Desarrollo de un Prototipo para Monitorear el Sistema de Bombeo Electro Sumergible de un Pozo Productor de Petróleo

Muñoz R. Andrés*, Flores C. Fernando*
*Departamento de Electrónica, Telecomunicaciones y Redes de Información (DETRI) de la Escuela Politécnica Nacional (EPN)

***Resumen*— En este artículo se describe el desarrollo de un prototipo, que en el futuro permita monitorear de forma autónoma el sistema de bombeo electro sumergible de un pozo productor de petróleo (sistema BES), tomando como referencia una arquitectura de Internet de las cosas (IoT). El resultado de este trabajo es un prototipo que consta de un componente de adquisición de datos desarrollado sobre un computador de placa reducida (en inglés, single board computer o SBC), el cual envía la información extraída del sistema BES a una aplicación web de almacenamiento, análisis y monitoreo de datos desarrollada como parte del prototipo propuesto. El problema que se plantea resolver, es la demora en la detección de incidentes en los pozos productores de petróleo, debido al retardo introducido en el proceso de lectura manual de los parámetros de un sistema BES. En primer lugar se describe el funcionamiento del equipo monitoreado, sus capacidades de comunicación y los elementos teóricos utilizados para el desarrollo del prototipo. Posteriormente se realiza el levantamiento de requerimientos y se genera el diseño del software necesario. Luego de ello, se describe el proceso de desarrollo del software para el prototipo. También se expone la configuración de un ambiente de pruebas y el proceso de despliegue de los elementos del prototipo. Finalmente, se configura un simulador de sistema BES y se ejecutan las pruebas del prototipo.**

***Palabras clave*—IoT, monitoreo, sistema BES, Modbus, Ruby on Rails, Raspberry Pi, Python**

I. Introducción

Un pozo productor de petróleo es una perforación profunda hecha para localizar o extraer este hidrocarburo, el cual se encuentra almacenado en reservorios. Más del 90% de los pozos productores de petróleo requieren alguna modalidad de levantamiento artificial para elevar el fluido desde el fondo del pozo hasta la superficie, cuando un reservorio no tiene la suficiente energía para producir petróleo de manera natural.

Uno de los tipos de sistemas de levantamiento artificial más versátiles y adaptables son los sistemas de bombeo electro sumergible (sistemas BES).

Un sistema BES es un sistema de levantamiento artificial instalado al fondo de un pozo productor de petróleo, que consiste de una bomba centrífuga de varias etapas dispuesta dentro de una carcasa y acoplada a un motor eléctrico sumergible [1]. Cuando el motor eléctrico sumergible gira, acciona la bomba centrífuga y esta a su vez empuja el petróleo hacia la superficie. El motor está conectado a sistemas eléctricos y de control en la superficie, a través de cables con armadura protectora.

La configuración típica de un sistema BES se muestra en la Fig. 1. El equipo instalado al fondo del pozo productor de petróleo consta de un motor, las protecciones, las etapas de la bomba, las entradas de la bomba, los cables de energía, el equipo de manejo de gas y el sensor de fondo. Este sensor es el dispositivo encargado de medir magnitudes de fondo tales como presión y temperatura, para posteriormente enviarlas a la superficie a través de cables de comunicación, donde estos datos son almacenados automáticamente en una unidad Modbus. A través de esta unidad, los datos monitoreados pueden ser leídos por medio de una interfaz física serial RS-485, o manualmente mediante un teclado y una pantalla digital incorporados en el equipo de superficie. Se pretende que el prototipo presentado se pueda conectar a futuro a la unidad de superficie Modbus del sistema BES, para realizar el proceso de adquisición de datos de forma autónoma.

El monitoreo permanente de pozos ayuda a mejorar el manejo de los yacimientos, lo que evita altos costos de una intervención del pozo. También ayuda a maximizar la producción y a optimizar la recuperación de petróleo. Por este motivo, la supervisión manual de los sistemas BES, suele ser realizada por operadores que recorren las diferentes facilidades con cierta frecuencia. Es posible también el uso de sistemas SCADA (Supervisión, Control y Adquisición de Datos), sin embargo estos sistemas no son comparables con sistemas IoT, por motivos que se explicarán más adelante durante la definición de la arquitectura IoT empleada y en la sección de conclusiones.

El problema a resolver a través del prototipo propuesto en este artículo, es la complejidad y el retardo introducidos debido a la lectura manual de los parámetros del sistema BES por parte de un operador humano. Estos factores afectan la capacidad de respuesta a incidentes y demoran el proceso de toma de decisiones ante posibles eventos de falla del sistema BES o del pozo productor de petróleo. Además de ello, se plantea eliminar las limitaciones del monitoreo de los datos a través de sistemas que no cumplan con una arquitectura IoT, como la imposibilidad de analizar los datos en tiempo real y

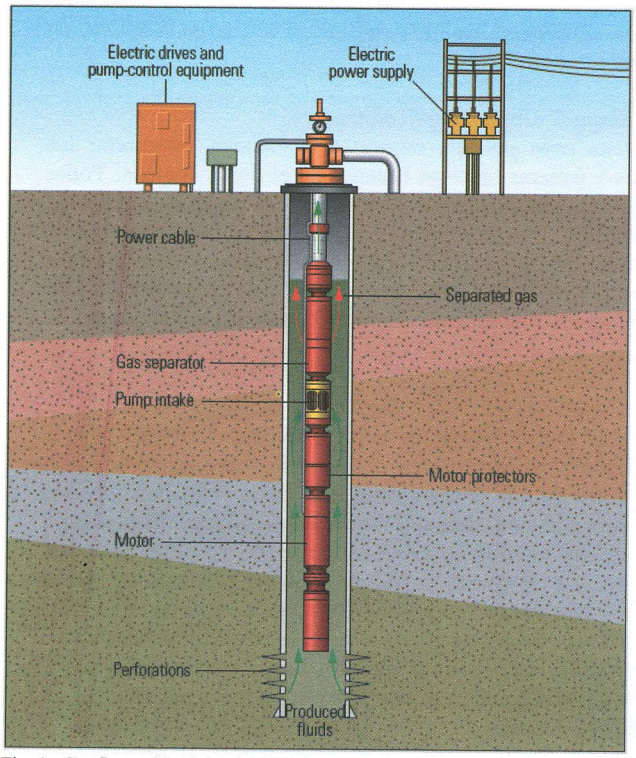

Fig. 1. Configuración típica de un sistema BES [1, Fig. 1]

de forma colaborativa. Para la solución de este problema, se aplica un enfoque de convergencia de Tecnologías de la Información (IT) con Tecnologías de Operaciones (OT).

