



Facultad de Ciencias Exactas
UNIVERSIDAD NACIONAL DE SALTA

**TECNICATURA ELECTRÓNICA UNIVERSITARIA
ELECTRÓNICA DIGITAL III**

**PROYECTO:
ANALIZADOR DE CONSUMO ELÉCTRICO**

Profesor: Maiver Wilfredo Villena.

Autor: Gabriel Nicolás Barrionuevo Ramirez.

Salta, Octubre de 2020.

RESUMEN DEL PROYECTO

INTRODUCCIÓN

Dando continuidad a la primera parte de éste proyecto como un “todo”, en esta segunda parte se pensó en el diseño de una interfaz que permita la consulta y el muestreo de los datos obtenidos por el analizador de consumo eléctrico de una manera más amigable con el usuario. También se busca almacenar los datos permitiendo la futura manipulación de los mismos dando verdadero sentido a la función del analizador.

El desafío para esta etapa del diseño es construir toda la infraestructura necesaria con componentes sencillos, disponibles en el mercado, y que permita la implementación de software y hardware al alcance tanto del diseñador como del usuario. Y que en lo posible funcione en cualquier plataforma de la que el usuario disponga.

OBJETIVOS

- 1.- Diseñar una infraestructura que permita el almacenamiento y muestreo de los datos.
- 2.- Escoger para el diseño dispositivos sencillos y disponibles en el mercado nacional.
- 3.- Escoger para el diseño software gratuito y en lo posible, libre.
- 4.- Minimizar los costos de construcción.
- 5.- Diseñar una interfaz multiplataforma amigable con el usuario.

ESTADO DEL ARTE

Como se mencionó en el primer informe, se han analizado diferentes dispositivos comerciales disponibles en el mercado nacional y proyectos libres que pretenden dar las mismas soluciones que el presente proyecto. Entre ellos, dispositivos como los que presenta Schneider Electric y proyectos libres como “Open Energy Monitor”.

Se evaluó principalmente cuánta información se entrega al usuario de las mediciones realizadas, la facilidad con la que el usuario interactúa con el dispositivo, la confiabilidad del funcionamiento y cuan robusto y seguro es el sistema de almacenamiento de datos.

Desde los aspectos que ahora se abordan, se observa que los dispositivos comerciales no permiten una gran manipulación de los datos, por el contrario, están diseñados con un fin específico y se limitan a presentar resultados que satisfagan sólo esa función. Esto los resguarda de errores humanos pero limita su potencial. Demandan la presencia física del usuario para la manipulación del dispositivo con interfaces poco intuitivas. Existe una relación muy estrecha entre la calidad operativa y de diseño, y su precio. Y en la mayoría de los casos, el diseño obliga a la compra de un dispositivo completo por cada máquina o sector de la red que se deseé monitorear.

ARQUITECTURA DEL SISTEMA

Se buscó un diseño simple pero robusto y confiable, que minimice los costos de producción y adquisición, que permitan la expansión y el ajuste de las funciones según los requerimientos del usuario. Todo esto rescindiendo de la necesidad de acudir hasta el dispositivo por parte del usuario para la consulta de los datos y por parte del administrador para realizar modificaciones o actualizaciones según los requerimientos del consumidor.

El siguiente diagrama describe a grandes rasgos la arquitectura del sistema:



RESULTADOS

Hasta el momento se ha alcanzado un diseño que permite el muestreo de los datos en tiempo real, es decir, al tiempo que los ACEs están midiendo, y el almacenamiento de ellos en una base de datos. El sistema también permite la consulta de los datos almacenados y a modo de ejemplo de cómo se podrían manipular esos datos, se los muestra en un gráfico histórico.

Por el momento sólo se han realizado pruebas dentro de una red privada debido a que darle salida a la nube al servidor supone cierto riesgo de seguridad para la red y se decidió no incursionar de forma insegura en un campo que supera los alcances del autor del proyecto en

estos momentos. Pero queda a la vista que una vez superada ésta limitación, el sistema debería comportarse de la misma manera.

CONCLUSIÓN

El resultado obtenido se debe en parte a los conocimientos adquiridos en las materias de electrónica digital I, II y III y a la investigación particular por parte del autor. En muchos casos, las temáticas a abordar y las herramientas a emplear superaron los alcances de la caja curricular de la carrera, pero debido a que se empleó software y hardware libres y muy populares en este tipo de proyectos, se dispuso con gran cantidad de información en blogs y artículos de la web. Permitiendo un avance ininterrumpido aun cuando se requirió de mucha autodidactica. Y aunque tal vez aún se puedan pulir más a fondo algunas cuestiones técnicas del diseño, los resultados arrojan una impresión de que el proyecto es totalmente viable, que está bien encaminado y que con algunos pequeños ajustes podría competir con otros dispositivos disponibles en el mercado.

También el diseño permite una gran una gran escalabilidad, y reajustes que ampliarían y optimizarían su funcionamiento según las necesidades del consumidor.

Cabe destacar que el proyecto es una parte de las dos que conforman el “todo” y que para darle continuidad inmediata a la primera parte es que se empleó el módulo WiFi basado en ESP8266 exclusivamente para dar comunicación inalámbrica a la placa de desarrollo Arduino NANO ya montada en el ACE. A futuro se podría implementar un módulo NodeMCU8266 en reemplazo de la placa Arduino cubriendo la función que actualmente ella cumple optimizando las prestaciones del dispositivo y abaratando los costos de construcción.

REFERENCIAS

OPEN ENERGY MONITOR:

<https://openenergymonitor.org/>

Analizador de consumo de potencia eléctrica con Arduino:

<https://e-archivo.uc3m.es/handle/10016/23628>

Equipo de medición de energía de bajo coste:

<https://repositorio.comillas.edu/jspui/bitstream/11531/6101/1/TFG001453.pdf>

LaMp:

<https://wiki.debian.org/LaMp>

Debian Jessie:

<https://www.debian.org/releases/jessie/>

MySQL:

www.mysql.com

Apache2:

<https://httpd.apache.org/>

Eclipse Mosquitto:

<https://mosquitto.org/>

Paho:

<https://www.eclipse.org/paho/>

jQuery:

<https://jquery.com/>

Bootstrap:

<https://getbootstrap.com/>

Arduino:

<https://www.arduino.cc/>

pubsubclient:

<https://pubsubclient.knolleary.net/>

ESP8266:

<https://www.espressif.com/en/products/socs/esp8266>

Python:

<https://www.python.org/>

MQTT:

<https://mqtt.org/>

*DESCRIPCIÓN DETALLADA DEL
DESARROLLO*

INTRODUCCIÓN

Las grandes industrias comúnmente solicitan el servicio de expertos que orienten las correcciones necesarias en la red para la optimización del consumo de la energía eléctrica y evite penalizaciones por el consumo de energía reactiva. El mismo requerimiento se traslada a los pequeños emprendimientos en los cuales se hace uso excesivo de máquinas eléctricas con motores y bobinas.

