TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE ACAPULCO

DESARROLLO DE PROTOTIPO SCADA PARA CONTROL DE NIVEL DE AGUA EN DOS CONTENEDORES PARA USO EN LA COMISIÓN DE AGUA POTABLE Y ALCANTARILLADO DEL MUNICIPIO DE ACAPULCO (CAPAMA)

TESIS PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:
MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA:
ING. HUGO ROJAS SALGADO

DIRECTOR DE TESIS:
M.T.I. ELOY CADENA MENDOZA

CODIRECTOR DE TESIS:
M.T.I. JUAN MIGUEL HERNÁNDEZ BRAVO

ACAPULCO, GRO., DICIEMBRE 2019.

El presente trabajo de tesis fue desarrollado en la *División de Estudios de Posgrado e Investigación* del *Instituto Tecnológico de Acapulco*, perteneciente al Programa Nacional de Posgrados de Calidad (PNPC-CONACYT).

Con domicilio para recibir y oír notificaciones en Av. Instituto Tecnológico de Acapulco s/n, Crucero del Cayaco, Acapulco, Guerrero, México. C.P. 39905.

Becario: Hugo Rojas Salgado.
CVU: 851861.
Núm. de apoyo: 474892.
Grado: Maestría

Declaración de responsabilidad

El que suscribe C. Hugo Rojas Salgado, alumno de Maestría en Sistemas Computacionales, en el Instituto Tecnológico de Acapulco, con el número de control: G17320006, declara que es el autor intelectual y el único responsable del presente trabajo de tesis, intitulado "**Desarrollo de prototipo SCADA para control de nivel de agua en dos contenedores para uso en la Comisión de Agua Potable y Alcantarillado del Municipio de Acapulco (CAPAMA)**", que fue desarrollado bajo la supervisión y dirección del asesor M.T.I Eloy Cadena Mendoza.

Hugo Rojas Salgado
Ingeniero en Sistemas Computacionales

Agradecimiento

A Dios por darme salud y fuerzas para seguir adelante.

A quienes con su conocimiento, experiencia y guía me formaron como profesional a lo largo de mi carrera académica.

Al equipo de trabajo de la Comisión de Agua Potable y Alcantarillado del Municipio de Acapulco por sus atenciones y por permitirme desarrollar el presente proyecto de tesis.

Al Conacyt por el apoyo económico a lo largo de dos años y por incentivarme a seguir conociendo y aprendiendo.

A mis padres por inculcarme el hábito del estudio.

Dedicatoria

Dedico esta obra con todo entusiasmo y cariño a mí amada esposa Araceli, por toda su paciencia y apoyo incondicional brindado durante el tiempo de desarrollo y escritura del presente trabajo.

Resumen

El presente objeto de estudio, tiene como finalidad a través de la investigación de artículos, herramientas, métodos y técnicas actuales para desarrollar sistemas de adquisición de datos enriquecidos con una interfaz de usuario (GUI) y de las muchas alternativas que existen en el mercado, formalicen la base para el desarrollo de un prototipo SCADA con el uso de tecnologías orientadas al Internet de las Cosas, que son económicas y que están orientadas al ámbito de "*hardware libre*", esto con el fin de que el prototipo se utilice en un futuro en la Comisión de Agua Potable y Alcantarillado del Municipio de Acapulco (CAPAMA), debido a que actualmente, CAPAMA presenta el inconveniente de mucha perdida de agua debido al excesivo derrame de agua en los contenedores por falta de control y automatización de sus bombas de agua, de no ser así, si no se atiende el problema se seguirá desperdiciando mucha agua.

Abstract

The present object of study, has like purpose through the investigation of articles, tools, methods and current techniques to develop systems of acquisition of data enriched with a user interface (GUI) and of the many alternatives that exist in the market, formalize the basis for the development of a prototype SCADA with the use of technologies oriented to the Internet of Things, which are economic and that are oriented to the field of “free hardware”, this in order that the prototype is used in the future in the Comisión de Agua Potable y Alcantarillado del Municipio de Acapulco (CAPAMA), because CAPAMA currently has the disadvantage of a lot of water loss due to excessive water spillage in the containers due to lack of control and automation of its water pumps , if not, if the problem is not addressed, a lot of water will be wasted.

Índice de contenido

Índice de figuras

Índice de tablas

Capítulo 1. Introducción al planteamiento de investigación

1.1 Antecedentes

Como dice (CAPAMA, 2019) en el año de 1946, dentro del periodo presidencial de Lic. **Miguel Alemán Valdez** se expidió un decreto mediante el cual se crea la primera *Junta Federal de Mejoras Materiales (J.F.M.M.)* con residencia en Acapulco, la cual estaba a cargo del señor **Melchor Perusquía** quien con el amplio apoyo económico del gobierno federal desplegó una gran actividad en la construcción de importantes obras materiales, para el puerto de Acapulco.

Una obra estaba orientada principalmente a resolver el *problema del agua*, por lo cual se iniciaron exploraciones y sondeos en varias zonas de la región, principalmente en el *Valle de la Sabana*, en donde se encontraron magníficos mantos acuíferos subálveos capaces de proporcionar agua en cantidades suficientes no solo para satisfacer las necesidades que había en ese entonces sino que existían volúmenes para incrementos de gastos mayores que pudieran requerirse en el futuro. Hacia el año de 1950 para avanzar rápidamente en los trabajos de construcción la junta contrató los servicios de gente especializada en hidráulica, iniciando en primer término la perforación de 7 pozos someros, en el margen derecho del rio de la sabana imponiendo a la zona el nombre de *campo de pozos de agua potable*. El sistema estuvo vigente hasta el año de 1960.

Para el año de 1969 la población de Acapulco había crecido alrededor de 174,800 habitantes aproximadamente, parte de ello debido a las emigraciones precipitadas, la mayoría de éstas personas llegaban y se asentaban sin algún orden, pues la mayoría de ellos se ubicaban de forma anárquica principalmente en el área del anfiteatro, por lo que esto originó a que se suscitaran *problemas relacionados fundamentalmente con el agua potable, el drenaje y las aguas pluviales*, por lo que las autoridades de los tres niveles de gobierno, consideraron que era urgente ordenar la reubicación de los asentamientos humanos hacia el valle de la ***Sabana***, formando de esta manera

la colonia que lleva por nombre ***Emiliano Zapata***. Para ese entonces las autoridades ya habían concluido los estudios de hidráulica que habían hecho con anterioridad con el objetivo de resolver el futuro del agua en Acapulco, concluyendo que la única manera de resolver el *problema del agua*, seria aprovechado las aguas del **Río Papagayo**, dichos estudios fueron realizados por los técnicos de la **Secretaria de Recursos Hidráulicos**, los cuales contemplaban la posibilidad de construcción de un pozo ***tipo Ranney*** [1]con un rendimiento total de 300 litros por segundo y la construcción de 12 pozos someros que en conjunto aportarían un gasto aproximado de 500 litros por segundo, y proyectaban la construcción de tres pozos más a futuro asegurando una producción de 1,000 litros por segundo. A dicha obra se le denominó **sistema Papagayo I**, esta obra tenía la finalidad de *mejorar el sistema de agua potable y alcantarillado*.

En el año de 1972, se desincorporaron los servicios de agua potable y alcantarillado sanitario, creándose una **Junta Administradora de Agua Potable y Alcantarillado** (**J.A.A.P.A.**), con el propósito de descentralizar los servicios del gobierno federal y lograr una mayor participación ciudadana por parte de los usuarios.

Posteriormente hasta el año de 1975 la Secretaria de Recursos Hidráulicos realizó los estudios correspondientes para determinar la factibilidad para construir una nueva captación de agua sobre la margen derecha del río Papagayo así como para determinar las obras por realizar y su costo aproximado. Tras un exhaustivo análisis propusieron como una solución más adecuada para la captación de agua la construcción de una obra de toma directa o ***bocatoma*** sobre la margen derecha del río papagayo a una distancia aproximada de un kilómetro, aguas abajo del sistema **Papagayo I**, denominándole a dicha construcción con el nombre de **Papagayo II.**

[1] Como dice Comisión Nacional del Agua se utiliza un pozo ***Ranney*** para captación de agua subsuperficial más eficiente y constan de un depósito central en donde se capta agua que recolectan tuberías radiales perforadas e inmersas en la zona saturada del acuífero.

En el año de 1977, se crea la Comisión de Agua Potable y Alcantarillado del Municipio de Acapulco (**CAPAMA**) como una entidad paraestatal de la administración pública del estado de Guerrero.

Para el año de 1989, por disposición de ley se crea la Comisión de Agua Potable y Obras Urbanas de Interés Social del Municipio de Acapulco (**CAPOUISMA**) como organismo público municipal. Pero el 29 de abril de 1994, por las nuevas reformas a la ley, retoma la denominación de **CAPAMA**, con carácter de organismo fiscal autónomo.

Alrededor del mes de septiembre del año 2013, se pone en operación la tercera captación sobre el rio Papagayo, denominada *Sistema de Lomas de Chapultepec*, trabajando a un 60% de su capacidad, con el objetivo de incrementar el caudal de agua potable, para el municipio de Acapulco en 1,200 litros por segundo.

La Comisión de Agua Potable y Alcantarillado del Municipio de Acapulco (**CAPAMA**), se encuentra integrada básicamente por una dirección general que administra cuatro principales subdirecciones, las cuales son:

- Secretaría particular.
- Contraloría general.
- Subdirección de informática.
- Subdirección jurídica.

Con el objetivo de delimitar los antecedentes del problema, dada la relevancia que tiene la subdirección de informática de CAPAMA para el objeto de estudio del presente proyecto se hace mención que se divide en tres departamentos que son los siguientes:

- Departamento de operaciones

Es el encargado de gestionar la impresión de recibos de agua, manejo de un sistema IBM AS400 (Sistema de control de recibos de agua, empleados y nómina).

- Departamento de análisis y desarrollo

Es el encargado de gestionar y desarrollar y mantener los portales de internet de CAPAMA, los pagos que hacen los usuarios a través del portal, y los cajeros automáticos denominados por **CAPAMA** con el nombre de capamáticos.

- Departamento de soporte técnico

Se encargan de toda la infraestructura de la red digital de **CAPAMA** (cableado de líneas de teléfonos, tendidos de cables de red ***UTP Ethernet*** y mantenimiento a cajeros capamáticos).

Dada la importancia que tiene el departamento de análisis y desarrollo de CAPAMA, para el presente objeto de estudio y con el objetivo de delimitar el planteamiento del problema, tenemos que hacer mención que dicho departamento se encarga de llevar el control de todos los sistemas informáticos de **CAPAMA** y el caso que nos acomete en éste estudio es el sistema **SCADA.**

El sistema en cuestión, que por sus siglas en inglés se llama *Supervisory Control And Data Acquisition* (**SCADA**), traducido al español como Supervisión, Control y Adquisición de Datos, fue implementado alrededor del año 2002 por la Comisión Nacional del Agua (**CONAGUA**) y la Comisión de Agua Potable Alcantarillado y Saneamiento del Estado de Guerrero (**CAPASEG**), su implementación se planeó por etapas y en su conjunto se encontraba instalado en alrededor de 53 sitios de los tanques de almacenamiento y distribución de agua potable de **CAPAMA**, localizados en diferentes zonas del municipio de Acapulco.

Uno de los objetivos por los que se implementó el sistema **SCADA** fue con la finalidad de automatizar el proceso que ayude a no desperdiciar agua potable en los tanques de almacenamiento y distribución, por lo que con este sistema pretendían realizar el proceso de monitoreo de niveles

de agua en los tanques, así como también el control de bombeo de agua de manera automática y remota, de esta manera el agua podría ser bombeada de un tanque a otro con base en su nivel de agua y apagando o encendiendo las bombas según sea su límite del nivel de agua establecido, evitando la perdida de agua por derramamiento en los tanques de agua, esto en contraste a como era en épocas anteriores, que todo se controlaba de forma manual, por lo que involucraba la pérdida de agua potable debido a que controlaban las bombas y tomas de agua de manera manual y el criterio para tomar la decisión de controlar las bombas, eran realizadas por un operador de **CAPAMA** y éste básicamente se basaba en la *observación,* cuando los tanques de almacenamiento y distribución se encontraban derramando agua potable, él tomaba la decisión de apagar y/o encender la bomba según era la situación, en este punto cabe hacer mención que el tiempo en el que apagaban la bomba, en un tanque lleno de agua, oscilaba entre los dos o tres días después debido a que un factor era las grandes distancias de los tanques de las zonas conurbadas, por lo que por consiguiente se generaba la pérdida de miles de litros de agua potable en dichos tanques.

Otra parte por la que fue implementado **SCADA**, era con la finalidad de ahorrar en el consumo de corriente eléctrica, ya que como se mencionó anteriormente, las bombas seguían funcionando tiempo después, de que el tanque destino al que le bombeaban agua estaba lleno por lo que por consiguiente se perdía agua potable y corriente eléctrica.

En la actualidad, de los 53 sitios en donde se encuentra implementado **SCADA**, solamente 2 sitios se encuentran en funcionamiento, y los demás sitios restantes presentan una serie de problemas como:

- Por el tiempo; la vida útil de los sensores y el desgaste del equipo, debido a que los sistemas se instalaron alrededor del año 2002, por lo que para la actualidad la mayoría esta inservible.

- Por la zona de alta salinidad que conlleva a la corrosión de los sensores y componentes electrónicos del sistema **SCADA**.
- Otro factor es que únicamente se instalaron los componentes de **SCADA** en los sitios de los tanques, pero nunca se utilizaron, ni se realizó la comunicación con la central de telemetría de **CAPAMA**.
- Se menciona también que el mantenimiento preventivo y correctivo, es un factor que afectó al funcionamiento de los sistemas **SCADA** porque algunos sitios en donde se instalaron se encuentran muy retirados de la ciudad y de difícil acceso, por lo que por consiguiente no se les da mantenimiento.
- Se hace hincapié en que la cobertura de la red de comunicación por radiofrecuencia, jugó un papel importante para el desempeño de los sistemas **SCADA**, desencadenando en su mal funcionamiento, dado que hay sitios que se encuentran muy retirados y no hay línea de vista.
- Un factor aunado al punto anterior es que, en horas de la media tarde se genera interferencia en el sistema **SCADA**, por lo cual desencadena en un tiempo de 5 minutos de desfase, generando que en la central de telemetría de **CAPAMA** no se muestre información del estatus de los niveles, y estado actual de las bombas en tiempo real.
- Otro factor influyente por lo que el sistema **SCADA** está detenido totalmente en la mayoría de los sitios es debido a que no fue una buena planeación de la implementación del sistema **SCADA**, (se menciona que algunos sitios se implementó pero a medias), ni tampoco se capacitó al personal para su uso, calibración y mantenimiento de los componentes.
- Este punto aunado al problema anterior desencadena en que el personal no cuenta con conocimiento para operar el sistema **SCADA**, por lo que por consiguiente no tiene

experiencia en el manejo del sistema, se hace mención que también no se involucró al personal de perfil adecuado en el proyecto, por lo que esto genera en el desinterés personal por mantener y operar el buen funcionamiento del sistema.

- La falta de corriente eléctrica en algunos sitios origina en que el sistema **SCADA** no funciona en su totalidad.
- Otro factor es con respecto a la coordinación de departamentos; entre del departamento de mecánica, el departamento eléctrico y el departamento de telemetría, porque no hay sincronía, por ejemplo el departamento de mecánica no avisa cuando hay bombas descompuestas en las zonas en donde se encuentra el sistema **SCADA**, de igual manera el departamento eléctrico tarda en reparar las instalaciones eléctricas de determinado sitio en donde se encuentra implementado **SCADA** dejando sin funcionamiento el sistema; porque no notifican; no hay un sistema que concentre las incidencias.

1.2 Planteamiento del problema

Se estima que del 100% de agua potable que se almacena en un tanque de almacenamiento y distribución de agua, el 70% del agua se pierde por derrame en el contenedor y por fugas, debido a que no cuentan con un sistema eficaz y eficiente que supervise los niveles de agua y administre el control.

Ha habido intentos por reparar y rehabilitar los sistemas **SCADA** en los sitios en donde no funciona, más sin embargo no se ha podido por que el sistema es desarrollado por un tercero, por tal motivo no se tiene acceso a los diagramas electrónicos ni al código del software para adaptarlo y escalarlo, además el costo de reimplementación por cada sitio es muy alto.

Falta de inversión para adquirir componentes para rehabilitar el sistema **SCADA** o adquirir nuevos sistemas de monitoreo que tienen un costo elevado.

Dado que en la actualidad la Comisión de Agua Potable y Alcantarillado del Municipio de Acapulco (**CAPAMA**) tiene implementado **SCADA** en 53 sitios, de los cuales solo 2 sitios están en funcionamiento con dicho sistema, si no se resuelve el control del encendido de bombas se seguirá desperdiciando mucha agua.

Por lo que con respecto a lo anteriormente descrito se pretende contar con un sistema de control y adquisición de datos que sea modular y escalable de acuerdo a los requerimientos de expansión de **CAPAMA**.

1.3 Objetivo general

Desarrollar un prototipo **SCADA** para control de nivel de agua en dos contenedores para su uso en la Comisión de Agua Potable y Alcantarillado del Municipio de Acapulco (**CAPAMA**).

1.4 Objetivos específicos

- Análisis de requerimientos.
- Desarrollar el sistema **SCADA** con la implementación de una tarjeta de desarrollo Intel Galileo 2 y sensores ultrasónicos.
- Desarrollar software servidor para el protocolo MQTT en ASP.NET Core.
- Desarrollar el software interfaz gráfica en un mapa que muestra información de tiempo real de los niveles de los contenedores en HTML5 y JavaScript.
- Integrar la comunicación del prototipo hardware y software para realizar pruebas de comunicación en tiempo real el mediante protocolo MQTT.

1.5 Hipótesis

El desarrollo de un prototipo electrónico **SCADA** utilizando tecnologías emergentes orientadas al Industrial Internet de las Cosas, y el desarrollo de una herramienta de software utilizando el protocolo **MQTT**, que muestra información en tiempo real de los niveles de agua en contenedores,

ayudará a disminuir significativamente el tiempo de comunicación de petición y respuesta remota, y a ofrecer seguridad en los datos y un óptimo desempeño en largas distancias.

1.6 Justificación

Este trabajo de tesis propone desarrollar un prototipo **SCADA** para controlar el nivel de agua de dos contenedores, que la Comisión de Agua Potable y Alcantarillado del Municipio de Acapulco (**CAPAMA**) puede utilizar en un futuro para controlar y monitorear los niveles de tanques de almacenamiento y distribución que tienen alrededor de la zona conurbada de Acapulco.

Por lo tanto la implementación de este sistema podría ayudar a disminuir las pérdidas de agua potable y ahorrar agua en los contenedores, por lo que el proyecto es pertinente en el contexto, dado que en la actualidad existe una pérdida significativa de agua potable en los tanques, debido a que su actual sistema **SCADA** demora mucho en ofrecer datos reales de sus niveles de agua en sus contenedores.

Impacto social. El sistema propuesto en este trabajo de tesis es relevante porque podría causar un impacto social, ya que está relacionado con el bienestar de las personas que habitan las zonas conurbadas de Acapulco, dada la importancia que tiene **CAPAMA** para almacenar y distribuir agua potable a las 649 mil ciudadanos aproximadamente a los cuales les ofrece el servicio en el municipio de Acapulco, que en la actualidad sufren por escases de agua en algunos casos entre uno y dos tres meses, de esta manera surge la necesidad de cuidar el agua potable con mecanismos que ayuden a ahorrar el vital líquido, por lo que si el sistema propuesta se implementa en otros sitios de tanques de almacenamiento y distribución de **CAPAMA**, podríamos contribuir a que llegue y alcance su distribución a más personas por tiempos más constantes y prolongados.

Impacto tecnológico. Puede causar un impacto tecnológico debido a que es una solución desarrollada y adaptada a las necesidades de **CAPAMA**, por lo que por consecuencia ayudará al

crecimiento de las necesidades de automatización y replicación de tanques de CAPAMA en el futuro.

Impacto económico. El proyecto es viable para CAPAMA, por la inversión monetaria en los componentes electrónicos y la inversión de la infraestructura de la red de comunicación que está orientada en la adquisición de componentes orientados a tecnologías de hardware emergentes *Open Source* orientadas al Industrial Internet de las Cosas, esto como tal, tiene un efecto potencial dado que pueden escalar el sistema a más tanques con menos gasto, cabe hacer mención que en esta parte juega un rol importante que el sistema propuesto trabaje en un ecosistema abierto, en perspectiva esa es la tendencia para **CAPAMA** y ellos puedan adecuarlo a sus necesidades de crecimiento y expansión.

1.7 Alcances y limitaciones

Los alcances y limitaciones para el proceso de desarrollo del sistema prototipo, involucra:

Como alcances el sistema pretende:

- Desarrollar un sistema de interfaz *hardware* que actuará como sistema de control y adquisición de datos, el cual está conectado con sensores ultrasónicos para medir el nivel del tanque de agua, sensor para medir la humedad, temperatura y un sensor para medir la corriente consumida por la bomba de agua.
- El dispositivo *hardware* envía información mediante la red Wi-Fi a un servidor de datos que interactuará con un operador mediante una plataforma de software de tiempo real.
- Desarrollar *software* que se comunica con el prototipo *hardware* para visualizar información en tiempo real y observar el estado del nivel de agua de los contenedores y manipular el encendido y apagado de la bomba, desde cualquier dispositivo conectado al punto de acceso.

Como límites del proyecto:

- La utilización de componentes sensores industriales de nivel de agua para la comunicación del software mediante el protocolo *Message Queue Telemetry Transport* (**MQTT**).
- La implementación del sistema **SCADA** en un tanque del mundo real, dado que la naturaleza de los componentes como los sensores ultrasónicos que están orientados al prototipado y prueba y no está recomendado para su uso rudo o industrial

Capítulo 2. Estado del arte

2.1 Evolución de los sistemas de automatización

A lo largo de la historia los sistemas **SCADA** han ido evolucionando conforme a las necesidades de automatización del ser humano.

Alrededor de los años sesenta la tendencia en automatización era la de que cada fabricante debía resolver sus problemas de control por sí solo. Quien se encontraba ante un problema de atomización desarrollaba un elemento electrónico específico para solventarlo. En los años setenta aparece la nueva generación de autómatas de la mano de fabricantes de equipos eléctricos como **Siemens**, **Square-D**, o **Allen-Bradley**, capaces de controlar grandes cantidades de entradas y salidas, ideales para industrias tales como la automoción, consecuencia de la evolución de la electrónica dada la reducción de los componentes, *lo que permitió realizar una disminución progresiva de tamaño, peso y coste en todos los niveles industriales de control*. En los años 80 se introdujeron los *micros **PLC*** que permitían realizar controles modulares que se adaptaban a las necesidades del momento e introducían *los sistemas de programación genéricos*. (Rodríguez Penin, Sistemas SCADA - Notas de diseño, Normática, Seguridad y comunicaciones industriales, primeros pasos con InTouch, 2007)

Con la introducción de las computadoras dentro del escenario, la automatización se convirtió más amigable para el operador, a pesar de que inicialmente el uso de la computadora fue restringido para el almacenamiento de datos y el cambio de punto de ajuste para controladores analógicos. Las computadoras digitales tempranas tenían serias desventajas tal como una mínima memoria, pobre confiabilidad y programación escrita en lenguaje máquina. (Thomas & McDonald, 2015)

Una de las necesidades de los sistemas **SCADA** es de mostrar o reflejar la información que está maquinando en tiempo real para hacerle saber al operador que tarea está realizando y como está funcionando, por tal razón surgió la necesidad de controlar y ver a distancias sistemas de control industrial, por lo que como dice (Rodríguez Penin, Sistemas SCADA - Notas de diseño, Normática, Seguridad y comunicaciones industriales, primeros pasos con InTouch, 2007) la aparición de la informática permitió realizar este tipo de control de manera más sencilla, de igual manera la evolución de los sistemas operativos ha incrementado las posibilidades de manipular, controlar y supervisar los sistemas **SCADA**.

Dos desarrollos importantes llevaron a la llegada del control distribuido: los avances en circuitos integrados y en sistemas de comunicación. Los sistemas de control distribuidos eran de estructura modular, con menús preprogramados, con una amplia selección de algoritmos de control para la ejecución. La autopista de datos se hizo posible con la introducción de nuevas técnicas de comunicación y medios. La redundancia a cualquier nivel fue posible, debido a la disponibilidad de componentes a tasas más baratas, y las herramientas de diagnóstico extensas se convirtieron en parte de los sistemas de control de supervisión y adquisición de datos (SCADA). (Thomas & McDonald, 2015)

Como dice (Rodríguez Penin, Sistemas SCADA - Notas de diseño, Normática, Seguridad y comunicaciones industriales, primeros pasos con InTouch, 2007) con el nacimiento del internet en el mundo de las comunicaciones industriales ahora es posible conectarse con un sistema de control situado en cualquier lugar del mundo gracias a la tecnología *Web-Server*: un ordenador dotado de un explorador y la dirección *IP* del sistema que queremos visualizar serán suficientes.