II. Marco teórico

A. *Modbus [2]*

Modbus es un protocolo de comunicaciones de capa aplicación, desarrollado por la empresa Modicon en 1979 para ser utilizado con sus controladores lógicos programables (PLC). Modbus es un método utilizado para intercambiar información entre diferentes dispositivos electrónicos industriales y sobre diferentes tipos buses seriales o redes. El maestro o cliente Modbus es el dispositivo que solicita la información. El esclavo o servidor Modbus, es el dispositivo que almacena y entrega la información solicitada. Modbus define una unidad de datos de protocolo o PDU independiente de las capas de comunicación inferiores. Los datos necesarios para el funcionamiento de Modbus sobre buses seriales o redes, pueden ser incluidos en campos adicionales en la unidad de datos de aplicación (ADU) como se aprecia en la Fig. 2.

La información es almacenada en el esclavo en cuatro diferentes tablas primarias que representan bloques de memoria. Dos de estas tablas almacenan datos booleanos (bobinas o contactos) y las dos restantes almacenan valores numéricos de tipo entero sin signo de 16 bits (registros), como se puede ver en la Tabla I.

B. *Computadores de placa reducida*

Un SBC es una computadora completa en una sola tarjeta de tamaño reducido. Una *Raspberry Pi* es una SBC que puede ser utilizada en proyectos de electrónica o para desempeñar

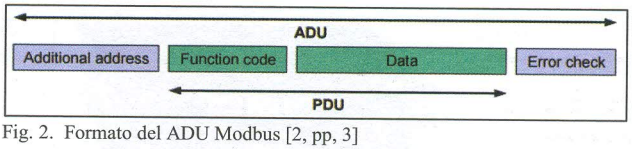

Fig. 2. Formato del ADU Modbus [2, pp, 3]

las labores de un computador de escritorio, como procesamiento de documentos, navegación en Internet, etc.

El uso de la Rasperry Pi, como componente de adquisición de datos para la implementación de este prototipo, es idóneo debido a su bajo costo, modularidad, capacidad de procesamiento, capacidades de conectividad importantes (pines GPIO o General Purpose Input Output), soporte para lenguajes de programación de alto nivel (lo cual permite la programación de tareas automáticas, como la adquisición periódica de datos), reducido tamaño y bajo consumo de energía. Estas capacidades permiten dotar de procesamiento y conectividad a objetos y dispositivos que originalmente no la tienen, como un sistema BES, habilitándolo para su conexión e interacción con aplicaciones de Internet.

Adicionalmente este dispositivo tiene la capacidad de realizar el formateo y el pre procesamiento, filtrado, validación de los datos adquiridos, por lo que puede ser utilizado como un dispositivo IoT de borde, como se explicará más adelante.

C. *Arquitectura de Internet de las Cosas (IoT) [3]*

En 2014 el IoTWF (Foro Mundial de IoT) liderado por Cisco, IBM, Rockwell Automation y otros, publicó una arquitectura referencial de IoT de siete capas. Esta arquitectura se puede observar en la Fig. 3. En general, los datos fluyen hacia la parte superior de la pila de capas, originándose en el borde (edge) hasta llegar al centro (center). Este enfoque permite lograr lo siguiente:

- Descomponer un problema IoT en partes más pequeñas
- Identificar las diferentes tecnologías utilizadas en cada capa y cómo están relacionadas con las otras capas
- Definir un sistema en el cual las diferentes partes puedan ser provistas por distintos fabricantes
- Contar con un proceso de definición de interfaces, que conduzca hacia la interoperabilidad
- Definir un modelo de seguridad escalonado que se aplique en las interfaces de cada capa

1) *Capa 1: Capa de dispositivos físicos y controladores*

Esta capa está conformada por las "cosas" u objetos en

TABLA I
TABLAS PRIMARIAS DEL MODELO DE DATOS MODBUS

Tablas primarias	Números de bobinas, contactos o registros	Direcciones de los datos	Tipo de datos	Nivel de acceso de maestro
Bobinas de salida discretas	1-9999	0x0000[1] a 0x270E	Booleano	Lectura - Escritura
Contactos de entrada discretos	10001-19999	0x0000 a 0x270E	Booleano	Solo lectura
Registros de entrada analógicos	30001-39999	0x0000 a 0x270E	Palabra de 16 bits	Solo lectura
Registros de retención de salida analógicos	40001-49999	0x0000 a 0x270E	Palabra de 16 bits	Lectura - Escritura

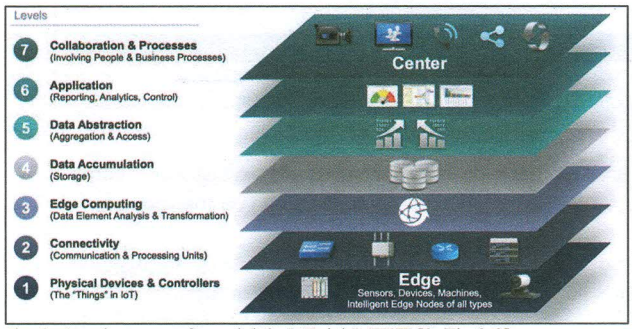

Fig. 3. Arquitectura referencial de IoT del IoTWF [3, Fig 2-2]

IoT, incluyendo varios endpoints y sensores, cuya función es generar datos y ser capaces de ser consultados y/o controlados sobre una red. Estos objetos pueden ser desde sensores microscópicos hasta maquinaria industrial (como un *sistema BES*).

2) Capa 2: Capa de conectividad

Esta capa es la responsable de la transmisión confiable y oportuna de los datos desde la capa 1 (objetos) hacia la red, y desde la red hacia la capa 3 (computación de borde). Esta capa abarca todos los elementos de conectividad de la red IoT, sin distinción entre la última milla, puertas de enlace y redes de retorno. La *conexión serial RS-485* es la capa de conectividad del prototipo propuesto.