Tales correcciones se diseñan partiendo por el relevamiento del consumo eléctrico del local. Lo cual implica una inspección exhaustiva y tediosa que resulta en horas de trabajo que se reducirían con el empleo de algún equipo de medición que brinde las mediciones necesarias.

Por otro lado, en el sector inmobiliario resulta de interés conocer el consumo de ciertos sectores de la red que permitan el cobro preciso y detallado del servicio. O en los hogares, algunos usuarios pueden desear monitorizar el consumo mensual.

Para todos estos requerimientos, existen analizadores de consumo eléctrico comerciales. Pero en ciertos casos, éstos no se destacan ser sencillos para un usuario inexperto ni por sus prestaciones. En general, los dispositivos comerciales son diseñados para un determinado fin, y se limitan a cumplir sólo esa tarea. Desde el punto de vista del diseñador electrónico, no son versátiles y no permiten su manipulación para adaptarlos para otras tareas a medida, o ajustar y optimizar su funcionamiento para determinado fin.

Para entender el origen y en qué se centra el diseño será necesaria una introducción teórica.

INTRODUCCIÓN TEÓRICA

MQTT:

El protocolo MQTT se ha convertido en uno de los principales pilares del IoT por su sencillez y ligereza. Ambos son condicionantes importantes dado que los dispositivos de IoT, a menudo, tienen limitaciones de potencia, consumo, y ancho de banda.

MQTT son las siglas MQ Telemetry Transport, aunque en primer lugar fue conocido como Message Queuing Telemetry Transport. Es un protocolo de comunicación M2M (machine-to-machine) de tipo message queue.

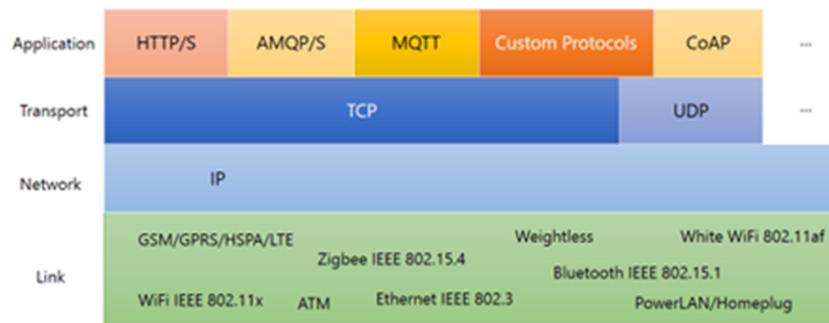
MQTT fue creado por el Dr. Andy Stanford-Clark de IBM y Arlen Nipper de Arcom (ahora Eurotech) en 1999 como un mecanismo para conectar dispositivos empleados en la industria petrolera.

Aunque inicialmente era un formato propietario, en 2010 fue liberado y pasó a ser un estándar en 2014 según la OASIS (Organization for the Advancement of Structured Information Standards).

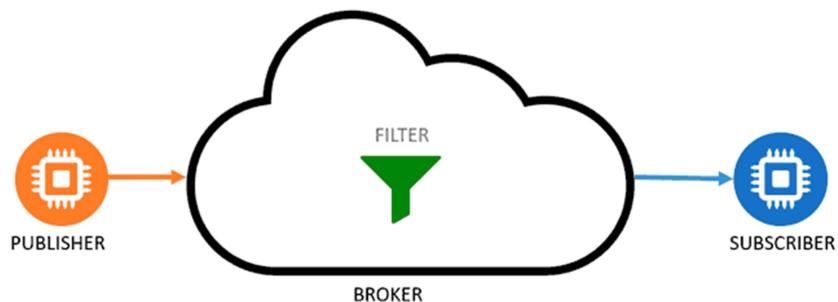
Funcionamiento de MQTT:

MQTT es un servicio de mensajería con patrón publicador/suscriptor (pub-sub) a diferencia de HTTP que es petición/respuesta. La arquitectura de MQTT sigue una topología de estrella, teniendo un nodo central que hace de servidor o “broker”.

MQTT es un protocolo pensado para IoT que está al mismo nivel que HTTP, o sea que se encuentra entre las capas superiores del modelo OSI.

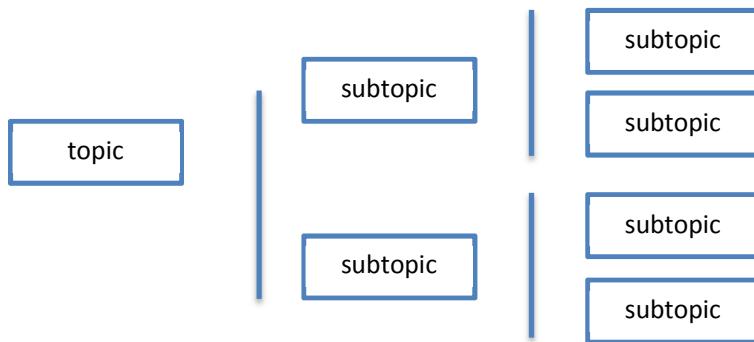


Para filtrar los mensajes que son enviados a cada cliente los mensajes se disponen en topics organizados jerárquicamente. Un cliente puede publicar un mensaje en un determinado topic. Otros clientes pueden suscribirse a este topic, y el broker le hará llegar los mensajes.

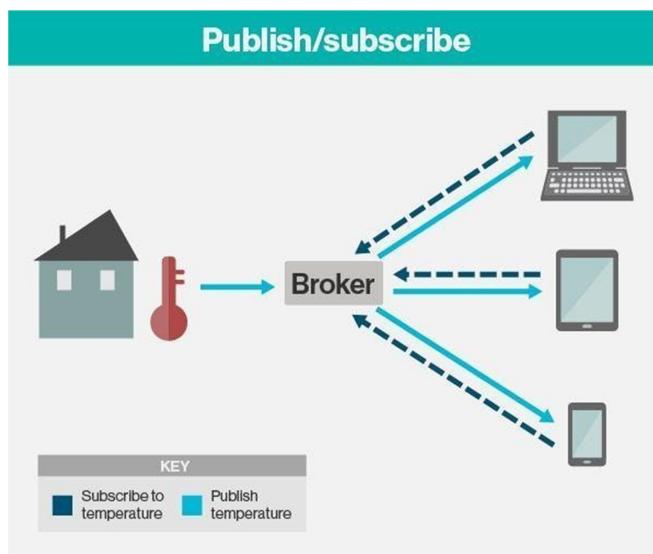


Organización jerárquica de los topics:

Un cliente puede publicar un mensaje en determinado topic, y otro cliente puede estar suscripto a ese topic para recibir ese mensaje. Quien publica el mensaje no sabe a quién va dirigido, es decir, quien lo leerá, sino que el broker es quién lo sabe y se encarga de enviar el mensaje a los clientes suscriptos a ese topic. Además como los topics tienen una organización jerárquica los clientes pueden suscribirse a un topic particular o también pueden suscribirse a un topic y a todos los siguientes aguas abajo.