2.2 Sistema SCADA

La automatización se utiliza en todo el mundo en una variedad de aplicaciones que van desde la industria del gas y el petróleo, la automatización del sistema de energía, la automatización de edificios, hasta la automatización de pequeñas unidades de fabricación. La terminología **SCADA** se usa generalmente cuando el proceso a controlar se extiende sobre un área geográfica amplia, como los sistemas de energía. Los sistemas **SCADA**, aunque son ampliamente utilizados por muchas industrias, están experimentando cambios drásticos. (Thomas & McDonald, 2015)

Se denomina *Scada* (Supervisory Control and Data Acquisition o Supervisión, Control y Adquisición de Datos) a cualquier software que permita el acceso a datos remotos de un proceso y permita, utilizando las herramientas de comunicación necesarias en cada caso, el control del mismo. No se trata de un sistema de control, sino de una utilidad software de monitorización o supervisión, que realiza la tarea de interface entre los niveles de control que son los Controladores Lógico Programable (**PLC)** y los de gestión, a un nivel superior. (Rodríguez Penin, Sistemas SCADA - Notas de diseño, Normática, Seguridad y comunicaciones industriales, primeros pasos con InTouch, 2007)

Los sistemas *SCADA* se desarrollan por la necesidad de monitorear, supervisar y controlar, por lo que para que pueda funcionar perfectamente y anclarse a las necesidades de la empresa debe cumplir los siguientes requerimientos:

- Funcionalidad completa de manejo y visualización en un sistema operativo y sobre cualquier **PC** estándar.
- Arquitectura abierta que permita combinaciones con aplicaciones estándar y de usuario, que permitan a los integradores crear soluciones de mando y supervisión optimizadas

(comunicaciones con bases de datos, lenguaje integrado como VB, C# o C, acceso a funciones y datos mediante *Application Programming Interface* **API**)

- Fácilmente configurable y escalable, debe ser capaz de crecer o adaptarse según las necesidades cambiantes de la empresa.
- Comunicaciones flexibles para poder comunicarse con total facilidad y de forma transparente al usuario con el equipo de planta y el resto de la empresa.

Entre los objetivos de por qué desarrollar un sistema SCADA principalmente como una herramienta de supervisión y de mando se encuentra lo siguientes:

- Económico
- Accesible
- Mantenible
- Ergonómico
- Gestionable
- Flexible
- Conectividad

Como dice (Rodríguez Penin, Comunicaciones Industriales - Guía práctica, 2008) el sistema *SCADA*, comprende toda una serie de funciones y utilidades encaminadas a establecer una comunicación lo más clara posible entre el proceso y el operador, por lo que algunas de prestaciones destacan:

- Monitorización.
- Supervisión.
- Adquisición de datos.
- Mando.

- Seguridad en los accesos.

Por lo tanto, como dice (Thomas & McDonald, 2015), la implementación de SCADA implica dos actividades principales: adquisición de datos (monitoreo) de un proceso o equipo y el control de supervisión del proceso, lo que lleva a la automatización completa. La automatización completa de un proceso se puede lograr automatizando el monitoreo y las acciones de control. La automatización de la parte de monitoreo se traduce en un operador en una sala de control, pudiendo "*ver*" el proceso remoto en la consola del operador, completa con toda la información requerida mostrada y actualizada en los intervalos de tiempo apropiados.

Por lo que el conjunto de elementos de medición del equipo ayuda a adquirir los datos del campo, (por ejemplo instalado en un tanque), y el conjunto de elementos de control del equipo implementa los comandos de control en el campo, como lo demuestra el flujo en la Figura 2-1

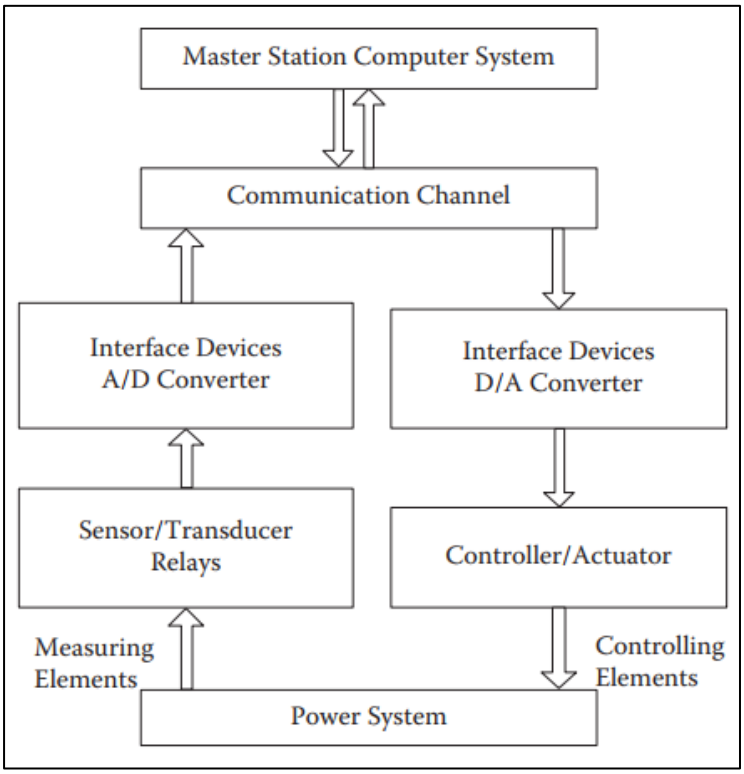

Figura 2-1. Flujo del proceso de monitoreo y control de un sistema SCADA.

2.3 Arquitectura de un sistema SCADA

Como dice (Rodríguez Penin, Sistemas SCADA - Notas de diseño, Normática, Seguridad y comunicaciones industriales, primeros pasos con InTouch, 2007) el desarrollo del ordenador personal ha permitido su implantación en todos los campos del conocimiento y a todos los niveles imaginables.

Las primeras incursiones en el campo de la automatización localizaban todo el control en el PC y tendían progresivamente a la distribución del control en planta. De esta manera, el sistema queda dividido en tres bloques principales, como se puede apreciar en la Figura 2-2, que son:

- Software de adquisición de datos y control (**SCADA**)
- Sistemas de adquisición y mando (sensores y actuadores).
- Sistema de interconexión.

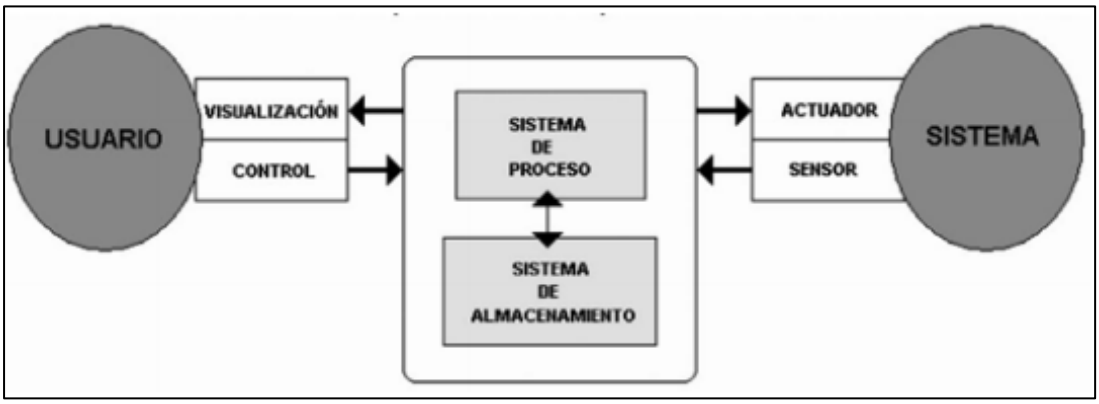

Figura 2-2. Arquitectura de sistema SCADA

El usuario, mediante herramientas de visualización y control, tiene acceso al Sistema de Control de Proceso, generalmente un ordenador donde reside la aplicación de control y supervisión (se trata de un sistema servidor). La comunicación entre estos dos sistemas se suele realizar a través de redes de comunicaciones corporativas (**Ethernet**).

El Sistema de Proceso capta el estado del Sistema a través de los elementos sensores e informa al usuario a través de las herramientas *Human Machine Interface* (**HMI**). Basándose en los

comandos ejecutados por el Usuario, el Sistema de Proceso inicia las acciones pertinentes para mantener el control del Sistema a través de los elementos actuadores.

La transmisión de los datos entre el Sistema de Proceso y los elementos de campo (sensores y actuadores) se lleva a cabo mediante los denominados buses de campo. La tendencia actual es englobar los sistemas de comunicación en una base común, como Ethernet Industrial. Toda la información generada durante la ejecución de las tareas de supervisión y control se almacena para disponer de los datos a *posteriori*.

Mediante el software de adquisición de datos y control, el mundo de las máquinas se integra directamente en la red empresarial, pasando a formar parte de los elementos que permitirán crear estrategias de empresa globales. Aparece el concepto de Fabricación Integral Informatizada (*Computer Integrated Manufacturing*). Un sistema Scada es una aplicación de software especialmente diseñada para funcionar sobre ordenadores en el control de producción que proporciona comunicación entre los dispositivos de campo, llamados también **RTU** (*Remote Terminal Units o Unidades Remotas*), donde se pueden encontrar elementos tales como controladores autónomos o autómatas programables, y un centro de control o Unidad Central (*MTU, Master Terminal Unit*), donde se controla el proceso de forma automática desde la pantalla de uno o varios ordenadores.

2.3.1 Interfaz Humano-Máquina.

Como dice (Thomas & McDonald, 2015) SCADA es una integración de varias tecnologías, pero generalmente cuatro son los componentes fundamentales

1. RTU.
2. Sistema de Comunicación
3. Estación maestra

4. Interfaz máquina-humano

La interfaz hombre-máquina del acrónimo en inglés *Human Machine Interface* (*HMI*) permite a un operador humano manipular y mostrar datos de las máquinas industriales, acepta los comandos del operador humano hacia las máquinas, a través de *HMI* un operador monitorea y responde a la información del sistema. El *HMI* de generación moderna también puede mostrar el estado actual del control industrial utilizando visualizaciones avanzadas en gráficos.

A menudo un operador humano controla un sistema *SCADA* a través de la interfaz hombre máquina, como se puede observar en la Figura 2-3 la interfaz *HMI* mostrando el estatus relevante para administrar el líquido almacenado en un tanque. (Nath, Stackowiak, & Romano, 2017)

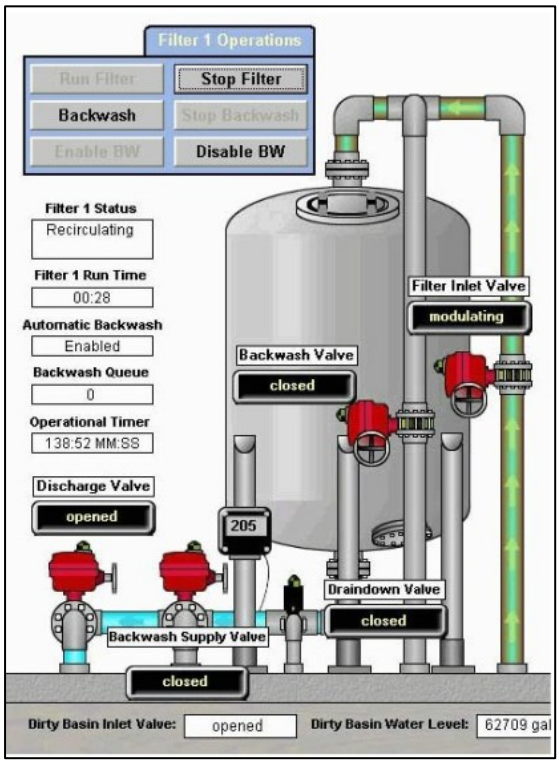

Figura 2-3. Software gráfico de Interfaz Humano-Máquina de un tanque.

Como dice (Rodríguez Penin, Sistemas SCADA - Notas de diseño, Normática, Seguridad y comunicaciones industriales, primeros pasos con InTouch, 2007) uno de los ejemplos claro que desde los años 90's es muy utilizado para desarrollar interfaces *HMI*, es Visual BASIC de

Microsoft que permite crear, con gran facilidad, controles gráficos e interfaces de usuarios gracias al conjunto de utilidades que soporta.

2.3.2 Comunicación de un sistema SCADA.

La comunicación de control de supervisión y adquisición de datos (SCADA) se refiere a los canales de comunicación empleados entre el equipo de campo y la estación maestra. El canal hace posible que un centro de control remoto acceda a los datos de campo *en tiempo real* para evaluar el estado del sistema. El canal de comunicación también transporta los comandos de control desde el centro de control al equipo apropiado en el campo para la implementación, para mantener el sistema de energía estable y seguro. (Thomas & McDonald, 2015)

2.3.3 Topología de distribución de sistema SCADA.

La topología para los sistemas SCADA. La topología define la disposición de los diferentes equipos alrededor del medio de transmisión de datos, determinando unas estructuras de red características:

- Redes Centralizadas (*Clusted systems*)

Todos los equipos están supeditados a un equipo central (*Host*) que controla todo el sistema. El host debe ser un equipo potente para gestionar el tráfico de datos con eficiencia.

- Redes distribuidas (*Distributed Systems*)

En este tipo de red, los equipos pueden ser máquinas sencillas que comparten las cargas de trabajo, los recursos y comunicaciones.

Las redes centralizadas se basan en la potencia de un único equipo y las redes distribuidas se basan en la distribución de los equipos menos potentes, por lo que las redes centralizadas se pueden configurar de las siguientes maneras:

- Anillo

- Estrella
- Bus
- Árbol
- Red

Como dice (Rodríguez Penin, Sistemas SCADA - Notas de diseño, Normática, Seguridad y comunicaciones industriales, primeros pasos con InTouch, 2007) la topología de un sistema Scada (su distribución física) variará adecuándose a las características de cada aplicación. Unos sistemas funcionarán bien en configuraciones de bus, otros en configuraciones de anillo, unos necesitaran equipos redundantes debido a sus características de proceso.

2.3.4 Protocolos de comunicación de un sistema SCADA.

Como lo especifica (Thomas & McDonald, 2015) el protocolo de comunicación define el formato en el que los datos se transfieren de un dispositivo a otro, por lo que la comunicación se vuelve más robusta, ambos dispositivos pueden decodificar los datos recibidos, y se pueden transferir múltiples datos. Cada protocolo define un conjunto de reglas mediante las cuales los datos que deben comunicarse están envueltos con direcciones de origen y destino y mecanismos de verificación de errores que se envían a través de los canales de comunicación, algunos de los protocolos de comunicación y tecnologías de comunicación de mayor auge y crecimiento en los sistemas SCADA son los siguientes:

- Protocolo Modbus
- Protocolo IEC 60870-5-101/103/104
- Protocolo de red distribuida 3 (DNP3)
- Protocolo de centro de inter-control (ICCP)
- Protocolo Ethernet

- Estándar IEC 61850 para comunicaciones de subestaciones eléctricas.
- Protocolo IEEE C37.118 Estándar *sincrofasor*
- Tecnologías para la automatización del hogar:
    - ZigBee
    - Dispositivos ZigBee
    - Wi-Fi

2.4 Aplicaciones de los sistemas SCADA

Los sistemas *SCADA* se utilizan ampliamente en un gran número de industrias, para su monitoreo y control. La industria del petróleo y el gas utiliza *SCADA* ampliamente para los campos petrolíferos, refinerías y estaciones de bombeo. Los grandes oleoductos y los gasoductos que recorren los océanos y continentes también son monitoreados por sistemas *SCADA* apropiados, donde se evalúa y controla el flujo, la presión, la temperatura, las fugas y otras características esenciales. Los sistemas de tratamiento de agua, distribución de agua y gestión de aguas residuales utilizan *SCADA* para monitorear y controlar los niveles de los tanques, las bombas de estaciones remotas y de elevación, y los procesos químicos involucrados. Los sistemas *SCADA* controlan la calefacción, ventilación y aire acondicionado de edificios como aeropuertos y grandes instalaciones de comunicación. El acero, el plástico, el papel y otras industrias manufactureras importantes utilizan el potencial de los sistemas *SCADA* para lograr productos más estandarizados y de calidad. La industria minera con *SCADA* integrado para los procesos de minería, como túneles, optimización de flujo de productos, logística de materiales, seguimiento de trabajadores y funciones de seguridad. (Thomas & McDonald, 2015)

2.4.1 Sistemas de control a distancia.

La forma en la cual las señales se intercambian entre el sistema a controlar y el sistema que controla, es denominado como *telemetría* (tele medida o medida a distancia), entendido como la transmisión a distancia de información sobre algún tipo de magnitud. Si además la presentación de los datos se realiza de forma inteligible ya nos proporciona la base para el *desarrollo de un sistema de control y monitorización a distancia* como se observa en la Figura 2-4. (Rodríguez Penin, Sistemas SCADA - Notas de diseño, Normática, Seguridad y comunicaciones industriales, primeros pasos con InTouch, 2007)

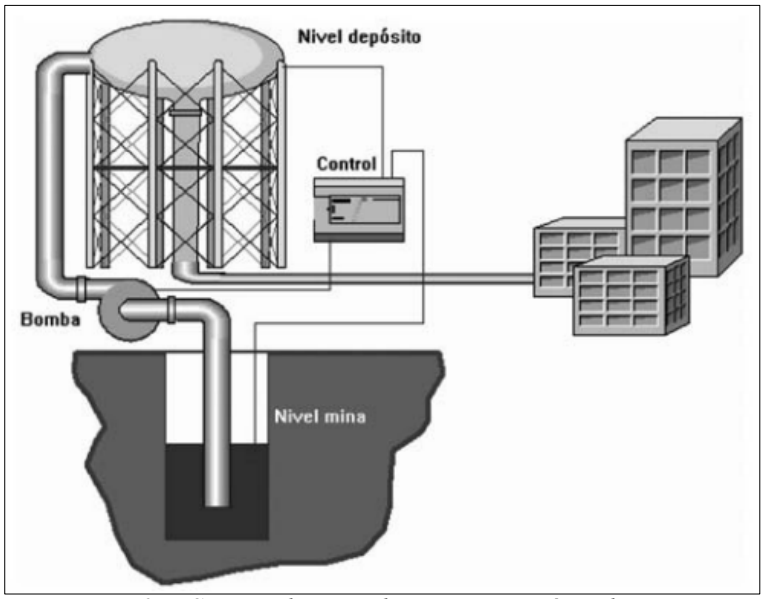

Figura 2-4. Sistema de control y monitorización a distancia.

La clasificación de los diferentes sistemas de intercambio de información, telemetría y monitorización podría hacerse basándose en el sistema de transmisión:

- Sistemas de marcación automática.

Utilizan las líneas telefónicas para transmitir en banda base (rango de voz). Los costes de comunicación son reducidos, puesto que las comunicaciones suelen ser puntuales. Por ejemplo, *un equipo de monitorización de nivel situado en un depósito de agua de una población bastará con*

que transmita el valor de nivel cada hora. Actualmente tiene bastante implantación la comunicación vía SMS (*Shor*t *Message Service*) o comunicación mediante caracteres ASCII (protocolo *Modbus*).

- Sistemas dedicados

Aquellos que tienen una línea de comunicación permanente con la central, se realizan mediante protocolos específicos de comunicación entre máquinas, permiten la monitorización continua de sistemas remotos y son muy rápidos en la captación de datos y en la emisión de comandos de control y configuración.

- Sistemas de canales multiplexados

A la técnica que permite enviar varias señales sobre un único canal de comunicación se denomina *multiplexado*; agrupa las señales de entrada en un único mensaje y en el otro extremo de la línea de transmisión estas señales son regeneradas, exactamente igual que si hubieran ido por cables dedicados.

2.5 Casos de estudio relacionados al proyecto de investigación

2.5.1 Detección de fugas en la tubería de la red principal del sistema de agua potable de la junta administradora de agua potable Sumak - Yaku - Araque – Otavalo.

En el artículo realizado en la **universidad técnica del norte** (**UTN**), en Ibarra, Ecuador, desarrollaron un tema que lleva por nombre detección de fugas en la tubería de la red principal del sistema de agua potable de la junta administradora de agua potable Suma - Yaku - Araque - Otavalo elaborado en año 2013, por los autores **Henry Cervantes Tafur** y **Jorge Terán Benalcázar** dice que para detectar fugas en un tramo de tubería, se coloca un dispositivo detector de fugas por caída de presión, éste inicia el trabajo de detección en cuanto recibe la orden del sistema principal de control de bombeo, y hasta cuando éste lo determine. Durante la detección, el dispositivo toma

datos de presión y la compara con el punto de control preestablecido, si es menor durante 20 segundos, se activa una bandera (alarma) con la cual se tomarán las respectivas acciones, por lo que para su elaboración requirieron la ***utilización de microcontroladores***, ***sensores de presión de 4-20 mA***, amplificadores para el tratamiento de señal, ***convertidor analógico-digital*** y un *generador fotovoltaico*, para la elaboración y diseño de las tarjetas electrónicas que integra toda la solución se utilizó el software ***PCB Wizard***, como prueba y resultado se obtuvieron dos debido a que se realizó una primera prueba en la estación dos y en la estación tres, por lo que para la primera fue que durante siete días se toman datos de presión mientras está en marcha el proceso de bombeo y se nota que la variación es aproximadamente a las 6:55 y es debido a que se debe a la manipulación de las válvulas para cambiar de dirección de flujo de agua hacia otro tanque, para evitar esa variación, por lo que basado en los datos obtenidos, la presión más baja es 31:00 **PSI**, y se vuelve a establecer el punto de control en 30 PSI, si el dispositivo detecta presiones menores a 30 **PSI** constantes durante 15 segundos seguidos, la alarma se activa en vista de que existe una posible fuga, para el resultado de la estación tres se dice que debido a que el valor de la señal de presión obtenida en esta estación es bajo, se tomaron muestras de presión durante y después del bombeo, para poder definir el valor del punto de control sin errores, durante el bombeo el valor promedio de presión es de 8 **PSI** y el valor después del bombeo baja a 6 **PSI** y disminuye con el tiempo, por lo que se decide establecer el punto de control en 7 **PSI**. Se concluye en que este proyecto puede servir como punto de referencia para la elaboración de prototipo electrónico, dado que para su elaboración se utilizaron componentes como *microcontroladores, sensores de presión y convertidores analógico-digital, así como también para desarrollar una placa PCB utilizando el programa de diseño* ***PCB Wizard*** y proporciona una idea de distribución de componentes por el tablero industrial propuesto.

2.5.2 Software localiza en tiempo real fugas en ductos de agua, petróleo o gas.

En la revista obtenida a través de internet, de fecha 20 de Abril de 2015, que lleva por título un software localiza en tiempo real fugas en ductos de agua, petróleo o gas, cuyo autor es **NCyT Amazing – Noticas de la ciencia y la tecnología – divulgando la ciencia por internet desde 1997,** y dice que es frecuente que las redes de distribución de agua, gas o petróleo tengan fugas en los tanques de almacenamiento, fallas de bombeo o existan tomas clandestinas. Con el propósito de evitar pérdidas económicas debido a estas causas, **Cristina Verde Rodarte**, investigadora del Instituto de Ingeniería de la UNAM, en México, diseño un vigilante virtual que detecta inmediatamente anomalías en cualquier tipo de ducto. El software llamado **VIVIUNAM** *realiza en tiempo real deducciones lógicas*, lo que permite identificar el tipo de falla y llegar hasta la raíz del problema y con ello se evita la pérdida de tiempo, al cavar o buscar manualmente el problema a lo largo de toda la tubería, comento la investigadora, quien también es miembro de la Academia de Ingeniería.

El *vigilante funciona con un algoritmo*, el cual a través de las leyes de la física y la aplicación de un modelo matemático de mecánica de fluidos, calcula una serie de datos que indican cuál es el comportamiento de los ductos de gas, agua o petróleo en condiciones normales de operación. Estos, a su vez *se comparan con el registro de mediciones de presión que la propia tubería genera y de la diferencia entre ellos se obtienen los síntomas de la presencia de fugas.* Cuando los resultados de *modelo matemático* **no coinciden** con las mediciones registradas de forma automática, es porque existe un **error** o **evento anormal**, y a partir de ello se buscan posibles escenarios; por ejemplo que el sensor de presión se desconectó, existe una fuga, toma clandestina o en general existe un disturbio que altera el comportamiento de los fluidos del ducto.

Por lo que consideramos como alternativa un ***algoritmo para el diagnóstico de fugas de agua*** utilizando para ellos una serie de sensores de presión que interactuarán en conjunto para arrojar un resultado lo más cercano a la detección de una fuga de agua en una tubería, aunque cabe hacer mención que en primer término, se tendría que hacer un tendido de sensores en los tanques, antes de llegar al punto del desarrollo del algoritmo.

2.5.3 Sistema monitoreo y cuantificación de flujos de lodo basado en una red inalámbrica de sensores.

En el artículo sistema para el monitoreo y cuantificación de flujos de lodo basado en una red inalámbrica de sensores, elaborado por los autores **Marcillo P.*; Bernal I.**; Macías C.*****, de la **Escuela Politécnica Nacional de Quito, Ecuador** y de la universidad **Pontificia Universidad Católica del Ecuador** y publicado en el mes de junio del año 2015, consultado a través de **ResearchGate**, en él se presenta el diseño e implementación de una red inalámbrica de sensores (RIS) que permitirá el monitoreo y cuantificación de flujos de lodo producidos por erupciones volcánicas. Esta red se integra en un sistema de monitoreo y cuantificación de flujos. El sistema ha sido concebido como un conjunto de dispositivos como módulos inalámbricos basados en **IEEE 802.15.4**, sensores de alta precisión, cámaras de video de alta resolución e iluminadores infrarrojos; infraestructura de comunicaciones; y, módulos de software que permitan visualizar los datos recolectados por los sensores, de acuerdo a lo señalado las **WSN** (Wireless Sensor Networks) tienen el potencial de beneficiar de forma sustantiva los estudios de actividad volcánica, los vulcanólogos frecuentemente utilizan arreglos de sensores interconectados con cables, las **WSN** tienen bajos requerimientos de energía, integrando una modesta cantidad de memoria, **CPU**, y comunicaciones inalámbricas pueden desempeñar un importante rol en el monitoreo volcánico, además este tipo de redes son más fáciles de desplegar y pueden integrar un mayor número de

sensores y una mayor cobertura, en este artículo se presenta el diseño e implementación de los nodos de una **WSN** con sensores especializados orientados al monitoreo y cuantificaciones de flujos de lodo, así como de la **WSN** propiamente dicha, por lo que también se desarrolló un sistema denominado sistema de monitoreo y cuantificación de flujos (**SMYCF**), este software hace uso de una red de acceso (RA) y de la red actual de microondas (**RM**) del IG para transportar la información generada por **WSN** hacia el centro de datos del IG, para el desarrollo de la propuesta se utilizaron las herramientas de la plataforma **imote2** desarrollada por **CrossbowTechnology Inc.** fue seleccionada por las características de su hardware, así como por su disponibilidad de trabajar con **TinyOS**, **.NET Micro Framework**, **Linux** y **SOS.** Para la parte de los sensores se utilizó un radar **réflex VG7**, un geófono **L-10AR**, una cámara de video de alta resolución, un iluminador infrarrojo, **GPS**, sensor de temperatura, sensor de humedad, por lo que en sus resultados explican que el funcionamiento del sistema requirió de pruebas realizadas a cada uno de los módulos que integran la **sRIS** y de pruebas en conjunto. Se establecieron valores por defecto o valores experimentales para los programas que fueron cargados en cada uno de los módulos que, en su momento, deberán ser analizados y modificados por los científicos y técnicos a cargo. De la misma manera se establecieron parámetros para la cámara de video que integra el Módulo de Visualización. Estos parámetros fueron configurados en los ***scripts bash*** creados para cambiar la frecuencia de transmisión de imágenes. Se podría citar, por ejemplo, los valores umbrales establecidos para los parámetros de temperatura, humedad y altura; de estos valores umbrales dependerá la generación de alertas en el sistema. Las pruebas al Módulo Sísmico consistieron en reemplazar el geófono por un generador de funciones, y así generar señales sinusoidales de diferente frecuencia y amplitud. Las pruebas al Módulo de Visualización consistieron en generar alertas por medio del Módulo de Sensores. Los mensajes de alerta fueron retransmitidos hacia el

Módulo de Visualización y se generaron los pulsos esperados que por un lado encendieron y apagaron el iluminador infrarrojo y por otro permitieron el cambio de la frecuencia de transmisión de imágenes. En lo que respecta a las imágenes transmitidas durante el estado de alerta, se usó el servidor **FTP** del **IG** para almacenarlas. Por último, se realizó una prueba de funcionamiento en conjunto en la que intervinieron cada uno de los módulos que integran la **sRIS**. La prueba fue enfocada especialmente al funcionamiento del Módulo Base y su interacción con los demás módulos

Por lo que considero que el artículo anteriormente descrito, me da una idea para realizar una red de sensores (**Wireless Sensor Networks**) que midan la presión de los tubos de agua y que envíen señal a través de alguna placa de desarrollado para procesar en tiempo real las lecturas, la ventaja que me nos da esto, es que para cada placa podemos instalar un sistema operativo lo que nos permite agregar más de un sensor y por la interconexión que ofrece se puede comunicar con otras placas o con un servidor para obtener y manipular los datos de sensores y actuadores.