3) Capa 3: Capa de computación de borde (edge)

Esta capa se encarga de la reducción de los datos y la conversión de flujos de datos en información lista para su procesamiento y almacenamiento por parte de las capas superiores. El procesamiento debe iniciarse tan cerca del borde y tan temprano como sea posible. Las funciones de la capa 3 se resumen como sigue:

- Evaluación y reformateo de los datos para el procesamiento en capas superiores
- Filtración de los datos para para el procesamiento en capas superiores
- Evaluación de los datos para enviar alertas, notificaciones u otras acciones

El *componente de adquisición de datos* realiza las funciones de dispositivo de borde en el prototipo propuesto.

4) Capa 4: Capa de acumulación

Esta capa permite el almacenamiento y la usabilidad de la información por parte de una aplicación, cuando sea necesario. En el caso del prototipo de monitoreo, esta capa es implementada por una base de datos relacional subyacente a la aplicación de web de monitoreo.

5) Capa 5: Capa de abstracción

Esta capa realiza la abstracción de la interfaz de datos para las aplicaciones de capas superiores. Esta abstracción es necesaria porque los datos pueden provenir de distintas fuentes de datos o por su volumen, estos datos pueden estar distribuidos en distintos dispositivos físicos. El *modelo* de la arquitectura MVC es el encargado de abstraer e independizar la aplicación del tipo de almacenamiento utilizado. En el caso específico de esta implementación, el modelo de Rails permite utilizar de forma subyacente varios tipos de bases de datos, sin la necesidad de modificar el código de la aplicación de monitoreo.

6) Capa 6: Capa de aplicación

Esta capa permite monitorear, analizar, interpretar los datos y generar reportes basados en el dicho análisis. Todas estas funcionalidades se llevan a cabo en la aplicación de monitoreo. Más adelante, en la sección de levantamiento de requerimientos se podrá verificar que el sistema debe permitir el monitoreo de los datos, la generación de reportes y alarmas dependiendo de los valores recibidos desde la aplicación de adquisición de datos.

7) Capa 7: Capa de colaboración y procesos

Esta capa involucra a las personas (usuarios de la aplicación) y los procesos del negocio. Un sistema IoT y la información que genera, tiene muy poco valor hasta que es puesta en acción durante la toma de decisiones. El prototipo planteado, habilita un esquema colaborativo al permitir que el proceso de operación del sistema BES pueda llevarse a cabo por varias personas. Por ejemplo, uno o varios ingenieros de producción, a podrían realizar el monitoreo remoto del sistema BES, sin necesidad de estar presentes en sitio. Si se llegara a registrar un evento (como por ejemplo un aumento excesivo de la temperatura de operación) el ingeniero de producción puede comunicar al operador en campo, para que este se traslade desde su ubicación actual hasta la ubicación del sistema BES monitoreado y tome una acción al respecto (apagado del sistema, revisión de parámetros de operación, etc.). Por otro lado, haciendo uso de alarmas configuradas para diferentes destinatarios, varias personas pueden estar al tanto de un evento de inmediato, aumentando la velocidad de respuesta ante incidentes. Por último, un ingeniero de reservorios puede obtener un reporte histórico de la presión del pozo, y observar las curvas de declinación de la presión y determinar una proyección de producción del pozo, programar trabajos de reacondicionamiento o determinar la necesidad del uso de otros mecanismos de recuperación de crudo, como la inyección de agua. Todo esto es posible debido al esquema de trabajo colaborativo de los sistemas IoT.

D. Convergencia de IT con OT [3]

El área de IT generalmente es la responsable de los sistemas de información de una empresa, como correo electrónico, archivos, bases de datos, etc., en tanto que OT es responsable de los equipos y procesos industriales, como máquinas, medidores, actuadores, dispositivos de automatización de distribución eléctrica, sistemas SCADA (control de supervisión y adquisición de datos), y otros. Tradicionalmente, OT ha utilizado redes dedicadas con protocolos de comunicaciones especializados para conectar estos dispositivos, y estas redes se han ejecutado completamente por separado de las redes de IT.

Algunas diferencias fundamentales entre las redes OT e IT se basan en la disponibilidad, por ejemplo si la red que conecta las máquinas en una fábrica falla, la producción puede paralizarse, lo cual puede traer repercusiones económicas enormes para las empresas. Por otro lado, si un servidor de correo electrónico administrado por el área de IT falla durante unas horas, la afectación puede ser grande pero no a un nivel comparable.

Con el surgimiento de IoT y los protocolos de IT basados en estándares como IPV6, las redes IT y OT convergen, es decir, OT está comenzando a adoptar los protocolos de red, transporte y métodos de la organización de IT, en tanto que IT, se está empezando a alinear con los requisitos operacionales de OT. La convergencia reduce la cantidad de infraestructura necesaria, facilita la operación de las redes, y el uso de estándares permite una escalabilidad y una capacidad de adaptación a las nuevas tecnologías, más rápidos.

El prototipo objeto de este artículo es un claro ejemplo de la convergencia de IT y OT. La adquisición de datos se la realiza utilizando un estándar de comunicaciones predominante en entornos industriales como es Modbus, pero para el envío de los datos adquiridos en el borde hacia capas superiores de la arquitectura IoT, se utilizan protocolos de la pila TCP/IP, esto habilita la posibilidad de que a futuro los datos puedan ser enviados a otros dispositivos en la red y empleen en diferentes tipos de aplicaciones. Adicionalmente, en un campo petrolero, donde existan varios sistemas BES, se puede utilizar una misma red TCP/IP para el monitoreo de varios equipos.