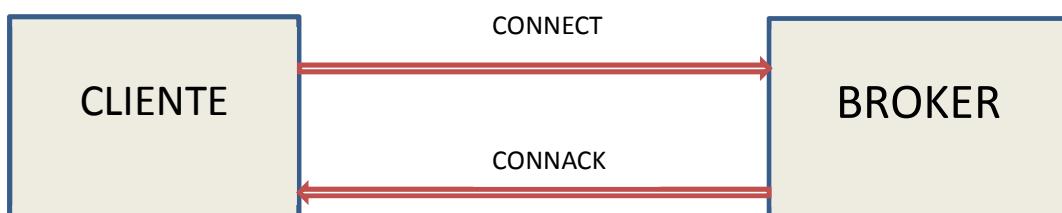


Publicación/Suscripción:

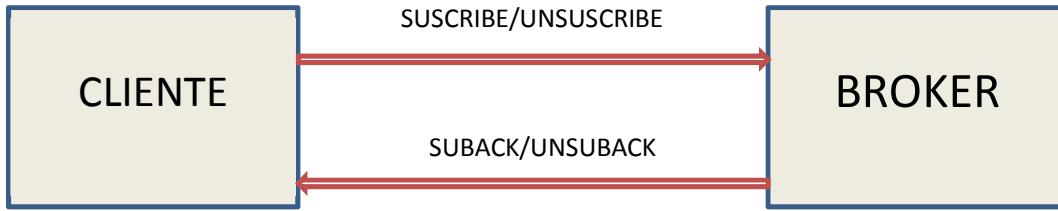


Los clientes inician una conexión TCP/IP con el broker, el cual mantiene un registro de los clientes conectados. Esta conexión se mantiene abierta hasta que el cliente la finaliza. Por defecto, MQTT emplea el puerto 1883 y el 8883 cuando funciona sobre TLS.

Para ello el cliente envía un mensaje CONNECT que contiene información necesaria (nombre de usuario, contraseña, client-id...). El broker responde con un mensaje CONNACK, que contiene el resultado de la conexión (aceptada, rechazada, etc).



Para suscribirse y desuscribirse se emplean mensajes SUBSCRIBE y UNSUBSCRIBE, que el servidor responde con SUBACK y UNSUBACK.



Para enviar los mensajes el cliente emplea mensajes PUBLISH, que contienen el topic y el payload.



Por otro lado, para asegurar que la conexión está activa los clientes mandan periódicamente un mensaje PINGREQ que es respondido por el servidor con un PINGRESP. Finalmente, el cliente se desconecta enviando un mensaje de DISCONNECT.

Estructura de un mensaje MQTT

Uno de los componentes más importantes del protocolo MQTT es la definición y tipología de los mensajes, ya que son una de las bases de la agilidad en la que radica su fortaleza. Cada mensaje consta de 3 partes:

Always	Optional	Optional	
Fixed Header		Optional Header	Payload
Control Header	Packet Length	0-Y Bytes	0-256Mbs
1 Byte	1-4 Bytes		

- Cabecera fija. Ocupa 2 a 5 bytes, obligatorio. Consta de un código de control, que identifica el tipo de mensaje enviado, y de la longitud del mensaje. La longitud se codifica en 1 a 4 bytes, de los cuales se emplean los 7 primeros bits, y el último es un bit de continuidad.
- Cabecera variable. Opcional, contiene información adicional que es necesaria en ciertos mensajes o situaciones.
- Contenido (payload). Es el contenido real del mensaje. Puede tener un máximo de 256 Mb aunque en implementaciones reales el máximo es de 2 a 4 kB.

QoS:

MQTT dispone de un mecanismo de calidad del servicio o QoS, entendido como la forma de gestionar la robustez del envío de mensajes al cliente ante fallos (por ejemplo, de conectividad).

MQTT tiene tres niveles QoS posibles.

- QoS 0 unacknowledged (at most one): El mensaje se envía una única vez. En caso de fallo puede que alguno no se entregue.
- QoS 1 acknowledged (at least one): El mensaje se envía hasta que se garantiza la entrega. En caso de fallo, el suscriptor puede recibir algún mensaje duplicado.
- QoS 2 assured (exactly one). Se garantiza que cada mensaje se entrega al suscriptor, y únicamente una vez.

Usar un nivel u otro depende de las características y necesidades de fiabilidad del sistema. Lógicamente, un nivel de QoS superior requiere un mayor intercambio de mensajes de verificación con el cliente y, por tanto, mayor carga al sistema.

Seguridad:

El protocolo MQTT dispone de distintas medidas de seguridad que podemos adoptar para proteger las comunicaciones. Esto incluye transporte SSL/TLS y autenticación por usuario y contraseña o mediante certificado. Sin embargo, hay que tener en cuenta que muchos de los dispositivos IoT disponen de escasa capacidad de proceso, por lo que el SSL/TLS puede suponer una carga de importante.

Otras características:

Hasta aquí se ha realizado una descripción sencilla del funcionamiento del protocolo, pero MQTT posee otras características que aseguran la calidad de la conexión, y su confiabilidad, como también la seguridad de los datos que se envían y la protección contra cualquier ataque desde el exterior. Para saber respecto al protocolo y sus estándares, se puede visitar la página oficial mqtt.org.

Infraestructura necesaria:

MQTT requiere de la pila TCP/IP, lo cual no representa un problema ya que hoy en día existen estas redes en la mayoría de hogares e industrias donde podría emplearse el dispositivo. Pero además, como se desean almacenar los datos y también manipularlos desde cualquier equipo disponible, se pensó en un servidor que además de servir de bróker para el servicio MQTT brinde servicios de base de datos y servicios web. Por lo tanto, un servidor LAMP instalado dentro de la misma red privada sería la solución perfecta. Luego, las consultas de los datos podrían realizarse desde cualquier plataforma que disponga de un navegador web, sea una computadora como un smartphone.

ESTUDIO DE TRABAJOS EXISTENTES

Desde un principio, se notó que las prestaciones y confiabilidad de los aparatos comerciales tienen una relación directa con sus precios. A más prestaciones, mayor precio. Y que en la mayoría de los casos, se debe adquirir un aparato completo por cada punto de la red o por cada máquina que se desea monitorizar. No son versátiles, están diseñados para determinado fin, y se limitan a eso.

Por otro lado, en la mayoría de los casos se requiere de una mano especializada para su manipulación y que el usuario experto se movilice hasta la ubicación del dispositivo para operarlo.