2.5.4 Monitoreo de sensores en aplicaciones web embebidas.

En el artículo que lleva por nombre monitoreo de sensores en aplicaciones web embebidas, de los autores **Andrés Felipe Gómez Rivera, Fabián Velásquez Clavijo** y **MSC. Andrés Fernando Jiménez López**, describen el proceso de diseño, implementación y evaluación de un sistema de monitoreo en tiempo real de sensores en una aplicación web embebida utilizando como plataforma hardware un ordenador de placa reducida basado en **SoC ARM**, dada la disponibilidad de plataformas de desarrollo embebidas de altas prestaciones, tales como los ordenadores de placa única y dimensiones reducidas han dado paso a una nueva tendencia de prototipado electrónico, aprovechando el bajo costo que supone su adquisición y la capacidad de trabajar sobre sistemas operativos como Linux, utilizando lenguajes de alto nivel como Python para controlar hardware

de una manera más sencilla en comparación con los sistemas basados en lógica reconfigurable como FPGA y como microcontroladores, las herramientas que se utilizaron para desarrollar el proyecto fueron una tarjeta de **SensorHub BoosterPack de Texas Instruments**, una **Raspberry Pi**, para la parte del diseño de la interfaz web que muestre la información en tiempo real se utilizó **PHP** alojado en el servidor de la Raspberry Pi mediante **APACHE**, el uso de **HTML** y **jQuery** para acceder a la información de los sensores y visualizarla en dos gráficos en tiempo real mediante **AJAX.** También se utilizó la tecnología de HTML5 para trabajar con **WebSocket** que facilitan la interacción entre programas, haciendo que el diseño de un sistema sea más modular y fácil de mantener que si se realizara en un solo entorno.

Por lo que consideramos que este articulo nos puede servir como base para cómo desarrollar aplicaciones web embebidas para mostrar la información en tiempo real mediante la implementación de un servidor **APACHE** y programado en **PHP** sobre una microcomputadora **Raspberry P**I a la cual se le conectan sensores a través de los ***WebSockets*** para detectar la presión del agua y diagnosticar una fuga o medir el nivel de un contenedor de agua.

2.5.5 Diseño de una interfaz para la detección de fugas de agua.

En el artículo que lleva por nombre diseño de una interfaz para la detección de fugas de agua, elaborado por los autores **Luis Antonio Gama Moreno, Marco Antonio Sánchez Rodríguez y Christopher de Jesús Ochoa Franco**, publicado por la **Revista digital universitaria de la DGSCA-UNAM**, el día 1 de febrero de 2010 en el volumen 11, número 2, dicen que el desperdicio de agua representar una fuerte problemática para las comunidades, organizaciones y gobiernos, ya que su mal uso, la mala distribución y los eventos naturales como; sequias, calentamiento global, contribuyen a la escases de este vital liquido, por lo que ellos diseñan una interfaz para la detección de fugas de agua en una casa-hogar denominado **aplicación para la detección de fugas de agua**

(AIDA), la cual se compone de dos elementos principales, el primero es un conjunto de dispositivos electrónicos instalados en los depósitos de agua los cuales permitirán medir los flujos y cantidades del líquido, el segundo componentes es la aplicación de software que llevara el control de los patrones de consumo y así detectar posibles fugas de agua enviando alertas de mensajes de texto corto mediante la red de telefonía **GSM**..

AIDA es una interfaz que ayudará a que los consumos de agua potable sean más racionados, y también mostrará estadísticas de estos, permitiendo la detección de fugas en base a los patrones de consumo almacenados, por lo que su diseño se encuentra integrado por **un módulo de información de agua disponible (MD)** que es el encargado de obtener el volumen registrado en litros de un contenedor de agua; a través de un sensor de presión bajo el principio de manómetros U (mecánica de fluidos), la presión que ejerce la atmosfera sobre la superficie del agua, conlleva a tener una presión en el fondo del tinaco en proporción a la masa de su contenido, esta presión es encapsulada en un conducto sellado y lleno de aire (cámara de aire), donde el sensor detectará la presión que el aire ejerce con respecto al vacío y el nivel de agua, un **microcontrolador PIC** se encargará de calcular el volumen mediante una ecuación programada, y enviará la información hacia el módulo de registro de información, también cuenta con **un módulo de medidor de flujo de agua** el cual se encargará de medir la cantidad de agua que pasa a través de un determinado punto de una tubería, con el fin de conocer la cantidad de gasto de agua que fluye en determinado tiempo, con la ayuda de un **flujómetro** ubicado estratégicamente en la red de tubería. Por consiguiente también cuenta con **un módulo maestro (MM)** el cual recibirá toda la información proveniente de los módulos medidores de flujo y detector de nivel de agua para ser procesados y respaldados en un servidor central el cual está encargado de crear las estadísticas de consumo, después **un módulo de alertas** el cual consiste de un sistema que se enlaza con un dispositivo

móvil vía **SMS** para el envío de alertas y por último **un módulo de gráficas** el cual se encuentra construido por medio de una **API** basada en la plataforma **JAVA** con la finalidad de mostrar al usuario, la forma en que se consume o gasta el agua de su casa hogar, para los **resultados obtenidos y pruebas** los resultados son almacenados en un servidor central. Las lecturas obtenidas por los sensores son almacenadas en una base de datos, que posteriormente son enviados al módulo generador de gráficas de consumo, por lo que la prueba fue emular una fuga de agua, la cual se le abrió a la llave de agua dentro de la casa dejando correr el agua más de lo que se usaría normalmente, el sistema detecta esta anomalía e inmediatamente manda una alerta al móvil configurado.

El artículo anteriormente desarrollado nos aporta conocimiento de cómo se puede desarrollar una interfaz electrónica que utiliza software de usuario para visualizar graficas que muestren la detección de las fugas, por lo que puedo seguir su analogía, que es muy parecido a lo que se pretende realizar en el proyecto de tesis; como el tendido de sensores para capturar la información de los niveles de tanques, y que los datos se concentren en un servidor para ser procesados por una aplicación que muestra información de tiempo real sobre el estado de los tanques de agua.

2.5.6 Desarrollo de una red de monitoreo por sensores remotos de la calidad de agua.

En el artículo publicado por la revista **tecnología en marcha vol. 18 no. 2, edición especial,** que lleva por título desarrollo de una red de monitoreo por sensores remotos de la calidad de agua, elaborado por los autores **Adolfo Chávez Campos, Freddy Araya Rodríguez, Adolfo Chaves Jiménez y Víctor Yépez García**, en donde se explica que a partir de la integración de tecnologías, se diseña una red de sensores cuyos datos son transferidos de manera remota a una estación central a partir de un sistema inalámbrico de transmisión, para el monitoreo de la calidad del agua, debido a los problemas de supervisión de la calidad del agua, que se encuentran sometidos a graves

dificultades en todo el mundo, pues en general los gobiernos se ven obligados a reducir los presupuestos, recortar el personal y establecer nuevas prioridades, los gobiernos cada vez se resisten a asignar fondos para el seguimiento, paradójicamente, nunca ha sido mayor la necesidad de contar con información fiable sobre la calidad del agua, por lo que se considera desarrollar un proyecto que consiste en implementar una red de monitoreo por sensores remotos para la calidad del agua potable, en la **zona huetar norte de costa rica**, a fin de mejorar las bases de datos sobre la calidad de agua, que permitan tomar las decisiones adecuadas respecto a la gestión de dicho recurso, la tecnología por implementar debe tener la característica de ser; innovadora y sostenible desde el punto de vista ecológico, social y económico, utilizando para su elaboración un sistema de recolección de datos **LabPro** de la marca **Vernier**, este sistema tiene capacidad para manejo de cuatro sensores analógicos y dos sensores digitales, está diseñado para recolectar los datos y enviarlos por puerto serie (RS232), este protocolo abierto es el que permite la integración de las tecnologías de transmisión y de manejo de los datos en un computador, además se implementa en primaria instancia un dispositivo S-100 de **HW-Group**, este sistema convierte la señal RS232 en una señal Ethernet, lo cual permite su manejo adecuado a través de la tecnología IP, la transmisión se realiza mediante el protocolo **802.11b** lo que permite la recepción de los datos en cualquier receptor inalámbrico de la tecnología **WLAN**, también se desarrolló un sistema de recolección de datos el cual se diseñó en **Visual Basic**, dicho sistema monitoreo genera un archivo con los datos históricos de monitoreo de tal forma que puedan ser analizados en **Excel**. Como resultados obtenidos se tiene en primera instancia la instalación de una red de monitoreo automatizada instalada en producción por lo que el resultado esperado es que funcionará satisfactoriamente.

Como aportación al presente proyecto de investigación se puede concluir que puede servir como fundamento para la elaboración del prototipo utilizar el sistema de captura de datos **Vernier**,

así como también, el lenguaje **Visual Studio** para desarrollar el sistema software y obtener de alguna manera a través del protocolo **RS232** la información obtenida de los sensores para posteriormente procesarla.

Capítulo 3. Marco teórico

3.1 Implicaciones teóricas sobre diseño y desarrollo de sistemas

3.1.1 Análisis y diseño orientados a objetos (A/DOO)

Como afirma (Deitel, 2005) el diseño orientado a objetos (DOO) modela el software en términos similares a los que utilizan las personas para modelar los objetos del mundo real. El diseño orientado a objetos ofrece una manera natural e intuitiva de ver el proceso de diseño del **software**; a saber, se modelan los objetos por sus atributos, comportamientos e interrelaciones, de igual forma que describimos los objetos del mundo real. Los lenguajes como ***C#*** están ***orientados a objetos***. La programación en un lenguaje de este tipo se llama ***Programación Orientada a objetos (POO)***, y permite a los programadores de computadoras implementar en forma conveniente un ***diseño orientado a objetos*** como un sistema de **software** funcional.

Por lo que (Deitel, 2005) sugiere, que para crear las mejores soluciones es conveniente que se siga un proceso detallado para ***analizar*** los ***requerimientos*** de un proyecto (es decir ***qué*** es lo que hará el sistema) y desarrollar un ***diseño*** que lo satisfaga (es decir, debe decidir ***cómo*** lo hará su sistema). Si este proceso implica analizar y diseñar su sistema desde un punto de vista orientado a los objetos, se llamada ***análisis y diseño orientados a objetos (A/DOO)***, **A/DOO** es el término genérico para el proceso de analizar un problema y desarrollar un método para resolverlo. Por lo que hay un solo lenguaje gráfico para comunicar los resultados de cualquier proceso de **A/DOO**, este lenguaje conocido como Lenguaje Unificado de Modelado (**UML**), se desarrolló a mediados de la década de 1990 bajo la dirección inicial de tres metodologías de software: *Grady Booch*, *James Rumbaugh* e *Ivar Jacobson*.

3.1.2 Lenguaje Unificado de Modelado (UML)

Como define (Deitel, 2005) el **Lenguaje Unificado de Modelado (UML)** es el esquema de representación gráfica más utilizado para modelar sistemas orientados a objetos. Sin duda ha unificado los diversos esquemas de notación populares. Las personas que diseñan sistemas utilizan el lenguaje (en forma de diagramas) para modelar sus sistemas. Una característica atractiva del **UML** es su flexibilidad. El **UML** es **extensible** (es decir, capaz de mejorar mediante nuevas características) e independiente de cualquier proceso de **A/DOO** particular. Los modeladores de **UML** tienen la libertad de utilizar varios procesos al diseñar sistemas, pero todos los desarrolladores pueden ahora expresar sus diseños mediante un conjunto estándar de notaciones gráficas.

La finalidad de **UML** es describir gráficamente una serie de comportamientos del sistema que puede interpretar un programador para desarrollar el sistema, éstos modelos son como "los planos arquitectónicos en los que se desarrollará el software" como dice (Deitel, 2005), los modelos que atienden a las necesidades de desarrollo de software y sistemas embebidos son: modelo de clase de dominio, modelo de caso de uso, modelo de secuencia, modelo de interacción, modelo de máquinas de estado y modelo de negocios como dice (Haugen, Moller-Pedersen, & Weigert, 2006)

3.1.3 La ingeniería de software

La ingeniería de **software** concentra las relaciones del análisis y diseño orientado a objetos con la programación orientada a objetos (POO) y Lenguaje Unificado de Modelado (UML) de tal manera que las combina para desarrollar soluciones de **software**, para ello inicialmente se presenta **un documento de requerimientos** el cual especifica el propósito general del sistema y *qué* es lo que debe de hacer. Tal documento es el resultado de un proceso detallado de **recopilación de**

requerimientos, de tal manera que un **analista** utilizaría la información recopilada para compilar una lista **requerimientos del sistema** para guiar a los diseñadores de sistemas.

Como dice (Deitel, 2005) el proceso de recopilación de requerimientos es una tarea clave de la primera etapa del ciclo de vida del **software**. El **ciclo de vida del software** especifica las etapas a través de las cuales el **software** evoluciona desde el tiempo en que fue concebido hasta el tiempo en que se retira de su uso. Por lo general, estas etapas incluyen el **análisis, diseño, implementación, prueba** y **depuración, despliegue, mantenimiento** y **retiro**. Existen varios modelos de ciclo de vida del software, cada uno con sus propias preferencias y especificaciones con respecto a cuándo y qué tan a menudo debe llevar a cabo los ingenieros de **software** las diversas etapas. Los **modelos de cascada** realizan cada etapa una vez en sucesión, mientras que los **modelos iterativos** pueden repetir una o más etapas varias veces a lo largo del ciclo de vida de un producto.

Un producto por lo regular es un sistema y un sistema es un conjunto de componentes que interactúan entre sí para resolver un problema. La **estructura del sistema** describe los objetos del sistema y sus interrelaciones. El **comportamiento del sistema** describe la manera en que cambia el sistema a medida que sus objetos interactúan entre sí. Todo sistema tiene tanto estructura como comportamiento; los diseñadores deben especificar ambos. Existen diversos tipos de estructuras y comportamientos de un sistema. UML 2 especifica 13 tipos de diagramas para documentar los modelos de un sistema, cada tipo de diagrama modela una característica distinta de la estructura o del comportamiento de un sistema.

3.1.4 Patrones de diseño de software

El autor (McLean Hall, 2014) dice que los patrones en el desarrollo de software orientado a objetos es un esfuerzo relativamente nuevo. En las últimas décadas, algunas colaboraciones

repetibles entre clases e interfaces se han identificado y codificado como ***patrones***. Hay muchos patrones de desarrollo de software, cada uno proporciona una solución genérica que puede reutilizarse en un dominio de problema específico. Algunos patrones se pueden usar en conjunto para producir soluciones elegantes a problemas complejos. Por supuesto, no todos los patrones son aplicables todo el tiempo, y se necesita experiencia y práctica para reconocer cuándo y dónde se pueden aplicar ciertos patrones. Algunos patrones no son tan benévolos. De hecho, son todo lo contrario. Se consideran anti-patrones, por lo que perjudican la adaptabilidad de su código y deben evitarse.

Hay varios patrones de estratificación comunes entre los cuales elegir para cualquier proyecto. Se deben utilizar como una guía que se adaptará a los requisitos y restricciones específicos de su situación. El factor diferenciador entre los patrones de ***estratificación*** es simplemente ***el número de capas utilizadas***. Por lo regular se comienza con una arquitectura simple formada por solo dos capas, luego inserta una tercera capa en el medio y, finalmente, extrapola a un número arbitrario de capas.

El número de capas requeridas se correlaciona con la complejidad de la solución; la complejidad de la solución se correlaciona con la complejidad del problema. Por lo tanto, cuanto más complejo sea el problema, más inclinado estará a invertir en una arquitectura de varias capas. La complejidad, en este caso, se mide por factores como: las limitaciones de tiempo impuestas al proyecto, su longevidad requerida, la frecuencia con la que los requisitos cambian y la importancia que el equipo de desarrollo asigna a los patrones y prácticas.

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software, en otras palabras, brindan una solución ya probada y documentada a problemas de

desarrollo de software que están sujetos a contextos similares, los patrones se pueden dividir en las siguientes categorías:[2]

- **Patrones creacionales:** inicialización y configuración de objetos
- **Patrones estructurales**: separan la interfaz de la implementación, se ocupan de cómo las clases y objetos se agrupan para formar estructuras más grandes.
- **Patrones de comportamiento**: más que describir objetos o clases, describen la comunicación entre ellos.

3.1.4.1 El patrón de diseño Model View View-Model (MVVM)

Como dice el autor (Garofalo, 2011), el patrón *Model View ViewModel* (**MVVM**) fue presentado por **John Gossman** (arquitecto de software en Microsoft para Windows Presentation Foundation y tecnologías Silverlight) en el año 2005, describe que **MVVM** es una especialización del modelo Presentation Model (**PM**) que fue presentado en 2004 por **Martin Fowler**.

Uno de los objetivos principales del patrón de PM es separar y abstraer la Vista -la interfaz de usuario (**IU**) visible -de la lógica de presentación para hacer que la **IU** sea comprobable. Los objetivos adicionales pueden incluir la elaboración de lógica de presentación reutilizable para diferentes interfaces de usuario (**IU**) y diferentes tecnologías de **IU**, reduciendo el acoplamiento entre la **IU** y otro código, y permitiendo a los diseñadores de **IU** trabajar de una manera más independiente. *MVVM* es una interpretación especializada del patrón PM diseñado para satisfacer los requisitos de *Windows Presentation Foundation* (WPF) y Silverlight.

Estructuralmente, una aplicación MVVM consta principalmente de tres componentes principales: el modelo, la vista y el modelo de vista, como se puede apreciar en la Figura 3-1 su interacción, a continuación se describe cada uno de sus componentes:

[2] (Tedeschi, 2010)

- **El modelo** es la entidad que representa el concepto de negocio; puede ser cualquier cosa, desde una simple entidad cliente hasta una compleja entidad comercial.
- **La vista** es el control gráfico o conjunto de controles responsable de mostrar los datos del modelo en la pantalla. Una vista puede ser una ventana de *WPF*, una página de *Silverlight* o simplemente un control de plantilla de datos *XAML*.
- **ViewModel** contiene la lógica de la interfaz de usuario, los comandos, los eventos y una referencia al modelo. En MVVM, *ViewModel* no se encarga de actualizar los datos que se muestran en la interfaz de usuario y esto es porque la View o *Vista* es un observador del *ViewModel*, así que tan pronto como el *ViewModel* cambie, la UI se actualiza a sí misma.

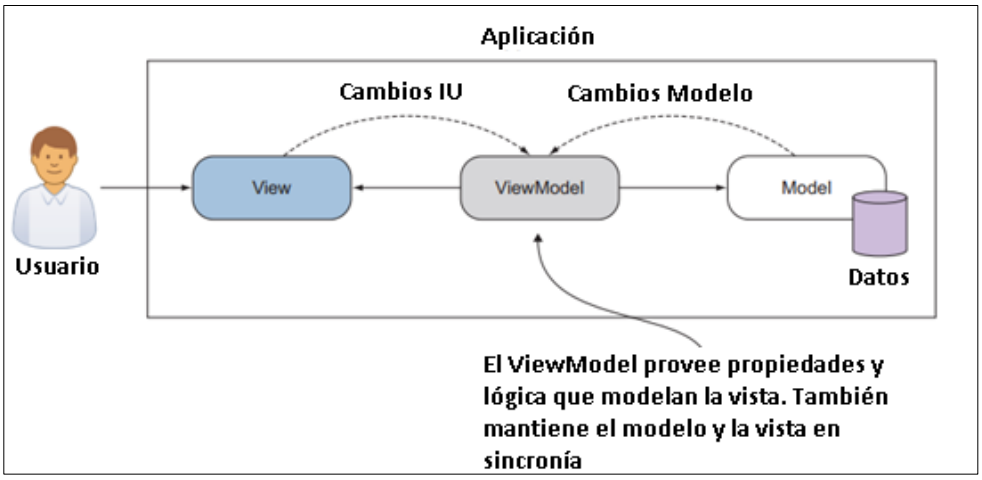

Figura 3-1. Interacción patrón MVVM.

3.2 Tecnologías de desarrollo de software utilizado

Como consideraciones teóricas para el diseño del software del sistema SCADA, se considera dividirlo en la parte **backend** que contempla la parte de acceso a datos y procesamiento en el servidor y la parte del **frontend** para la interfaz que se muestra al usuario con gráficos de tiempo real.

3.2.1 Tecnologías para el desarrollo backend

Parte del backend se considera utilizar el lenguaje C# porque es un lenguaje orientado a *cross-platform*, y multiparadigma, además de ser de acceso libre para construir y desarrollar sistemas, así como por su robustez en grandes sistemas que se encuentran en producción a nivel mundial.

3.2.1.1 El lenguaje C#

Como dice (Ferguson, Patterson, Beres, Boutquin, & Gupta, 2003) durante los últimos 20 años, C y C++ han sido los lenguajes elegidos para desarrollar aplicaciones comerciales y de negocios. Estos lenguajes proporcionan un altísimo grado de control al programador permitiéndole el uso de puntero y muchas funciones de bajo nivel. Sin embargo, cuando se comparan lenguajes, como Microsoft Visual Basic con C/C++, uno se da cuenta de que aunque C y C++ son lenguajes mucho más potentes, se necesita mucho más tiempo para desarrollar una aplicación con ellos. Muchos programadores de C/C++ han temido la idea de cambiar a lenguajes como Visual Basic porque podrían perder gran parte del control de bajo nivel al que estaban acostumbrados.

Lo que la comunidad de programadores necesitaba era un lenguaje que estuviera entre los dos. Un lenguaje que ayuda a desarrollar aplicaciones rápidas pero que también permitiese un gran control y un lenguaje que se integrase bien con el desarrollo de aplicaciones Web, *XML* y muchas de las tecnologías emergentes.

C# combina las mejores ideas de lenguajes como C, C++ y Java con las mejoras de productividad de .NET Framework de Microsoft y brinda una experiencia de codificación muy productiva tanto para los nuevos programadores como para los veteranos.

3.2.1.2 .ASP.NET Core

Como dice (Lock, 2018) el desarrollo de ASP.NET Core fue motivado por el deseo de crear un marco web con cuatro objetivos principales:

- Para ser ejecutado y desarrollado multiplataforma.
- Tener una arquitectura modular para facilitar el mantenimiento.
- Para ser desarrollado completamente como software de código abierto.
- Para ser aplicables a las tendencias actuales en el desarrollo web, como las aplicaciones del lado del cliente y la implementación en entornos de nube.

Para lograr todos estos objetivos, **Microsoft** necesitaba una plataforma que pudiera proporcionar bibliotecas subyacentes para crear objetos básicos como listas y diccionarios, y realizar, por ejemplo, operaciones de archivos simples. Hasta este punto, el desarrollo de ASP.NET siempre se había centrado y dependía de *.NET Framework* solo para *Windows*. Para ASP.NET Core, Microsoft creó una plataforma ligera que se ejecuta en *Windows*, *Linux* y *macOS* llamada .NET Core, como se muestra en la Figura 3-2 *.NET Core* comparte muchas de las mismas API que .NET Framework, pero es más pequeño y actualmente solo implementa un subconjunto de las características que proporciona .NET Framework, con el objetivo de proporcionar una implementación y un modelo de programación más simples. Es una plataforma completamente nueva, en lugar de una bifurcación de .NET Framework, aunque utiliza un código similar para muchas de sus API.