Normalmente las redes OT no cuentan con un esquema de seguridad comparable a las redes IT, ya que el principal mecanismo de seguridad en ellas, es el aislamiento de los equipos industriales, es decir, se requiere acceso físico para controlar o monitorear los equipos. El prototipo planteado, implementa el monitoreo unidireccional de los datos registrados por el sensor de fondo del sistema BES, sin embargo, en caso de que a futuro se plantee el diseño de un sistema que bajo el mismo esquema permita el control del sistema BES, las consecuencias de un ataque serían catastróficas. Se proponen cinco mecanismos para minimizar los riesgos de seguridad, estos se describen a continuación:

- Actualización y parchado constante de los componentes de software del dispositivo (actualización del sistema operativo, librerías de Python y Ruby utilizadas, etc.).
- Encripción de los datos en transporte, utilizando protocolos como TLS o SSL para las comunicaciones entre el dispositivo de borde (Raspnerry Pi) y la aplicación web de monitoreo.
- Se pueden user ACL (listas de control de acceso, del inglés Access Control Lists) a nivel de red, para restringir el tráfico proveniente tanto de los equipos de borde, como de los clientes de monitoreo autorizados.
- A nivel de aplicación se pueden utilizar claves que identifiquen a los dispositivos monitoreados y que habiliten el acceso a la aplicación de monitoreo.
- Se puede levantar una VPN (red privada virtual) en la aplicación de monitoreo a la cual se puedan conectar los dispositivos de borde para garantizar un túnel de comunicaciones encriptado. Sin embargo, es posible que se deba realizar una evaluación previa de este mecanismo, debido a la sobrecarga introducida durante el proceso de comunicación.

Por otro lado, es importante tomar en cuenta el medio en el que están desplegados los equipos industriales. En el caso puntual de los sistemas BES, en su gran mayoría en Ecuador estos se encuentran instalados en pozos productores de petróleo ubicados en lugares remotos del oriente ecuatoriano. Esto implica que puede no existir acceso a redes públicas de energía eléctrica o de Internet. En el primer caso, se debe considerar que en el ambiente petrolero se suelen utilizar fuentes alternativas de energía como generadores a diésel o a gas, los cuales están sujetos a grandes fluctuaciones en su operación ya que alimentan a dispositivos industriales (como los mismos sistemas BES). Este tipo de fluctuaciones pueden ser perjudiciales para los equipos electrónicos que no fueron concebidos necesariamente para desempeñarse bajo estas condiciones. En el caso del prototipo de monitoreo, este problema se podría neutralizar utilizando mecanismos de almacenamiento y estabilización de energía como UPS y con la instalación de sistemas de conexión a tierra confiables. En el caso de la conexión a Internet requerida para el envío de datos, se pueden utilizar enlaces de datos satelitales para este fin, ya que este medio prácticamente no tiene restricciones de cobertura o alcance.

III. Desarrollo del Prototipo

A. *Arquitectura del prototipo*

En la Fig. 4 se muestra el diagrama de componentes del prototipo propuesto. El *PC simulador de sistema BES*, cuenta con un software que emula un esclavo Modbus, por otro lado se conecta a este computador un *adaptador físico USB / RS-485*, para simular también la interfaz serial de comunicaciones del sistema BES. A los pines de entrada/salida de la *Raspberry* se conecta un adaptador RS-485 para permitir la comunicación serial con el *PC simulador de sistema BES*. Finalmente, este componente es el encargado ensamblar una estructura JSON [4] con los datos monitoreados y enviarlos hacia el *PC servidor web*.

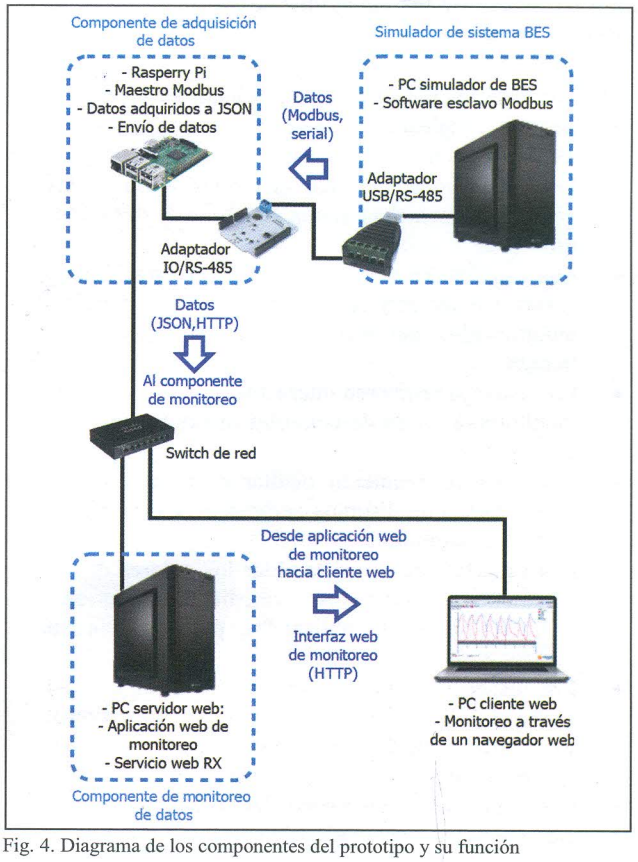

Fig. 4. Diagrama de los componentes del prototipo y su función

El envío del objeto JSON se ejecuta con un comando HTTP POST, para lo cual utilizará un servicio web REST, disponible en una *aplicación de web monitoreo*, desarrollada para este fin. El *componente de monitoreo de los datos*, consta de un *PC servidor web* sobre el cual se implementa una aplicación web para la visualización de los parámetros del *PC emulador del sistema BES* y también se implementa un servicio web que permite la recepción de datos desde el componente de adquisición de datos. Por último, la aplicación de monitoreo web es accedida a través de un *PC cliente web*, a través de un navegador web.

B. Levantamiento de requerimientos

Para la etapa de *levantamiento de requerimientos* de la aplicación web de monitoreo, se utilizan *historias de usuario*. Una historia de usuario describe una funcionalidad que tiene valor para el usuario del software, además es una característica descrita desde el punto de vista del usuario, en lenguaje natural. Los usuarios de la aplicación web de monitoreo se diferencian a través de dos roles, *user*, que es la persona que realizará el monitoreo del sistema BES, es decir, quien será el operador del sistema y el rol *administrator*, el cual además de poder realizar las tareas permitidas al rol *user*, podrá crear y editar los objetos del sistema (usuarios, dispositivos monitoreados, permisos, etc.), a excepción de las mediciones recibidas desde el componente de adquisición de datos (el rol *user* y *administrator* se denominan de esta manera en el código fuente de la aplicación, para fines prácticos de este artículo se utilizarán los términos *usuario* y *administrador*, y el plural *usuarios* para referirse a los dos roles). En base a las historias de usuario, a continuación se muestra un listado de los requerimientos funcionales de la aplicación web de monitoreo.