No están diseñados como herramientas de laboratorio, por lo que su empleo en ensayos puede resultar incómodo, desprolijo, costoso y tedioso. Requieren de una instalación estructurada y cableado. Los datos se deberán consultar uno por uno para cada dispositivo. Y si se quiere realizar cálculos a partir de los datos obtenidos deberán realizarse manualmente.

Ninguno de los dispositivos estudiados brinda alguna ventaja sobre el diseñado para ninguno de los aspectos mencionados.

OBJETIVOS

Se pretende diseñar un dispositivo capaz de medir los diferentes parámetros eléctricos que permitan el cálculo de la potencia disipada por una determinada carga para monitorizar el consumo y diagnosticar el suministro eléctrico. Que envíe los datos a un servidor que los almacene o los muestre en tiempo real a cualquier cliente autorizado para tal fin. Que permita incrementar los sensores sin necesidad de realizar cambios mayores al sistema y sin que los costos de producción y adquisición se disparen. Para abaratrar costos se pretende que la consulta de los datos pueda realizarse mediante software desde diferentes dispositivos, independiente de su organización y arquitectura, lo cual al mismo tiempo evita la necesidad de implementar una pantalla u otro tipo de interfaz al dispositivo. Que presente una interfaz amigable e intuitiva. Y permita el ajuste y actualización del software de forma remota y sencilla. Que se minimicen los costos de producción.

- 1.- Crear un dispositivo capaz de medir frecuencia, tensión, intensidad, potencia activa y factor de potencia.
- 2.- Proporcionar un sistema simple, pero robusto y confiable.
- 3.- Escalabilidad.
- 4.- Obtener un software simple, amigable y multiplataforma.
- 5.- Minimizar los precios de construcción.
- 6.- Construcción de un prototipo que materialice el diseño y permita realizar ensayos.

RECURSOS / HERRAMIENTAS

- 1.- ACE.
- 2.- Módulo WiFi ESP8266, Raspberry Pi2 o similar, router, Smartphone, computadora.
- 3.- Material bibliográfico de consulta referidos a varios lenguajes de programación, pilas TCP/IP, protocolo MQTT, HTTP, WebSockets, etc.
- 4.- Internet y diferentes recursos informáticos como SO, aplicaciones, librerías, etc.

DISEÑO Y CONSTRUCCIÓN DEL SISTEMA

El sistema integra diferentes dispositivos que lo componen. En este apartado se analizará cada dispositivo por separado para una descripción limpia y detallada.

Analizador de Consumo Eléctrico:

Es el dispositivo que se construyó con anterioridad con el fin de medir frecuencia, tensión, intensidad, potencia y factor de potencia. Para conocer más sobre este dispositivo se puede consultar el primer informe del proyecto en el cual se detalla su funcionamiento, diseño, exactitud en las mediciones y demás cuestiones que no tienen relación con ésta segunda parte por lo que no se profundizará más respecto a él.

Lo importante a destacar de éste dispositivo es que en primera instancia, los datos eran entregados mediante el puerto serie de una placa de desarrollo Arduino NANO. Como se pretendió dar continuidad inmediata al proyecto, se agregó un módulo ESP8266 para que reciba los datos que entrega la placa Arduino y a partir de allí los entregue a un broker mediante MQTT.

Módulo WiFi ESP8266:

ESP8266 es un chip de bajo costo WiFi con un stack TCP/IP completo y un microcontrolador. En este caso funciona como expansión para la placa de desarrollo Arduino del ACE brindándole conexión WiFi, aunque un dispositivo como el NodeMCU8266 podría por sí solo reemplazar la otra placa y además brindar conectividad. De todas formas, como se pretendía dar continuidad al proyecto sin realizar cambios en él, se lo empleó un módulo de expansión sólo para la conexión inalámbrica.

Como la Arduino NANO originalmente entregaba un tren de datos por puerto serie que correspondían a todas las mediciones realizadas, los cuales eran visualizados en el monitor serie de la IDE de Arduino, se debe trabajar estos datos para identificarlos, separarlos y publicarlos por topics para darle estructura respetando los estándares de MQTT.

Para ello se escribió un código en Arduino IDE que sencillamente conecta el módulo a la red WiFi que brinda un router, se conecta con el broker, recibe los datos provenientes del ACE, los separa y clasifica, y los publica en los topics correspondientes.

Para entender mejor el funcionamiento se puede consultar el código fuente, como también las librerías empleadas. En este caso ESP8266WiFi.h para la conexión con el router y PubSubClient.h para el servicio MQTT.

Router:

Se decidió trabajar con un router comercial para brindar la red WiFi durante el diseño del prototipo ya que se supone que hoy en día existe un router WiFi en cualquier hogar o instalación industrial y en un futuro, se podría aprovechar la red preexistente en el local donde se instale el sistema y se emplee el dispositivo de medición. No es un requisito imprescindible disponer de un router WiFi ya que la Raspberry que acompaña el sistema también brindará una zona WiFi, pero sí se requerirá de un modem, o algún tipo de dispositivo que permita la conexión a internet si se desea monitorear el sistema de forma remota. Éste será requerido o proporcionado por el prestador del servicio de internet.

En cuanto a la configuración del router, éste deberá brindar una red para todos los dispositivos intervenientes. Por conveniencia brindando servicio DHCP para facilitar la conexión de los ACEs y usuarios a la red pero reservando una dirección IP para el servidor LAMP del cual se hablará a continuación.

Servidor LAMP:

Éste es uno de los aspectos fundamentales que hace al sistema diferente a los demás y que da forma al desarrollo completo del proyecto.

Sin profundizar demasiado en la teoría, ya que se puede encontrar mucha información al respecto en bibliografía específica y en internet, como ser en las páginas oficiales de los componentes de software que componen el servidor Linux Apache MySQL Php. Se mencionarán los aspectos más relevantes para el presente proyecto.

En principio, la plataforma elegida para la implementación de éste servidor fue la SBC Raspberry Pi 2, pero al no disponer de ella en el momento de iniciar con el desarrollo del proyecto, la mejor alternativa fue trabajar sobre una máquina virtual en la que se instaló Debian Jessie, distribución de la cual nace Raspbian Jessie en su momento para Raspberry Pi 2 (aunque en la actualidad existen nuevas distribuciones de Raspbian, ahora Raspberry Pi OS basados en Debian Buster) asegurando que todo el desarrollo fuera fácilmente implementado posteriormente en la SBC elegida. Luego por cuestiones ajenas a la voluntad del autor, ya que se atravesó una pandemia que complicó la adquisición de la SBC, el servidor quedó implementado en la máquina virtual hasta finalizar el desarrollo y realizar las pruebas del sistema. De todas maneras, esto no altera los resultados obtenidos.