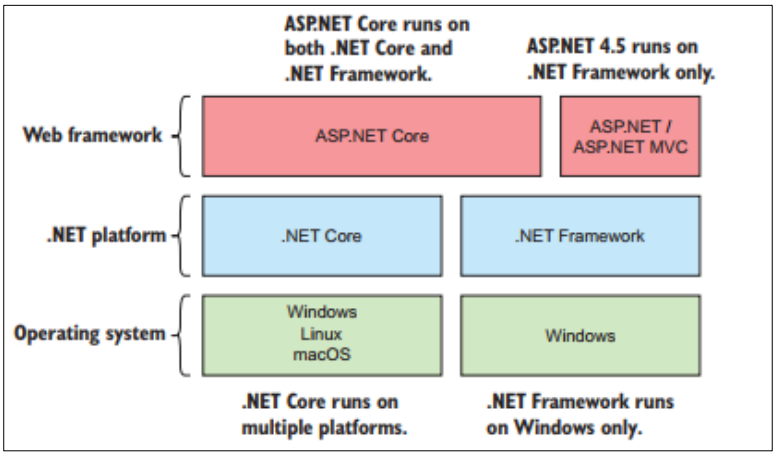

Figura 3-2. Arquitectura de ASP.NET Core.

3.2.1.3 Base de datos NoSQL MongoDB

Como dice (Sarasa, 2016) uno de los principales problemas hace referencia a la diferencia entre el modelo relacional y las estructuras de datos en memoria que algunos autores denominan «impedancia». Todas las operaciones en SQL consumen y retornan relaciones de acuerdo con el álgebra relacional. En particular, los valores en una ***tupla*** relacional tienen que ser simples (no pueden tener estructura tal como un registro anidado o una lista). Esta limitación no se da en las estructuras de datos en memoria, donde existe un conjunto de tipos de estructuras mucho más ricas que las relaciones. Como consecuencia, si se quiere usar una estructura de datos más compleja que la relación, hay que traducirla a una representación relacional para poderla almacenar en disco.

Otra limitación presente en las bases de datos relacionales es su integración en un clúster. Un clúster es un conjunto de máquinas que permite implementar soluciones para escalar una aplicación (escalado horizontal). Es una solución más resistente que disponer de una única máquina más potente (escalado vertical), dado que, cuando se produce un fallo en un clúster, la aplicación podrá continuar funcionando, proporcionando, así, una alta fiabilidad.

Debido a las limitaciones mencionadas con anterioridad, algunas empresas decidieron usar alternativas a las bases de datos relacionales. Este fue el caso de Google y Amazon, que desarrollaron sistemas de almacenamiento basado en el uso de clústeres: las *Big Tables* de **Google** y *Dynamo* de **Amazon**. Estos sistemas permitían gestionar grandes cantidades de datos en ambientes distribuidos y llevar a cabo su procesamiento, por lo que se consideran el origen de las denominadas «bases de datos *NoSQL*», sus principales ventajas son las siguientes:

- Productividad del desarrollo de aplicaciones. En los desarrollos de muchas aplicaciones se invierte un gran esfuerzo en realizar mapeos entre las estructuras de datos que se usan en memoria y el modelo relacional. Las bases de datos ***NoSQL*** proporcionan un modelo

de datos que encaja mejor con las necesidades de las aplicaciones simplificando así la interacción, lo que resulta en tener que codificar, depurar y evolucionar menos las aplicaciones.

- Datos a gran escala. Las organizaciones han encontrado muy valiosa la posibilidad de tener muchos datos y procesarlos rápidamente. En general, es caro (en el caso de ser posible) hacerlo con una base de datos relacional. La primera razón es que las bases de datos relacionales están diseñadas para ejecutarse en una única máquina, y por otro lado es mucho más barato ejecutar muchos datos y procesarlos si se encuentran cargados sobre clústeres de muchas máquinas pero más pequeñas y baratas. La mayoría de las bases de datos ***NoSQL*** están diseñadas para ejecutarse sobre clústeres, por lo que encajan mejor para estas situaciones.

Una de las aplicaciones orientadas al diseño y creación de base de datos ***NoSQL*** es ***MongoDB*** el cual es muy potente y fácil de trabajar, como dice (Chodorow, 2013), básicamente se integra de las siguientes características:

- Un documento es la unidad básica de datos para MongoDB y es aproximadamente equivalente a una fila en un sistema de administración de base de datos relacional (pero mucho más expresivo).
- De manera similar, una colección puede considerarse como una tabla con un esquema dinámico.
- Una sola instancia de MongoDB puede alojar múltiples bases de datos independientes, cada una de las cuales puede tener sus propias colecciones.
- Cada documento tiene una clave especial, "_id", que es única dentro de una colección.

- MongoDB viene con un *shell* de **JavaScript** simple pero poderoso, que es útil para la administración de instancias de MongoDB y la manipulación de datos.

3.2.2 Tecnologías para el desarrollo frontend

El *frontend* es la parte de tecnologías que se ejecutan del lado del cliente, esto quiere decir que un usuario ejecuta abre o instancia una página web en el navegador y el navegador *renderiza* el contenido de tal manera que interpreta el contenido y lo muestra al usuario, por lo tanto aquí se describen los lenguajes y tecnologías alrededor del desarrollo del *backend*.

3.2.2.1 Tecnologías de desarrollo

Las tecnologías del desarrollo que se ejecutan e interpretan en el navegador, comprende el lenguaje de marcado (***HTML***) o *Lenguaje de Marcado de Hipertexto*, que fue inventado por **Tim Berners Lee**, y el principio fundamental es trabajar con ***etiquetas*** que un navegador entiende y lo muestra según su semántica, por otro lado existe también el (**CSS**) o *Cascade Style Sheet* Hojas de estilo en cascada que nos ayuda dar formato a un documento HTML, de tal manera que pueda ser más vistoso al usuario, con respecto a la interactividad esta es dotada a través del lenguaje ***JavaScript*** que de igual manera se ejecuta en el navegador y mediante los *scripts* se agrega interactividad, como animaciones, comportamientos y eventos a los objetos de *HTML*., nativamente desarrollar una aplicación compleja con ***JavaScript***, HTML y CSS conlleva mucho tiempo, por otra parte existen los muy conocidos *frameworks*, que son un conjunto de librerías, clases, interfaces y componentes pre-construidos que se puede utilizar para simplificar el desarrollo de las aplicaciones, y por consecuencia desarrollarlos en menor tiempo.

3.2.2.2 Bootstrap

Hasta hace unos años, poner el estilo en un sitio web era un mal necesario para los desarrolladores. CSS no es exactamente un lenguaje de programación, y no fue diseñado teniendo

en cuenta la capacidad de mantenimiento. Diseñar un sitio web fue una pesadilla para los diseñadores web, que lo usaban todos los días y tenían que repetir su trabajo con cada nuevo proyecto. Esto llevó a la creación de cientos de micro bibliotecas con el propósito de reducir la repetición (o copiar y pegar operaciones) en las definiciones de CSS básicas y estándar.

Afortunadamente, una compañía, ***Twitter***, decidió publicar su "proyecto" interno como un proyecto de código abierto conocido como *Bootstrap*. Esto se convirtió rápidamente en la "biblioteca" de CSS más popular disponible, debido a sus estilos y componentes predeterminados, su modularidad y la facilidad con la que se puede personalizar a través del lenguaje de preprocesamiento de ***CSS Less*** (y recientemente también con ***Sass***). *Bootstrap* responde de forma predeterminada, lo que significa que los sitios y las aplicaciones web creadas con él se adaptan automáticamente al tamaño de la pantalla del dispositivo, ya sea una pantalla de TV, computadora de escritorio, computadora portátil, tableta o teléfono inteligente. (Chiaretta, 2018)

3.2.2.3 KnockoutJS

Para (Munro, 2015) KnockoutJS es una biblioteca de código abierto de JavaScript, que fue construido para permitir crear aplicaciones web dinámicas y enriquecidas, está construido para aplicar y trabajar con el modelo ***Model-View-ViewModel*** *(MVVM)*, *Knockout* hace que sea realmente sencillo implementar una interfaz de usuario compleja que responsa a las interacciones de los usuarios.

La implementación de Knockout implica tres cosas distintas: una vista que contiene elementos HTML y CSS que se vinculan a los datos, un ViewModel que contiene los datos para vincularse a la vista, y Knockout es como una especie de "*pegamento*" que realiza el enlace de los datos a la vista con el ViewModel.

Como dice (Ferrando, 2015), Knockout es muy liviano para cargar, por lo que también establece cadenas de relaciones entre los datos del modelo para transformarlos y combinarlos implícitamente con la vista. Las ventajas de usar ***KnockoutJS*** son las siguientes:

- Es gratis y de código abierto.
- Está construido utilizando JavaScript puro.
- Puede funcionar junto con otros marcos.
- No tiene dependencias.
- Es compatible con todos los navegadores convencionales, incluso los antiguos como IE 6+, Firefox 3.5+, Chrome, Opera y Safari (de escritorio / móvil).
- Está completamente documentado con documentos API, ejemplos en vivo y tutoriales interactivos.
- La función de Knockout es específica: unir vistas y modelos. No gestiona **DOM** ni maneja solicitudes *AJAX*.
- Nos da la libertad de desarrollar nuestro código de la manera que queremos.

3.2.3 Dispositivos electrónicos utilizados

Como referencia teórica se describen los componentes que cubren los conceptos básicos acerca de los componentes utilizados en el proyecto de investigación.

3.2.3.1 Tarjeta de desarrollo Intel Galileo gen 2

Para (Richardson, 2014) Intel Galileo es una placa de desarrollo hardware, la cual es una tarjeta de circuito electrónico que ayuda a desarrollar objetos interactivos leyendo información del mundo físico, procesándolo y entonces tomando una acción en el mundo físico. Intel Galileo es una tarjeta compatible con Arduino, esto significa que puede ser programada con el Arduino IDE usando el lenguaje de programación Arduino, es también compatible con los pines de salida de Arduino v1.0

La tarjeta de desarrollo Intel Galileo está orientada para "*makers*"[3], fue anunciada el 4 de Octubre de 2013 en la feria de ***Makers de Roma***. La tarjeta se basa en el procesador de aplicaciones ***Intel Quark SoC X1000***, con el sistema operativo Linux, y es compatible con las **APIs** de referencia de Arduino. La tarjeta Intel Galileo introdujo varios software extra y novedades de hardware que excedían a otras placas Arduino disponibles en el mercado.

Como dice el autor (De Sousa, 2015) dependiendo de la naturaleza del proyecto, se puede obtener mucha potencia de procesamiento y ahorrar económicamente si se usa en comparación con tarjetas de Arduino o Raspberry, como se puede observar en la Tabla 3-1 la comparativa de principales tarjetas populares de desarrollo.

Tabla 3-1. Comparación de Intel Galileo con las tarjetas de desarrollo más populares.

Características	Intel Galileo	Arduino Yun	Raspberry Pi model B
Velocidad de CPU	400 MHz	400 MHz	700 MHz
Memoria	256 MB	64 KB (AR9331) y 2.5. KB (ATmega)	512 MB
Almacenamiento interno	8 MB	16 MB (AR9331) y 32 KB (ATmega)	-
Almacenamiento externo	MicroSD	MicroSD	SD Card
Red	Ethernet y Wi-Fi (Adaptador Wi-Fi es vendido por separado)	Ethernet y Wi-Fi	Ethernet y Wi-Fi (Antena Wi-Fi es vendido por separado)
Salida de video	-	-	HDMI and 1080p RCA compuesto
Salida de audio	-	-	HDMI y Jack de audio de 3.5mm
Pines de I/O	14 a 3.3 V o 5V	20 a 5V	17 a 3.3V
Entrada analógica	6 (12-bit ADC)	12 (10-bit ADC)	-
Salida PWM	6	7	1
Reloj de tiempo real	Si	-	-
SPI	1	1	2
I2C	1	1	1

[3] Termino definido por la revista Make, 2019 que se utiliza para hacer referencia a una persona que fabrica algo, basado en la cultura de "***hágalo usted mismo***".

Como dice el autor (Ramon, 2014) Para obtener una idea clara de por qué se debe usar las tarjetas Intel Galileo, asumir que necesita desarrollar un proyecto con los siguientes requerimientos:

- Guardar información en una tarjeta SD para ***logging***
- Conectar y transmitir colección de datos usando el internet.
- Usuarios deberían ser capaces de transmitir archivos **log** y archivos **logs** de monitoreo en demanda. Así, un servidor web debería ser desarrollado.
- Un periférico USB especifico como una webcam va ser usada en tu tarjeta Arduino que va ser un host. Las imágenes capturadas por esta webcam van a ser parte de los datos que van a ser transmitidos.
- El acceso a internet debería ser usando Ethernet o conexiones Wifi. Se debe d tener el tiempo correcto y la fecha para los datos que estas procesando en **logging** en la tarjeta SD sean correcto, incluso cuando tus tarjetas se reinician y el sistema es restaurado, por lo que un Reloj de Tiempo Real es necesario.

3.2.3.2 Sensores

Según la definición de (Serna Ruiz, Ros García, & Rico Noguera, 2010) los sensores imitan la capacidad de percepción de los seres humanos, por ello es cada vez más usual encontrarlos incorporados a cualquier área tecnológica. Debido a esta característica de imitar la percepción humana, podemos encontrar sensores relacionados con los diferentes sentidos: vista, oído, tacto, es decir, que reaccionan a la luz, el sonido, el contacto, etc. De igual manera que nuestro cerebro reacciona a la información que recibe de nuestros sentidos, los dispositivos que incorporan sensores reacciones a la información que reciben de ellos. Los sensores son por tanto dispositivos electrónicos que nos permiten interactuar con el entorno, de forma que nos proporcionan

información de ciertas variables que nos rodean para poder procesarlas y así generar órdenes o activar procesos.

3.2.3.3 Sensor ultrasónico HC-SR 04

El fabricante (Elec Freaks, 2013) dice que el módulo de rango ultrasónico *HC-SR04* proporciona una función de medición sin contacto de 2 cm a 400 cm, la precisión del rango puede reaccionar hasta 3mm. Los módulos incluyen transmisores ultrasónicos, receptor y circuito continuo.

- Tensión de alimentación: 5V DC
- Corriente durante la medición: 15mA o corriente de trabajo
- Frecuencia ultrasónica: 40 KHz
- Ángulo de medida: 15º
- Señal de entrada del *trigger*: 10uS pulso alto
- Señal de salida de *echo*: Entrada señal nivel TTL y el rango en proporción.
- Distancia máxima de lectura: 400 cm
- Distancia mínima de lectura: 2 cm
- Resolución: 1 cm
- Dimensiones: 45x20x15 milímetros

Como se observa en la Figura 3-3 un breve impulso ultrasónico se transmite en el momento 0, reflejado por un objeto. El sensor recibe esta señal y la convierte en una señala eléctrica. El siguiente pulso se puede transmitir cuando el eco (*echo*) se desvanece. Este periodo de tiempo se llama periodo del ciclo. El periodo del ciclo recomendado no debe ser inferior a 50ms. Si se envía un pulso de disparo de 10 μs de ancho al pin de señal, el módulo ultrasónico emitirá ocho señales ultrasónicas de 40 kHz y detectará el eco de vuelta. La distancia medida es proporcional al ancho

del pulso del eco y puede calcularse mediante la fórmula: ***µs/58 = centímetros o µs/148 = pulgadas.*** Si no se detecta ningún obstáculo, el pin de salida dará una señal de nivel alto de 38ms.

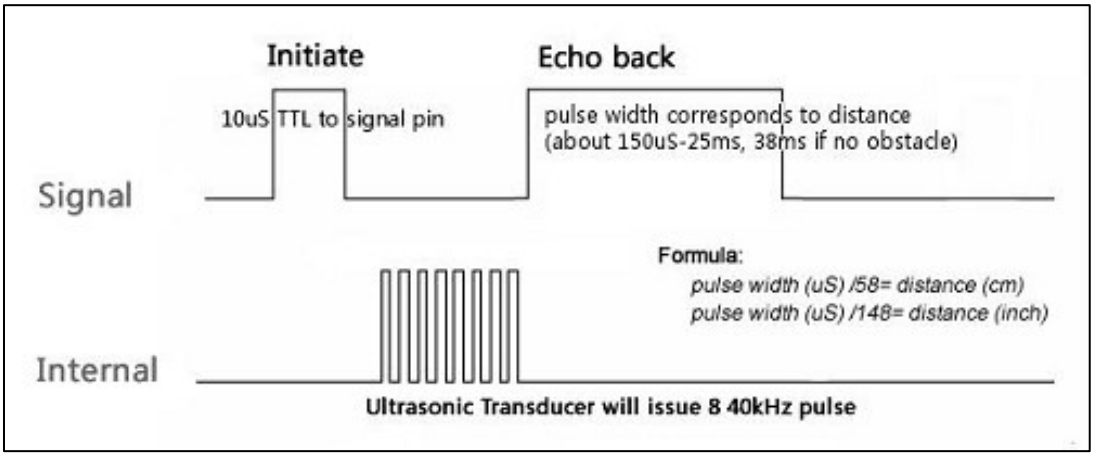

Figura 3-3. Proceso de funcionamiento de HC-SR04.

3.3 El protocolo MQTT

MQTT del inglés **Message Queuing Telemetry Transport** y traducido al español como ***Transporte de Telemetría de Cola de Mensajes,*** como dice (Gastón, 2017) es un protocolo de tipo **M2M** del inglés *Machine to Machine,* traducido al español como **máquina a máquina,** que está orientado al **Internet de las Cosas**, es muy ligero y se maneja sobre la capa de aplicación, está recomendado para la industria en el uso de los sistemas SCADA, además es un protocolo especificado y estandarizado por el consorcio **OASIS** y el **ISO/IEC 20922** en el año 2016. También ofrece varios mecanismos de seguridad y es muy robusto y flexible en áreas en donde las redes de comunicaciones sobre el protocolo TCP/IP manejan altas latencias

Capítulo 4. Modelado del sistema

4.1 Especificación de requerimientos del sistema

En este apartado se pretende especificar los requerimientos del sistema con respecto al planteamiento del problema con la finalidad de proponer un prototipo **SCADA** que permita coadyuvar el problema, por lo que se propone desarrollar un prototipo **SCADA** para contralar el nivel de agua en dos contenedores para su uso futuro en la Comisión de Agua Potable y Alcantarillado de Municipio de Acapulco, la intención fundamental de éste prototipo es formalizar y fundamentar la base en la cual **CAPAMA** se pueda apoyar para aplicarlo en sus tanques de agua reales.

El prototipo propuesto simula los requerimientos básicos de **CAPAMA** para la supervisión, control y adquisición de datos de sus tanques de almacenamiento y distribución, por lo tanto los requerimientos son los siguientes:

- El prototipo simula dos contenedores de agua potable, uno simula ser el tanque de almacenamiento y el otro de distribución, cada tanque simula tener un sistema SCADA que enviará la información de los niveles de los tanques, así como los parámetros de humedad, temperatura y amperaje, cuando el tanque de almacenamiento tenga más del 70% de agua y el tanque de distribución tenga menos del 20% y esté activado el control automático la bomba funcionará hasta llenar el tanque de distribución como se puede observar en la Figura 4-1, cuando por alguna circunstancia se requiere el control manual se ofrece la opción de apagar o encender la bomba a medida de la decisión que se tome con base a los niveles de los tanques de agua capturado por los sensores.

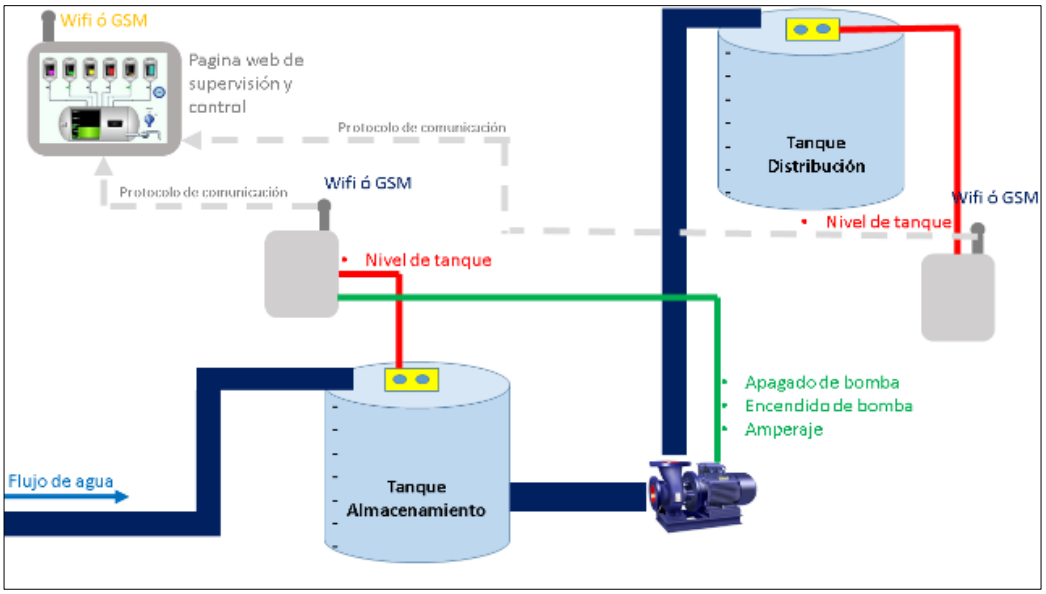

Figura 4-1. Bosquejo general de funcionamiento de los tanques del prototipo.

- Se requiere diseñar y desarrollar un sistema para supervisar, controlar y adquirir la información de los niveles de agua en los tanques en tiempo real así como la obtención de otros parámetros como el amperaje consumido por la bomba, la humedad y temperatura del tanque enviados a una página web para consultar la información y tomar decisiones como lo puede observar en la Figura 4-1. Utilizando para su construcción componentes de bajo costo, orientados al ***Internet of Things*** (IoT), traducido al español como ***Internet de las cosas***, dado que existe un antecedente acerca de la diseminación de esta tecnología en México aprovechando la integración de la telemetría con sensores especializados para el control de la calidad de agua en las redes de distribución, utilizando tecnologías como *Arduino*, *Raspberry Pi* y *Cloudino* para transferir información a la nube.

El documento de especificación de requerimientos tiene como finalidad representar la necesidades sobre el problema, en este caso se requiere desarrollar un sistema SCADA con el propósito de supervisar el estado de las bombas y el estado de los niveles de agua de los contenedores

4.2 Análisis de prototipo SCADA

Para realizar el análisis se modela el sistema SCADA como un ente unificando tanto la parte de software y la parte de la electrónica, por lo que en el siguiente apartado mostraremos los modelos desarrollados con el *Lenguaje Unificado de Modelado (UML)*, que permite modelar y diseñar éste sistema para su posterior construcción.

Para el análisis y diseño del presente trabajo se considera iniciar principalmente con el desarrollo del sistema hardware que involucra; el desarrollo y elaboración de una interfaz electrónica que leerá la información de los sensores y que se conectará a una tarjeta de desarrollo *Intel Galileo gen 2,* para obtener los datos, procesarlos y enviarlos mediantes un protocolo ***MQTT***. Por lo que por consiguiente se tiene planeado trabajar el desarrollo del software de tiempo real paralelamente, ya que la interacción es mutua y se converge en la integración de éstas dos junturas.

Como dice (Haugen, Moller-Pedersen, & Weigert, 2006) para desarrollar un sistema embebido se requiere empezar por *la declaración de dominio*, y el *modelo de dominio*, el *modelo de uso*, diagrama de colaboración de uso, *el diagrama de interacción* y el *diagrama máquinas de estado*, así como los diagramas que modelan el sistema desde una perspectiva de software como lo dice (Deitel, 2005) que recomienda modelar la *estructura del sistema*; la cual describe los objetos del sistema y sus interrelaciones, y el *comportamiento del sistema que describe;* la manera en que cambia el sistema a medida que sus objetos interactúan entre sí, por lo que en los siguientes puntos se enumeran los modelos más relevantes que ayudaran al programador a construir el prototipo SCADA.

4.3 Metodología de desarrollo

Para desarrollar el prototipo ***SCADA***, se consideró trabajar en paralelo el desarrollo del sistema embebido prototipo electrónico, junto con el diseño del software interfaz web que muestra la

información en tiempo real, como el flujo de la metodología orientada al diseño de sistemas embebidos clásica lo sugiere (Ghenassia, 2005), para una mejor verificación, depuración e integración de ambas partes, como se puede observar en la Figura 4-2.

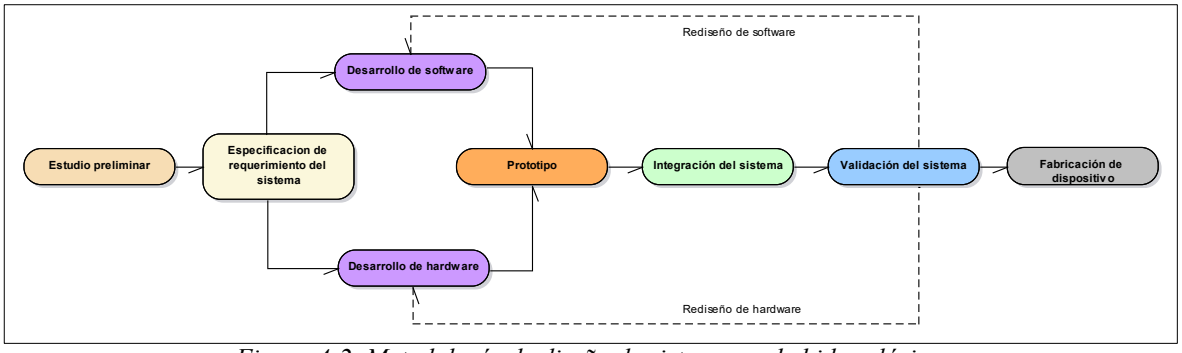

Figura 4-2. Metodología de diseño de sistemas embebidos clásica.

4.4 Declaración de dominio

Empezamos por declarar el modelo de dominio, el cual consiste en modelar como una ente las operaciones más relevantes del sistema; en este caso, se dice que un operador de **CAPAMA** está asociado en una relación de uno a uno con un sistema de supervisión y ese sistema se encuentra relacionado de uno a uno o muchos sistemas de control y adquisición de datos el cual en su conjunto conforman lo que es el sistema **SCADA** y esto se da porque solo podemos tener un sistema que monitoree la información que adquiere de uno o más sistemas de control y adquisición de datos que obtienen la información de los sensores y que la envían mediante el servicio **REST**, por lo tanto el control y adquisición de datos se encuentra ligado de uno a uno a la adquisición de datos de sensores y control de encendido y apagado de bombas, por consiguiente se considera que el sistema de control y adquisición de datos se encuentra instalado en un tanque de **CAPAMA** por lo que la expresión en una relación es de uno por que un sistema de control y adquisición debe estar instalado en un tanque, para apreciar las relaciones, observe la Figura 4-3.