- Los usuarios requieren la posibilidad de acceder al sistema empleando únicamente un nombre de usuario y contraseña correctos
- Los usuarios requieren visualizar las especificaciones del sistema monitoreado durante el proceso de monitoreo
- Los usuarios requieren acceder a una gráfica con varias curvas correspondientes a los parámetros monitoreados del sistema BES en función del tiempo
- Los usuarios requieren interactuar con el gráfico de monitoreo a través de controles incorporados en la aplicación
- Los usuarios requieren ocultar o mostrar una o varias curvas de distintos parámetros en un mismo gráfico de monitoreo
- Los usuarios requieren cambiar los colores de los parámetros graficados, mediante controles incorporados en la aplicación, para facilitar su visualización
- Los usuarios requieren modificar la escala del gráfico de monitoreo a través de un control incorporado en la aplicación, para facilitar su visualización
- Los usuarios requieren controlar las escalas de cada curva presente en el gráfico de monitoreo, de manera independiente
- Los usuarios requieren contar con la opción de una escala automática para cada parámetro graficado
- Los usuarios requieren modificar la escala del tiempo a través de controles incorporados en la aplicación
- Los usuarios requieren que el gráfico de monitoreo se actualice automáticamente a medida de que los datos lleguen desde el equipo monitoreado
- Los usuarios requieren la opción de generar el gráfico de monitoreo, especificando un intervalo de fechas para la generación de dicho gráfico
- Los usuarios requieren guardar con un nombre las configuraciones de monitoreo (escalas, colores, parámetros mostrados, intervalos de tiempo, etc.) como "perfiles de monitoreo" y poder recuperarlos más tarde
- Los usuarios requieren poder editar o eliminar un "perfil de monitoreo" previamente guardado
- Los usuarios requieren una opción de configuración de rangos de valores máximos y mínimos, para los parámetros monitoreados en cada sistema BES
- Los usuarios requieren recibir una alarma vía correo electrónico en cuanto un parámetro alcance un valor máximo o mínimo configurado
- Los usuarios requieren visualizar un registro de errores de comunicación o tiempo de espera agotado, en caso de que estos errores ocurran entre el componente de adquisición de datos y el equipo monitoreado
- Los usuarios requieren la opción de descargar en un archivo de valores separados por comas, los parámetros monitoreados para un intervalo de fechas determinado
- Los administradores requieren los permisos exclusivos de modificación de objetos del sistema (usuarios, equipos monitoreados, perfiles de monitoreo, etc.) a excepción de los valores monitoreados
- Los administradores requieren que se genere un código único llamado "api key", cuando creen un sistema BES a ser monitoreado, en la aplicación de monitoreo

Una buena parte de las funciones específicas llevadas a cabo por el prototipo en su conjunto y cada uno de los elementos de software que lo componen, permanecen transparentes para el usuario final de la aplicación de monitoreo, es decir que para el usuario permanece transparente el cómo se llevará a cabo el proceso de monitoreo. Este "cómo" está definido en la arquitectura propuesta para el desarrollo del prototipo y en los requerimientos de calidad del prototipo, los cuales se listan a continuación:

- El prototipo debe estar basado en la arquitectura referencial del IoTWF, descrita anteriormente
- Se debe utilizar software libre para su desarrollo del software requerido
- Las aplicaciones de monitoreo y de adquisición de datos utilizarán para su comunicación servicios web REST, con la finalidad de garantizar su independencia, modularidad y reusabilidad

- El componente de adquisición de datos debe permitir la configuración de la dirección de la unidad de superficie Modbus del sistema BES, el período de monitoreo, las direcciones de memoria Modbus donde se almacenan los parámetros monitoreados y la dirección del servicio web donde se requiere realizar el envío de los datos
- Se requiere que la configuración del componente de adquisición de datos se lleve a cabo a través de un archivo de texto.
- No se requiere que el componente de adquisición de datos tenga una interfaz de usuario, sin embargo un usuario que se conecte vía SSH al SBC, debe tener la posibilidad de visualizar un log de ejecución de la aplicación de datos
- La aplicación de adquisición de datos debe iniciar automáticamente en cuanto se encienda el SBC
- En caso de pérdida de comunicaciones entre el componente de adquisición de datos y el de monitoreo, el primero deberá almacenar los datos monitoreados hasta el restablecimiento de las comunicaciones
- En caso de errores de comunicaciones entre el componente de adquisición e datos y el equipo monitoreado (sistema BES), o un tiempo de espera agotado el sistema BES

Antes de concluir esta sección es importante mencionar que el modelo de proceso de desarrollo de software en cascada, en el cual los requerimientos se conocen bien desde un principio tiene varios problemas de aplicabilidad, entre los cuales se listan los siguientes:

- Es complejo para los usuarios de un sistema enunciar en forma explícita todos los requerimientos desde un inicio
- No es común que los proyectos reales sigan un flujo secuencial, ya que una gran cantidad de requerimientos se descubren durante el proceso de desarrollo del software
- El modelo de la cascada presenta dificultades para aceptar la incertidumbre y gestionar el cambio [5]

Una buena cantidad de requerimientos funcionales y de calidad fueron descubiertos durante el desarrollo del prototipo. En esta sección se listan los requerimientos funcionales y de calidad del software requeridos por el prototipo, de una forma secuencial y ordenada, únicamente con la finalidad de facilitar la lectura y comprensión de este artículo.

C. *Diseño de la aplicación de adquisición de datos*

En la Fig. 5 se presenta el diagrama de clases de la aplicación de adquisición de datos.

Las clases indicadas en el diagrama, han de interpretarse como sigue:

- La clase Master, es una abstracción que representa al maestro Modbus, el cual se comunica con el esclavo de la unidad de superficie del sistema BES.

