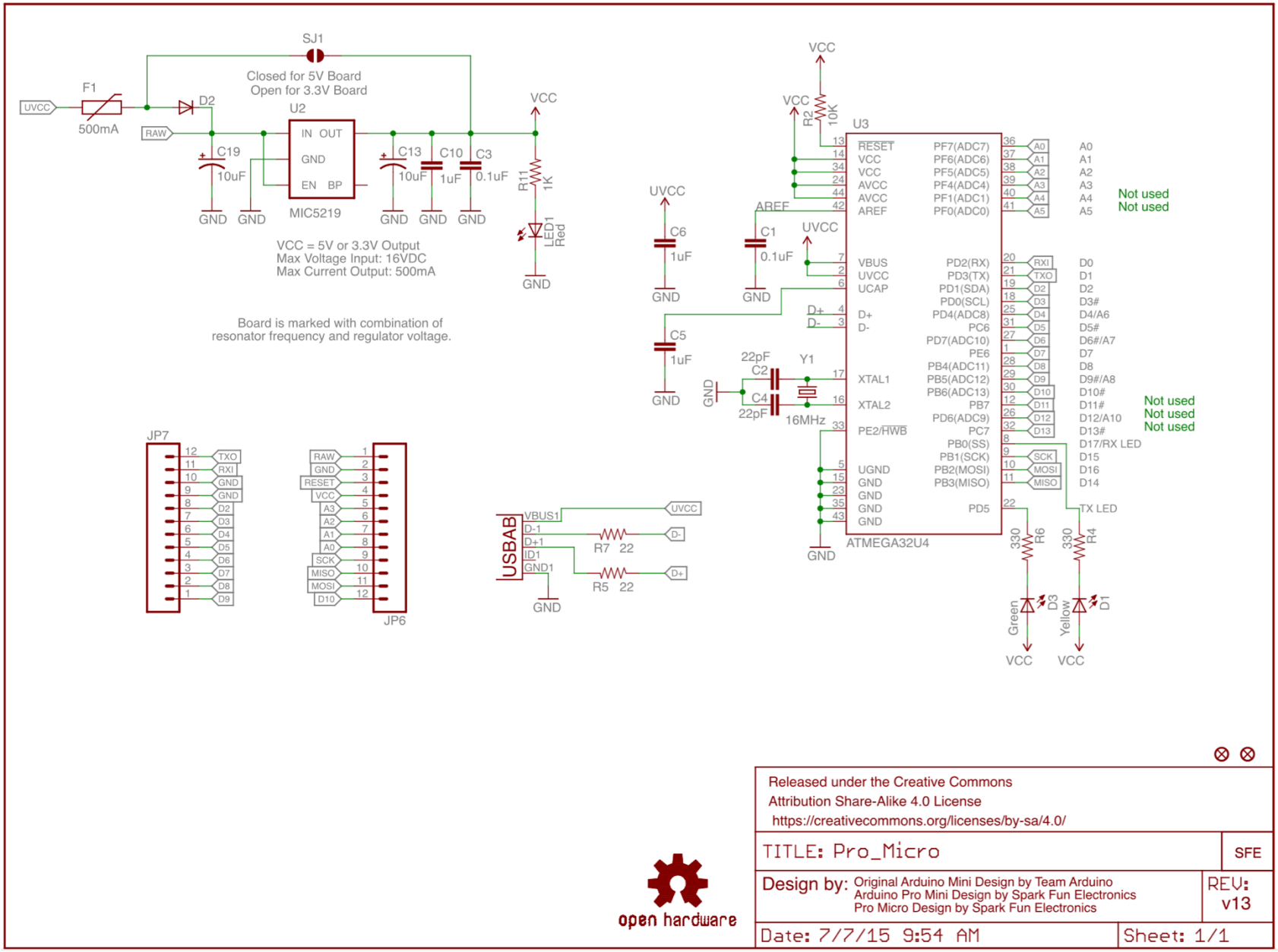
Por norma debería de ser especificado que las subestaciones eléctricas de potencia cuenten un método de notificación de este tipo, ya que muchas de ellas no se encuentran lo suficientemente cerca de las áreas de trabajo como para que se les brinde atención oportuna y de forma periódica.

Apéndice A. Diagramas Esquematicos de Componentes de Hardware



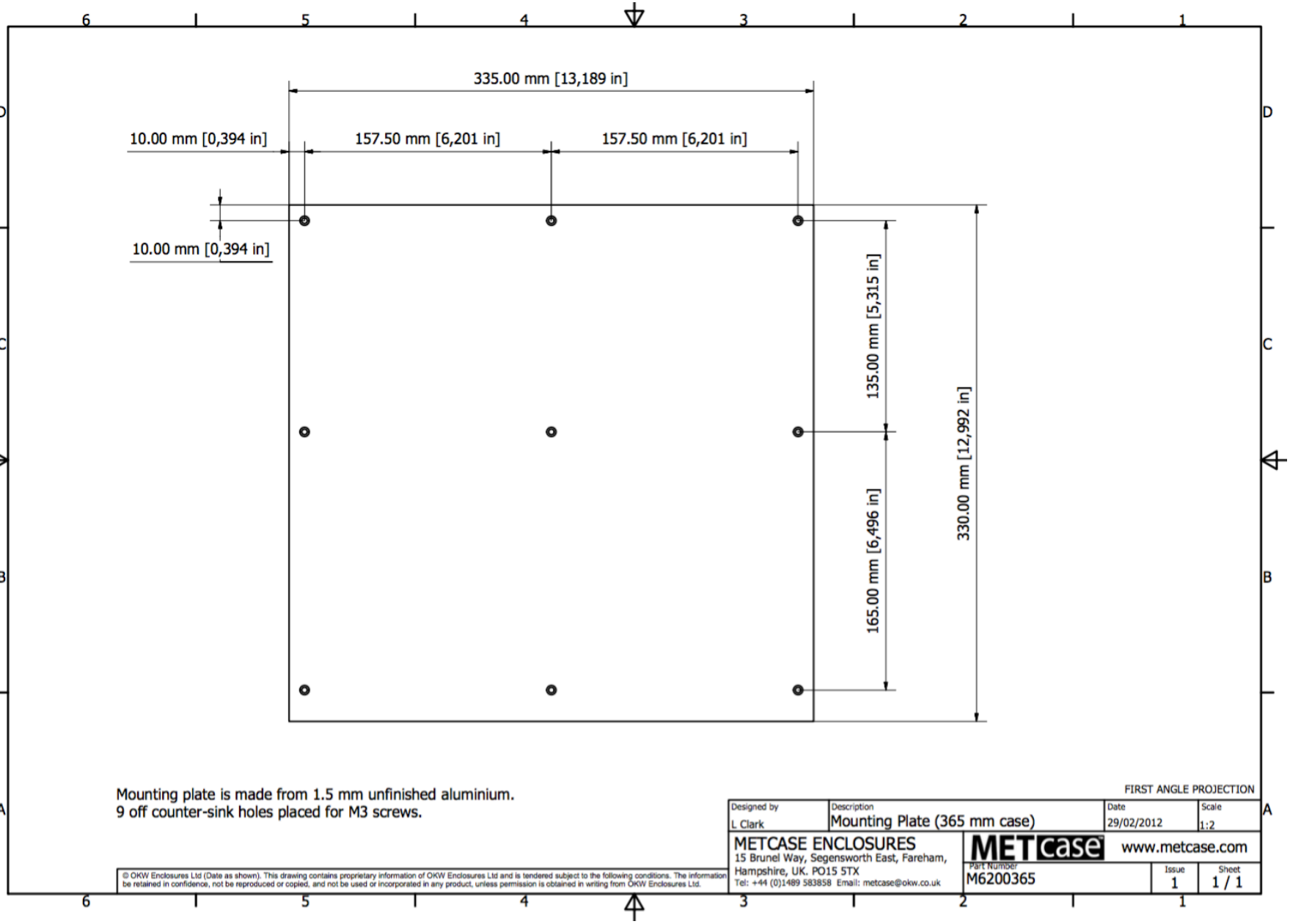

ARDUINO® ETH06 SHIELD
Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS". Arduino DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED,
REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE
Arduino may make changes to specifications and product descriptions at any time, without notice. The Customer must not
rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino reserves
these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.
The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.
"Arduino" name and logo are trademarks registered by Arduino S.r.l. in Italy, in the European Union and in other countries of the world.
Author: G.Martino
Shield_Eth06-REV3
23.01.2015 15:54:21
Sheet: 1/1
IC4
W5100
IC1
CAT811TTBLCT3
JLOW
JANALOG
JHIGH
ICSP
POWER
ZZ1
AG9XX0
IC11
74LVC1G14DBU
IC8
IC9
IC10
74LVC1G125DCK
IC2
MC33269ST-3.3
IC11P
IC8P
IC9P
IC10P





TITLE: Pro_Micro	SFE
Design by: Original Arduino Mini Design by Team Arduino Arduino Pro Mini Design by Spark Fun Electronics Pro Micro Design by Spark Fun Electronics	REV: v13
Date: 7/7/15 9:54 AM	Sheet: 1/1

open hardware



Mounting plate is made from 1.5 mm unfinished aluminium.
9 off counter-sink holes placed for M3 screws.

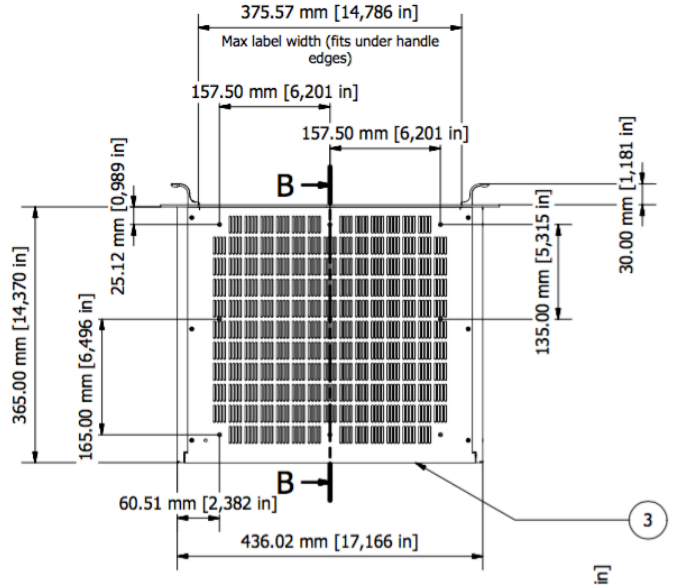
FIRST ANGLE PROJECTION

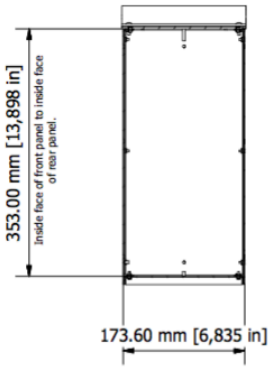
Designed by	Description	Date	Scale
L Clark	Mounting Plate (365 mm case)	29/02/2012	1:2