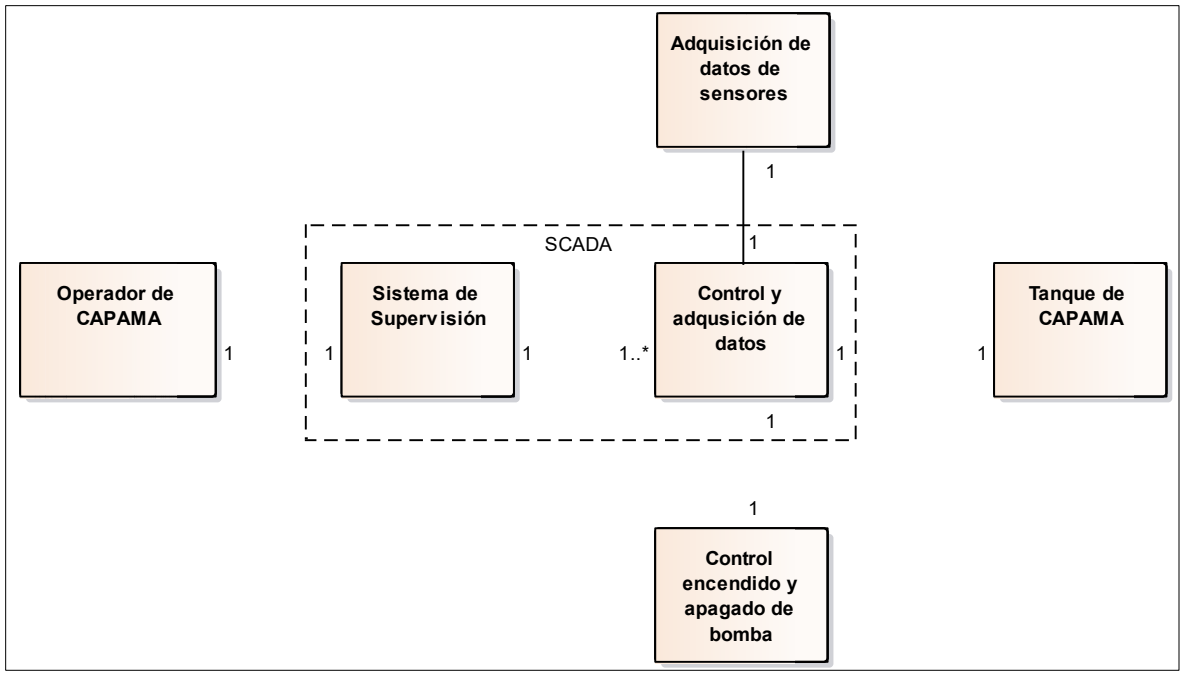

Figura 4-3. Modelo de dominio de clases global.

Por otra parte también se modela el dominio de clases específico para el sistema de supervisión, control y adquisición de datos, el cual puede ser observado en la Figura 4-4, el cual se expresa de la siguiente manera; un sistema de supervisión de control y adquisición de datos se encuentra compuesto por un módulo de conexión inalámbrica que comunica la microcomputadora *Intel Galileo* con la red metropolitana de **CAPAMA**, también a ese sistema se agrega una interfaz gráfica de tiempo real que se puede desacoplar de dicho de sistema; el objetivo es que sea interoperable. También está compuesto de una entidad de control que a su vez se encuentra compuesta por un *contactor* de carga de 120v, para la entidad que modela la adquisición de datos y que se encuentra integrada a la clase de supervisión, control y adquisición de datos, se compone de un sensor de nivel de agua, sensor de humedad, sensor de temperatura, sensor de amperaje, con la respectiva relación explicita de que puede ser cero o uno, (puede o no estar integrado al sistema de adquisición de datos).

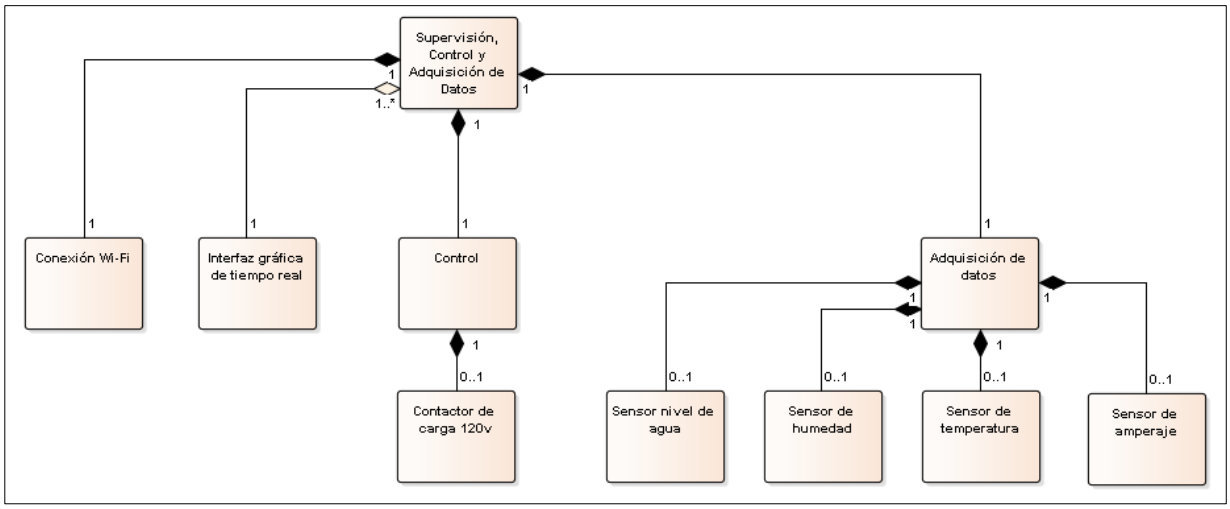

Figura 4-4. Modelo de dominio de clases específico.

4.5 Escenarios y modelo de caso de uso

En el escenario principal se considera el sistema prototipo *SCADA* como una entidad que unifica tanto el software que muestra la información de tiempo real, así como la interfaz electrónica que es una microcomputadora *Intel Galileo gen* 2 que adquiere la información de los sensores, por lo que al escenario se le considera como sistema de supervisión, control y adquisición de datos; en este escenario se manejan tres actores, el operador, el supervisor y el usuario administrador, y sus funciones se describen a continuación:

El operador: actor que se encarga de abrir el sistema software de tiempo real, autenticarse e iniciar en su menú principal o ***dashboard*** para ver la información de los niveles de los tanques de agua, él tiene control de los motores de cada tanque y puede elegir entre iniciarlo de manera automática en que las reglas están dadas por los niveles de agua que encienden o apagan los motores, o el modo manual que indica que hay un operario (o tanquero que apaga las bombas de manera manual en la instalación en donde se encuentra el tanque, supervisado por *SCADA*), así

como también puede ver el status de todas las bombas; si estas están apagadas, encendidas o se encuentran en reposo.

El supervisor: actor que tiene como finalidad autenticarse en la plataforma, y únicamente visualizar la información relacionada acerca de los niveles de los tanques y de ver los estados de las bombas.

Administrador: actor encargado de realizar todas las operaciones que realiza el operador, excepto que es el encargado de autorizar las operaciones de modo manual o automático de los motores y de agregar, quitar y configurar los tanques de agua.

En la Figura 4-5 puede ver el diagrama de **caso de uso**, que modela el escenario y los tres actores.

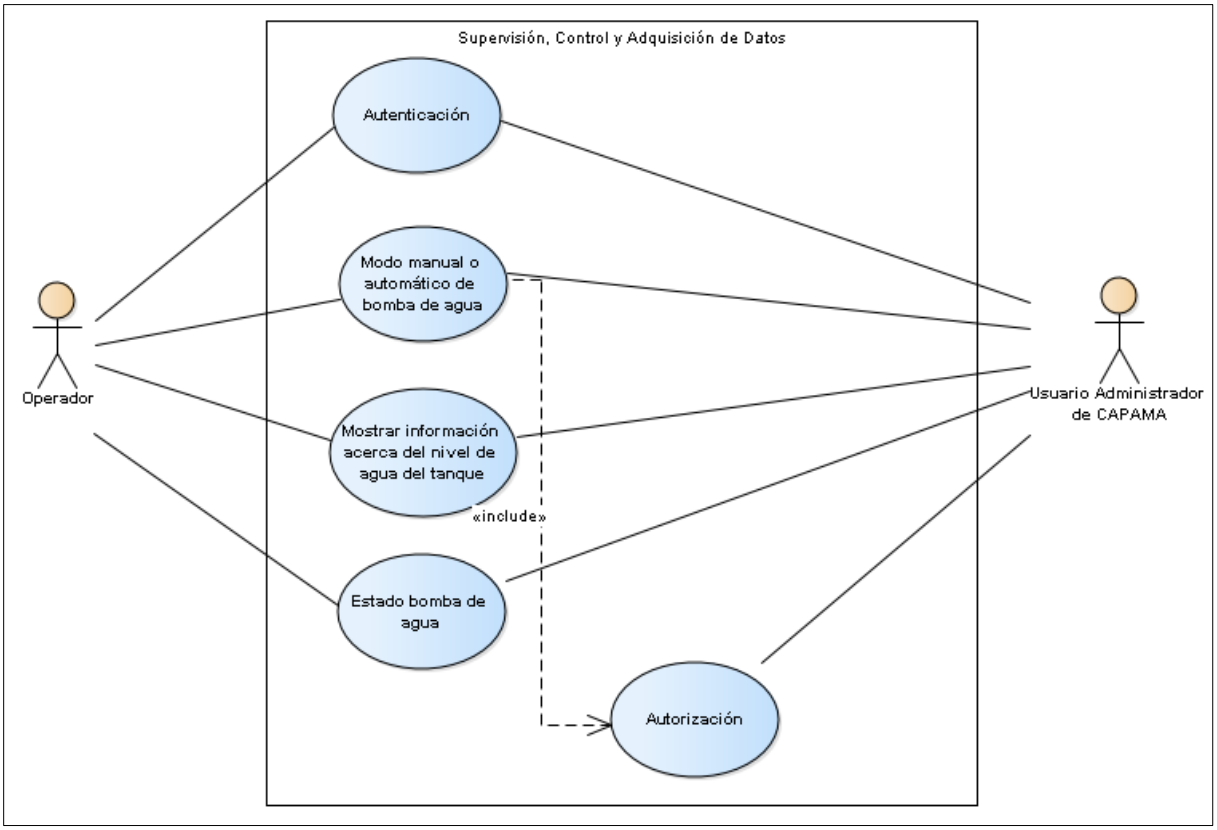

Figura 4-5. Modelo de caso de uso propuesto.

4.6 Modelo de negocio

El flujo de modelo de negocio se muestra en la Figura 4-6 y se encuentra segmentado en tres etapas principalmente; la etapa de adquisición de datos, la etapa de control y la etapa de supervisión. Cada una cumple con un propósito específico y se encuentran interrelacionadas dada la naturaleza de un conjunto de componentes que interactúan para crear un sistema.

La etapa de adquisición de datos: En esta etapa se tiene contemplado el inicio de la operación de la adquisición de datos de los sensores a través de la microcomputadora microcomputador *Intel Galileo gen 2*, dicha información es enviada a través del protocolo **MQTT** que se encuentra hospedado y regido por la aplicación de monitoreo y supervisión que monitorea los contenedores en tiempo real.

La etapa de control: En la etapa de control se considera que la microcomputadora integra un GPIO (**General Porpuse Input Ouput**) el cual es programado para iniciar un driver que controla el encendido y apagado del motor que bombea el agua, de igual manera notifica en tiempo real mediante el servicio **RESTful**, con un verbo **post** el envío del estado de la bomba (si se encuentra encendida o apagada).

La etapa de supervisión: En la etapa de supervisión se considera la aplicación que hospeda el servicio **RESTful** y la conexión con **la base de datos**, de tal manera que muestra la información en tiempo real que el servicio **RESTful** provee a través de los verbos **post, get, put, delete, path**, según, sean los tipos de estados devueltos por estos verbos se habilitaran o deshabilitaran los iconos que muestran los tanques en la interfaz de tiempo real.

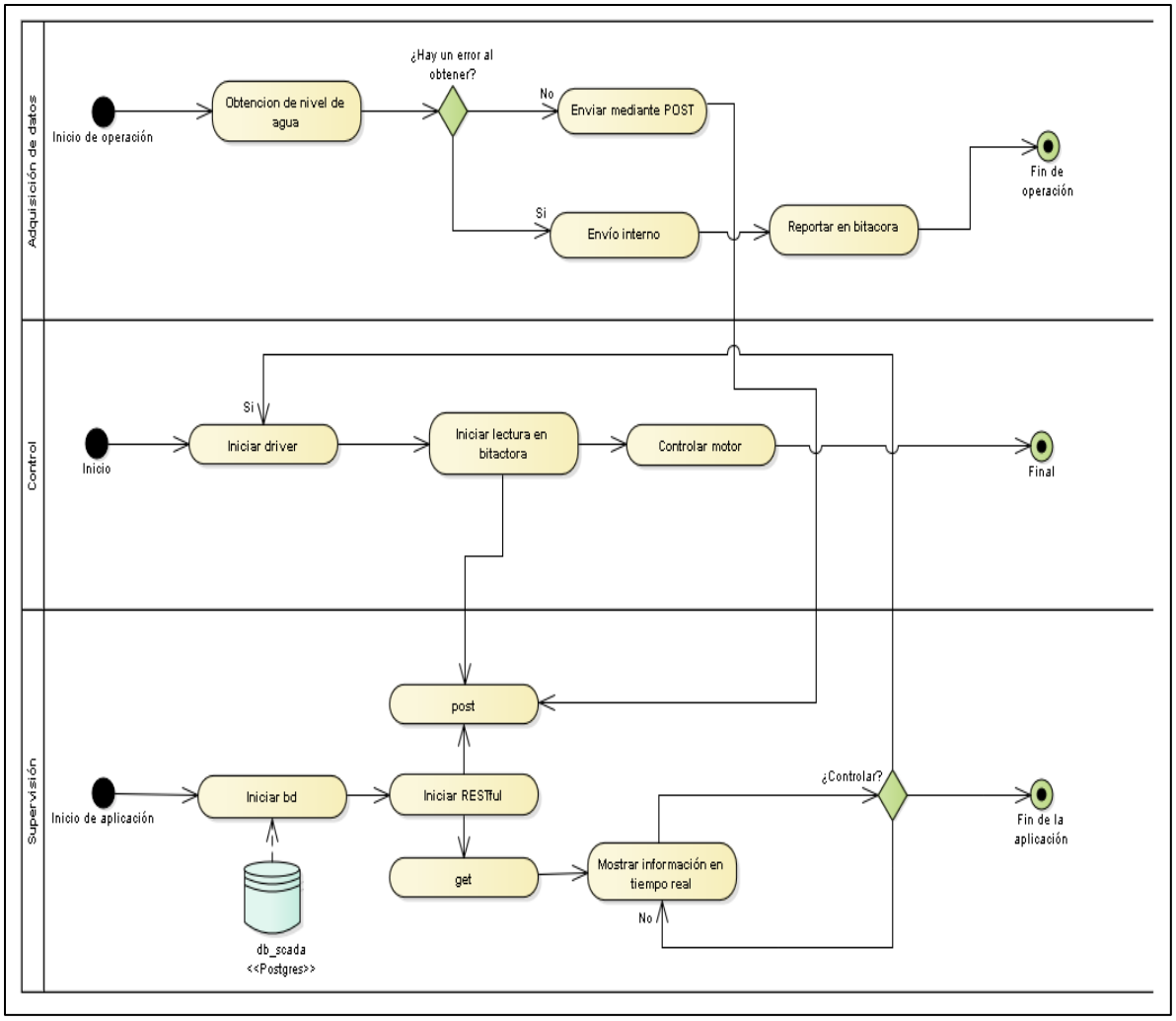

Figura 4-6. Modelo de negocios de sistema SCADA en un tanque de CAPAMA.

4.7 Diagrama de clases

Una vez que expresamos el modelo de dominio, se denotan los sustantivos u objetos más importantes para generar las clases que integraran el sistema, para ello, los siguientes nombres de clases fueron seleccionados, usuarios, operadores, tanques, control, adquisición datos, motor, bomba y sensor, por lo tanto como puede ver en la Figura 4-7, algunos de los *sustantivos* fueron modelados y algunos que no existen se integraron para darle integridad y seguridad.

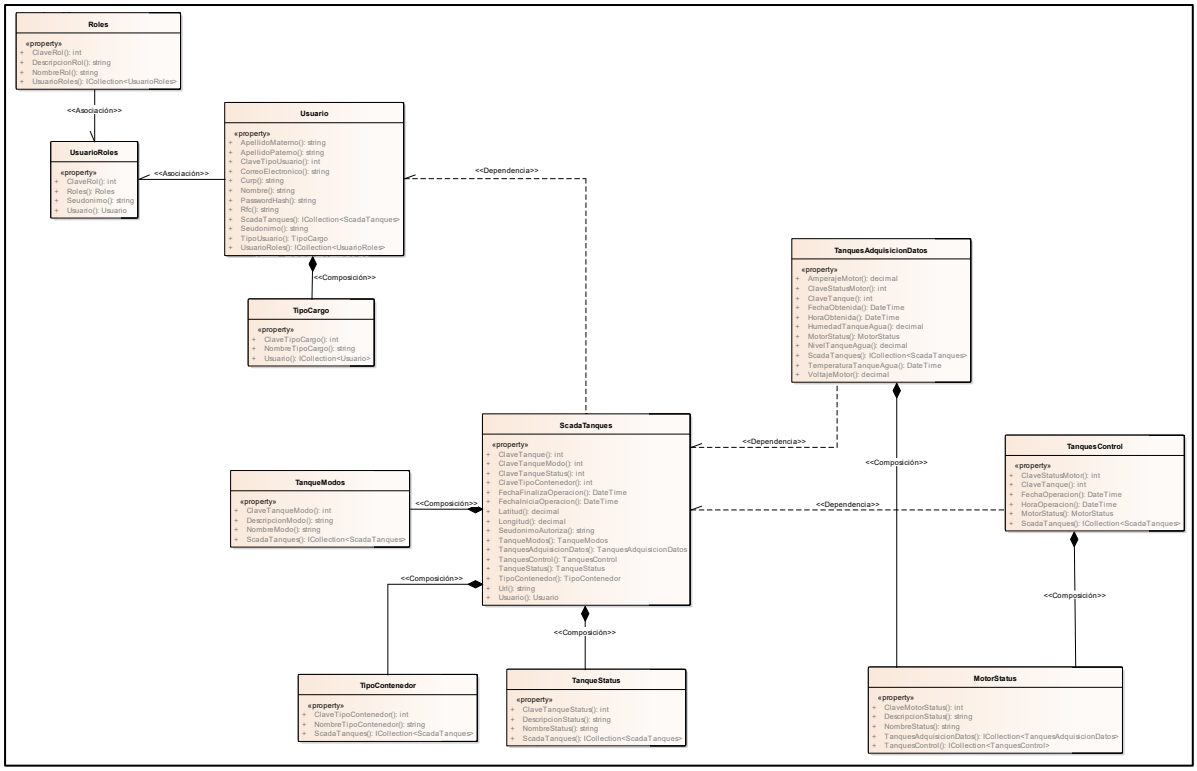

Figura 4-7. Modelo de clases.

La descripción de las clases es la siguiente; primeramente modelamos un usuario que puede ser de tipo: administrador, operador y supervisor, por lo tanto se tiene contemplado que ese usuario se autenticará en la aplicación y por lo tanto debería de visualizar ciertas cosas según sea la condición de privilegios del usuario; por lo tanto para eso se anexaron los **roles** y los **usuariosroles** que modelan la parte de los privilegios, posteriormente una clase llamada **scadatanques** tiene una dependencia a usuarios, pues en esta clase se modela la representación de un tanque que solo el usuario administrador puede agregar al sistema, esa clase se encuentra compuestas por los modos de tanques representado por **tanquemodos**, tipos de contenedores, que es modelado por **tipocontenedor** y estados de los tanques que es modelado por **tanquestatus**, por consiguiente tenemos una dependencia de **scadatanques** a **tanquesadquisiciondatos** pues en esta clase modelamos la información que obtenemos de los sensores y se encuentra conformada por el estado del motor representado por **motorstatus**, de igual manera también **tanquescontrol** que tiene una

dependencia a **tanquesadquisiciondatos** con la finalidad de identificar el tanque en el cual se controla un motor, de igual modo se compone de **motorstatus** el cual se utiliza para notificar el estado del motor a la clase control.

Siguiendo las convenciones y buenas prácticas, propuestas en el libro de (Deitel, 2005) y el patrón de repositorio genérico propuesto por (Microsoft, 2019) para aislar los datos de tal manera que se forme una capa de acceso y manipulación de datos nada más, se pasa a formalizar las clases propuestas en la Figura 4-7 para acceder a un repositorio por cada una de las clases; esto es con el fin de ahorrar código que se repite en todas las clases y que son operaciones comunes como: insertar información, obtener información, obtener información específica, actualizar la información y eliminar la información, por lo que en la diagrama de la Figura 4-8 puede apreciar su distribución y operaciones.

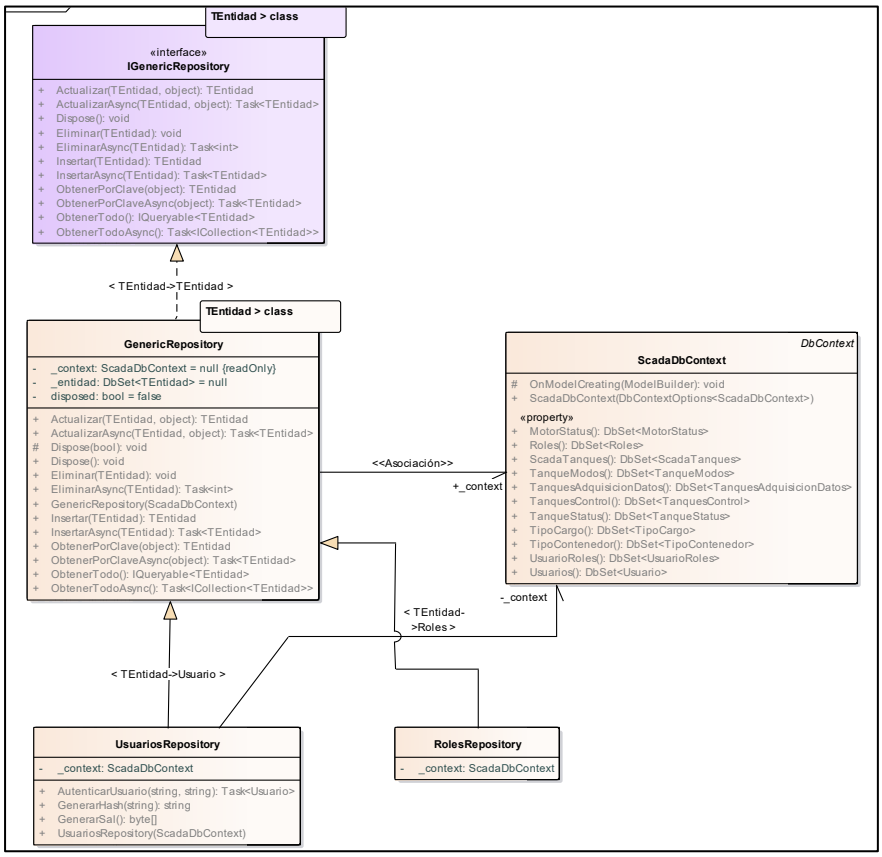

Figura 4-8. Clases que implementan el patrón de repositorio genérico para abstracción de datos.

4.8 Diagrama de Entidad-Relación

Se propone el uso de un diagrama que modela la interacción y las relaciones entre entidades del sistema **SCADA** ver Figura 4-9, por lo que se sugiere utilizarlo como referencia dado que como se pretende utilizar bases de datos ***NoSQL*** puede cambiar sustancialmente, este diagrama como tal, se encuentra integrado por dos tablas bases que son los usuarios junto con su autenticación, los roles de usuarios concierne a la autorización de ciertas secciones del sistema **SCADA**, y la otra tabla base es con respecto a los tanques del sistema, que integra por ejemplo el tipo de contenedor, el estatus del tanque, el almacenamiento de la adquisición de datos, el control y estado de las bombas.