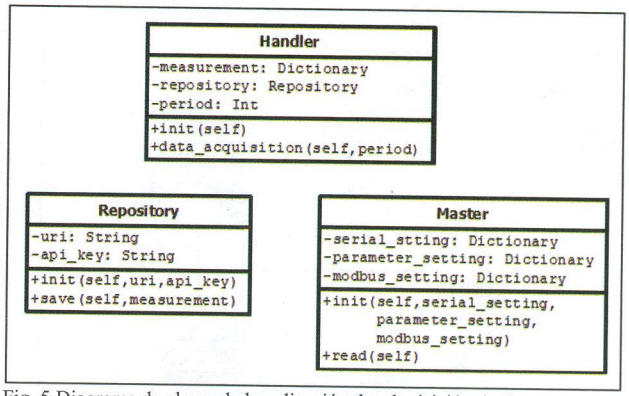

Fig. 5 Diagrama de clases de la aplicación de adquisición de datos

- La clase Repository es una abstracción del recurso donde se va a enviar la información adquirida, que para esta implementación es un servicio web.
- La clase Handler es desde donde se instancian las clases Repository y Master para leer los datos desde el esclavo Modbus del PC simulador del sistema BES, formatear las lecturas de datos en JSON y enviarlas al servicio web de recepción de datos, utilizando el método save de la clase Repository.

D. *Diseño de la aplicación web de monitoreo*

La aplicación web de monitoreo se desarrolla bajo el patrón de arquitectura de software MVC. El modelo es la parte encargada del almacenamiento de los datos de la aplicación y el controlador es el encargado de recibir las peticiones, solicitar datos al modelo y generar las vistas.

En la Fig. 6 se muestra el diagrama de clases modelo y en la Fig. 7 se muestra el diagrama de las clases controlador.

Las clases indicadas en los diagramas anteriores representan lo siguiente:

- User: Esta clase representa los usuarios del sistema.
- Device: Esta clase representa los dispositivos a ser monitoreados. Un objeto de la clase Device, representa a un sistema BES.
- Measurement: Esta clase modelo representa los objetos de mediciones que llegan desde el componente de adquisición de datos. Los objetos medición deben tener información del dispositivo (clase Device) al que pertenecen. Un dispositivo puede tener muchas mediciones.
- ParameterRange: Esta clase representa el rango máximo y mínimo que puede tomar un parámetro medido. Para un dispositivo se pueden definir un rango de valores máximo y mínimo por cada parámetro monitoreado.
- MonitoringProfile: Esta clase representa un perfil de monitoreo, que son los parámetros de configuración de los gráficos de monitoreo y la información del eje del tiempo. Los perfiles de monitoreo generan un alto valor al usuario de la aplicación, ya que almacenan la configuración de gráficos que el usuario requiere generar de manera recurrente.

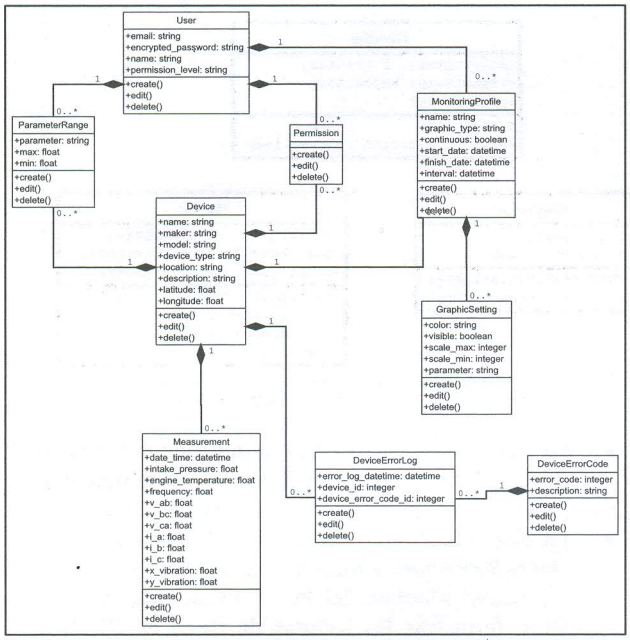

Fig. 6 Diagrama de clases de modelos de la aplicación web de monitoreo

- GraphicSetting: Esta clase contiene la configuración para la generación de una gráfica con los valores de monitoreo de un parámetro específico. Esta configuración incluye escala, color y visibilidad. Un perfil de monitoreo puede tener varios objetos de tipo GraphicSetting.
- Permission: Esta clase representa el permiso que tiene un usuario para visualizar los datos de monitoreo de un determinado dispositivo.
- DeviceErrorLog: Esta clase representa los logs de errores que se generen en la aplicación.
- DeviceErrorCode: Esta clase representa los códigos de error que pueden ser generados por errores en el proceso de adquisición de datos.

E. Diseño del servicio web de recepción de datos

Para el diseño del servicio web para la recepción de datos, se toman en cuenta los siguientes criterios. La función de este componente del prototipo, es recibir los datos, validarlos y finalmente almacenarlos usando el modelo Measurement. En el patrón MVC el rol de recibir las peticiones, procesarlas y enviarlas al modelo para finalmente almacenar los datos, es desempeñado por las clases controlador, por este motivo el servicio web se diseña como un controlador de la aplicación web [6, pp. 29-35].

F. Diseño de la interfaz gráfica [7]

En la Fig. 8 se muestra un bosquejo de la vista en la que el usuario realiza el proceso de monitoreo. En este bosquejo se pone énfasis en la visibilidad del gráfico de monitoreo y los controles de los parámetros, ubicados debajo del gráfico. En la vista de monitoreo propuesta, se muestra de forma detallada la información del sistema que se está monitoreando.