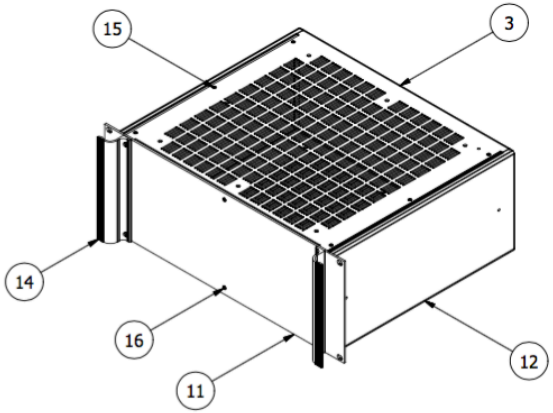
METCASE ENCLOSURES
15 Brunel Way, Segensworth East, Fareham, Hampshire, UK. PO15 5TX
Tel: +44 (0)1489 583858 Email: metcase@okw.co.uk

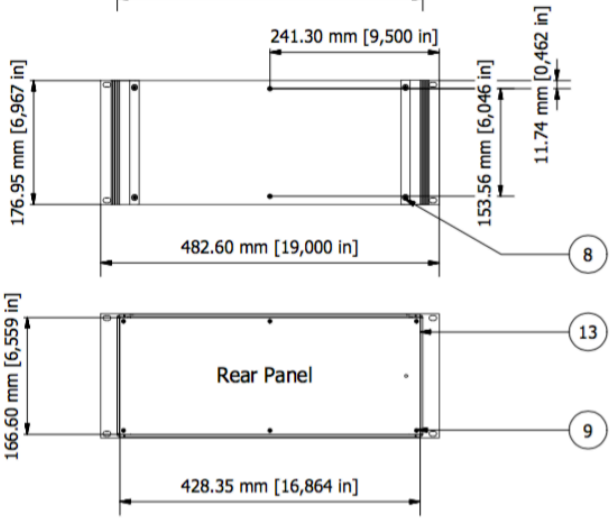
METcase www.metcase.com

Part Number	Issue	Sheet
M6200365	1	1 / 1









NOTES.
Case metalwork is made from 1.5 mm aluminium and painted RAL 9005 fine texture black.
Front panel is made from 3 mm aluminium with a clear etch & anodise finish.
Handles are made from an aluminium extrusion & also painted RAL 9005 fine texture black.
All parts are fitted with M4 studs for earth use.

PARTS LIST

ITEM	QTY	PART NUMBER	DESCRIPTION
3	2	M6203069	Top/Base Cover
8	4	M4x16 CSK Screw (B)	Front Panel Screws
9	6	M3x8 Pan Screw (B)	Rear Panel Screws
11	1	M6219040	Front Panel
12	2	M6202459	Side Panels
13	1	M6201049	Rear Panel
14	2	M6200409	Handles
15	12	M3x8 CSK Screw (B)	Cover Screws
16	2	M3x8 CSK Screw	Front Panel Screws

FIRST ANGLE PROJECTION

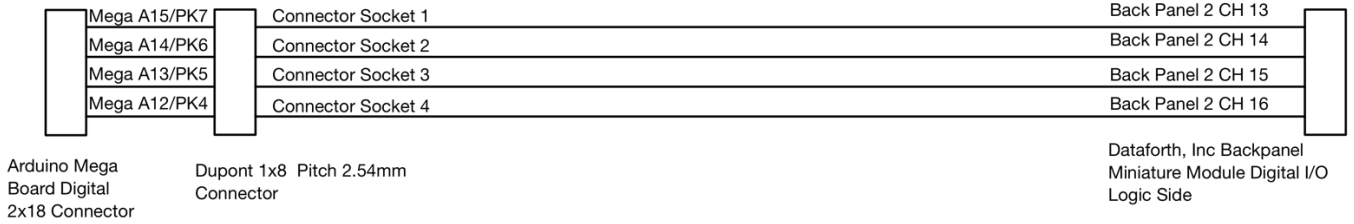
Designed by	Description	Date	Scale
L Clark	Combimet 4U x 19" x 365 mm Vented	11/10/2013	1:6

METCASE ENCLOSURES
15 Brunel Way, Segensworth East, Fareham, Hampshire, PO15 5TX, UK.
Tel: +44 (0)1489 583858 Email: metcase@okw.co.uk

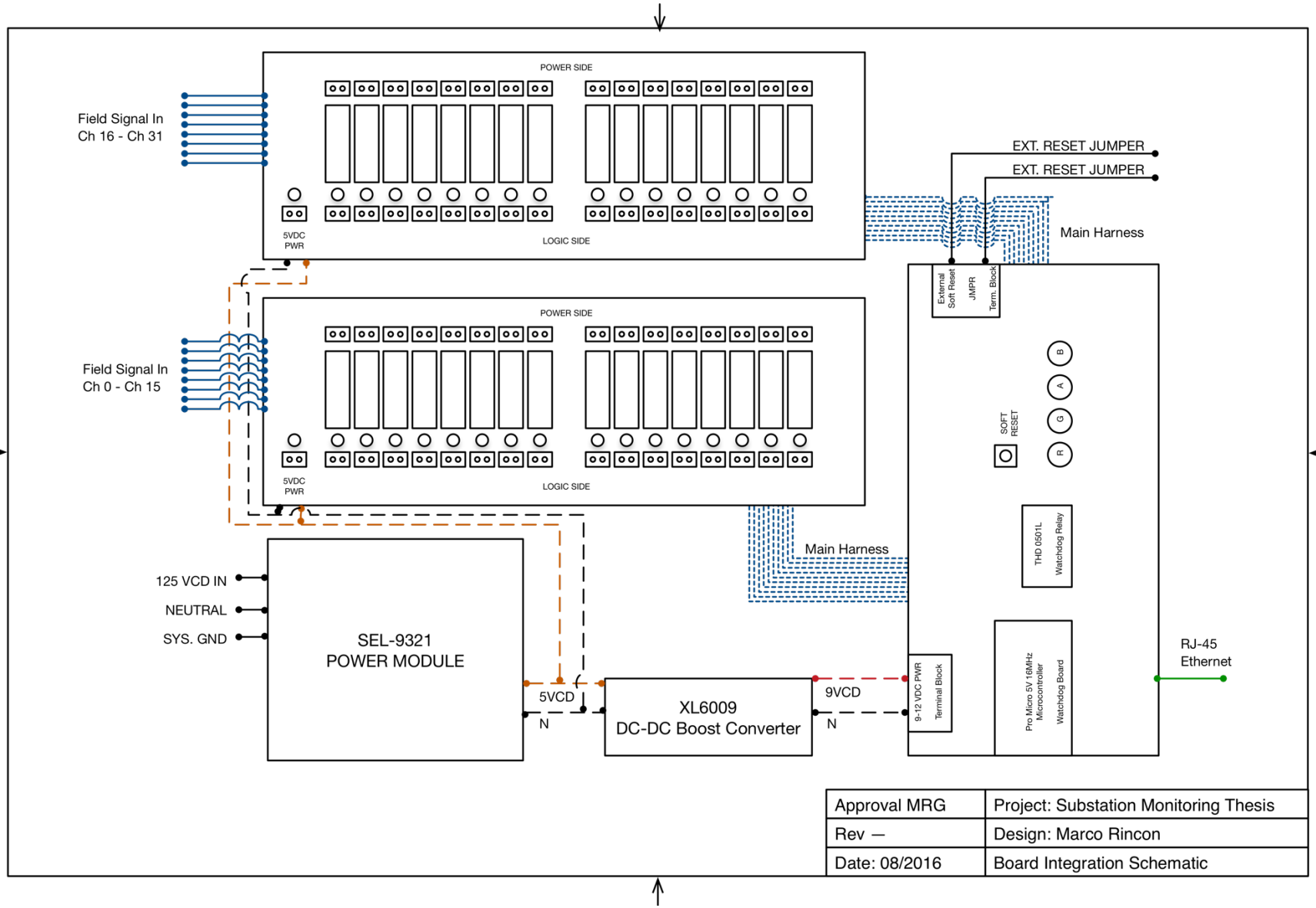
METCase www.metcase.com

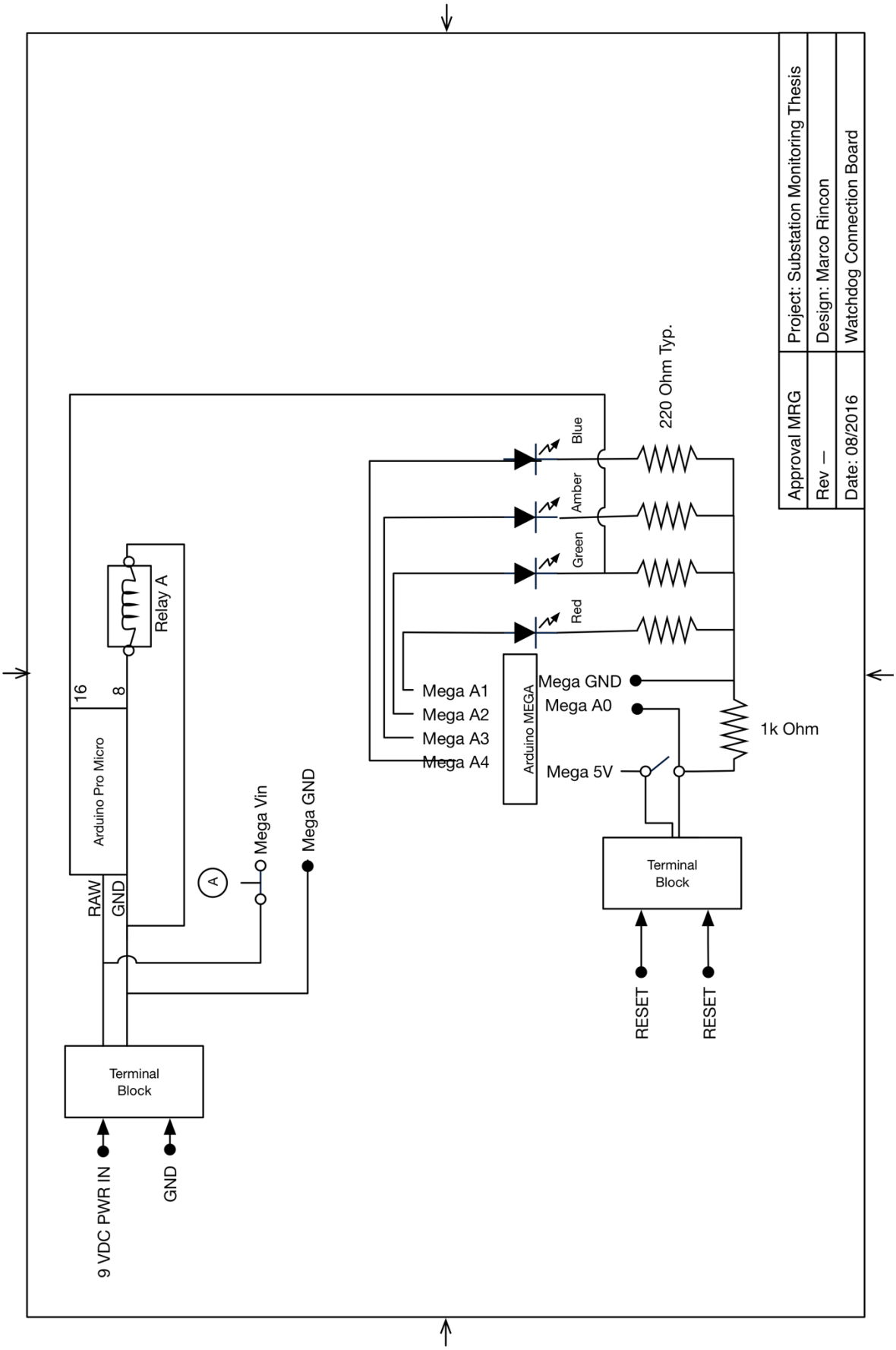
Part Number	Issue	Sheet
M6219469	1	1 / 1