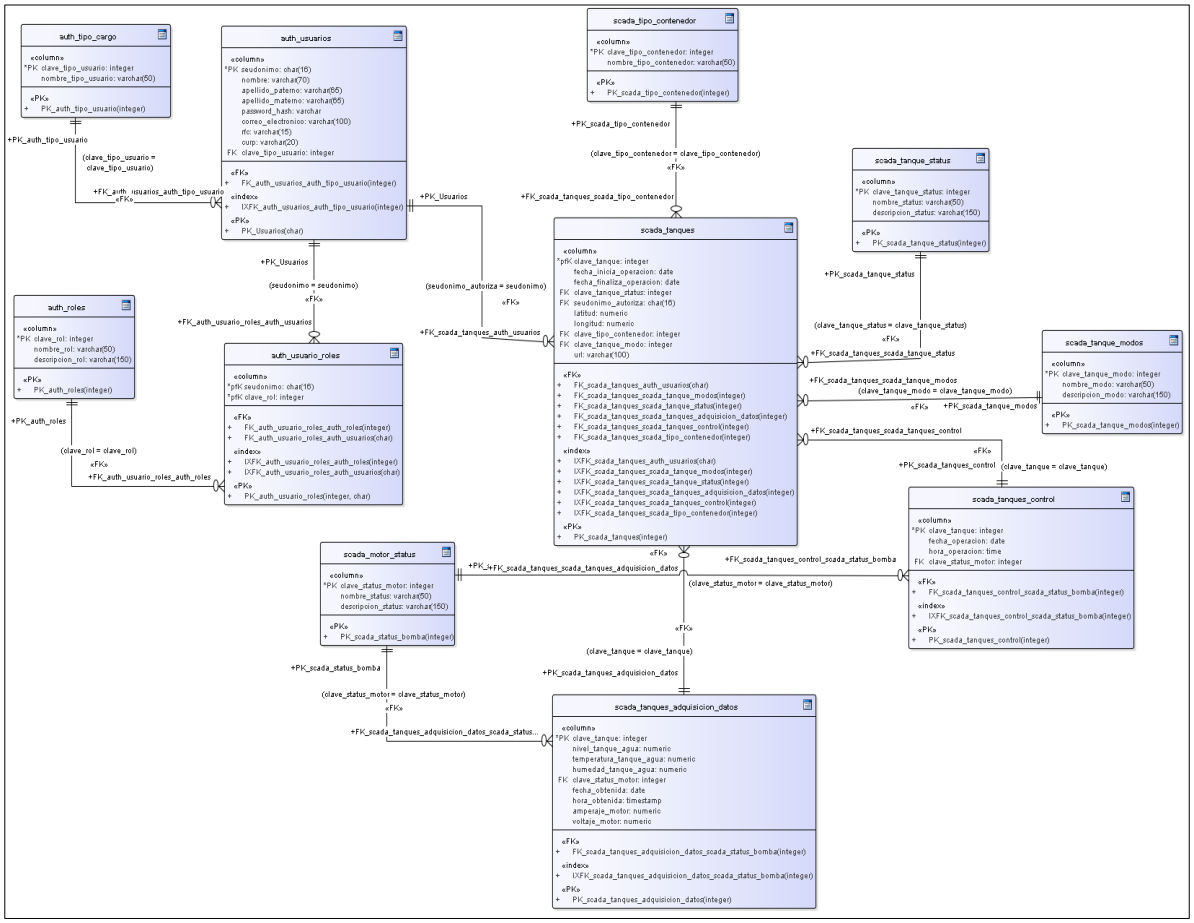

Figura 4-9. Diagrama de Entidad-Relación del Sistema SCADA.

4.9 Diagrama de máquina de estado

Se propone el uso de un diagrama de máquina de estado para ver los estados de un objeto en un momento dado en el tiempo, como se puede observar en la Figura 4-10 se indica mediante los valores de los atributos del objeto que *estado clave* está sucediendo y por qué o bajo que circunstancia está pasando y por qué circunstancia cambia de estado, en este caso el sistema ***SCADA*** modela estado de un *usuario*; si esta *autenticado* accede al sistema de lo contrario regresa a su estado como usuario *no autenticado*, si es autenticado entonces accede al sistema ***SCADA***, entonces el sistema verifica la información del sensor para ver si un *contenedor está vacío*, si lo está entonces cambia al estado de *encender bomba*, cuando el sensor detecte el tope límite del líquido cambiará al *estado de apagar bomba*, por el contrario sí no ha alcanzado su límite entonces cambia de *estado encender bomba*, por el contario *finaliza la operación*. Con respecto del contenedor *si el contenedor está vacío* puede tomar *el estado de contenedor lleno* cuando el sensor ha registrado un datos que indica que se ha llenado, por el contrario si detecta que no hay liquido cambia *de estado a contenedor vacío*, cuando el contenedor está lleno *finaliza la operación*.

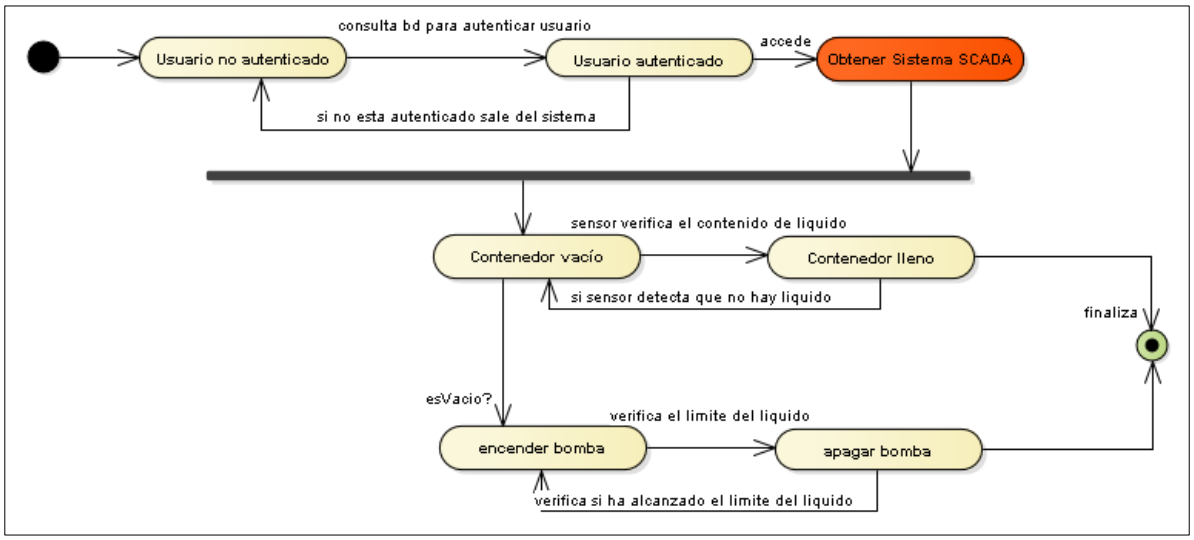

Figura 4-10. Diagrama de máquina de estado.

4.10 Diagrama de despliegue

Para instalar e implementar nuestro sistema SCADA se sugiere el uso de un diagrama de despliegue para implantarlo en producción para el uso del sistema por los usuarios, por lo se sugiere implementar como en la Figura 4-11 sugiere, que segmenta el servidor web y el servidor de base datos, en cada servidor físico único y solo se usa un medio como un "***router***" para interconectar ambas partes, con respecto al servidor de base de datos se recomienda el uso de Linux Ubuntu Server 18 y un motor de base de datos orientado a bases de datos no relaciones, como es el caso de MongoDB, para el servidor web se recomienda un sistema operativo Linux Ubuntu Server 18, con el entorno de ejecución del .NET Framework de Microsoft para hospedar la aplicación web, así como también la instalación de un servidor **NGINX** para la escucha en el puerto 80 para la publicación de la página web del sistema **SCADA**.

Con respecto al dispositivo que se conecta con la página web, tiene que ser mediante un protocolo, por lo que se sugiere el uso del protocolo **MQTT**, el cual en su conjunto va a ser publicado por las interfaces del dispositivo **SCADA** como tal, que a su vez está conectado a sensores y actuadores para obtener y procesar la información, para escribir la interacción entre los dispositivos electrónicos se recomienda elaborar el firmare en **C** y utilizar librerías "***Open Source***", como ***SubClient*** para la implementación del protocolo y la comunicación.

Por otro lado esta segmentación se servidores nos da mayor potencia a la hora de procesar datos, dado que cada servidor cumple como única función la asignada por la arquitectura esto nos da holgura para encontrar problemas técnicos más rápidos y mantenimientos mucho más rápidos.

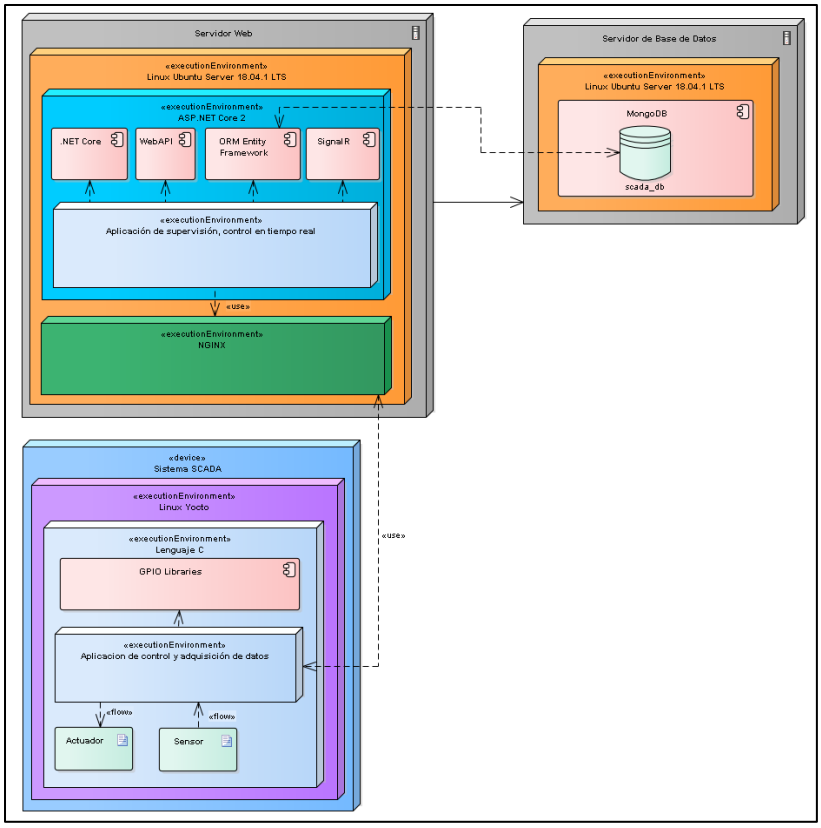

Figura 4-11. Diagrama de despliegue.

4.11 Arquitectura

Se presenta el sistema **SCADA** como un sistema que integra dos capas principalmente; la primera conforma el sistema en el tanque de agua, la segunda integra el sistema en las oficinas de telemetría de CAPAMA, por lo que como puede apreciar en la Figura 4-12, de lado izquierdo podemos encontrar los componentes que conforman la interfaz electrónica que se encarga de adquirir y procesar los datos de los sensores de nivel de agua, posteriormente esta información es enviada a través de internet por medio de las antenas ***ubiquiti*** que actualmente forman la infraestructura metropolitana de **CAPAMA**, de lado derecho podemos apreciar que en las oficinas se tiene contemplado un servidor que hospeda la aplicación que muestra la información de tiempo real, para ello el servidor, integra un servicio de base de datos **MongoDB**, que a su vez interactúa

con un servicio **RESTful** que obtiene, procesa y muestra en la interfaz **dashboard** del usuario operario que este activo y autenticado.

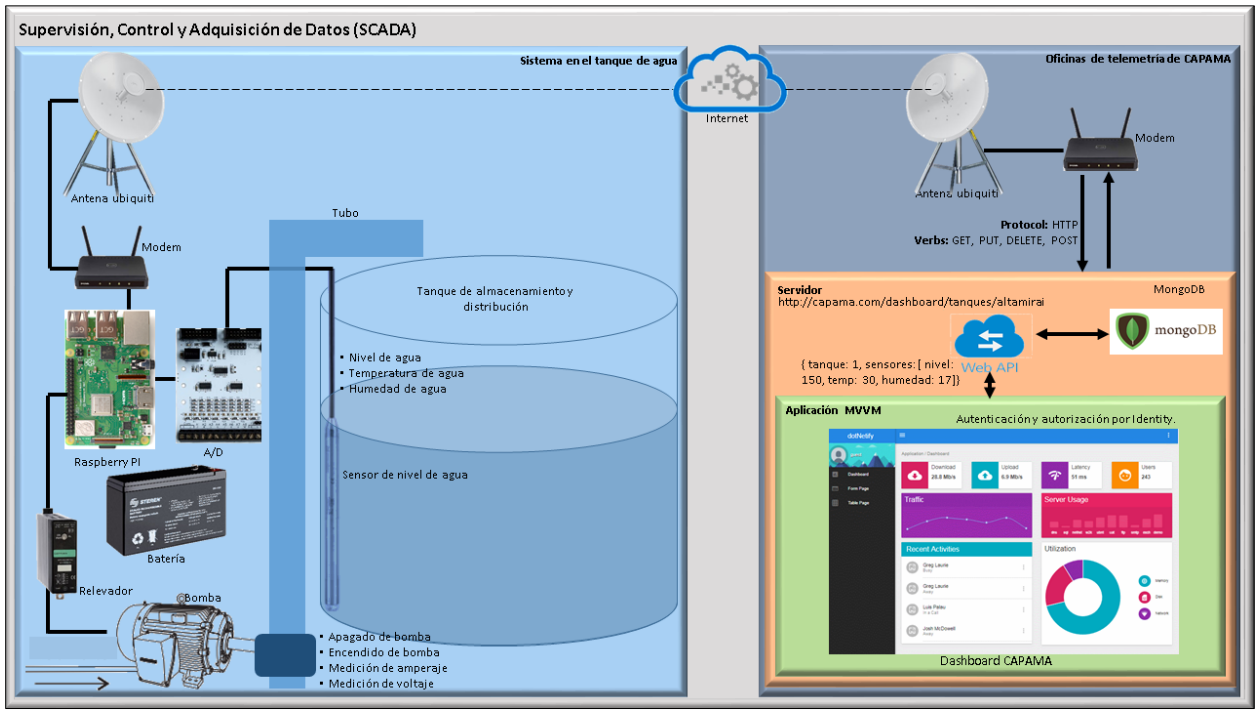

Figura 4-12. Arquitectura de sistema SCADA para un tanque.

Capítulo 5. Desarrollo prototipo SCADA

En este capítulo se detalle el proceso de desarrollo del prototipo ***SCADA,*** con base al modelado del sistema del capítulo anterior, por lo que como el presente proyecto involucra dos tipos de desarrollo; el prototipo electrónico del sistema SCADA y el propio software que gestiona las operaciones básicas del sistema SCADA, una de las consideraciones acerca de su elaboración fue desarrollar ambas partes paralelamente como lo dice la metodología clásica de desarrollo de sistemas embebidos de (Ghenassia, 2005) ya que cada parte depende de la otra

5.1 Desarrollo de interfaz electrónica SCADA

Se considera el desarrollo de un sistema hardware con componentes electrónicos que interactúan juntos para enviar al software información acerca de los niveles de agua en los contenedores de tal manera que la información que se muestra es de tiempo real, por lo que su desarrollo atiende a las necesidades del modelado del sistema del capítulo 4.

5.1.1 Herramientas utilizadas.

Dado que una de las consideraciones como tal que el proyecto de investigación es un *prototipo* se hace hincapié en que para el desarrollo de la parte electrónica, se utilizaron componentes electrónicos de bajo costo, orientados al ***Internet de las Cosas*** y ***Open Source Hardware*** y considerando la parte del ***planteamiento del problema*** y con fundamento en el diseño del modelado del sistema, se considera el uso de las siguientes herramientas:

- **Fritzing 0.9.3**: La cual es la herramienta que nos permite realizar el diseño prototipado del sistema SCADA.
- **Visual Studio Code con el plugin de Arduino**: Es el editor de código para desarrollar el firmware de la placa de desarrollo *Intel Galileo gen 2.*

5.1.2 Diseño de prototipo SCADA

El prototipo electrónico SCADA comprende el uso de dos sensores ultrasónicos para obtener el nivel de agua de los contenedores de manera no invasiva, un relevador que se conecta a un contactor industrial para accionar una bomba y una placa de desarrollo *Intel Galileo gen 2* para gobernar los componentes y almacenar la lógica de negocios que el sistema SCADA requiere, atendiendo a las necesidades del *planteamiento del problema y el modelado del sistema*, por lo que el resultado del diseño de la interconexión de los componentes se muestra en la Figura 5-1, formando de esta manera un esquemático que será la base para desarrollar el sistema prototipo *SCADA*.

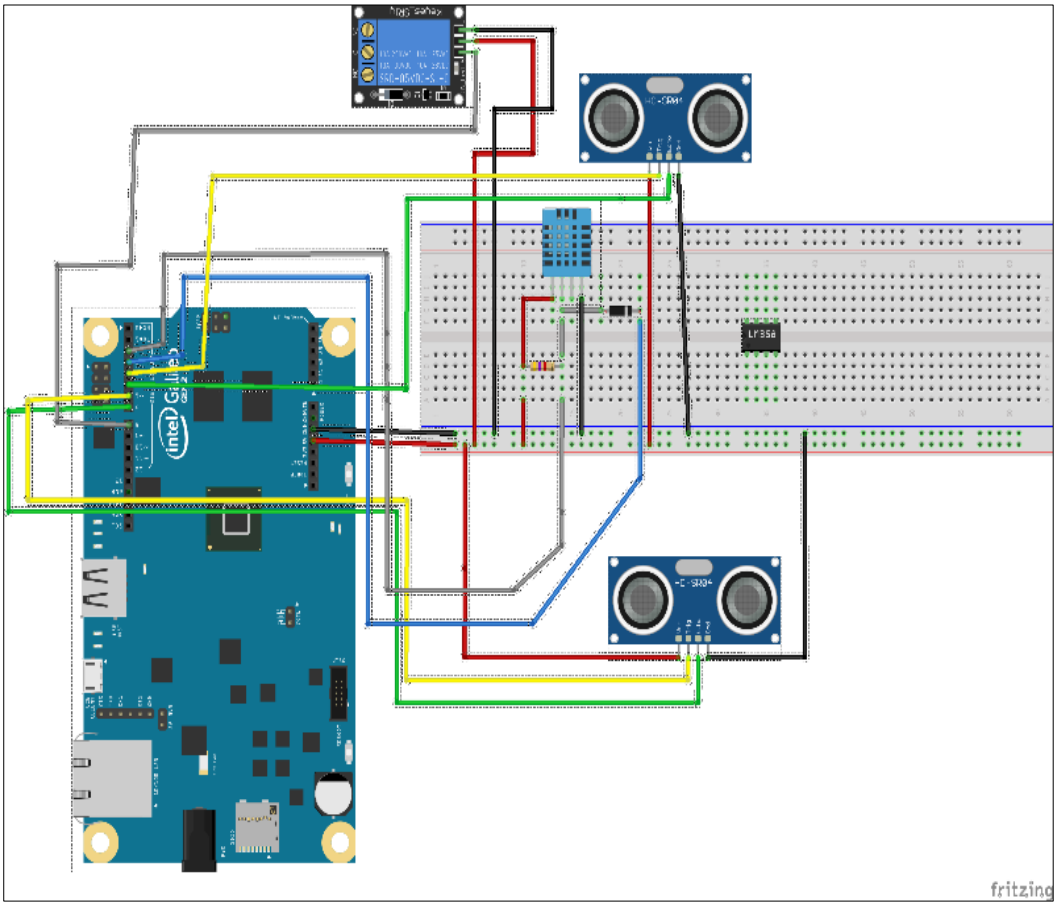

Figura 5-1. Resultado del diseño del prototipo SCADA.

5.1.3 Componentes utilizados y elaboración de prototipo SCADA

Atendiendo al diseño del esquemático electrónico discutido en el punto anterior se desarrolló la maqueta del prototipo SCADA, considerando los componentes electrónicos fundamentales para su funcionamiento se eligieron los siguientes:

- 2 Sensores ultrasónicos HC-SR 04.
- 1 Router Linksys.
- 2 Interruptores termomagnéticos.
- 1 Mini contactor de 18A industrial.
- 1 Mini Relevador de 18A industrial.
- 1 Sensor de corriente alterna 30A no invasivo.
- 1 Relevador de 5v.
- 1 Sensor DHT 11.
- 1 Intel Galileo Gen 2.

Concluida la elección de los materiales elegidos, se procede a desarrollar el prototipo con las herramientas y otros aditamentos utilizables para sujetar los cables y componentes, uniendo los componentes electrónicos de acuerdo al diagrama electrónico generado en el punto ***diseño de prototipo SCADA***, se obtiene como resultado la Figura 5-2.

Figura 5-2. Resultado de prototipo SCADA para electrónica.

Para el desarrollo del firmware se utilizó la herramienta de *Visual Studio Code con el plugin de* Arduino microcomputadora ***Intel Galileo Gen 2***, utilizando para su programación como lenguaje base ***C***, para dotar al prototipo de las operaciones y comportamientos que tiene que realizar con base en los requerimientos del sistema modelados en el *capítulo 4*.

Y una parte fundamental en la elaboración del *firmware* fue la utilización de librerías como ***PuSubClient*** para Arduino adaptada a la placa de desarrollo ***Intel Galileo gen 2*** para la comunicación con la interfaz web que muestra la información en tiempo real mediante el uso del protocolo **MQTT**, y esto lo hace mediante un patrón denominado *publicador*/*subscriptor*, utilizando para ello los denominados *tópicos*, que básicamente son URI's del inglés (**Unified Resource Identifier),** traducido al español como *Identificador de Recurso Unificado,* por lo que dentro del *loop* principal de firmware, como se puede apreciar en la Figura 5-3 los publicadores

están “enviando o publicando la información” hacia el servidor o *bróker* que es nuestra aplicación de tiempo real que recibe los datos, para mostrarlos al usuario y dar un transparencia al usuario de “*tiempo real*”.

```
// Loop until we're reconnected
  while (!client.connected())
  {
    Serial.print("Intentando conectar servidor MQTT...");
    // Attempt to connect
    if (client.connect("tanqueCapama1"))
    {
      Serial.println("Conectado");
      // publicadores y subscriptores para tanque de distribucion
      client.publish("tanqueCapama1/sensores/nivel", "0");
      client.publish("tanqueCapama1/sensores/temperatura", "0");
      client.publish("tanqueCapama1/sensores/humedad", "0");

      // publicadores y subscriptores para tanque de almacenamiento
      client.publish("tanqueCapama2/sensores/nivel", "0");
      client.publish("tanqueCapama2/sensores/temperatura", "0");
      client.publish("tanqueCapama2/sensores/humedad", "0");
      client.publish("tanqueCapama2/sensores/bomba/potencia", "0");

      client.subscribe("tanqueCapama2/sensores/bomba/control");
      client.subscribe("tanqueCapama2/sensores/bomba/modo");
    }
    else
    {
      Serial.print("fallo, rc=");
      Serial.print(client.state());
      Serial.println(" intentando de nuevo en 5 segundos");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
```

Figura 5-3. Código principal de comunicación con protocolo MQTT.

Con respecto a la integración y ubicación de los sensores, como se aprecia en la Figura 5-4 estos se ubican en la parte superior de los contenedores de agua su finalidad es medir el nivel de agua de los contenedores y enviar la información al sistema de interfaz web que muestra la información de tiempo real, cabe hacer mención que se elaboró una tubería que conecta dos contenedores de agua, simulando un contenedor de “*almacenamiento*” y otro de “*distribución*” con una bomba que rebombea el agua del contenedor de almacenamiento al de distribución, la bomba y los sensores ultrasónicos están conectados a la maqueta del prototipo ***SCADA*** mostrado en la Figura 5-2.

Figura 5-4. Ubicación sensores ultrasónicos en contenedores de almacenamiento y distribución.

5.2 Desarrollo prototipo software SCADA

Para desarrollar el software del prototipo SCADA, se consideró trabajar en paralelo el desarrollo del sistema embebido prototipo electrónico, junto con el diseño del software interfaz web que muestra la información en tiempo real.

5.2.1 Herramientas de desarrollo utilizadas

Las herramientas que fueron utilizadas para desarrollar el prototipo SCADA fue la siguiente:

- **Visual Studio 2017:** Es el Entorno de Desarrollo Integrado del acrónimo en inglés IDE que se utiliza por defecto para desarrollar aplicaciones orientadas al ecosistema .NET,

trabaja con los lenguajes de programación de Microsoft, como: C#, Visual Basic, XAML y tecnologías como el .NET Framework y el ASP.NET Core, este último utilizado para desarrollar toda la aplicación software del sistema SCADA.

5.2.2 Desarrollo de software SCADA

Siguiendo la recomendación de (Ferrando, 2015) en el desarrollo del software del prototipo *SCADA* se utilizó una regla denominada *KISS* (***Keep It Simple, Stupid***) traducido al español como mantenlo simple y tonto; aprovechando esta regla se procedió a diseñar el software de tal manera que el programa que muestra los datos de tiempo real se integre a la interfaz electrónica del prototipo SCADA, de una manera que en futuro se permitan acoplamiento y desacoplamientos de otros sistemas SCADA que se puedan integrar en un futuro de manera amena y sencilla, diseñados con el mismo principio, sin involucrar mucha complejidad de conexiones con el software, por lo que su diseño está basado principalmente en **URI's** para la localización e identificación de los tópicos de los contenedores de agua integrando para ello la parte del patrón publicador/subscriptor que ofrece el protocolo ***MQTT.***

5.2.3 Almacenamiento y acceso a datos

Para la implementación de la base de datos y el acceso a datos, considerando que el proyecto en primer lugar trabajará con el procesamiento de mucha cantidad de información de datos al almacenar la información en la base de datos se opta por utilizar ***MongoDB***, como se denota el por qué utilizar este tipo de base de datos orientados a ***Documentos*** en el *capítulo 3 marco teórico*, por lo que el código de acceso a la base de datos se realiza como se muestra en la Figura 5-5.

```
public class DbContext : ICollectionContext
    {
        private readonly IMongoDatabase _db;

        public DbContext(IOptions<Settings> opciones)
        {
            MongoClient client = new MongoClient(opciones.Value.ConnectionString
);
            _db = client.GetDatabase(opciones.Value.Database);
        }

        public IMongoCollection<Usuario> Usuarios => _db.GetCollection<Usuario>(
"auth_usuarios");
        public IMongoCollection<Tanque> Tanques => _db.GetCollection<Tanque>("sc
ada_tanques");
        public IMongoCollection<TanqueAdquisicionDato> TanquesAdquisicionDatos =
> _db.GetCollection<TanqueAdquisicionDato>("scada_tanques_adqui_datos");
        public IMongoCollection<TanqueControl> TanquesControl => _db.GetCollecti
on<TanqueControl>("scada_tanques_control");
    }
```

Figura 5-5. Configuración de acceso a la base de datos MongoDB.