Se eligió Debian por sus grandes prestaciones para sistemas de recursos limitados. Es mucho menos costoso implementar un servidor LAMP en una SBC que adquirir un verdadero servidor innecesariamente más potente. Además la disponibilidad de incontables repositorios y software libre y/o gratuito, más herramientas como apt para gestionar la instalación y descarga de software y librerías desde los repositorios, facilitan en gran medida la implementación del servidor. El software requerido e implementado fue, Apache2, MySQL,

Python, entre otros. Basta una instalación mínima, sin entorno gráfico para configurar y poner a funcionar el servidor por completo.

A continuación se detallan los aspectos más importantes de la configuración y el funcionamiento de cada programa para este proyecto.

MySQL:

Una base de datos es una forma segura y robusta de almacenar y manipular los datos. Para el prototipo construido, se implementó una base de datos llamada `proyecto_final` que almacena las mediciones tomadas por el ACE y detalla la procedencia de la medición, el topic, la medición y la fecha y hora en la que la medición fue tomada, todo esto guardado en una única tabla llamada “datos” con formato InnoDB.

datos

Columna	Tipo	Nulo	Predeterminado
id	int(11)	No	
usuario	varchar(4)	No	
topic	varchar(12)	No	
payload	varchar(8)	No	
fecha	timestamp	No	0000-00-00 00:00:00

Luego las consultas se filtrarán por ACE, tópico, y fecha o según como sea conveniente mediante código SQL.

Apache2:

El servidor web posibilita la implementación de la interfaz con el usuario. Cabe destacar que en este aspecto fue necesaria mucha autodidáctica ya que la formación de la tecnicatura electrónica universitaria no se concentra en lenguajes de programación web, pero si entrega sólidos conocimientos de las bases de la programación que permiten a los estudiantes incursionar en nuevos lenguajes con éxito. También la formación académica de los técnicos enriquece sus conocimientos en cuanto a la teoría interviniente por lo que cuentan con un gran respaldo y un sinfín de herramientas que permiten ampliar los horizontes. Por eso, aunque tal vez se puedan pulir muchos aspectos técnicos del diseño, lo logrado hasta aquí satisface las expectativas del autor.

La interfaz con el usuario cuenta de tres páginas web, una de inicio (`index.php`) en la cual el usuario elige si desea consultar el historial de mediciones almacenadas en la base de datos o si desea monitorear en tiempo real las mediciones que está realizando algún ACE conectado. Cada opción lleva al usuario a cada una de las dos páginas restantes (`realtime.php`) y (`anteriores.php`).

Index.php:

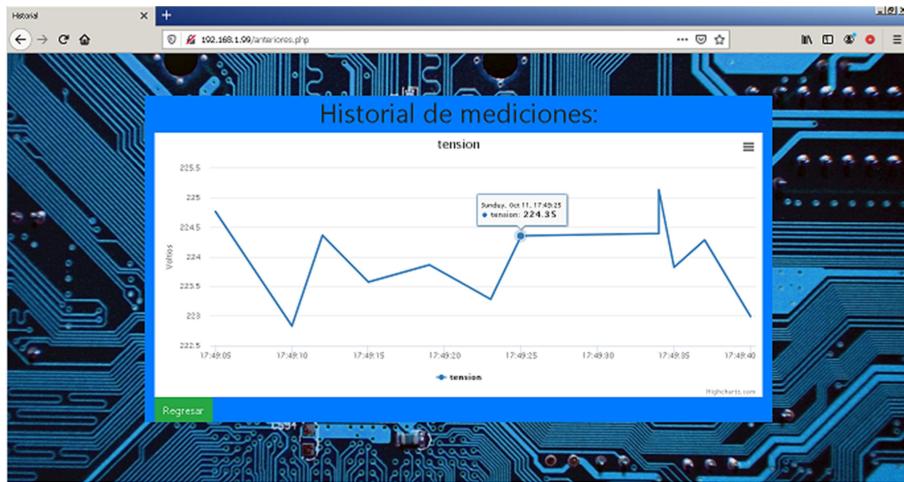
Esta página principal realiza consultas a la base de datos para listar en la tarjeta “Consultar historial de mediciones” los ACEs de los cuales se tiene información almacenada. Por otro lado, como el broker implementado no permite consultar por los usuarios

conectados, se detecta a los ACEs activos consultando en la base de datos si algún ACE envió información en los últimos segundos y los enlista en la tarjeta “Monitorear en tiempo real”. Esta función se ejecuta al cargar la página o al hacer click en el botón “refrescar”. La interacción entre en cliente y el servidor se implementó mediante funciones escritas en JavaScript, métodos para los cuales se requirió de la librería jQuery.



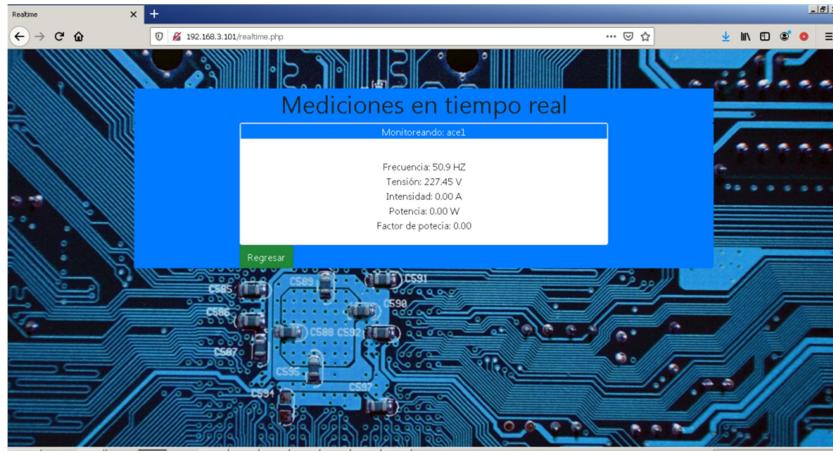
anteriores.php:

Solo para ejemplificar cómo se pueden manipular los datos almacenados, en ésta página se le presenta al usuario un gráfico con las mediciones tomadas en el tiempo. Los ACEs entregan una medición cada dos segundos aproximadamente, y si se realizara una consulta para un día entero de mediciones, no sería muy útil una tabla tan extensa, por eso, un gráfico parecía algo más adecuado y cercano a un escenario real. Las consultas se realizan mediante php y el gráfico mediante JavaScript, con el uso de librerías de Highcharts que brinda éste tipo de soluciones.



realtime.php:

En ella se implementa principalmente código JavaScript y la librería paho-mqtt que brinda Eclipse para la suscripción al servicio MQTT que permite la monitorización de los datos enviados por el ACE previamente seleccionado.



Para lograr un entorno amigable y sencillo se hizo uso de las librerías de diseño BootStrap que además hace que las páginas sean “responsive” por lo cual se puede consultar los datos desde plataformas con diferentes tamaños de pantalla sin que eso sea una mala experiencia. Se han realizado pruebas desde smartphones, netbooks y notebooks corroborando que la experiencia es muy buena independientemente del dispositivo que se emplee.