Dataforth, Inc Backpanel
Miniature Module Digital I/O
Logic Side
Back Panel 1 CH 1
Connector Socket 2,1
Mega 22/PA0
Back Panel 1 CH 2
Connector Socket 2,2
Mega 23/PA1
Back Panel 1 CH 3
Connector Socket 3,1
Mega 24/PA2
Back Panel 1 CH 4
Connector Socket 3,2
Mega 25/PA3
Back Panel 1 CH 5
Connector Socket 4,1
Mega 26/PA4
Back Panel 1 CH 6
Connector Socket 4,2
Mega 27/PA5
Back Panel 1 CH 7
Connector Socket 5,1
Mega 28/PA6
Back Panel 1 CH 8
Connector Socket 5,2
Mega 28/PA7
Back Panel 1 CH 9
Connector Socket 6,1
Mega 30/PC7
Back Panel 1 CH 10
Connector Socket 6,2
Mega 31/PC6
Back Panel 1 CH 11
Connector Socket 7,1
Mega 32/PC5
Back Panel 1 CH 12
Connector Socket 7,2
Mega 33/PC4
Back Panel 1 CH 13
Connector Socket 8,1
Mega 34/PC3
Back Panel 1 CH 14
Connector Socket 8,2
Mega 35/PC2
Back Panel 1 CH 15
Connector Socket 9,1
Mega 36/PC1
Back Panel 1 CH 16
Connector Socket 9,2
Mega 37/PC0
Back Panel 2 CH 1
Connector Socket 10,1
Mega 38/PD7
Back Panel 2 CH 2
Connector Socket 10,2
Mega 39/PG2
Back Panel 2 CH 3
Connector Socket 11,1
Mega 40/PG1
Back Panel 2 CH 4
Connector Socket 11,2
Mega 41/PG0
Back Panel 2 CH 5
Connector Socket 12,1
Mega 42/PL7
Back Panel 2 CH 6
Connector Socket 12,2
Mega 43/PL6
Back Panel 2 CH 7
Connector Socket 13,1
Mega 44/PL5
Back Panel 2 CH 8
Connector Socket 13,2
Mega 45/PL4
Back Panel 2 CH 9
Connector Socket 14,1
Mega 46/PL3
Back Panel 2 CH 10
Connector Socket 14,2
Mega 47/PL2
Back Panel 2 CH 11
Connector Socket 15,1
Mega 48/PL1
Back Panel 2 CH 12
Connector Socket 15,2
Mega 49/PL0
Back Panel 1 CH Common
Connector Socket 18,1
Mega GND
Back Panel 2 CH Common
Connector Socket 18,2
Mega GND
Dupont 2x18 Pitch 2.54mm
Connector
Arduino Mega
Board Digital
2x18 Connector
Approval MRG
Project: Substation Monitoring Thesis
Rev —
Design: Marco Rincon
Date: 08/2016
MAIN DIGITAL I/O HARNESS

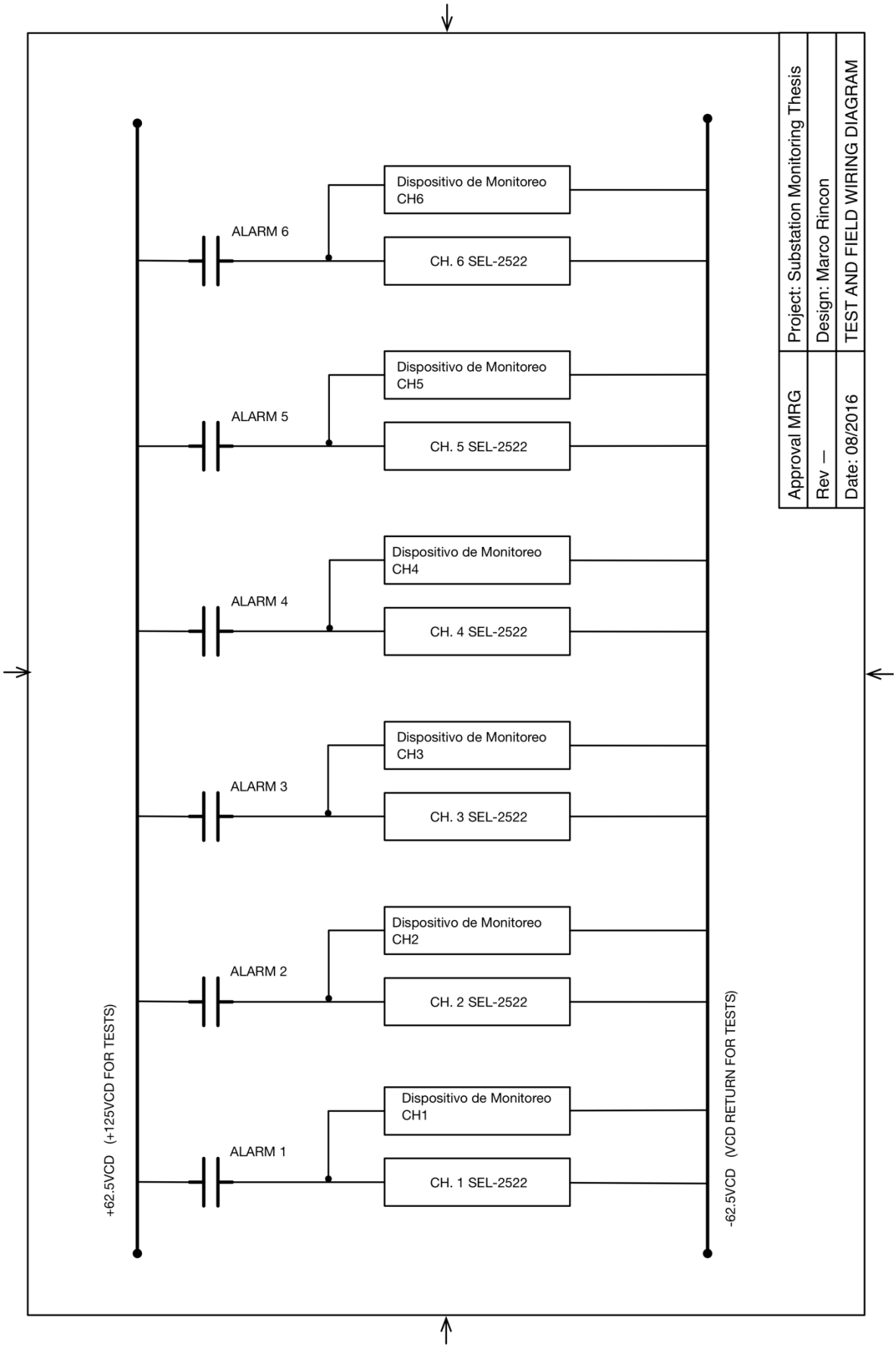


Approval MRG	Project: Substation Monitoring Thesis
Rev —	Design: Marco Rincon
Date: 08/2016	SECONDARY DIGITAL I/O HARNESS


POWER SIDE
Field Signal In
Ch 16 - Ch 31
5VDC
PWR
LOGIC SIDE
EXT. RESET JUMPER
EXT. RESET JUMPER
Main Harness
External
Soft Reset
JMPR
Term. Block
POWER SIDE
Field Signal In
Ch 0 - Ch 15
5VDC
PWR
LOGIC SIDE
SOFT
RESET
B
A
G
R
THD 0501L
Watchdog Relay
Main Harness
125 VCD IN
NEUTRAL
SYS. GND
SEL-9321
POWER MODULE
5VCD
N
XL6009
DC-DC Boost Converter
9VCD
N
9-12 VDC PWR
Terminal Block
Pro Micro 5V 16MHz
Microcontroller
Watchdog Board
RJ-45
Ethernet
Approval MRG
Project: Substation Monitoring Thesis
Rev —
Design: Marco Rincon
Date: 08/2016
Board Integration Schematic


Project: Substation Monitoring Thesis
Design: Marco Rincon
Watchdog Connection Board
Approval MRG
Rev —
Date: 08/2016
220 Ohm Typ.
Blue
Amber
Green
Red
Relay A
16
8
Arduino Pro Micro
RAW
GND
Mega Vin
Mega GND
A
Mega A1
Mega A2
Mega A3
Mega A4
Arduino MEGA
Mega GND
Mega A0
Mega 5V
1k Ohm
Terminal Block
Terminal Block
9 VDC PWR IN
GND
RESET
RESET

Eth. Comm.
10/100 BASE-T RJ-45
Common
Ch. 16
Ch. 17
Ch. 18
Ch. 19
Ch. 20
Ch. 21
Ch. 22
Ch. 23
Ch. 24
Common
Ch. 0
Ch. 1
Ch. 2
Ch. 3
Ch. 4
Ch. 5
Ch. 6
Ch. 7
Ch. 8
Ch. 25
Ch. 26
Ch. 27
Ch. 28
Ch. 29
Ch. 30
Ch. 31
NOT USED
Ext. Reset
Ext. Reset
Ch. 9
Ch. 10
Ch. 11
Ch. 12
Ch. 13
Ch. 14
Ch. 15
PWR+
PWR Comm
SYS. GND
Approval MRG
Rev —
Date: 08/2016
Project: Substation Monitoring Thesis
Design: Marco Rincon
Enclosure Backplane


Dispositivo de Monitoreo CH6
ALARM 6
CH. 6 SEL-2522
Dispositivo de Monitoreo CH5
ALARM 5
CH. 5 SEL-2522
Dispositivo de Monitoreo CH4
ALARM 4
CH. 4 SEL-2522
Dispositivo de Monitoreo CH3
ALARM 3
CH. 3 SEL-2522
Dispositivo de Monitoreo CH2
ALARM 2
CH. 2 SEL-2522
Dispositivo de Monitoreo CH1
ALARM 1
CH. 1 SEL-2522
+62.5VCD (+125VCD FOR TESTS)
-62.5VCD (VCD RETURN FOR TESTS)
Approval MRG
Rev —
Date: 08/2016
Project: Substation Monitoring Thesis
Design: Marco Rincon
TEST AND FIELD WIRING DIAGRAM