Para la parte de la gestión de acceso y procesamiento de la información se utiliza el patrón de acceso a datos denominado *patrón de repositorios genéricos*, el cual permite crear una capa de abstracción de datos, esto significa que en esta capa únicamente y exclusivamente se trabaja con los datos y su procesamiento en la base de datos, este patrón implementa de manera genérica lo que se le conoce como modelo **CRUD** (*Create, Read, Update, Delete*) en cada una de las clases repositorios del acceso a datos, como se muestra en la interfaz que dicta las operaciones en la Figura 5-6.

```
public interface IGenericRepository<TEntidad> where TEntidad : class
    {
        Task<ICollection<TEntidad>> ObtenerTodoAsync();

        Task<TEntidad> ObtenerPorIdAsync(object id);

        Task<TEntidad> InsertarAsync(TEntidad entidad);

        Task<bool> ActualizarAsync(TEntidad entidad, object id);

        Task<bool> EliminarAsync(object id);
    }
```

Figura 5-6. Interfaz que implementa modelo CRUD.

El soporte de acceso y almacenamiento de datos va de la mano con métodos **async** o asíncronos en el *server-side* utilizando el framework web api, como se puede observar en la Figura 5-7 para exponer un servicio **RESRful** que funciona utilizando peticiones asíncronas y procesándolas mediante *hilos* utilizando *Tasks* para dar rápida respuestas de almacenamiento de datos, utilizando para ello la capa de abstracción de datos mediante el patrón de *repositorios genéricos*.

```
// POST: api/TanquesAdquisicionDatos
        [HttpPost]
        public async Task<IActionResult> Post([FromBody] TanqueADDto model)
        {
            if (model == null)
                return BadRequest("El objeto fué nulo");

            if (!ModelState.IsValid)
                return BadRequest(model);

            TanqueAdquisicionDato tanqueAd = _mapper.Map<TanqueAdquisicionDato>(
model);
            await _tanqueAdRepository.InsertarAsync(tanqueAd);

            return StatusCode(201);
            //return CreatedAtRoute("Get", new { id = tanqueAd.Id }, tanqueAd);
        }
```

Figura 5-7. Método asíncrono utilizando hilos haciendo petición a la base de datos MongoDB.

Se opta por utilizar estos mecanismos porque la parte de adquisición de datos contempla que aproximadamente cada fracción de milisegundo está adquiriendo información de los sensores, por lo que se requiere rapidez y fluidez en el almacenamiento de la información.

5.2.4 Desarrollo de servidor MQTT

Una de las cosas más importantes para el desarrollo de la aplicación ***SCADA*** es la comunicación con la contraparte hardware que contempla la parte electrónico con la interacción de los componentes sensores y actuadores, en este apartado se revisa la configuración mediante programación que habilita a la aplicación a utilizar el protocolo **MQTT** para que esté a la escucha en el puerto *1883* teniendo como consecuencia que la aplicación devuelva un servidor que hospeda un servidor **MQTT**, como se puede apreciar en la Figura 5-8 lo más relevante, es el soporte para

habilitar el puerto y establecer la configuración mínima de un servidor MQTT, así como como el soporte para *WebSockets* que nos ayudará a "*parsear*" la información del protocolo MQTT a WebSockets dado que como tal el protocolo MQTT no puede ser escuchado e interpretado por un navegador, también como se puede notar se agrega soporte para el patrón MVC (Modelo Vista Controlador).

```
public void ConfigureServices(IServiceCollection services)
        {
            // optios for configuration mqtt
            var mqttServerOptions = new MqttServerOptionsBuilder()
                .WithoutDefaultEndpoint()
                .Build();

            //this adds a hosted mqtt server to the services
            services.AddHostedMqttServer(mqttServerOptions);

            //this adds tcp server support based on Microsoft.AspNetCore.Connect
ions.Abstractions
            services.AddMqttConnectionHandler();

            //this adds websocket support
            services.AddMqttWebSocketServerAdapter();

            services.AddConnections();

            // agregamos al pipeline de .net core el automapper
            services.AddAutoMapper();

            // agregamos al pipeline de .net core el mvc
            services.AddMvc()
                .SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
```

Figura 5-8. Configuración del servidor MQTT y patrón MVC.

Como se dijo anteriormente para las operaciones de comunicación e intercambio de información de tiempo real entre el prototipo SCADA electrónico y el software aplicación web que muestra la información de tiempo real mediante gráficos, se utilizó el protocolo **MQTT** del inglés **Message Queuing Telemetry Transport** y traducido al español como ***Transporte de Telemetría de Cola de Mensajes,*** es un protocolo de tipo **M2M** del inglés *Machine to Machine,*

traducido al español como **máquina a máquina,** que está orientado al **Internet de las Cosas**, es muy ligero y se maneja sobre la capa de aplicación, está recomendado para la industria en el uso de los sistemas SCADA, además es un protocolo especificado y estandarizado por el consorcio **OASIS** y el **ISO/IEC 20922** en el año 2016. También ofrece varios mecanismos de seguridad y es muy robusto y flexible en áreas en donde las redes de comunicaciones sobre el protocolo TCP/IP manejan altas latencias.

5.2.5 Desarrollo de la aplicación de una sola página (SPA)

En el desarrollo de la aplicación se utiliza las librerías del protocolo **MQTT** para obtener los datos de los sensores ultrasónicos del nivel de agua y los de los demás sensores especificados en los *requerimientos del sistema*, por lo que para el para obtener los datos del servidor *bróker* o servidor MQTT, se utiliza una libraría que se ejecuta del lado del cliente denominada *mqtt.js* y lo más relevante de esta librería es escuchar al servidor **MQTT** para obtener la información de los sensores mediante los denominados *"tópicos"* como se puede observar en la Figura 5-9.

```
client.on('message', function (topic, message) {

        if (topic == 'tanqueCapama1/sensores/nivel') {
            dataset[0] = message;
        }
        if (topic == 'tanqueCapama1/sensores/temperatura') {
            temperaturaTanque1 = message;
        }
        if (topic == 'tanqueCapama1/sensores/humedad') {
            humedadTanque1 = message;
        }

        if (topic == 'tanqueCapama2/sensores/nivel') {
            datasetTanque2[0] = message;
        }
        if (topic == 'tanqueCapama2/sensores/temperatura') {
            temperaturaTanque2 = message;
        }
        if (topic == 'tanqueCapama2/sensores/humedad') {
            humedadTanque2 = message;
        }
        if (topic == 'tanqueCapama2/sensores/bomba/potencia') {
            amperajeTanque2 = message;
        }

        console.log(('mensaje ' + topic + ": " + message.toString()));
    });
```

Figura 5-9. Tópicos para obtener la información de los sensores.

Una vez que se obtiene la información, estos son procesados de tal manera que esa información capturada se refleje mediante figuras animadas que hacen analogía al mundo real, en este caso dos contenedores de líquido, por lo que para desarrollar esta parte se utilizó *d3.js* una librería muy potente para desarrollar gráficos y figuras a través de código *JavaScript* como se muestra en la Figura 5-10; el código de un contenedor es convertido a imagen ***svg*** en tiempo de ejecución como lo puede observar en la Figura 5-16.

```
            svg = d3.select("#svgcontainer").append("svg:svg").attr("width", wid
th + 2 * margin)
                .attr("height", height + 2 * margin)
                .attr("class", "fondoTanque");

            svg.append("g")
                .attr("transform", "translate(" + margin + "," + margin + ")")
                .selectAll("rect")
                .data(dataset)
                .enter().append("rect")
                .attr("width", 100)
                .attr("height", function (d) { return height - y(d); })
                .attr("x", function (d, i) { return x(i); })
                .attr("y", function (d) { return y(d); });

            var yAxis = d3.svg.axis()
                .scale(y)
                .orient("right")
                .ticks(4);

            svg.append("g")
                .attr("transform", "translate(" + margin + "," + margin + ")")
                .attr("class", "axis")
                .call(yAxis);

            svg.append("text")
                .attr("text-anchor", "middle")
                .attr("font-size", "9")
                .attr("x", 30)
                .attr("y", 100)
                .text("tanque # 1");
```

Figura 5-10. Código de SVG para mostrar un tanque con datos de tiempo real.

Para la parte de la aplicación web que muestra gráficos con información de los niveles de los contenedores en tiempo real así como de los demás parámetros especificados en el modelado del

prototipo **SCADA**, se desarrolló la aplicación utilizando el .NET Framework con la tecnología de ASP.NET Core, utilizando el concepto de desarrollo de las **SPA (Single Page Application)** o aplicaciones de una sola página la cual ofrece mayor rendimiento y su carga en el navegador es mucho más rápido, dado que ya no se hacen peticiones directamente al servidor para cargar todo el **DOM** (***Document Object Model***) de HTML, si no que este concepto utiliza llamadas asíncronas mediante **AJAX** (**Asynchronous JavaScript and XML**) o basada en **WebSockets**, como puede observar en la Figura 5-11 una llamada ***AJAX*** para actualizar únicamente las partes necesarias en el **DOM** de HTML, por lo para su uso en aplicaciones de nueva generación que muestran información de tiempo real es recomendado. (Emmit , 2016)

```
self.registrarUsuario = function (form) {
        console.log(form);
        console.log(ko.toJSON(self.usuario));

        $.ajax({
            url: '/api/usuarios/registrar',
            data: ko.toJSON(self.usuario),
            type: 'POST',
            contentType: 'application/json',
            headers: {
                'Authorization': 'Bearer ' + self.getAccessToken()
            },
            dataType: 'json',
            success: self.successUsuarioRegistrar,
            error: self.errorUsuarioRegistrar
        });
    };
```

Figura 5-11. Ejemplo de una llamada AJAX a un servicio RESTful.

5.2.6 Integración del patrón MVVM (Modelo Vista Vista-Modelo)

Para la parte de diseño de la aplicación y la integración del bróker MQTT y la arquitectura de una *Single Page Application* , se aplicó el patrón **MVVM** (*Model View View-Model*) mediante la librería denominada como ***KnockoutJS***, como lo sugiere (Emmit , 2016), esto nos ofrece una ventaja a la hora de mantener y depurar el código, dado que todo se concentra en un *View-Model*, este forma "*el pegamento*" entre el código *HTML* y las operaciones de JavaScript que están en el

View-Model mediante los *databindings*, en este caso el *view-model* contiene todas las operaciones de nuestro sistema SCADA; la autenticación y autorización ver Figura 5-12, el acceso y almacenamiento de datos de los tanques ver Figura 5-13, así como las *routes* para el Single Page Application ver Figura 5-14.

```
self.comprobarAutenticacion = function () {

        if (self.getAccessToken() !== null) {
            console.log(self.getAccessToken());
            self.esAutenticado(true);
            self.limpiarDatosUsuario();
            self.obtenerTodosUsuarios();

        }
        else {
            self.esAutenticado(false);
        }
    };
```

Figura 5-12. Comprobación de autenticación y autorización de usuario.

```
self.obtenerTodosTanques = function () {
        $.ajax({
            method: 'get',
            url: '/api/tanques',
            contentType: 'application/json; charset=utf-8',
            headers: {
                'Authorization': 'Bearer ' + self.getAccessToken()
            },
            success: self.successTodosTanques,
            error: self.errorTodosTanques
        });
    };
```

Figura 5-13. Acceso y obtención de información.

```
//Rutas del lado del cliente
    Sammy(function () {

        this.get('#login', function () {
            // sugerido minusculas
            self.establecerPaginaActiva("login");
        });

        this.get('#tanques', function () {
            self.obtenerTodosTanques();
            self.establecerPaginaActiva("tanques");
        });

        this.get('#usuarios', function () {
            self.establecerPaginaActiva("usuarios");
        });

        this.get('#home', function () {
            self.establecerPaginaActiva("mapa");
        });

        this.get("", function () {
            this.app.runRoute("get", "#home");
        });
    }).run();
```

Figura 5-14. Rutas de la aplicación de una sola página.

5.2.7 Secciones

Como resultado del desarrollo de la aplicación web se tiene las siguientes interfaces gráficas del sistema SCADA.

5.2.7.1 Sección de inicio de sesión.

Página de inicio de sesión para ingresar al sistema *SCADA,* la funcionalidad es introducir las credenciales de un usuario autorizado dado de alta en la base de datos del sistema, para acceder y controlar el sistema SCADA ver Figura 5-15.

Figura 5-15. Página de inicio de sesión para ingresar al sistema SCADA.

5.2.7.2 Sección dashboard principal

Si el usuario está autenticado y autorizado, la página de inicio se mostrará la funcionalidad principal del sistema *SCADA* esto es, que muestra un mapa con dos gráficos de contenedores que muestran información de tiempo real, en éste caso; el nivel del líquido simulado de los contenedores; información enviada por los sensores ultrasónicos, como lo muestra la Figura 5-16 que inicializan vacíos e inmediatamente empiezan a mostrar la información del nivel del líquido Figura 5-17, información enviada a través por el protocolo **MQTT**, capturada por los sensores ultrasónicos. Además el usuario puede seleccionar el icono del tanque, en este caso el tanque # 1 y puede visualizar en una ventana *modal* los datos de los datos adquiridos por el tanque, como la temperatura, la humedad y el nivel del líquido en tiempo real como puede apreciar en la Figura 5-18 y además como se aprecia en la Figura 5-19 también puede manipular el tipo de operación; manual o automático, cuando la operación es manual la bomba de rebombeo de agua se manipula conforme a la decisión del operador, cuando está en modo automático, cuando el tanque de rebombeo de agua tiene más del 70% de contenido de agua y el tanque de distribución tiene menos

del 10% o igual el sistema automáticamente toma la decisión de rebombear el agua, las opciones antes mencionadas se aprecian en la Figura 5-19.

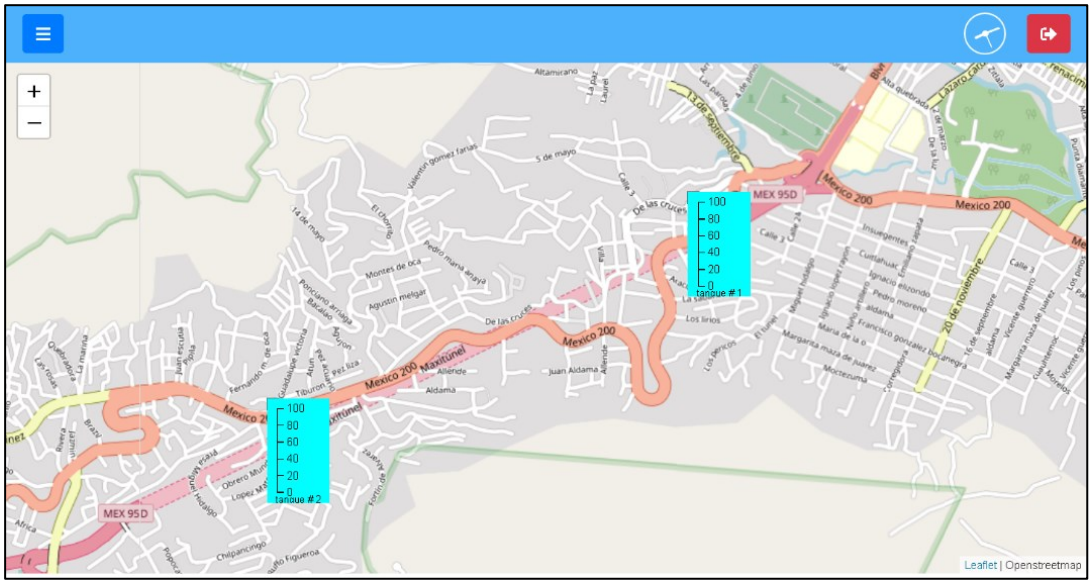

Figura 5-16. Página principal que muestra dos contenedores del sistema SCADA.

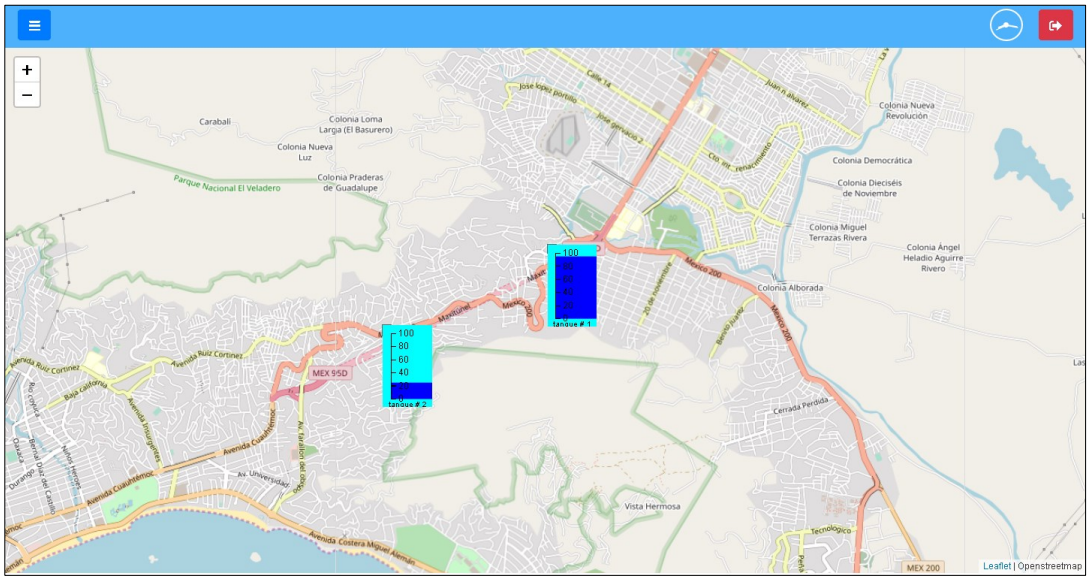

Figura 5-17. Página principal que muestra información de tiempo real del líquido de los contenedores.

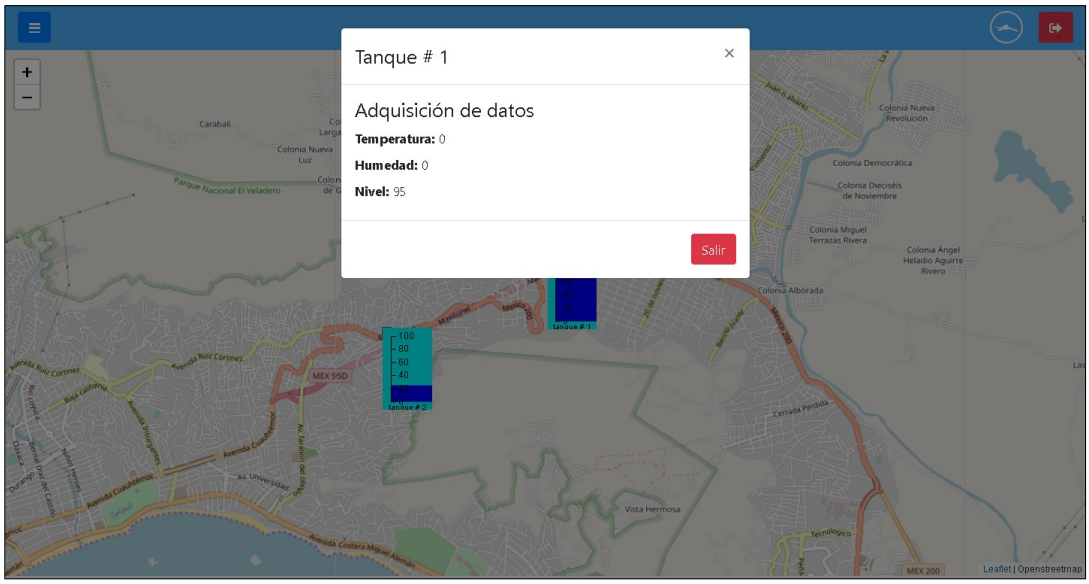

Figura 5-18. Tanque 1, muestra información en tiempo real de temperatura, humedad y nivel.

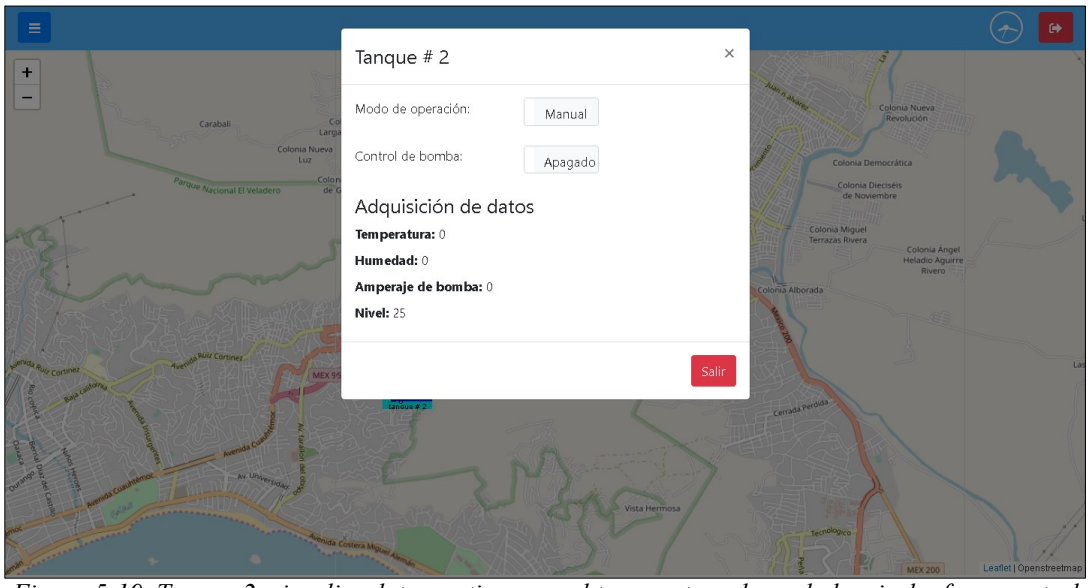

Figura 5-19. Tanque 2, visualiza datos en tiempo real temperatura, humedad y nivel, ofrece control de bomba y modo de operación.

5.2.7.3 Sección de registro de usuarios

Otra funcionalidad que ofrece el sistema *SCADA* es contener una página de registro de los usuarios que pueden acceder al sistema ver Figura 5-20. Cuando un usuario se encuentra registrado puede acceder en la ventana de inicio de sesión del sistema *SCADA* ver Figura 5-15.

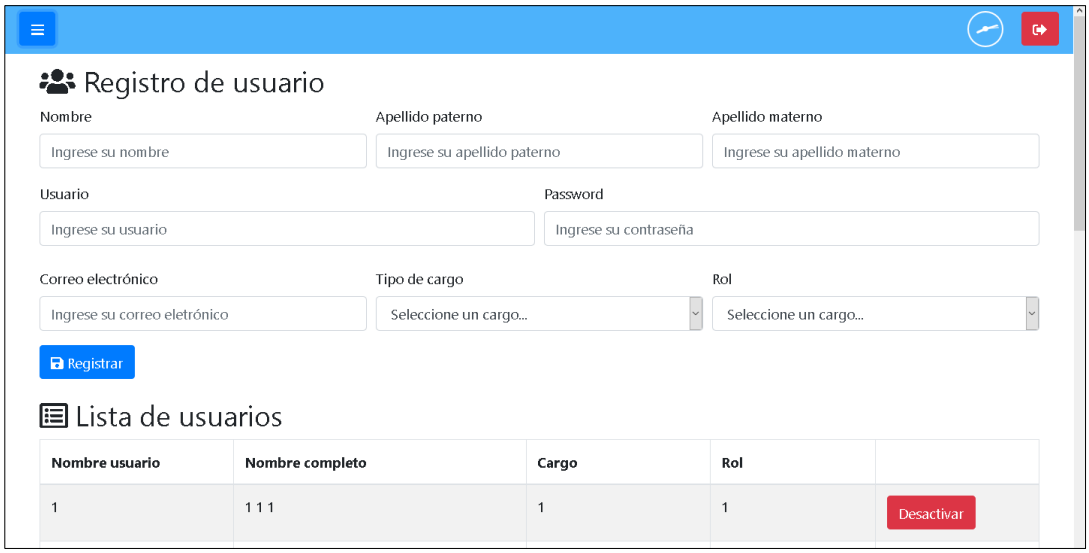

Figura 5-20. Página de registro de usuario para ingresar al sistema SCADA.

5.2.7.4 Sección de registro de contenedores

El sistema *SCADA* contiene una sección de registro de los tanques (ver Figura 5-21*)* que tiene como finalidad llevar un registro de los tanques del sistema, ya que está información es requerida para guardar la información en la base de datos sobre la adquisición de los datos de los sensores relacionados al tanque registrado.

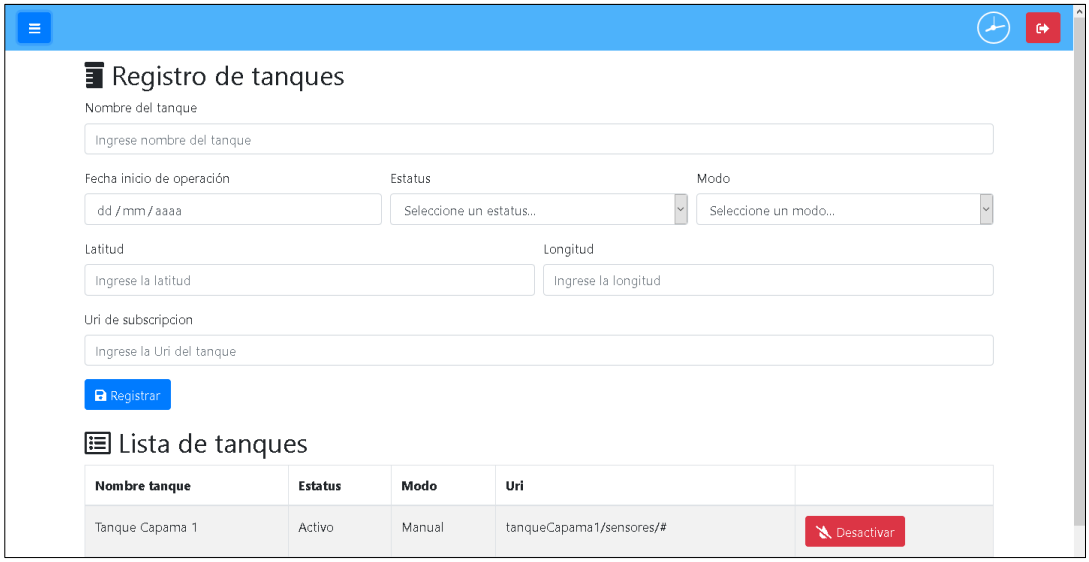

Figura 5-21. Página de registro de tanques del sistema SCADA.