Tomando en cuenta la accesibilidad, se propone en este diseño, un menú superior que permita acceder rápidamente a las vistas de listado, creación, edición y eliminación de los

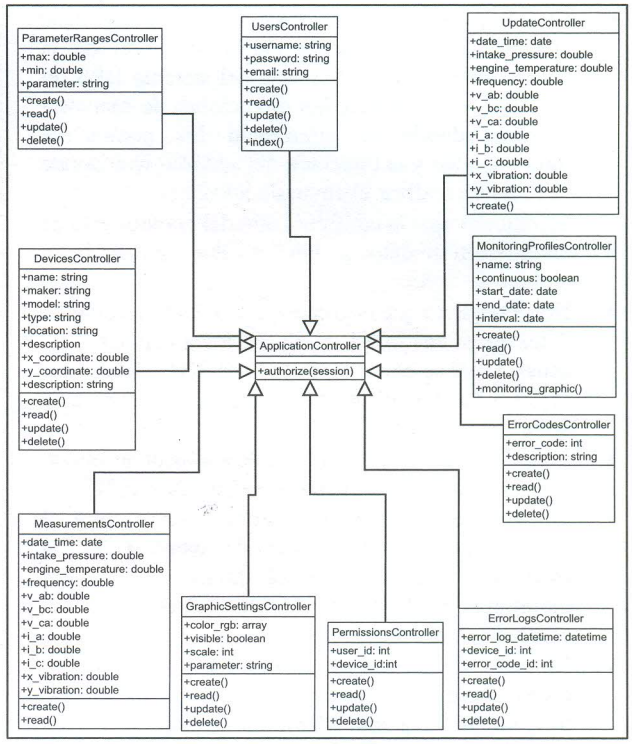

Fig. 7 Diagrama de clases de los controladores de la aplicación web de monitoreo

objetos del sistema (dispositivos, perfiles de monitoreo, usuarios, permisos, alarmas, etc.). Este menú también contiene los enlaces de inicio o cierre de sesión.

G. Implementación de la aplicación de adquisición de datos

Para esta implementación se emplea el lenguaje de programación orientado a objetos Python y se construyen cada una de las clases de la aplicación tomando en cuenta el diseño descrito anteriormente.

El flujo de trabajo de la aplicación se lleva a cabo como sigue: la clase Handler se encarga de leer un archivo de configuración que contiene los parámetros de la aplicación (dirección del esclavo Modbus, direcciones Modbus de los datos de monitoreo, la URL del servicio web, el intervalo de monitoreo, etc.) y con estos parámetros se instancian las clases Master y Repository.

Fig. 8. Vista principal de monitoreo. En la parte izquierda del gráfico y los controles, se puede apreciar la información completa del equipo monitoreado

Fig. 9. Interfaz gráfica de monitoreo

Se puede emplear el procedimiento especificado en https://s3.amazonaws.com/public.andresmunozit.com/docs/configuracion-simulador-bes.pdf para configurar un ambiente de pruebas para el prototipo.

IV. Resultados

En la sección de desarrollo del prototipo, se encuentran referenciados varios enlaces externos que contienen en extremo detalle el despliegue y operación de los componentes del prototipo (en total 53 páginas). En estos documentos se puede apreciar el cumplimiento de los requerimientos funcionales y de calidad de las aplicaciones, los cuales se encuentran establecidos en la sección de levantamiento de requerimientos.

Con respecto a uno de los requerimientos de calidad del prototipo, el cual indica que el mismo debe ajustarse a la arquitectura IoT del IoTWF, este análisis se realiza en la sección de definición de Arquitectura de Internet de las Cosas. La única observación que puede derivar de este análisis, es que la capa de abstracción de los datos definida para este prototipo, implementada en este prototipo mediante el modelo de la aplicación web, tiene soporte para el uso de distintos tipo de bases de datos relacionales sin incorporar necesariamente un soporte en la aplicación web de monitoreo, para otro tipo de mecanismos de acumulación de datos, como por ejemplo Hadoop, utilizado con en Big Data. Esto se da porque no existe como requerimiento específico de esta implementación, dotar de un mecanismo de análisis de un *volumen* colosal de datos, de *varios* tipos y fuentes, a una alta *velocidad* de procesamiento, características que se constituyen en los tres pilares fundamentales de Big Data. Sin embargo, existen gemas de Ruby para habilitar la comunicación entre una aplicación desarrollada en este lenguaje y Hadoop como es el ejemplo de Rubydoop (https://github.com/iconara/rubydoop). Esta funcionalidad se puede desarrollar en el futuro en caso de que se requiera.

Otro resultado importante, es la rapidez en la detección de eventos suscitados en los sistemas monitoreados. Durante el proceso de revisión manual de los parámetros monitoreados, como se indicó en la descripción del problema, se introduce un retardo de detección de incidentes en los pozos productores. Con la implementación de este prototipo, este retardo prácticamente se elimina debido a dos factores. El primero es la posibilidad de crear alarmas para valores críticos de los parámetros o errores de comunicación entre el sistema BES y el equipo de borde, o entre el equipo de borde y la aplicación de adquisición de datos. En segundo lugar, estas alarmas pueden ser enviadas al creador de las mismas y a otros destinatarios alrededor del mundo, vía correo electrónico, esto abre la posibilidad de un tipo de monitoreo pasivo y colaborativo, ya que varias personas pueden estar atentas de estas alarmas e indicar acciones al personal en sitio. Otra forma de mitigar este problema es emplear personal humano para la revisión constante de problemas en los sistemas BES, sin embargo esto es extremadamente ineficiente y sería un trabajo precario para un ser humano.

Otro resultado importante, derivado de los requerimientos de calidad del prototipo, es que se elimina la necesidad de un operador humano que realice el monitoreo en sitio del sistema BES de forma periódica. Sin embargo, al estar el equipo de borde operando en una zona remota, sin intervención humana, su funcionamiento debería ser resilente y confiable.

En el caso de la implementación de este prototipo, el sistema tiene la capacidad de recuperarse de errores de comunicación de dos formas, la primera es que en caso de un corte de comunicación, el componente de adquisición de datos almacena la información hasta que la comunicación esté disponible nuevamente, y por otro lado, la aplicación de monitoreo notifica al usuario mediante un correo electrónico, acerca de estas fallas de comunicación (como se indicó anteriormente).

Por otro lado, en caso de fallas de energía, normalmente la misma fuente de energía del sistema BES se puede utilizar para la operación del componente de adquisición de datos (generadores a diésel o gas), por lo que el usuario de monitoreo web, conocerá de inmediato de esta falla e informará al personal en sitio para que tome cartas en el asunto.