Apéndice B. Pseudocodigo

```
#PIN Number
define PINNUMBER "***"

define TEMBOO_ACCOUNT "*********"  # Your Temboo account name
define TEMBOO_APP_KEY_NAME "********"  # Your Temboo app key name
define TEMBOO_APP_KEY "**********"  # Your Temboo app key
define ETHERNET_SHIELD_MAC {0x00, 0x00, 0x00, 0x00, 0x00, 0x00}

#Google Authentication
const String ClientIDValue = "*******"
const String ClientSecretValue = "********"
const String RefreshTokenValue = "********"
const String LICENSE_FILE = "*******"
const String STATUS_FILE = "*********"
const String STATUS_FILE_TAB = "******"
const String STATUS_FILE_CLIENT_ROW = "**"
const String LOG_FILE = "*******"
const String LICENSE_TAB = "*******"
const String LICENSE_LOCATION = "*****"
const String VERSION = "20"
const int GSMEnable = 1
const int daily_status_hour = 13

#Service License
String T_APP_KEY =""
String T_APP_KEY_NAME = "TEC"

const long Interval = 900000 # time out interval in miliseconds
const long BatchInterval = 120000 # time to wait before sending latched faults
const long LOSTCOMInterval = 120000

const long wait_to_confirm = 1 # ms to confirm latched alarm
const int timeZone = -6  # Central Daylight Time (USA)    time zone used for the time stamp
const String CLIENTID = "TEC" #CLIENT id to be transmitted

char remoteNumber[20]= "*********"
char remoteNumber_mmto1[20]="*******"
char remoteNumber_mmto2[20]="*********"
char remoteNumber_cust1[20]="**********"
char remoteNumber_cust2[20]="*******"


# IC FOR CODE AND COMMUNICATION PURPOSES, NOT RECOMMENDED TO MODIF   Y

# NTP Servers:
IPAddress timeServer(132, 163, 4, 101) # time-a.timefreq.bldrdoc.gov

unsigned int localPort = 8888  # local port to listen for UDP packets

# ***** MASK / UNMASK DIGITAL INPUTS (will ignore an alarm IF    set to 0, alarm will not be transmitted and alarm code for that channel
will be null) *********#####

                  /It basically enables or disable (bypass) the digital input
#pin 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
int MaskUnmaskPINS [32] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}

                  #Which of the detected alarms (not masked) can trigger email alerts?
#pin 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
int MaskUnmaskEMAIL[32] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}

                  #Which kind logic should each chanel follow?
int MaskUnmaskLOGIC[32] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
                        "\n Alta Temp de aceite en T1",
const String ALARM_CODE_NAME[32] = {  "\n Bajo Nivel aceite T1",
```

```
                                "\n Disparo por 86T1 en T1",
                                "\n Sobrepresion en T1",
                                "\n Baja presion de N2 en T1",
                                "\n Alta Temp en devanado T1",
                                "\n Bajo Nivel aceite T2",
                                "\n Alta Temp de aceite en T2",
                                "\n Disparo por 86T2 en T2",
                                "\n Sobrepresion en T2",
                                "\n Baja presion de N2 en T2",
                                "\n Alta Temp en devanado T2",
                                "\n Bajo Nivel aceite T3",
                                "\n Alta Temp de aceite en T3",
                                "\n Disparo por 86T3 en T3",
                                "\n Sobrepresion en T3",
                                "\n Baja presion de N2 en T3",
                                "\n Alta Temp en devanado T3",
                                "\n Baja presion SF6 52L",
                                "\n Baja presion aire 152L",
                                "\n Disparo Int 152L",
                                "\n Apertura Cuchilla 189T1",
                                "\n Apertura Cuchilla 189T2",
                                "\n Apertura Cuchilla 189T3",
                                "\n Disparo Int 52T1",
                                "\n Disparo Int 52T2",
                                "\n Disparo Int 52T3",
                                "\n Bajo voltaje CA cargador",
                                "\n Bajo voltaje CD cargador",
                                "\n Falla a Tierra Sistema CD",
                                "\n Alarma Gral T4",
                                "\n DI31
                               }

const String EMAIL_SERVER = "*******"
const String EMAIL_USERNAME = "*******"
const String EMAIL_PASSWORD = "******"
const String EMAIL_PORT = "****"
const String EMAIL_SSL = "*******"
const String EMAIL_SUBJECT = String(CLIENTID) + " SUBSTATION PANEL STATUS"
const String EMAIL_FROM = "********"
const String EMAIL_TO = "**********"
const String EMAIL_BODDY_MSG_HEADER = String(CLIENTID) + " SUBSTATION ALARM STATUS, details \n"
const String EMAIL_BODDY_MSG_FOOTER = "Regards"

# GLOBAL VARIABLES USED IN THE CODE, NOT RECOMMENDED TO MODIF   Y
unsigned long  previousMillis = 0 # last time update # Initialization for timeout routine
unsigned long  previousMillis_HB = 0 # last time update HB
unsigned long  previousMillis_BATCH = 0
unsigned long  previousMillis_LOSTCOM = 0
const long interval_HB = 500 # hearth beat blinking period
int setupinprogress = 0
int healthlevel_global = 2
int healthlevel_actual = healthlevel_global
int alertsent = 0
int echoReply = 0
int daily_email_sent = 0

int latchedfault = 0
int majorfault = 0
int latchedfaults[32] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
String previous_alarmcode = ""
int state = HIGH
int resetcondition = 0

#  PINS INITIALIZATION
const int resetPBpin = A0
const int redLEDpin = A1
const int greenLEDpin = A2
const int amberLEDpin = A3
const int blueLEDpin = A4

FUNCTION setup() {
  Serial.begin(9600)
  Display(F("Initializing..."))
  setupinprogress=1
  Display(F("Software version: "))
  Display(VERSION)
  Display(F("Client: "))
  Display(CLIENTID)
  SET_MODE(resetPBpin, INPUT)
  SET_MODE(redLEDpin, OUTPUT)
```

```
SET_MODE(greenLEDpin, OUTPUT)
SET_MODE(amberLEDpin, OUTPUT)
SET_MODE(blueLEDpin, OUTPUT)

delay(2000);
for (int x=22, y=0x<54x++)
{
 IF    (x>=50)
 {
    IF    (x==50)
    SET_MODE(A15, INPUT)
    IF    (x==51)
    SET_MODE(A14, INPUT)
    IF    (x==52)
    SET_MODE(A13, INPUT)
    IF    (x==53)
    SET_MODE(A12, INPUT)
 }
 else
 {
 SET_MODE(x, INPUT)
 }

 IF    (x<=49)
 {
 Display(F("Digital Input "))
 Display(x)
 }
 else
 {
 Display(F("Analogue Input "))
 Display(x)
 }
 Display(F(" Alarm Number "))
 Display(y)

 IF    (MaskUnmaskPINS[y] == 1)
 {
  Display(F(" READY"))
 }
 else
 {
  Display(F(" BYPASSED!!"))
 }
 y++
 SET(amberLEDpin, LOW)
 delay(125)
 SET(amberLEDpin, HIGH)
 delay(125)
}

Display(F("Testing LEDs..."))
SET(redLEDpin, LOW)
SET(greenLEDpin, LOW)
SET(amberLEDpin, LOW)
SET(blueLEDpin, LOW)

delay(100)
SET(redLEDpin, HIGH)
SET(greenLEDpin, HIGH)
SET(amberLEDpin, HIGH)
SET(blueLEDpin, HIGH)

delay(4000)
SET(redLEDpin, LOW)
SET(greenLEDpin, LOW)
SET(amberLEDpin, LOW)
SET(blueLEDpin, LOW)

delay(500)

IF    (GSMEnable == 1)
{
    Display(F("Connecting to GSM Carrier"))
    scannerNetworks.begin()
    SET(amberLEDpin, HIGH)
    # connection state
    boolean notConnected = true

  long beginWait = millis()
```

```
    while ((millis() - beginWait < 900000) AND (notConnected))
    {
       IF   (gsmAccess.begin(PINNUMBER)==GSM_READY)
       {
         SET(greenLEDpin,HIGH)
         SET(amberLEDpin,LOW)
         String SMSMess = CLIENTID+" connected to GSM network"
         char mensajeSMS[200]=""
         SMSMess.toCharArray(mensajeSMS,200)
         notConnected = false
         sendSMS(mensajeSMS, remoteNumber)
         sendSMS(mensajeSMS, remoteNumber_mmto1)
         sendSMS(mensajeSMS, remoteNumber_mmto2)
         sendSMS(mensajeSMS, remoteNumber_cust1)
         #sendSMS(mensajeSMS, remoteNumber_cust2)
       }
       else
       {
         Display(F("Not connected"))
         delay(1000)
       }
    }
    # get modem parameters
    Display(F("Modem IMEI: "))
    IMEI = modemTest.getIMEI()
    IMEI.replace("\n","")
    IF   (IMEI != NULL)
      Display(IMEI)

    # currently connected carrier
    Display(F("Current carrier: "))
    Display(scannerNetworks.getCurrentCarrier())

    # returns strength and ber
    # signal strength in 0-31 scale. 31 means power > 51dBm
    # BER is the Bit Error Rate. 0-7 scale. 99=not detectable
    Display("Signal Strength: ")
    Display(scannerNetworks.getSignalStrength())
    Display(" [0-31]")
    SET(greenLEDpin, LOW)
    SET(amberLEDpin, LOW)
    IF    (notConnected == false)
    {
      gsmAccess.shutdown()
    }
}

Display(F("Connecting to the network..."))
SET(amberLEDpin, HIGH)

delay(4000)

while(!Serial)
Display(F("Negotiating IP DHCP:"))
IF    (Ethernet.begin(ethernetMACAddress) == 0) {
  Display(F("FAIL"))
  while(true)
  {
    delay(100)
    SET(amberLEDpin, HIGH)
    delay(100)
    SET(amberLEDpin, LOW)
    delay(100)
    SET(redLEDpin, HIGH)
    delay(100)
    SET(redLEDpin,LOW)
  }
}
Display(F("OK"))

SET(redLEDpin, LOW)
SET(amberLEDpin, LOW)
SET(greenLEDpin, HIGH)
delay(2000)

# Get NTP Time
Udp.begin(localPort)
Display(F("Synchronizing..."))
delay(1000)
setSyncProvider(getNtpTime)
```

```
  delay(1000)
  echoReply = testconnection()

  IF    (echoReply == 1)
  {
    Display(F("Connection OK"))
    healthlevel_actual = healthlevel_global
  }
  else
  {
    Display(F("Connection failed HL: "))
    healthlevel_actual =  healthlevel_actual - 1
    Display(healthlevel_actual)
  }

  delay(1000)
  CheckLicenseChoreo()
  delay(500)
  String alarmcode = generatealarmcode()
  Display(alarmcode)
  delay(500)
  SendEmailAlertsChoreo(alarmcode)
  Display(F("Setup complete.\n"))
  setupinprogress = 0
  delay(500)
  SET(greenLEDpin, HIGH)
  SET(redLEDpin, LOW)
  SET(amberLEDpin, LOW)
}

time_t prevDisplay = 0 # when the digital clock was displayed

FUNCTION loop() {
  # put your main code here, to run repeatedly:
  unsigned long currentMillis = millis()
  unsigned long currentMillis_BATCH = millis()
  unsigned long currentMillis_LOSTCOM = millis()
  int panel_status = 0
  int panel_status_echo = 0
  int email_req = 0
  int actual_status
  String alarmcode = ""
  String datatotest = ""
  char c
  char senderNumber[20]
  String texto = ""
  int hour_num = hour()
  int minute_num = minute()

  checkIO()
  panel_status = checkLatched()

IF    ((currentMillis - previousMillis > Interval) OR (currentMillis < previousMillis))     # CHECK FOR TIMEOUT TO REPORT STATUS, IF
timeout expired... or FUNCTIONed due overlow
  {
    previousMillis = currentMillis
      IF    (timeStatus() != timeNotSet)
      {
        IF    (now() != prevDisplay)
          { #update the display only IF    time has changed
            prevDisplay = now()
            digitalClockDisplay()
          }
      }

    Display(F("Transmission opened since timeout expired"))
    alarmcode = generatealarmcode()

    Display(alarmcode)

    echoReply = testconnection()

      IF    (echoReply  == 1)
      {
        Display(F("Connection OK"))
        healthlevel_actual = healthlevel_global
        delay(2000)
        IF    (T_APP_KEY == "CADUCADA")
        {
```

```
      Display(F("No license"))
     }
     else
     {
     SendDataChoreo(alarmcode)
     }
    }
    else
    {
     Display(F("Connection failed HL: "))