Conector MySQL-Python:

Dado que se trata de un lenguaje familiar y sencillo, fue escogido para realizar la conexión entre el broker y la base de datos. Porque además cuenta con la herramienta “pip” que permite gestionar e instalar paquetes, por lo que la conexión entre el broker y la base de datos, así como la instalación de las librerías necesarias para implementar código SQL resultan ser tareas muy sencillas.

Basta con ejecutar los siguientes comandos para instalar los paquetes necesarios:

- i. apt mysql-client
- ii. apt python
- iii. apt python-pip
- iv. apt python-dev
- v. pip install MySQL-python
- vi. pip install paho-mqtt

Luego el código escrito en python para la conexión entre el cliente MQTT y la base de datos hará uso de las librerías MySQL-python y paho-mqtt el cual se hará cargo de suscribirse al topic correspondiente, recibir los mensajes y enviarlos a la base de datos para su almacenamiento.

Además, el programa se declaró como servicio y se lo habilitó con systemctl para que se ejecute automáticamente en el arranque.

Broker:

Mosquitto es uno de los brokers más elegidos entre los proyectos con Raspberry por tratarse de un software liviano, ideal para ser montado en equipos de baja potencia como las

SBC. Se trata de un proyecto Open Source que forma parte de los proyectos IoT de Eclipse. Soporta las versiones de MQTT 3.1, 3.1.1 y 5.0. Y cuenta con un gran apoyo de la comunidad.

Este es un punto central. Todo el proyecto se sustenta sobre este protocolo de comunicación, y el broker es el pilar más importante. Por eso, resulta conveniente comentar las dificultades que se atravesaron a la hora de implementarlo.

Como se pretende que los datos sean monitorizados desde una página web, se requirió montar MQTT sobre el protocolo WebSockets, por lo tanto, el broker debía brindar soporte a éste protocolo de comunicación. Mosquitto, a partir de su versión 1.4 da soporte a WebSockets pero para éstas versiones antiguas existe cierta complicación en su instalación. Las versiones de Mosquitto disponibles en los repositorios de Debian no tienen implementado WebSockets.

Dos caminos para instalar Mosquitto con soporte para WebSockets:

1) Instalando una versión precompilada desde los repositorios de Mosquitto:

Ésta es la forma sencilla de hacerlo. Empezamos por agregar a la lista de fuentes los repositorios de mosquitto importando e instalando la clave de acceso al paquete. Esto puede realizarse usando las herramientas wget y apt como se muestra a continuación:

- i. wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
- ii. apt-key add mosquitto-repo.gpg.key

Luego nos situamos en la carpeta donde se encuentran las listas de repositorios de apt:

- iii. /etc/apt/sources.list.d/

Aquí se descarga la lista con wget:

- iv. wget http://repo.mosquitto.org/debian/mosquitto-jessie.list

*Donde jessie puede reemplazarse por wheezy o stretch dependiendo de la versión de debían con la que se esté trabajando.

A continuación se actualiza la información de apt y se instala mosquitto:

- v. apt update
- vi. apt install mosquitto

Esta versión precompilada da soporte a WebSockets y fue resultado de las innumerables complicaciones que existían a la hora de realizar la instalación desde la fuente a causa de las dependencias de mosquitto y errores en el código. Muchos de esos errores fueron reparados tras las actualizaciones, pero al mismo tiempo, cuando se actualizaron otras librerías de las cuales depende Mosquitto volvieron a aparecer diferencias en la sintaxis.

Por una cuestión de simple preferencia, y como ya se había trabajado con esta versión, se optó por compilar desde la fuente la versión 1.4 de Mosquitto.

2) Compilando Mosquitto desde la fuente:

Como se mencionó anteriormente, la dificultad para la instalación de mosquitto radica en sus dependencias. Por eso, para éste caso en el que se compiló la versión 1.4 fue necesario eliminar la librería libwebsockets3 y reinstalar una versión anterior (si no se realiza esta acción, se obtendrán errores de compilación) el motivo es que en libwebsockets3 se renombró un objeto anteriormente llamado libwebsockets por libwebsocket, éste pequeño cambio de una letra resulta en que el compilador nos indique a la hora de compilar que libwebsockets no está definido.

Entonces, empezamos por resolver las dependencias de mosquitto con los comandos:

- i. apt install build-essential
- ii. apt install libc-ares-dev
- iii. apt install uuid-dev
- iv. apt install libssl-dev
- v. apt install openssl

Luego, convenientemente en una carpeta temporal que se haya creado, se descarga libwebsockets1.3. Es posible obtenerlo desde diferentes fuentes, lo importante es que sea la versión indicada. Una vez dentro de la carpeta de libwebsockets1.3 se compila e instala con cmake y make:

- vi. cmake
- vii. make
- viii. make install

Realizado lo anterior, es momento de obtener mosquitto1.4 con la herramienta wget:

ix. wget http://mosquitto.org/files/source/mosquitto-1.4.tar.gz

Se descomprime el archivo, y aquí un paso importante. Dentro de la carpeta de mosquitto se encuentra el archivo “config.mk”. En él, debemos editar la línea “WITH_WEBSOCKETS = no” cambiando el valor “no” por “yes”.

A continuación se compila e instala con los comandos:

- x. make
- xi. make install

Es conveniente crear un usuario para la ejecución de mosquitto:

xii. useradd -r -m -d /var/lib/mosquitto -s /usr/sbin/nologin -g nogroup mosquitto

Y luego editar el archivo de configuración para habilitar WebSockets. El archivo de configuración “mosquitto.conf” se encuentra en la carpeta “/etc/mosquitto”. Lo editamos con un editor de texto modificando las líneas 277 y 297 donde le indicaremos:

277.- Listener 9001

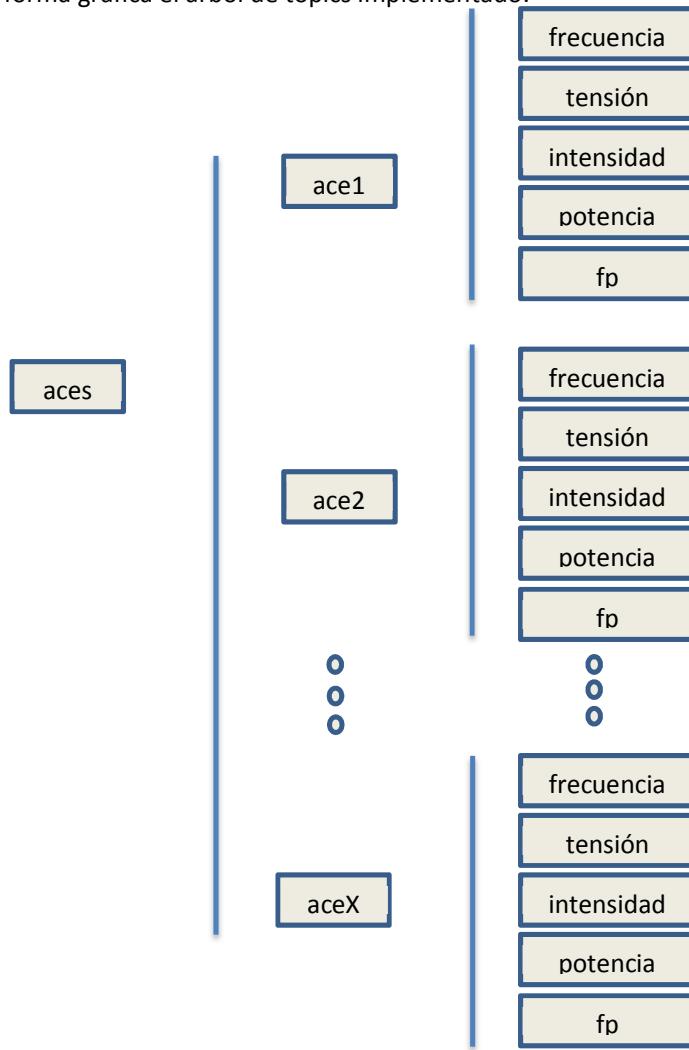
297.- protocol websockets

Finalmente, para que el broker se ejecute en el arranque como un servicio se debe copiar el archivo “mosquitto.service” que se encuentra en la carpeta desde la que se hizo la instalación en el directorio “/etc/systemd/system/” y se ejecutan los siguientes comandos:

- xiii. systemctl daemon-reload
- xiv. systemctl enable mosquitto.service

El detalle tan minucioso del procedimiento de instalación del broker parece solo satisfacer un capricho del autor. Pero en realidad fue incorporado al informe porque representó uno de los desafíos más grandes en el proceso. Requirió de mucha lectura, tanto en los blogs oficiales de Mosquitto como foros y blogs de terceros. También un seguimiento de las modificaciones documentadas en el proyecto Mosquitto en “git”. Conocer estos detalles evitará futuros inconvenientes a quienes deseen implementar un sistema similar al que se presenta.

De la introducción teórica al protocolo MQTT se aprecia que una de las características más importantes del protocolo es su estructura de topics. Por eso, a continuación se describe de forma gráfica el árbol de topics implementado:



Como se puede ver, los últimos subtopics corresponden a los parámetros que mide el ACE. Un nivel aguas arriba se encuentran los ACEs conectados, es decir, este topic permite identificar qué ACE es el que envía las mediciones. Y el siguiente nivel es la raíz donde se identifica la “familia” de sensores, lo que permite que el sistema se aadecue a una eventual ampliación del mismo con la aplicación de otros sensores, aunque esa no sea la intención del proyecto.

Es importante mencionar que las conexiones se realizan mediante autentificación con nombre de usuario y contraseña. Para ello se creó un usuario para los aces, otro para las páginas web, otro para el conector mqtt-mysql y un último usuario para mosquitto_sub. Brindando una primera barrera de seguridad y permitiendo desde el lado del broker diagnosticar posibles problemas de conexión.

PhpMyAdmin:

Si bien este software no forma parte vital del sistema, resultó ser una herramienta muy útil para la administración de la base de datos recordando que se había realizado una instalación mínima de debian por lo que no se contaba con interfaz gráfica. Aunque se puede gestionar sin problemas la base de datos desde las líneas de comandos, resultaba un esfuerzo innecesario. También PhpMyAdmin brinda la posibilidad de gestionar la base de datos de forma remota, por lo que es una buena opción mantenerlo instalado en el servidor.

DHCP, DNS y Enmascaramiento:

Raspberry incluye una tarjeta para la conexión inalámbrica desde su versión Pi3. Por lo tanto se la puede emplear además como punto de acceso. En los repositorios de Debian existen programas livianos que permiten aplicar los servicios necesarios. Teniendo en cuenta que el prototipo se monta en una máquina virtual, se implementaron estos servicios a la red interna de VirtualBox, mediante una placa de red conectada a dicha red interna. Por lo tanto se brinda servicio a cualquier máquina virtual extra que se conecte a la dicha red de manera que se puede probar el correcto funcionamiento de los servicios.

Dnsmasq: diseñado para proporcionar servicio DNS y opcionalmente DHCP, es una herramienta muy sencilla y fácil de configurar. Una vez instalado, en su archivo de configuración (dnsmasq.conf) se editan las líneas necesarias para indicar la tarjeta de red en la cual escuchará las peticiones DHCP y DNS, como también el rango de direcciones que se brindarán.

- Específicamente en este proyecto, la misma tarjeta tiene implementado el servidor LAMP y el broker por lo tanto si un usuario se conecta a la red mediante este punto de acceso y luego desea consultar los datos de algún ACE solicitará acceso a la página web albergada en la misma máquina, por lo tanto se puede agilizar dicho acceso permitiendo que Dnsmasq resuelva la traducción de nombre. Para eso se edita el archivo “/etc/hosts” agregando la dirección de IP de la propia máquina para el dominio que se haya registrado para su consulta remota.

Shorewall: es una herramienta de firewall de código abierto que se basa en el sistema Netfilter integrado en el kernel de Linux, proporciona un mayor nivel de abstracción para describir las reglas usando archivos de texto. La instalación de esta herramienta incluye unas carpetas de ejemplos que proporcionan todos los archivos de texto pre-escritos para la configuración del cortafuego. En particular, la carpeta “two-interfaces” ubicada en “usr/share/doc/shorewall/examples/” proporciona todos los archivos necesarios para este caso. Entonces se copia el contenido de la carpeta en “/etc/shorewall/”. Se requiere la edición de algunos archivos para el correcto funcionamiento del firewall, como la adición del servicio

DHCP para la interface de red conectada a la red interna en el archivo “interfaces”, y definir el sentido del enmascaramiento en el archivo “masq” (para versiones posteriores a Jessie el archivo en cuestión será “snat”). Pero es crucial para el proyecto prestar especial atención al archivo “rules”.

El archivo “rules” define las excepciones a las políticas por defecto del firewall. Y como en el propio firewall funciona el servidor LAMP y el broker, es fundamental permitir el acceso a las peticiones de los servicios DNS, web, MQTT, y WebSockets. Que por defecto son: 53, 80, 1883, 9001 respectivamente. Sin esta edición, el sistema sería inútil ya que ningún usuario podrá acceder a los servicios que dan forma al proyecto.

Existe suficiente documentación en la página oficial de shorewall para más información, y de todas maneras se proporcionarán los archivos de interés para ser consultados y evacuar cualquier duda al respecto.