Finalmente, en caso de que el dispositivo de adquisición de datos sufra un apagado inesperado, la aplicación de adquisición de datos iniciará automáticamente una vez se encienda el SBC, enviará un mensaje de error al sistema de monitoreo web y continuará operando con normalidad.

V. Conclusiones

Es factible desplegar el prototipo desarrollado en un ambiente de producción, tomando en cuenta los siguientes condicionamientos explicados a lo largo del documento:

- Una fuente de energía estable, un equipo de protección eléctrica (como un UPS) y una conexión a tierra confiable.
- Es importante que durante un proceso de implementación en producción de este prototipo, se consideren aspectos de seguridad de redes IT con la finalidad de minimizar los riesgos, ya que en el caso de este prototipo, un ataque de acceso no autorizado a la red IT podría implicar la escucha no autorizada de los datos de monitoreo, pero en otras aplicaciones de IoT industrial podría derivar en la toma de control no autorizado de los equipos industriales, lo cual tendría consecuencias catastróficas.

- Es necesario tomar en cuenta que el entorno de aplicabilidad de esta solución, suelen ser sitios remotos sin acceso o con acceso mínimo a redes públicas de telecomunicaciones, por lo que en el mejor de los casos se podría emplear para la comunicación entre la capa de borde y las capas superiores un servicio GPRS (General Packet Radio Service) o directamente enlaces satelitales.
- Cabe mencionar que en un escenario donde los sistemas monitoreados se encuentren en una misma ubicación geográfica, se puede compartir el mecanismo de comunicaciones desde el borde hacia capas superiores, debido al uso del stack de protocolos TCP/IP por parte del dispositivo de borde.

Las ventajas contra una recopilación manual de información son evidentes y ello se puede observar en la sección de resultados. Sin embargo, al realizar una breve comparación de IoT industrial y otros mecanismos de monitoreo como sistemas SCADA, se puede concluir lo siguiente:

- La integración entre distintos sistemas SCADA puede llegar a tener un alto grado de complejidad debido al uso de soluciones propietarias, dependientes de los fabricantes. IoT industrial garantiza la interoperabilidad entre dispositivos de distintos fabricantes, al hacer uso de los protocolos de comunicación de red de IT. Uno de los requerimientos de calidad del prototipo (comunicación a través de un servicio web REST), está enfocado justamente al uso de mecanismos que permitan la modularidad e independencia de sus componentes.
- Los sistemas SCADA implican un TCO (coste total de propiedad, proveniente del inglés Total Cost of Ownership) mucho más alto para las empresas, ya que por el mencionado uso de componentes propietarios, se requiere la adquisición de equipo especializado y licencias de software. IoT industrial habilita la opción del uso de software libre para el almacenamiento y análisis de los datos, eliminando la necesidad de licencias. Adicionalmente permite reducir drásticamente el TCO de un sistema de supervisión, al abrir la posibilidad del uso de computación en la nube, a medida y bajo demanda. En el caso de la aplicación web de monitoreo, se puede realizar su despliegue en servidores virtuales en la nube.
- De los puntos anteriores se deriva la mejora considerable desde el punto de vista de la escalabilidad, de IoT industrial con respecto a los sistemas SCADA. El despliegue en la nube de las aplicaciones de análisis y almacenamiento, habilita opciones de aprovisionamiento dinámicos de recursos computacionales bajo demanda y proporciona mecanismos de redundancia a nivel de regiones geográficas. Incluso existen opciones de habilitación de auto escalamiento de los recursos computacionales asignados. Utilizando SCADA este aprovisionamiento debe ser planificado y dimensionado de forma anticipada y de la manera más prolija posible.

REFERENCIAS

[1] Flattern, R (n.d.). *Electrical Sumersible Pumps*. 2015, Accedido en: Sep. 17, 2018, [En línea] Disponible: http://www.slb.com/~/media/Files/resources/oilfield_review/defining_series/Defining-ESP.ashx

[2] Modbus Organization, Inc., *MODBUS Application Protocol Specification V1.1b3*. 2012. Accedido en: Sep. 17, 2018, [En línea] Disponible: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[3] Hanes D., *"IoT fundamentals: networking technologies, protocols, and use cases for the internet of things"*. Indianapolis, IN: Cisco Press, 2017

[4] ECMA International (n.d.). *The JSON Data Interchange Synthax*. 2017, Accedido en: Sep. 17, 2018, [En línea] Disponible: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf.

[5] Pressman, R. "*Ingeniería de Software. Un Enfoque Práctico*," New México, D. F., México: McGraw-Hill, 2010, pp. 33–34.

[6] Ruby, D., Thomas, D., y Hasson, D. Building an Application, *"Agile Web Development with Rails"*, Estados Unidos de América: Programatic Programmers, LLC, 2012

[7] Bourque, P., & Fairley, R (n.d.). User Interface Design. *Guide to the Software Engineering Body of Knowledge, Version 3.0, SWEBOK*, 2014. Accedido en: Sep. 17, 2018, [En línea] Disponible: http://www.swebok.org

Muñoz Rosero Andrés Sebastián: Nacido en Quito, Ecuador, el 16 de diciembre de 1986. En 2005 obtiene su título de Bachiller Técnico Electrónico en el Instituto Tecnológico Superior Central Técnico de Quito. Desde el 2011 hasta la actualidad ha desempeñado actividades de administración de infraestructura de red y desarrollo de software en importantes empresas del sector de la energía como Tecnie y Consorcio Pegaso. En julio de 2017 presenta el proyecto "Desarrollo de un Prototipo para Monitorear el Sistema de Bombeo Electro Sumergible de un Pozo Productor de Petróleo", el cual desarrolla con el auspicio de Consorcio Pegaso, operador petrolero del Bloque 45, Dayuma, Ecuador, para obtener el título de Ingeniero Electrónico y de Redes de Información. (am@andresmunozit.com)

Flores Cifuentes Williams Fernando: Graduado en la Escuela Politécnica Nacional en Electrónica y Telecomunicaciones en el año 1984. Actualmente y desde abril de 1982 se desenvuelve como docente del DETRI (Departamento de Electrónica, Telecomunicaciones y Redes de Información). Trabajó como Asesor de Teleinformática en ASETA desde Febrero de 1987 hasta Mayo de 2001. (fernando.flores@epn.edu.ec)