     healthlevel_actual = healthlevel_actual - 1
     Display(healthlevel_actual)
    }

     IF    (GSMEnable == 1)
     {
        SET(amberLEDpin, HIGH)
       boolean notConnected = true
       long beginWait = millis()
       Display(F("Checking for incoming SMS"))
       while (notConnected)
       {
              IF   (gsmAccess.begin(PINNUMBER)==GSM_READY)
              {
              notConnected = false
              }
              else
              {
              Display(F("Connection to carrier failed"))
              delay(1000)
              }
       }

       delay(500)
       IF    (notConnected == false)
       {

         delay(10000)
         Display("1")
        for (int r=1 r<10 r++)
        {
         Display(r)
         while (sms.available()) {
           Display("2")
         Display(F("Message received from:"))

         # Get remote number
         sms.remoteNumber(senderNumber, 20)
         Display(senderNumber)
         texto = ""

         while (c = sms.read()) {
           texto = texto + c
         }
         Display(texto)
         Display(F("\nEND OF MESSAGE"))

         # Delete message from modem memory
         sms.flush()
         Display("MESSAGE DELETED")

         delay(1000)

IF    (texto == "Tic"  OR  texto == "TIC"  OR  texto == "tic"  OR  texto == "Tic "  OR  texto == "TIC "  OR  texto == "tic ")
         {
         Display(F("Sending SMS"))

         String SMSMess = "Toc!! " + CLIENTID+ " is Alive!. Status: "

         alarmcode = generatealarmcode()
         SMSMess = SMSMess + alarmcode + "_R"+String(resetcondition)
         char mensajeSMS[200]=""
         SMSMess.toCharArray(mensajeSMS,200)
         sendSMS(mensajeSMS, senderNumber)
         }
         }
         delay(2000)
        }
 gsmAccess.shutdown()
        SET(amberLEDpin, LOW)
```

```
        }
      }
 }

IF    ((currentMillis_BATCH - previousMillis_BATCH > BatchInterval) OR (currentMillis_BATCH < previousMillis_BATCH))     # CHeck IF
latched faults are present during the batch timout interval, IF     positive SEND
{
 previousMillis_BATCH = currentMillis_BATCH
  IF    ((panel_status == 1)  AND  latchedfault == 0)   # if alarm, and not already alarmed
  {
    delay(wait_to_confirm) #wait defined ms seconds to confirm alarm
    panel_status_echo = checkLatched() #check again in echo

    IF    (panel_status_echo == 1  AND  latchedfault == 0)  # if  really alarmed and not previously alarmed
     {
     latchedfault = 1                        # alarm main panel
     alarmcode = generatealarmcode() # get alarm code
     previous_alarmcode = alarmcode
     IF    (timeStatus() != timeNotSet)           #publish in serial timestamp
      {
        IF    (now() != prevDisplay)
          { #update the display only IF    time has changed
            prevDisplay = now()
            digitalClockDisplay()
          }
      }
     Display(F("Transmission opened because alarm triggered"))
     Display(F("Alarm Code "))
     Display(alarmcode)

  echoReply = testconnection()
  IF    (echoReply == 1)
     {
       Display(F("Connection OK"))
       healthlevel_actual = healthlevel_global
       delay(2000)
       SendDataChoreo(alarmcode)
       MakeLogEntryChoreo(alarmcode)
       email_req = CheckIF    _EMAIL_required(alarmcode)

       IF    (email_req == 1)
       {
       SendEmailAlertsChoreo(alarmcode)
       }
     }
     else
     {
       Display(F("Connection failed HL:"))
       healthlevel_actual = healthlevel_actual - 1
       Display(healthlevel_actual)

       Display(F("Lost network connection"))
      IF    (GSMEnable == 1)
      {
          SET(amberLEDpin, HIGH)
         boolean notConnected = true
         long beginWait = millis()
         while ((millis() - beginWait < 900000) AND (notConnected))
         {
                 IF   (gsmAccess.begin(PINNUMBER)==GSM_READY)
                 {
                 notConnected = false
                 }
                 else
                 {
                 Display(F("Connection to carrier failed"))
                 delay(1000)
                 }
         }


         IF    (notConnected == false)
         {
         Display(F("Sending SMS"))
         String SMSMess = CLIENTID+" Communication lost. Alarm Detected Status: "
         alarmcode = generatealarmcode()
         SMSMess = SMSMess + alarmcode + "_R"+String(resetcondition)
         char mensajeSMS[200]=""
         SMSMess.toCharArray(mensajeSMS,200)
           sendSMS(mensajeSMS, remoteNumber)
```

```
          sendSMS(mensajeSMS, remoteNumber_mmto1)
          sendSMS(mensajeSMS, remoteNumber_mmto2)
          sendSMS(mensajeSMS, remoteNumber_cust1)
          #sendSMS(mensajeSMS, remoteNumber_cust2)
        gsmAccess.shutdown()

        }
        SET(amberLEDpin, LOW)
      }

      alertsent = 1
    }

  }
 }
IF    (panel_status == 1  AND  latchedfault == 1)   #IF    alarm and already alarmed
 {
   delay(wait_to_confirm) #wait defined ms to confirm alarm
   panel_status_echo = checkLatched() #check again in echo

   IF    (panel_status_echo == 1  AND  latchedfault == 1)  # IF    really alarmed and already alarmed
    {
     alarmcode = generatealarmcode() # get alarm code
     IF    (alarmcode != previous_alarmcode)           # check IF    its a new alarm
      {
          previous_alarmcode = alarmcode         #save new alarm as previous
          latchedfault = 1      # alarm main panel
            IF    (timeStatus() != timeNotSet)           #publish in serial timestamp
              {
                IF    (now() != prevDisplay)
                  { #update the display only IF    time has changed
                    prevDisplay = now()
                    digitalClockDisplay()
                  }
              }
          Display(F("Transmission opened because another alarm triggered"))
          Display(F("Alarm Code "))
          Display(alarmcode)
          echoReply = testconnection()
          IF    (echoReply == 1)
          {
            Display(F("Connection OK"))
            healthlevel_actual = healthlevel_global
            delay(2000)
            SendDataChoreo(alarmcode)
            MakeLogEntryChoreo(alarmcode)

            email_req = CheckIF   _EMAIL_required(alarmcode)

            IF    (email_req == 1)
            {
            SendEmailAlertsChoreo(alarmcode)
            }
          }
          else
          {
            Display(F("Connection failed HL: "))
            healthlevel_actual = healthlevel_actual - 1
            Display(healthlevel_actual)
            Display(F("Lost network connection"))
    IF    (GSMEnable == 1)
     {
         SET(amberLEDpin, HIGH)
        boolean notConnected = true
        long beginWait = millis()
        while ((millis() - beginWait < 900000) AND (notConnected))
        {
                IF   (gsmAccess.begin(PINNUMBER)==GSM_READY)
                {
                notConnected = false
                }
                else
                {
                Display(F("Connection to carrier failed"))
                delay(1000)
                }
        }

        IF    (notConnected == false)
        {
```

```
          Display(F("Sending SMS"))
          String SMSMess = CLIENTID+" Communication lost. Alarm Detected Status: "
          alarmcode = generatealarmcode()
          SMSMess = SMSMess + alarmcode + "_R"+String(resetcondition)
          char mensajeSMS[200]=""
          SMSMess.toCharArray(mensajeSMS,200)
            sendSMS(mensajeSMS, remoteNumber)
            sendSMS(mensajeSMS, remoteNumber_mmto1)
            sendSMS(mensajeSMS, remoteNumber_mmto2)
            sendSMS(mensajeSMS, remoteNumber_cust1)
            #sendSMS(mensajeSMS, remoteNumber_cust2)
          gsmAccess.shutdown()
          }
            SET(amberLEDpin, LOW)

       }
              alertsent = 1
             }
        }
     }
   }
}

  IF    (panel_status == 0  AND  panel_status_echo == 0  AND  latchedfault == 1) # unlatch IF    latched and condition back normal operation
  {
   latchedfault = 0
   SET(redLEDpin, LOW)
  }

   IF    (READ(resetPBpin) == HIGH)
   reset()

IF    ((healthlevel_actual < 1) AND (alertsent == 0))  #check connection status to internet and IF    lost and not already notIF   ied send alert
{
  Display(F("Lost network connection HL Down: "))
  Display(healthlevel_actual)

      IF    (GSMEnable == 1)
       {
           SET(amberLEDpin, HIGH)
          boolean notConnected = true
          long beginWait = millis()
          while ((millis() - beginWait < 900000) AND (notConnected))
          {
                  IF   (gsmAccess.begin(PINNUMBER)==GSM_READY)
                  {
                  notConnected = false
                  }
                  else
                  {
                  Display(F("Connection to carrier failed"))
                  delay(1000)
                  }
          }

          IF    (notConnected == false)
          {
          Display(F("Sending SMS"))
          String SMSMess = CLIENTID+" Communication lost. Status: "
          alarmcode = generatealarmcode()
          SMSMess = SMSMess + alarmcode + "_R"+String(resetcondition)
          char mensajeSMS[200]=""
          SMSMess.toCharArray(mensajeSMS,200)
            sendSMS(mensajeSMS, remoteNumber)
            sendSMS(mensajeSMS, remoteNumber_mmto1)
            sendSMS(mensajeSMS, remoteNumber_mmto2)
            sendSMS(mensajeSMS, remoteNumber_cust1)
            #sendSMS(mensajeSMS, remoteNumber_cust2)
          gsmAccess.shutdown()
          }
         SET(amberLEDpin, LOW)
       }

  alertsent = 1
}

IF    ((healthlevel_actual > 0) AND (alertsent == 1)) #check connection status to internet and IF    recovered and  already notIF   ied alert, send
recovery notIF   ication
{
  Display(F("Recovered network connection HL Up: "))
```

```
 Display(healthlevel_actual)

     IF    (GSMEnable == 1)
      {
         SET(amberLEDpin, HIGH)
        boolean notConnected = true
        long beginWait = millis()
        while ((millis() - beginWait < 900000) AND (notConnected))
        {
                IF   (gsmAccess.begin(PINNUMBER)==GSM_READY)
                {
                notConnected = false
                }
                else
                {
                Display(F("Connection to carrier failed"))
                delay(1000)
                }
        }

        IF    (notConnected == false)
        {
        Display(F("Sending SMS"))
        String SMSMess = CLIENTID+" Communication Recovered. Status: "
        alarmcode = generatealarmcode()
        SMSMess = SMSMess + alarmcode + "_R"+String(resetcondition)
        char mensajeSMS[200]=""
        SMSMess.toCharArray(mensajeSMS,200)
          sendSMS(mensajeSMS, remoteNumber)
          sendSMS(mensajeSMS, remoteNumber_mmto1)
          sendSMS(mensajeSMS, remoteNumber_mmto2)
          sendSMS(mensajeSMS, remoteNumber_cust1)
          #sendSMS(mensajeSMS, remoteNumber_cust2)
        gsmAccess.shutdown()
        }
       SET(amberLEDpin, LOW)
      }
 alarmcode = generatealarmcode()
 SendEmailAlertsChoreo(alarmcode)
 alertsent = 0
 delay(500)
}

IF    ((healthlevel_actual < 1) AND (alertsent==1)) #IF    connection lost and alert sent check every often connection status
{
  IF    ((currentMillis_LOSTCOM - previousMillis_LOSTCOM > LOSTCOMInterval) OR (currentMillis_LOSTCOM < previousMillis_LOSTCOM))