Una vez instalados y configurados Dnsmasq y Shorewall, cualquier dispositivo conectado al punto de acceso será capaz de navegar en internet y de consultar las mediciones de los ACEs. Pero en un escenario real, en el que se emplee una Raspberry, se requiere de un paso extra para la verdadera implementación del punto de acceso. Esto es, crear el servidor de autenticación que hará visible la red WiFi para los usuarios que deseen conectarse. Dicho servicio puede brindarse a partir de herramientas como Hostapd, tal vez una de las más populares entre los proyectos de Raspberry. Su instalación y configuración es sencilla y existe suficiente documentación y ejemplos en diversos sitios, foros y blogs por lo que completar este paso no debería resultar difícil.

ENSAYOS

Se repitió todo el proceso de instalación en una nueva máquina virtual, con un router diferente para evaluar con cuánta facilidad se instala el sistema ya conociendo el procedimiento y contando con todos los archivos necesarios, simulando un escenario real en el que un cliente requiere el producto. También se conectó una máquina virtual con entorno gráfico a la red interna simulando un usuario conectado al punto de acceso implementado.

Se模拟aron desperfectos que se asumen posibles en un escenario real:

- interrupción del servicio de red por reinicio del router.
- Interrupción del suministro eléctrico para el ACE.
- Interrupción del suministro eléctrico para el servidor.
- Interrupción del suministro eléctrico para el ACE y el servidor.

Los clientes fueron programados para reconectarse en caso de una eventual desconexión. Y el servidor fue programado para iniciar automáticamente los servicios ante un reinicio.

Con todo el sistema funcionando y el ACE midiendo una carga conectada a la red se realizaron consultas de los diferentes parámetros desde computadoras y celulares inteligentes.

CONCLUSIÓN

Todo proyecto tecnológico pretende dar solución a una necesidad. Dicha necesidad es el disparador que activa la creatividad del diseñador del proyecto. Si no se tiene una hipótesis clara y firme, es difícil encaminar el desarrollo. Los técnicos somos formados para hacer uso de las técnicas y la tecnología disponible para crear productos y servicios que solventen los requerimientos de esa necesidad. Como éste es un proyecto universitario y no existió una verdadera demanda para el producto se plantearon dos hipótesis como punto de partida:

- Construir una herramienta de laboratorio que permita monitorear el consumo eléctrico de una carga en tiempo real y el estado del suministro eléctrico como también comprobar la corrección del factor de potencia de la carga.
- Construir un dispositivo que permita monitorear el consumo eléctrico de cada departamento de un complejo de departamentos para realizar el cobro mensual de acuerdo al consumo.

Ambas hipótesis convergen en la construcción de un mismo dispositivo y requieren la medición de los mismos parámetros. De allí nació el “ACE”.

Ya sea como herramienta de laboratorio o como medidor de consumo, el siguiente desafío fue la transmisión de los datos de forma inalámbrica y el almacenamiento de los mismos para su procesamiento.

Finalmente, se presentó un prototipo versátil, capaz de adecuarse para cada fin con mínimos cambios. Y una interfaz a modo de ejemplo que muestra cómo pueden manipularse esos datos. Dejando abierto el proyecto para ser ajustado a medida, según los requisitos del futuro usuario.

Por eso, este sistema, no es un proyecto de domótica. No se espera que interactúe con otros tipos de sensores, ni que actúe de otra manera aunque podría adaptarse. En cuanto a la escalabilidad del proyecto, éste crece a medida que se agreguen ACEs que monitoreen más de una máquina o sector de la red eléctrica al mismo tiempo. Condición que se cumple sin modificación alguna.

Respecto de la confiabilidad del sistema, de los ensayos se concluye que el sistema es capaz de recuperarse a las fallas ensayadas sin problemas. Por otro lado, la sencillez de la interfaz con el usuario oculta completamente todos los procesos y mecanismos más complejos y sensibles. Lo que impide que una mano inexperta realice alguna modificación catastrófica.

La única limitación importante hasta el momento es la seguridad del sistema y del resto de la red ante ataques externos. Humildemente y como se mencionó en un principio, los escasos conocimientos referidos a la seguridad informática llevaron a tomar la decisión de no darle salida a la red hacia la nube por lo pronto, para no exponer el resto de los dispositivos a los riesgos que eso implica.

“Si lo sé hacer, lo hago. Si no lo sé hacer, lo aprendo y después lo hago” lema de mi autoría que rige la forma en la que realizo cualquier trabajo y pone límite a mis incursiones en temas desconocidos. Entonces, antes de vulnerar la red privada sobre la que se trabajó, se

optó por dar como finalizado el trabajo en el punto hasta donde los conocimientos adquiridos durante la carrera y la posterior investigación de temas relacionados lo permitieron. Sigue aprender sobre seguridad informática y luego aplicarlo. O en caso de ser necesario, poner esa tarea en manos de una persona idónea.

También de los ensayos se concluyó que la interfaz es lo suficientemente intuitiva como para que una persona ajena a la tecnología empleada y a los conceptos que envuelven al proyecto pueda realizar las consultas sin problemas. Por lo tanto será un buen modelo a seguir para las modificaciones necesarias que ajusten la interfaz a los requerimientos de un futuro cliente.

El proyecto es viable. El sistema es económico. Los dispositivos son accesibles y fáciles de obtener ya que están disponibles en el mercado nacional.

AGRADECIMIENTOS:

Ocuparé estas últimas líneas en un intento de compartir mis sensaciones luego de haber abordado este trabajo:

Recuerdo que una vez, presentando un proyecto con compañeros para la materia Taller, uno de ellos comenzó su exposición diciendo “*Como todo técnico, a prueba y error, comencé a medir hasta adivinar que cables corresponde a cada bobina...*”. No me pareció que el momento fuera oportuno para objetar y además mi tiempo de exposición ya había concluido.

La formación académica recibida en esta institución, forma a los estudiantes y futuros profesionales con un perfil sólido, de buen carácter y criterio. El azar y la magia no son herramientas que nos conduzcan necesariamente al éxito al abordar un desafío. Ningún examen se aprueba marcando la respuesta correcta de pura suerte. Adquirimos las herramientas necesarias para encarar un proyecto o una investigación. El camino seguro y certero es la pesquisa y la pericia.

Este proyecto me ha demandado mucha autodidáctica e investigación. Y pude demostrarme a mí mismo que he sabido aprovechar todo el conocimiento que mis profesores compartieron y volcaron sobre mí. Por eso, les estaré eternamente agradecido, recordaré a la institución que me formó como la más generosa que he conocido y podré afirmar abiertamente que “mi sabiduría viene de esta tierra”.

Agradezco también a mi familia, a mi novia y a mis compañeros, personas con las que compartí fundamentalmente ésta última etapa y que me han empujado para llegar hasta aquí.