# CHECK FOR TIMEOUT TO check comm status, IF    timeout expired... or FUNCTIONed due overlow
  {
    previousMillis_LOSTCOM = currentMillis_LOSTCOM
  echoReply = testconnection()
  IF    (echoReply == 1)
     {
       Display(F("Connection OK"))
       healthlevel_actual = healthlevel_global
     }
     else
     {
       Display(F("Connection not recovered HL: "))
       Display(healthlevel_actual)

       healthlevel_actual = 0
     }
  }

}
IF    ((hour_num == daily_status_hour) AND (minute_num < 10))
{
  IF    (daily_email_sent == 0)
  {
      Display(F("Transmission opened - daily health level check..."))
      alarmcode = generatealarmcode()
      Display(alarmcode)
     daily_email_sent = 1

     IF    (echoReply == 1)
     {
       Display(F("Connection OK"))
       healthlevel_actual = healthlevel_global
       delay(2000)
       CheckLicenseChoreo()
```

```
      SendEmailAlertsChoreo(alarmcode)
    }
    else
    {
      Display(F("Connection failed HL: "))
      healthlevel_actual = healthlevel_actual - 1
      Display(healthlevel_actual)

    }

  }
}
else
{
  daily_email_sent = 0
}

hearthbeat()
}

FUNCTION hearthbeat()
{
  unsigned long currentMillis_HB = millis()

  IF    ((currentMillis_HB - previousMillis_HB > interval_HB) OR (currentMillis_HB < previousMillis_HB))     # CHECK FOR TIMEOUT TO
REPORT STATUS, IF    timeout expired... or FUNCTIONed due overlow
  {
   previousMillis_HB = currentMillis_HB

   IF    (state==HIGH) state=LOW
   Else state=HIGH
   SET(greenLEDpin,state)
  }
}

FUNCTION reset()
{
  resetcondition = 1
  String alarmcode = generatealarmcode()

  SET(amberLEDpin, HIGH)
  Display(F("Unlatching Faults..."))

  echoReply = testconnection()
  IF    (echoReply == 1)
    {
      Display(F("Connection OK"))
      healthlevel_actual = healthlevel_global
      delay(2000)
      MakeLogEntryChoreo(alarmcode)
      delay(2000)
      SendEmailAlertsChoreo(alarmcode)
    }
    else
    {
      Display(F("Connection failed HL: "))
      healthlevel_actual = healthlevel_actual - 1
      Display(healthlevel_actual)
      Display(F("Lost network connection"))

     IF    (GSMEnable == 1)
     {
         SET(amberLEDpin, HIGH)
        boolean notConnected = true
        long beginWait = millis()
        while ((millis() - beginWait < 900000) AND (notConnected))
        {
                IF   (gsmAccess.begin(PINNUMBER)==GSM_READY)
                {
                notConnected = false
                }
                else
                {
                Display(F("Connection to carrier failed"))
                delay(1000)
                }
        }

        IF    (notConnected == false)
        {
```

```
        Display(F("Sending SMS"))
        String SMSMess = CLIENTID+" Communication lost. Reset Requested. Status prior reset: "
        alarmcode = generatealarmcode()
        SMSMess = SMSMess + alarmcode + "_R"+String(resetcondition)
        char mensajeSMS[200]=""
        SMSMess.toCharArray(mensajeSMS,200)
          sendSMS(mensajeSMS, remoteNumber)
          sendSMS(mensajeSMS, remoteNumber_mmto1)
          sendSMS(mensajeSMS, remoteNumber_mmto2)
          sendSMS(mensajeSMS, remoteNumber_cust1)
          #sendSMS(mensajeSMS, remoteNumber_cust2)
        gsmAccess.shutdown()
        }
     SET(amberLEDpin, LOW)
      }
      alertsent = 1
    }

 latchedfault = 0 # clear faults
 majorfault = 0  # clear faults

 for (int x=0 x<32 x++)
 {
   latchedfaults[x] = 0  # clear latched faults
 }

alarmcode = generatealarmcode()

 IF    (echoReply == 1)
    {
      healthlevel_actual = healthlevel_global
      SendDataChoreo(alarmcode)
    }
    else
    {
      healthlevel_actual = healthlevel_actual - 1
     IF    (GSMEnable == 1)
     {
         SET(amberLEDpin, HIGH)
        boolean notConnected = true
        long beginWait = millis()
        while ((millis() - beginWait < 900000) AND (notConnected))
        {
                IF   (gsmAccess.begin(PINNUMBER)==GSM_READY)
                {
                notConnected = false
                }
                else
                {
                Display(F("Connection to carrier failed"))
                delay(1000)
                }
        }

        IF    (notConnected == false)
        {
        Display(F("Sending SMS"))
        String SMSMess = CLIENTID+" Communication lost. Reset Completed. Status: "
        alarmcode = generatealarmcode()
        SMSMess = SMSMess + alarmcode + "_R"+String(resetcondition)
        char mensajeSMS[200]=""
        SMSMess.toCharArray(mensajeSMS,200)
          sendSMS(mensajeSMS, remoteNumber)
          sendSMS(mensajeSMS, remoteNumber_mmto1)
          sendSMS(mensajeSMS, remoteNumber_mmto2)
          sendSMS(mensajeSMS, remoteNumber_cust1)
          #sendSMS(mensajeSMS, remoteNumber_cust2)
        gsmAccess.shutdown()
        }
     SET(amberLEDpin, LOW)
      }
      alertsent = 1
    }

 SET(redLEDpin, LOW)
 resetcondition = 0
 delay(1000)
 SET(amberLEDpin, LOW)
 Display(F("Faults cleared!"))
}
```

```
FUNCTION checkLatched()
{
  int alarm = 0

  for (int x=0 x<32 x++)
  {
    IF    (latchedfaults[x] == 1)
    {
       alarm = 1
    }
  }

  #Temperature latching alarm deactivated since its time consuming
  /*sensors.requestTemperatures()
  float sensorValue = sensors.getTempCByIndex(0)#(((analogRead(A2)/1024.0)*5.0)-0.5)*100.0       #READ Analog Inputs and convert to
Temperature
  IF    (sensorValue > OVHT_THRESHOLD)                #CHECK IF    Temperature is OK, IF    not trigger alarm.
  {
    alarm=1
    #Display("Temperature alarm latched! Actual Temp: ")
    #Display(sensorValue)
  }
  */
  IF    (alarm == 1)
  return 1
  else
  return 0
}

FUNCTION checkIO()        # THIS FUNCTION CHECKS FOR ALARM CONDITIONS, ALL ALARM CONDITIONS SHOULD BE
PROGRAMMED IN THIS SECTION
{
    int alarm=0
                                #CHECK Digital Inputs based on status and IF    enabled in mask_unmask section on the header
    for (int x=22, y=0x<54x++)
    {
      IF    (x>=50)
      {
        IF    (MaskUnmaskLOGIC[y]==1)
        {
        IF    (READ(A15)==LOW  AND  MaskUnmaskPINS[y] == 1)
        {
            alarm=1
            latchedfaults[y] = 1
        }
        }
        else
        {
        IF    (READ(A15)==HIGH  AND  MaskUnmaskPINS[y] == 1)
        {
            alarm=1
            latchedfaults[y] = 1
        }
        }

        IF    (MaskUnmaskLOGIC[y]==1)
        {
        IF    (READ(A14)==LOW  AND  MaskUnmaskPINS[y] == 1)
        {
            alarm=1
            latchedfaults[y] = 1
        }
        }
        else
        {
        IF    (READ(A14)==HIGH  AND  MaskUnmaskPINS[y] == 1)
        {
            alarm=1
            latchedfaults[y] = 1
        }
        }

        IF    (MaskUnmaskLOGIC[y]==1)
        {
        IF    (READ(A13)==LOW  AND  MaskUnmaskPINS[y] == 1)
        {
            alarm=1
            latchedfaults[y] = 1
        }
```

```
      }
      else
      {
      IF    (READ(A12)==HIGH  AND  MaskUnmaskPINS[y] == 1)
      {
           alarm=1
           latchedfaults[y] = 1
      }
      }
     }
     else
     {
           IF    (MaskUnmaskLOGIC[y]==1)
           {
           IF    (READ(x)==LOW  AND  MaskUnmaskPINS[y] == 1)
           {
           alarm=1
           latchedfaults[y] = 1
           }
           }
           else
           {
           IF    (READ(x)==HIGH  AND  MaskUnmaskPINS[y] == 1)
           {
           alarm=1
           latchedfaults[y] = 1
           }
           }
     }
     y++
  }

  IF   (alarm)
  SET(redLEDpin, HIGH)
}

FUNCTION generatealarmcode()      # CHECKS FOR ALL PINS IN BOARD AND GENERATES A BINARY STRING APPENDING ALL PIN
STATUS TOGETHER... of course IF    not bypassed
{
  String binaryalarm = ""
  String post_binaryalarm = ""

  for(int x=0x<32x++)
  {
    IF    (latchedfaults[x] == 1  AND  MaskUnmaskPINS[x] == 1)
    binaryalarm = binaryalarm + "1"
    else
    binaryalarm = binaryalarm + "0"
  }

  for (int y=0,z=31y<32y++,z--)
  {
    post_binaryalarm = post_binaryalarm + binaryalarm.charAt(z)
  }

return post_binaryalarm
}

FUNCTION generate_timestamp()
{

 #Prepare Datagram
   #format year
   String year_datagram = String(year())
   #format month
   int month_noformat = month()
   String month_datagram
   IF    (month_noformat<10)
   {
     month_datagram = "0"+String(month_noformat)
   }
   else
   {
     month_datagram = String(month_noformat)
   }
   #format day
   int day_noformat = day()
   String day_datagram
   IF    (day_noformat<10)
   {
```

```
    day_datagram = "0"+String(day_noformat)
   }
   else
   {
    day_datagram = String(day_noformat)
   }
    #format hour
   int hour_noformat = hour()
   String hour_datagram
   IF    (hour_noformat<10)
   {
    hour_datagram = "0"+String(hour_noformat)
   }
   else
   {
    hour_datagram = String(hour_noformat)
   }

    #format minute
   int minute_noformat = minute()
   String minute_datagram
   IF    (minute_noformat<10)
   {
    minute_datagram = "0"+String(minute_noformat)
   }
   else
   {
    minute_datagram = String(minute_noformat)
   }

    #format second
   int second_noformat = second()
   String second_datagram
   IF    (second_noformat<10)
   {
    second_datagram = "0"+String(second_noformat)
   }
   else
   {
    second_datagram = String(second_noformat)
   }

   String TimeStamp = year_datagram + month_datagram + day_datagram + "T" + hour_datagram + minute_datagram + second_datagram
  return TimeStamp
}

FUNCTION generate_datagram(String alarmcode)
{

   String TimeStamp_datagram = generate_timestamp()
   #IdentIF   y customer for the TimeStamp
   String CLIENTID_datagram = CLIENTID+"_"


   #Get the code
   String alarmcode_datagram = alarmcode

   # Concat all together
   String var = TimeStamp_datagram+ "_" + CLIENTID_datagram + alarmcode_datagram

   return var
}

FUNCTION LogEntry_datagram(String alarmcode)
{
  String timestamp = generate_timestamp()
  #sensors.requestTemperatures()
  #float sensorValue = sensors.getTempCByIndex(0)#(((analogRead(A2)/1024.0)*5.0)-0.5)*100.0
  #int temp = sensorValue
  String sensor_data = "NA"#String(temp)

  String datagram = String(CLIENTID) + "," + String(timestamp) + "," + sensor_data

  for (int x=31x>=0x--)
   {
    datagram = datagram + ","+ alarmcode.charAt(x)
   }

  datagram = datagram + "," + String(resetcondition)
```

```
  return datagram

}


FUNCTION CheckIF   _EMAIL_required(String alarmcode)
{
  int required = 0
    for (int x=31,y=0x>=0x--,y++)
    {
      IF    (alarmcode.charAt(x)=='1'  AND  MaskUnmaskEMAIL[y] == 1)   # check IF    any of the latched alarm triggers an emai IF    it does
returns a signal that can be used to send an email
       {
        required = 1
       }
    }
   return required
}

FUNCTION Compose_EMAIL(String alarmcode)
{
  String EMAIL_MESSAGE = EMAIL_BODDY_MSG_HEADER  + "\n" +"Time Stamp: \n"

  String timestamp = generate_timestamp()
 Display(timestamp)

  int alarm_qty = 0
  int cont = 0

  EMAIL_MESSAGE = EMAIL_MESSAGE + timestamp + "\n"

  IF    (resetcondition == 0)
  {
    IF    (setupinprogress==1)
    {

     EMAIL_MESSAGE = EMAIL_MESSAGE + "\n SYSTEM REBOOT SUCCESSFULY COMPLETED"
    }
    else
    {
         for (int x=31,y=0x>=0x--,y++)
         {
            IF    (alarmcode.charAt(x)=='1')  # builds an email with all the latched alarms
            {
              IF    (alarm_qty<10)
              {
              EMAIL_MESSAGE = EMAIL_MESSAGE + ALARM_CODE_NAME[y]
              cont++
              }
              alarm_qty++
            }
         }
         IF    (alarm_qty>9)
         {
           EMAIL_MESSAGE = EMAIL_MESSAGE +"\n and other "+String(alarm_qty-cont)+" latched alarms detected"
          }

        IF    (alarm_qty == 0)
        {
          EMAIL_MESSAGE = EMAIL_MESSAGE + "\n SYSTEM IN NORMAL OPERATION "
        }
    }

  }
  else
  {
    EMAIL_MESSAGE = EMAIL_MESSAGE + "\n PANEL ALARM RESET ACTIVATED "
  }
    EMAIL_MESSAGE = EMAIL_MESSAGE + "\n" + EMAIL_BODDY_MSG_FOOTER
    Display((EMAIL_MESSAGE))
  return EMAIL_MESSAGE
}

FUNCTION digitalClockDisplay(){
  # digital clock display of the time
  Display(hour())
  printDigits(minute())
  printDigits(second())
  Display(" ")
  Display(day())
```

```
  Display(" ")
  Display(month())
  Display(" ")
  Display(year())
  Display()
}

FUNCTION printDigits(int digits){
  # utility for digital clock display: prints preceding colon and leading 0
  Display(":")
  IF   (digits < 10)
    Display('0')
  Display(digits)
}

FUNCTION SendEmailAlertsChoreo(String alarmcode) {
 Display(F("\n Sending Email"))
 SET(blueLEDpin, HIGH)
 TembooChoreo SendEmailChoreo(client)
    Display(F("Begins"))
    String EMAIL_BODY = Compose_EMAIL(alarmcode)
    Display(F("Ends"))
    # Invoke the Temboo client
    SendEmailChoreo.begin()

    # Set Temboo account credentials
    SendEmailChoreo.setAccountName(TEMBOO_ACCOUNT)
   SendEmailChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME)
    SendEmailChoreo.setAppKey(TEMBOO_APP_KEY)
    # Set Choreo inputs
    String FromAddressValue = EMAIL_FROM
    SendEmailChoreo.addInput("FromAddress", FromAddressValue)
    String ServerValue = EMAIL_SERVER
    SendEmailChoreo.addInput("Server", ServerValue)
    String UsernameValue = EMAIL_USERNAME
    SendEmailChoreo.addInput("Username", UsernameValue)
    String PortValue = EMAIL_PORT
    SendEmailChoreo.addInput("Port", PortValue)
    String UseSSLValue = EMAIL_SSL
    SendEmailChoreo.addInput("UseSSL", UseSSLValue)
    String ToAddressValue = EMAIL_TO
    SendEmailChoreo.addInput("ToAddress", ToAddressValue)
    String SubjectValue = EMAIL_SUBJECT
    SendEmailChoreo.addInput("Subject", SubjectValue)
    String PasswordValue = EMAIL_PASSWORD
    SendEmailChoreo.addInput("Password", PasswordValue)
    String MessageBodyValue = EMAIL_BODY
    SendEmailChoreo.addInput("MessageBody", MessageBodyValue)
    /*Display(FromAddressValue)
    Display(ServerValue)
    Display(UsernameValue)
    Display(PortValue)
    Display(UseSSLValue)
    Display(ToAddressValue)
    Display(SubjectValue)
    Display(PasswordValue)
    Display(MessageBodyValue)
    */
    # IdentIF   y the Choreo to run
    SendEmailChoreo.setChoreo("/Library/Utilities/Email/SendEmail")
  # Run the Choreo
  unsigned int returnCode = SendEmailChoreo.run()

  # A non-zero return code indicates an error
  IF    (returnCode != 0)
  {
    Display(F("Failed!"))
  }
  else
  {
    Display(F("Success!"))
  }

   SendEmailChoreo.close()
   SET(blueLEDpin, LOW)
}

FUNCTION MakeLogEntryChoreo(String alarmcode)
{
    Display(F("\n Logging Data in Remote Station"))
```

```
    SET(blueLEDpin, HIGH)
    String datagram = LogEntry_datagram(alarmcode)

    TembooChoreo AppendRowChoreo(client)

    # Invoke the Temboo client
    AppendRowChoreo.begin()

    # Set Temboo account credentials
AppendRowChoreo.setAccountName(TEMBOO_ACCOUNT)
    AppendRowChoreo.setAppKeyName(T_APP_KEY_NAME)
    AppendRowChoreo.setAppKey(T_APP_KEY)

    # Set Choreo inputs
    AppendRowChoreo.addInput("SpreadsheetTitle", LOG_FILE)
    AppendRowChoreo.addInput("RowData", datagram)
    AppendRowChoreo.addInput("RefreshToken", RefreshTokenValue)
    AppendRowChoreo.addInput("ClientSecret", ClientSecretValue)
    AppendRowChoreo.addInput("ClientID", ClientIDValue)

AppendRowChoreo.setChoreo("/Library/Google/Spreadsheets/AppendRow")

    # Run the Choreo when results are available, print them to serial
  # Run the Choreo
  unsigned int returnCode = AppendRowChoreo.run()

  # A non-zero return code indicates an error
  IF    (returnCode != 0)
  {
    Display(F("Failed!"))
  }
  else
  {
    Display(F("Success!"))
  }

    AppendRowChoreo.close()
    SET(blueLEDpin, LOW)
}


FUNCTION SendDataChoreo(String alarmcode)
{
    Display(F("\n Sending Actual Status"))
    SET(blueLEDpin, HIGH)
    String datagram = LogEntry_datagram(alarmcode)
    TembooChoreo UpdateRowChoreo(client)
    # Invoke the Temboo client
    UpdateRowChoreo.begin()

    # Set Temboo account credentials   UpdateRowChoreo.setAccountName(TEMBOO_ACCOUNT)
UpdateRowChoreo.setAppKeyName(T_APP_KEY_NAME)
    UpdateRowChoreo.setAppKey(T_APP_KEY)
    String RowDataValue = datagram
    UpdateRowChoreo.addInput("RowData", RowDataValue)
    UpdateRowChoreo.addInput("RefreshToken", RefreshTokenValue)
    UpdateRowChoreo.addInput("ClientSecret", ClientSecretValue)
    UpdateRowChoreo.addInput("SpreadsheetName", STATUS_FILE)
    UpdateRowChoreo.addInput("ClientID", ClientIDValue)
    UpdateRowChoreo.addInput("Row", STATUS_FILE_CLIENT_ROW)
    UpdateRowChoreo.addInput("WorksheetName", STATUS_FILE_TAB)
UpdateRowChoreo.setChoreo("/Library/Google/Spreadsheets/UpdateRow")
unsigned int returnCode = UpdateRowChoreo.run()

  # A non-zero return code indicates an error
  IF    (returnCode != 0)
  {
    Display(F("Failed!"))
  }
  else
  {
    Display(F("Success!"))
  }

    UpdateRowChoreo.close()
    SET(blueLEDpin, LOW)
}

FUNCTION CheckLicenseChoreo()
{
```

```
  SET(amberLEDpin, HIGH)
  SET(blueLEDpin, HIGH)
  Display(F("Retrieving License..."))
    TembooChoreo RetrieveCellValueChoreo(client)

    # Invoke the Temboo client
    RetrieveCellValueChoreo.begin()

    # Set Temboo account credentials
    RetrieveCellValueChoreo.setAccountName(TEMBOO_ACCOUNT)
    RetrieveCellValueChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME)
    RetrieveCellValueChoreo.setAppKey(TEMBOO_APP_KEY)
    # Set Choreo inputs
    RetrieveCellValueChoreo.addInput("RefreshToken", RefreshTokenValue)
    String SpreadsheetNameValue = LICENSE_FILE
    RetrieveCellValueChoreo.addInput("SpreadsheetName", SpreadsheetNameValue)
    RetrieveCellValueChoreo.addInput("ClientSecret", ClientSecretValue)
    RetrieveCellValueChoreo.addInput("ClientID", ClientIDValue)
    String CellLocationValue = LICENSE_LOCATION
    RetrieveCellValueChoreo.addInput("CellLocation", CellLocationValue)
    String WorksheetNameValue = LICENSE_TAB
    RetrieveCellValueChoreo.addInput("WorksheetName", WorksheetNameValue)
RetrieveCellValueChoreo.setChoreo("/Library/Google/Spreadsheets/RetrieveCellValue")
    # Run the Choreo when results are available, print them to serial
    # Run the Choreo when results are available, print them to serial
    unsigned int returnCode = RetrieveCellValueChoreo.run()
   # a response code of 0 means success print the API response
    IF   (returnCode == 0) {

      String ValorCelda # a String to hold the CellValue
      String NewAccToken # a String to hold the NewAccToken

      while(RetrieveCellValueChoreo.available()) {
        # read the name of the output item
        String name = RetrieveCellValueChoreo.readStringUntil('\x1F')
        name.trim()

        # read the value of the output item
        String data = RetrieveCellValueChoreo.readStringUntil('\x1E')
        data.trim()

        # assign the value to the appropriate String
        IF    (name == "NewAccessToken") {
          NewAccToken = data
        } else IF    (name == "CellValue") {
          ValorCelda = data
        }
      }

      T_APP_KEY = ValorCelda

    } else {
      # there was an error
      # print the raw output from the choreo

      while(RetrieveCellValueChoreo.available()) {
        char c = RetrieveCellValueChoreo.read()
        Display(c)
      }
    }

  RetrieveCellValueChoreo.close()
  SET(amberLEDpin, LOW)
  SET(blueLEDpin, LOW)
  Display(F("License: "))
  Display(T_APP_KEY)
}

FUNCTION sendSMS(char txtMsg[200], char Number[20]){
  Display(F("Message to mobile number: "))
  Display(Number)
  # sms text
  Display(F("SENDING"))
  Display()
  Display(F("Message:"))
  Display(txtMsg)
  # send the message
  sms.beginSMS(Number)
  sms.print(txtMsg)
  sms.endSMS()
```

```
  Display(F("\nCOMPLETE!\n"))
}

FUNCTION testconnection() {
  TembooChoreo HelloWorldChoreo(client)

  # Set Temboo account credentials
HelloWorldChoreo.setAccountName(TEMBOO_ACCOUNT)
HelloWorldChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME)
  HelloWorldChoreo.setAppKey(TEMBOO_APP_KEY)
  # Set Choreo inputs
  String ValueValue = "OK"
  HelloWorldChoreo.addInput("Value", ValueValue)
HelloWorldChoreo.setChoreo("/Library/Utilities/Test/HelloWorld")
  # Run the Choreo
  unsigned int returnCode = HelloWorldChoreo.run()
  # A return code of zero means everything worked
  IF    (returnCode == 0) {
    while (HelloWorldChoreo.available()) {
      String name = HelloWorldChoreo.readStringUntil('\x1F')
      name.trim()
      String data = HelloWorldChoreo.readStringUntil('\x1E')
      data.trim()

      IF    (name == "Result") {
        IF    (data == "Hello, OK!") {

          return 1
        }
        else{

          return 0
        }
      }
    }
  }
  HelloWorldChoreo.close()
}

const int NTP_PACKET_SIZE = 48 # NTP time is in the first 48 bytes of message
byte packetBuffer[NTP_PACKET_SIZE] #buffer to hold incoming & outgoing packets

FUNCTION getNtpTime()
{
  while (Udp.parsePacket() > 0)  # discard any previously received packets
  Display(F("Transmit NTP Request"))
  sendNTPpacket(timeServer)
  uint32_t beginWait = millis()
  while (millis() - beginWait < 1500) {
    int size = Udp.parsePacket()
    IF    (size >= NTP_PACKET_SIZE) {
      Display(F("Receive NTP Response"))
      Udp.read(packetBuffer, NTP_PACKET_SIZE)  # read packet into the buffer
      unsigned long secsSince1900
      # convert four bytes starting at location 40 to a long integer
      secsSince1900 =  (unsigned long)packetBuffer[40] << 24
      secsSince1900 |= (unsigned long)packetBuffer[41] << 16
      secsSince1900 |= (unsigned long)packetBuffer[42] << 8
      secsSince1900 |= (unsigned long)packetBuffer[43]
      return secsSince1900 - 2208988800UL + timeZone * SECS_PER_HOUR
    }
  }
  Display(F("No NTP Response :-("))
  return 0 # return 0 IF    unable to get the time
}

FUNCTION sendNTPpacket(IPAddress &address)
{
  # set all bytes in the buffer to 0
  memset(packetBuffer, 0, NTP_PACKET_SIZE)
  # Initialize values needed to form NTP request
  # (see URL above for details on the packets)
  packetBuffer[0] = 0b11100011   # LI, Version, Mode
  packetBuffer[1] = 0     # Stratum, or type of clock
  packetBuffer[2] = 6     # Polling Interval
  packetBuffer[3] = 0xEC  # Peer Clock Precision
  # 8 bytes of zero for Root Delay & Root Dispersion
  packetBuffer[12]  = 49
  packetBuffer[13]  = 0x4E
  packetBuffer[14]  = 49
```

```
  packetBuffer[15]  = 52
  # all NTP fields have been given values, now
  # you can send a packet requesting a timestamp:
  Udp.beginPacket(address, 123) #NTP requests are to port 123
  Udp.write(packetBuffer, NTP_PACKET_SIZE)
  Udp.endPacket()
}
```

Bibliografía

[1] D. Lineweber y S. McNulty, «The Cost of Power Disturbances to Industrial & Digital Economy Companies,» PRIMEN, Madison WI, 2001.

[2] M. Ristic, M. Thakur y M. Vaziri, «The major differences between electromechanical and microprocessor base technologies in relay setting rules for transformer current differential protection,» de *58th Annual Conference for Protective Relay Engineers*, 2005.

[3] E. Schweitzer III, J. J. Kumm, M. S. Weber y D. Hou, «Philosophies for testing protective relays,» de *20th Annual Western Protective Relay Conference*, Spokane WA, 1993.

[4] D. Hou, H. J. Altuve y E. Schweitzer III, «Distribution System Protection, Automation, and Monitoring,» de *Modern Solutions for Protection, Control, and Monitoring of Electric Power Systems*, Pullman , WA: Quality Books, 2010, p. 29.

[5] H. J. Altuve, *Protecting Power Systems for Engineers,* Pullman, WA: SEL, 2015.

[6] V. Skendzic, «Power System Communications,» de *Modern Solutions for Protection, Control, and Monitoring of Electric Power Systems*, Pullman, WA: Quality Books, 2010, p. 211.

[7] A. Idris, N. Anuar, M. Misron y F. Fauzi, «Back to Results: The readiness of Cloud Computing, A case study in Politeknik Sultan Salahuddin Absul Aziz Shah, Sha Alam,» de *2014 International Conference on Computational Science and Technology*, Kota Kinabalu, 2014.

[8] A. Dionicio, M. Monjaras, S. Sanchez, M. Guel, G. Gonzalez, O. Vazquez, J. Estrada, H. Altuve, I. Muoz, I. Yanez, P. Loza y J. Needs, «Using spread-spectrum radio as a cost-effective teleprotection channel - field experience in Mexico,» de *Developments in Power System Protection (DPSP 2010). Managing the Change, 10th IET International Conference on*, Manchester, 2010.

[9] J. B. Mooney, E. O. Schweitzer III y H. J. Altuve, «Transmission Line Protection,» de *Modern Solutions for Protection, Control, and Monitoring of Electric Power Systems*,

E. O. Schweitzer III y H. J. Altuve Ferrer, Edits., Pullman, WA: Quality Books, 2010, p. 57.

[10] J. McDonald, «Substation automation. IED integration and availability of information,» *Power and Energy Magazine,* vol. 1, nº 2, pp. 22-31, 03/04 2003.

[11] W. Siriwatworasakul y N. Hoonchareon, «Conceptual Design of Wide Area Protection in Transmission System,» de *10th International Conference on Electrical Engineering, Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, Bankok, 2013.

[12] J. Arinez, «Integration requirements for manufacturing-based Energy Management Systems,» de *Innovative Smart Grid Technologies (ISGT)*, Gaithersburg, MD, 2010.

[13] K. Stouffer, J. Falco y K. Kent, «Guide to Supervisory Control and Data Acquisition (SCADA) and Industrial Control Systems Security,» Gaithersburg, MD, 2006.

[14] E. O. Schweitzer III, *Protecting Power Systems for Engineers,* Pullman, WA, 2015.

[15] A. Idris, N. Anuar, M. Misron y F. Fauzi, «Back to Results The readiness of Cloud Computing: A case study in Politeknik Sultan Salahuddin Abdul Aziz Shah, Shah Alam,» de *Computational Science and Technology (ICCST), 2014 International Conference on*, Kota Kinabalu, 2014.

[16] Y. Jadeja y K. Modi, «Cloud computing - concepts, architecture and challenges,» de *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*, Kumaracoil, 2012.

[17] D. Ashlock y A. Warren, «The Engineer's Guide to Signal Conditioning,» 2015. [En línea]. Available: NI.

[18] A. S.R.L, «arduino,» Smart Projects SRL, [En línea]. Available: www.arduino.org/products/boards/arduino-mega-2560. [Último acceso: 04 07 2016].

[19] Arduino, «Arduino Ethernet Shield,» Smart Projects Srl, [En línea]. Available: www.arduino.org/products/shields/arduino-ethernet-shield. [Último acceso: 05 07 2016].

[20] WIZnet, «W5100 Datasheet,» WIZnet, 2011. [En línea]. Available: http://www.wiznet.co.kr/wp-

content/uploads/wiznethome/Chip/W5100/Document/W5100_Datasheet_v1.2.6.pdf. [Último acceso: 05 07 2016].

[21] Arduino, «Arduino GSM Shield,» Smart Projects Srl, 05 07 2016. [En línea]. Available: www.arduino.org/products/shields/arduino-gsm-shield-2.

[22] Arduino.CC, «Quectel M10 Hardware Design,» [En línea]. Available: https://www.arduino.cc/en/uploads/Main/Quectel_M10_datasheet.pdf. [Último acceso: 05 07 2016].

[23] Sparkfun, «Pro Micro - 5V / 16MHz,» Sparkfun, 06 07 2016. [En línea]. Available: https://www.sparkfun.com/products/12640.

[24] Atmel, *ATmega16U4/ATmega32U4 Datasheet,* Atmel, 2016.

[25] I. Schweitzer Engineering Laboratories, *SEL-9321 Low-Voltage DC Power Supply,* Pullman, Washington: SEL, 2015.

[26] L. Kylinchip Electronic (Shangai) Co., «Datasheet, 400kHz 60V 4A Switching Current Boost / Buck-Boost / Inverting DC/DC Converter,» [En línea]. Available: http://www.kylinchip.com/PDF/XL6009%20datasheet.pdf. [Último acceso: 07 07 2016].

[27] O. Impulse, «XL6009 Schematic Diagram,» [En línea]. Available: https://www.openimpulse.com/blog/wp-content/uploads/wpsc/downloadables/XL6009-Schematic-Diagram.pdf. [Último acceso: 07 07 2016].

[28] I. Dataforth, «SCMD-MIAC/MIDC Miniature Digital Input Modules,» [En línea]. Available: http://www.dataforth.com/catalog/pdf/scmdmiac_midc.pdf. [Último acceso: 07 07 2016].

[29] I. Dataforth, «Backpanels, SCMD Accessries Digital I/O Module,» 07 07 2016. [En línea]. Available: http://www.dataforth.com/catalog/pdf/SCMD-PB16TSM.pdf.

[30] T. Dierk y E. Rescorla, «The Transport Layer Security (TLS) Protocol Version 1.2,» IETF, 2008.

[31] National Institute of Standars and Technology NIST, «Announcing the Advanced Encryption Standard (AES),» NIST, 2001.

[32] I. Goldberg, D. Wagner, R. Thomas y E. Brewer, «A Secure Environment for Untrusted Helper Applications,» de *Proceedings of the Sixth USENIX UNIX Security Symposium*, San Jose, CA, 1996.

[33] Temboo, «Temboo, a dynamic code generation platform,» Temboo, [En línea]. Available: https://temboo.com/platform. [Último acceso: 11 7 2016].

[34] Arduino, «Arduino Introduction,» Smart Projects srl, [En línea]. Available: https://www.arduino.cc/en/guide/introduction. [Último acceso: 11 06 2016].

[35] D. Hardt, *The OAuth 2.0 Authorization Framework,* Internet Engineering Task Force, 2012.