5.2.8 Aspectos de seguridad

Con respecto a la seguridad de los sistemas SCADA, según la norma IEEE 1402-200, "*Guide for Electric Power Substation Physical and Electronic Security*", contempla los riesgos informáticos como problemas comunes de los equipos de control y monitorización, y hace hincapié en la necesidad de implantar sistemas de seguridad orientados a impedir las intrusiones de tipo electrónico. (Rodríguez Penin, Comunicaciones Industriales - Guía práctica, 2008)

Por lo que en el presente trabajo nos limitamos a desarrollar únicamente el sistema SCADA utilizando un protocolo MQTT sin seguridad, por lo que si se requiere implementar seguridad existen algunos de los mecanismos de seguridad para este protocolo como la encriptación mediante un certificado TLS/SSL, o mediante el *payload* encriptado o mediante el protocolo OAuth 2.0, cabe hacer mención que se utiliza la seguridad solo en la consulta e insertado a la base de datos mediante JWT o conocido *Json Web Tokens,* pasándole en su encabezado de *HTTP* el *token* generado del lado del servidor cuando el usuario se autentico de manera correcta, al utilizar el tipo de servicios *RESTful* no se manejan estados como a diferencia de las cookies, por lo que el mecanismo implementado en este proyecto es utilizando *SessionStorage API* de HTML5 para almacenar los datos de sesión temporalmente o hasta que se finalice la sesión en el navegador o en su caso se cierre la ventana.

Capítulo 6. Resultados

Con respecto a los resultados cabe destacar que como propósito general se desarrolló un prototipo **SCADA** que cumpliera con los requerimientos de la Comisión de Agua Potable y Alcantarillado del Municipio de Acapulco, de tal manera, que el prototipo simulará los eventos de cómo funcionaría implementado en el mundo real.

6.1. Prueba técnica de tiempo de envío de información protocolo MQTT

Con respecto a que en la actualidad **CAPAMA** tiene un sistema **SCADA** muy antiguo por lo que su red de comunicaciones básicamente se constituye de antenas de radiofrecuencia utilizando la banda **UHF** para intercambiar información de la estación de tanque a la central de datos, y que por lo que por experiencia se sabe, en hora pico de tráfico de automóviles que oscila entre 6 pm y 7:30 pm, se genera ruido e interferencia en el dispositivo **SCADA**, por lo que el tiempo de envío y recepción de información es muy prolongado y oscila de minutos a horas, esto ocasiona un desfase en la información de los niveles de agua capturados por los sensores en los tanques en tiempo real, teniendo como consecuencia el manejo de información imprecisa para los trabajadores de **CAPAMA**.

Por otro lado, cabe señalar que el protocolo **MQTT**, trabaja en la parte superior del modelo conceptual de interconexión de sistemas abiertos (**OSI**), denominado como capa de **Aplicación**, por lo tanto se categoriza como un protocolo de mensajes basado en el patrón ***publicador-subscriptor*** y su rendimiento depende mucho del hardware en donde se implemente y ejecute el servidor de **MQTT**, por lo tanto la prueba realizada consiste en la obtención del tiempo en que tarda el sensor ultrasónico (que denominamos ***publicador***) que a través de la tarjeta de desarrollo ***Intel Galileo*** *publica* la información y la envía como mensaje al servidor de ***MQTT*** a través de los *tópicos* que son destino y que pasa por una aplicación que muestra información de los tanques en

tiempo real y que se mantiene a la escucha en el puerto **1883**, a través de una computadora con características relevantes como un procesador **Intel Core i5 2.5 GHz**, **6Gb en memoria RAM** y tarjeta integrada video **Intel HD Graphics 4000** además de una tarjeta de conexión **WiFi** con conexión a internet a través de una IP pública estática suministrada por el operador **ISP IZZI**, localizándose la troncal de datos, en el Instituto Tecnológico de Acapulco, y a una distancia del prototipo con el sensor ultrasónico ***(publicador)*** de aproximadamente 6.08 km, localizado en las instalaciones de **CAPAMA**, carretera **Cayaco** – **Puerto Marqués s/n, colonia Llano Largo**, que envía la información de un sensor ultrasónico de nivel de agua en un contenedor de 200 litros, midiendo el proceso del tiempo que envía la información, con la librería **StopWatch** que se encuentra en el espacio de nombre de **System.Diagnostics** del ***.NET Framework***, recomendada por **Microsoft** para realizar procesos de **diagnóstico**, **pruebas** y **depuración**.

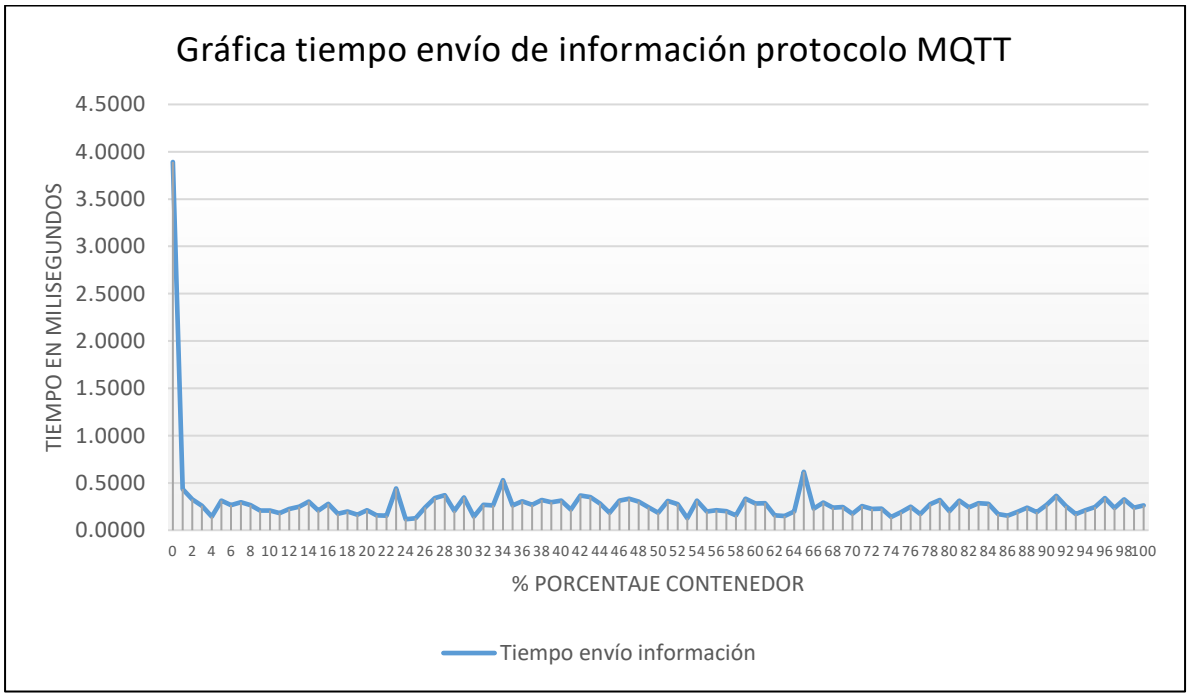

Figura 6-1. Gráfica de tiempo de envío de información mediante protocolo MQTT.

Como se puede apreciar en la gráfica Figura 6-1 el tiempo medido de envío del protocolo MQTT se representa por la línea azul e inicia en 3.8917 milisegundos, este retardo se presenta la

primera vez debido a que el *publicador* ubica el bróker o servidor **MQTT** y generará una única petición (***request***) del ***tópico***, para que enseguida envíe el valor del nivel del sensor, que en promedio resulta en un valor de envío de 0.2537 milisegundos, cabe destacar que la velocidad de envío y recepción de paquetes del protocolo **MQTT** dependerá en primer lugar de la latencia del tráfico de la red en donde está conectado el servidor **MQTT**, y de las características de procesamiento del equipo, dado que el protocolo trabaja en la capa de **aplicación**, otro factor que hay que mencionar es que la calidad del servicio, traducido del inglés como **QoS**, en el protocolo MQTT existen los niveles; 0, 1 y 2, predeterminadamente en las pruebas se optó por el valor predeterminado 0, pero como menciona (Gastón, 2017) para asegurar la calidad del servicio se puede optar por el nivel 2 o 3, y dado que estos nivel incrementan las posibilidades de que los paquetes lleguen a su destino con total seguridad, representan un mayor tiempo de espera, pues sacrifican tiempo por seguridad, por parte del servidor **MQTT**, lo que como sugiere el autor, dependiendo del tipo de aplicación que se desarrollará se puede ajustar los parámetros de calidad del servicio.

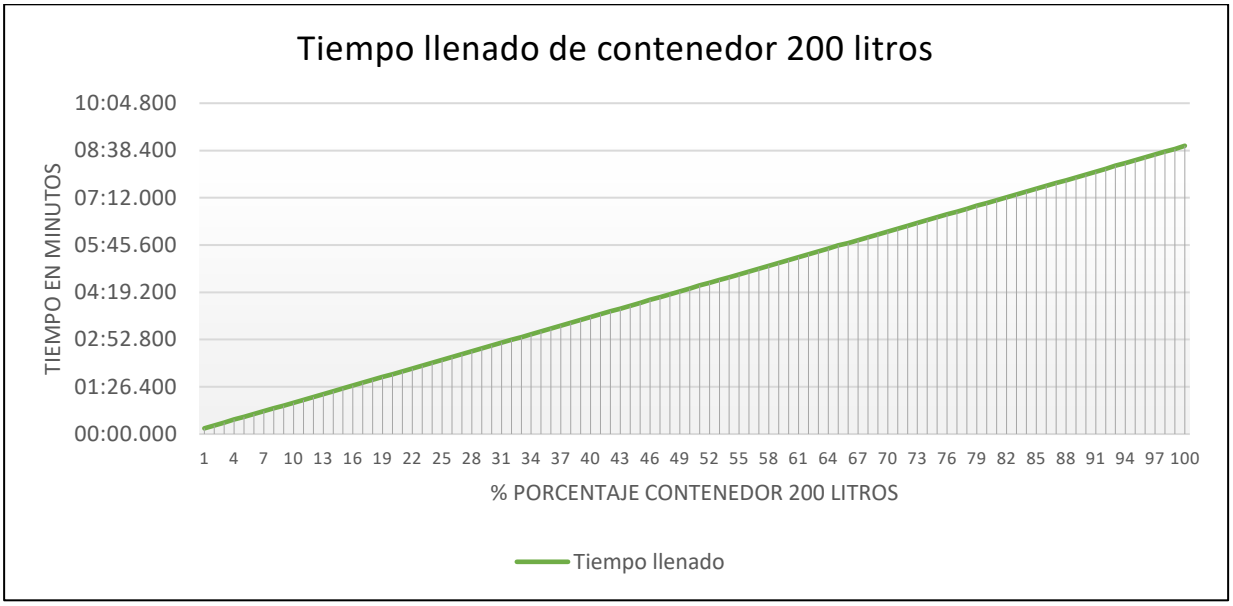

Figura 6-2. Gráfica de tiempo de envío de información mediante protocolo MQTT.

Como en la gráfica de la Figura 6-2 se puede apreciar el tiempo de llenado de un contenedor de 200 litros, inicia en un tiempo de 0 segundos y se mantiene en constante llenado llegando hasta un tiempo de 8 minutos, 38 segundos y 400 milisegundos, llegando a un 100% de su capacidad, por lo que en ese momento el prototipo **SCADA** apaga la bomba de agua para que no se derrame el agua del contenedor.

Con respecto a las medidas realizadas al sistema **SCADA** actual de la **CAPAMA**, el presente trabajo de investigación se limita a recabar información empírica de la experiencia de los trabajadores, pues como tal no se cuenta con los instrumentos necesarios para llevar a cabo la medición del tiempo de envío y recepción de datos a través de la banda radiofrecuencia **UHF**, en especial durante las horas de tráfico automovilístico y/o lluvias que provocan arcos eléctricos en los postes de energía de alta tensión que generan interferencia en el dispositivo **SCADA** y que por consecuencia la información tarda más en ser enviada, que oscila de unos cuantos minutos a horas, pero que en promedio la información tarda en ser enviada 30 segundos por lo que los datos graficados en la Figura 6-3 obedecen a los datos proporcionados por los trabajadores de **CAPAMA**.

Figura 6-3. Gráfica de tiempo de envío por radiofrecuencia por banda UHF

En contraste si se implementará el servidor **MQTT** y el envío de información mediante este protocolo, las pruebas dictan que el protocolo **MQTT** es mucho más rápido para mostrar la información en tiempo real, observe la Figura 6-4 si en promedio CAPAMA con su actual sistema SCADA estima que en promedio son 30000 milisegundos de envío de información, y el protocolo MQTT representa en promedio 0.2537 milisegundos en promedio, si es una cantidad de diferencia representativa, dado que la información del protocolo **MQTT** viaja por la red a través Internet utilizando una dirección IP publica, estática es mucho más rápido, que la información transmitida por radiofrecuencia utilizando la banda **UHF.**

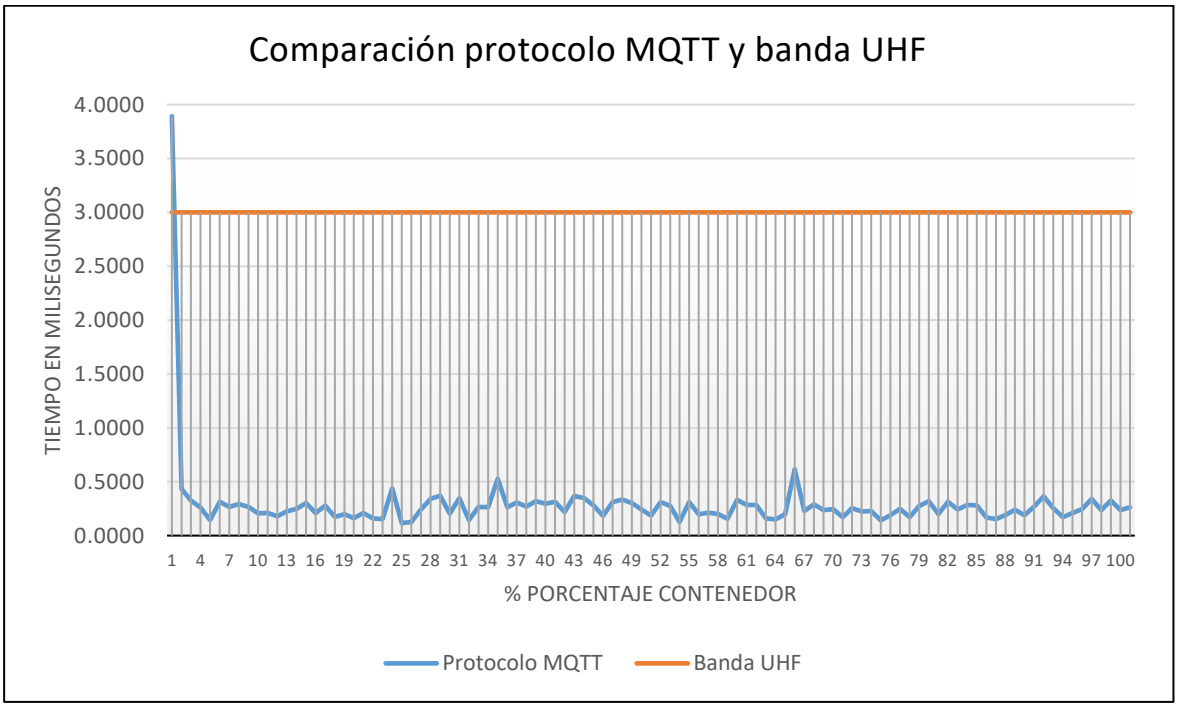

Figura 6-4. Comparación de protocolo MQTT y banda de radiofrecuencia UHF

Por otra parte cabe destacar que el autor (Gastón, 2017) menciona que el protocolo MQTT es muy eficiente y altamente recomendado para aplicaciones industriales de tipo como pruebas de fuego y gas, telemática automotriz, sistemas punto de venta **KIOSK**, detección química, monitoreo de entornos y tráfico, por ser altamente ligero y ser posible transmitir altos volúmenes

de datos sin altas sobrecargas, así como se capaz de emitir datos de un cliente a muchos clientes, soportar varios modelos de conexión como el modelo “siempre conectado” y “algunas veces conectado”, ser capaz de publicar información sobre redes poco confiables y proporciones entregas confiables a través de una conexión frágil, y además sobre todo ser capaz de proveer la escalabilidad necesaria para distribuir datos a cientos de miles de clientes, otro factor muy importante que cabe ser mención es que el protocolo **MQTT** por ser un protocolo que trabaja sobre la capa de aplicación puede ejecutarse en cualquier dispositivo orientado al Internet de las Cosas y al Industrial Internet de las Cosas, pues la especificación del protocolo **MQTT 3.1.1** esta estandarizada por el consorcio **OASIS** y se convirtió en el estándar **ISO/IEC 20922** en año 2016 lo que alienta a los fabricantes a utilizar dicho protocolo.

Otro punto a considerar son los costos, sobre todo en el costo de componentes electrónicos, dado que en la actualidad la terminal **RTU Moscad-L** con **PLC** integrado de **Motorola**, representa un costo total de $ 26, 111 pesos, comparado con un dispositivo orientado al Industrial Internet de las Cosas, como la **Raspberry Pi 3**, el **BeagleBone** o **Intel Galileo Gen 2,** representa un ahorro de aproximadamente 19,000 mil pesos pues es la parte del software la que gobierna la comunicación, y existen cada día más muchas librerías de software libre que ayudan a realizar tareas y por consecuencia a disminuir costos.

6.2 Ámbito profesional

Con respecto a los resultados se hace mención que se elaboró el prototipo con respecto del planteamiento del problema y los requerimientos del sistema para poder llegar a los objetivos y concretar la hipótesis, pero a lo largo de la elaboración del proyecto de investigación surgieron muchos cambios; como cambio de personal en donde desarrollé el proyecto de prototipo **SCADA,**

cambios de directivos en donde pedían otros prototipos ajenas al proyecto, pero que de igual manera todos se realizaron y presentaron a los directivos de **CAPAMA** satisfactoriamente.

Estos cambios y nuevas necesidades (como la elaboración de nuevos prototipos) que surgieron a lo largo de la vida del desarrollo del proyecto de alguna u otra manera generaron desfases en los tiempos, pero se cumplieron los objetivos de desarrollar el prototipo **SCADA** para la Comisión de Agua Potable y Alcantarillado del Municipio de Acapulco, presentándole a los directivos de la empresa un prototipo totalmente funcional que cumplía con los requerimientos de sus necesidades, cabe hacer mención que para llegar a ese punto fue un trabajo muy arduo dado que para desarrollar el proyecto requería de mucha investigación en la empresa como en los tanques del mundo real, para ver su funcionamiento en acción, por lo que considero que como profesional adquirí nuevos conocimientos en el desarrollo de sistemas de control remoto y nuevos mecanismos de protocolos de comunicación como el *MQTT* que es protocolo emergente en el Internet de las Cosas, que está tomando mucho popularidad dado que es muy robusto y seguro, en cuanto a componentes industriales; como los contactores y relevadores industriales para mí esa parte también fue nueva por que aprendí como se leen los esquemáticos industriales y como se conectan e interactúan, dado que ya trabajo con una tensión de corriente eléctrica más alta de lo que como se trabaja normalmente, en general me llene de experiencia de cómo funcionan los sistemas ***SCADA*** y cómo puedo desarrollar uno utilizando nuevos componentes emergentes y orientados a Internet de las Cosas y que como tal yo lo puedo aplicar en el ámbito profesional toda esa experiencia adquirida en estos dos años si hay una empresa que así lo requiera.

6.3 Experiencias futuras

Con respecto al tema del Internet de las Cosas, hasta la fecha de escritura del presente tesis, surge un nuevo concepto denominado *Industrial 4.0* o IIoT (**Internet Industrial de las Cosas**)

que según (Verma, 2019) es la solución integral para todas la necesidades industriales, como optimizar recursos y proporcionar un entorno de trabajo más seguro, las industrias están comenzando a adoptar estos dispositivos para mejorar su eficiencia operativa y garantiza la seguridad pues tiene muchas aplicaciones como en la industrial de agua, la industria de la hospitalidad, la industria del gas y del aceite, minería y transporte, pue según el autor los beneficios en un futuro son que se pueden reducir hasta el 55% el riesgo operacional, reducir hasta en 44% el tiempo de inactividad y expandirse dentro de nuevos mercados hasta en un 40%, debido en parte a la eficiencia y al mantenimiento, a la optimización de recursos activos y al incremento de ingresos.

Por lo que en mi experiencia si se profundiza en el estudio de esta materia y se adquiere la capacidad sobre aplicarlo, se puede emplear en los futuros desarrollos de dispositivos para la empresa **CAPAMA** y para otras ***empresas*** o ***startup*** que quieran implementar este tipo de sistemas a un bajo costo, que en esencia esto es el Internet de las Cosas pero con la experiencia de uso de componentes en el ámbito industrial cosa que se hizo en el presente desarrollo de tesis, componentes como la ***Intel Galileo*** orientada al ***Internet de las cosas*** y la integración del contacto y relevador industrial para controlar una bomba de agua.

Capítulo 7. Conclusiones y trabajos a futuro

7.1 Conclusiones

En la presentación de la funcionalidad y operación del prototipo *SCADA* a los directores responsables de la Comisión de Agua Potable y Alcantarillado del Municipio de Acapulco (**CAPAMA**) se demostró que utilizando componentes electrónicos de bajo costo orientados al Internet de las Cosas y hardware *opensource,* así como herramientas de nueva generación para desarrollar software y utilización de patrones de diseño de software, así como el uso de repositorios de librerías de desarrollo de software que simplifican el tiempo de codificación y la abundante documentación de éstas, se pueden construir sistemas de supervisión, control y adquisición de datos de tal manera que tenga un gasto mínimo significativo en la compra de los componentes electrónicos y en el uso del software ya que no representa un gasto excesivo en el pago de licencias de uso.

Además se demostró ante trabajadores de **CAPAMA** que utilizando el prototipo **SCADA** diseñado en el presente trabajo de investigación, en términos de rapidez en el tiempo de respuesta de la obtención de datos de los sensores de nivel de agua, fue mucho más rápido, seguro y totalmente en tiempo real, comparado con el actual sistema que tiene **CAPAMA** debido a que ese sistema por una lado es muy antiguo y por otro, que actualmente trabaja su comunicación sobre antenas de radio frecuencia, utilizando **Frecuencia Ultra Alta** (abreviado del inglés **UHF**) por lo que cuando hay mucha interferencia; generalmente ocasionada por el ruido de los carros generadas por lo regular en horas picos de tráfico u otras interferencias eléctricas provocadas por cables eléctricos de alta tensión cercanos a las antenas, muy seguido se pierde la señal de forma prolongada generando como consecuencia un desfase en la información de los niveles de los tanques mostrada en tiempo real.

En términos generales cabe aclarar que el presente trabajo se mostró como un prototipo, utilizando componentes orientados al Internet de las Cosas (traducido del inglés **IoT**) y como este tipo de dispositivos pueden ayudar a aminorar los costos de desarrollo e implementación, ayudado de un protocolo de comunicaciones que trabaja a nivel aplicación (en el propio software) como es el caso del protocolo **MQTT** respaldado por **IBM** para la fabricación de sistemas industriales **SCADA**, por lo que para llevarse a la implementación de un tanque del mundo real utilizando el mismo software que muestra la información de nivel de agua en los tanques en tiempo real, así como la misma tarjeta de desarrollo Intel Galileo, se recomienda un tipo de sensor industrial como el *Waterpilot FMX21* y aditivos mínimos en la interfaz electrónica, para poder adecuarlo.

7.2 Trabajos a futuro

Una de las aportaciones que se le pueden agregar al presente trabajo en la parte del desarrollo de software es agregar la funcionalidad de programar el firmware del sistema SCADA que actualmente se incrusta en el sistema operativo **Linux Yocto** y se programa mediante el entorno de desarrollo integrado de Arduino y conectado a la computadora, lo que se sugiere es *programarlo remotamente* a través de una conexión SSH, subiendo o descargando el *sketch hexadecimal* al sistema operativo de la tarjeta de desarrollo Intel Galileo.

Aditivo a esto se puede decir que para dotar de la funcionalidad antes mencionada, también se debería de agregar la funcionalidad de crear código en la misma interfaz gráfica del sistema SCADA a partir de una serie de configuraciones o atributos que el usuario tiene que seguir parar generar un resultado de código acorde a la configuración introducido, esto anclado a la funcionalidad anterior aumenta las posibilidades de hacer el sistema *SCADA* mucho más específico y robusto en la configuración propia de cada uno de los contenedores.

Con respecto a los sensores ultrasónicos utilizados, no se recomienda para uso rudo como tal en un tanque real, dado que la naturaleza de este trabajo fue desarrollar un prototipo, por tal razón se anima a que en un futuro se pueda utilizar y conectar un sensor ultrasónico industrial o un sensor de nivel por presión hidrostática como el *Waterpilot FMX21* de la empresa *Endress+Hauser* para uso en un tanque real, así como también el equipo eléctrico para alimentar el prototipo mediante fotoceldas solares, protección de seguridad para el prototipo como su caja de protección eléctrica y la línea de vista a la conexión mediante *WI-FI*, en caso de que se requiera supervisar, controlar y adquirir datos remotamente.

También en un futuro se puede considerar construir una red de sensores, tomando como base el prototipo SCADA, para capturar y supervisar el gasto de energía eléctrica de las bombas de agua, para evaluar y tomar decisiones con respecto a la eficiencia energética, lo que a largo plazo representaría un ahorro en el consumo de corriente eléctrica.

Otro de los aspectos que se pueden considerar para una futura implementación en este proyecto es la de implementar *Inteligencia Artificial* para tratar de encontrar patrones con respecto a que tiempos u horas del día hay mayor consumo de agua y apagar o encender las bombas con respecto a la decisión que tome la inteligencia artificial al experimentar con los patrones que haya detectado con el estudio de los datos adquiridos de los sensores de nivel de agua.

Referencias bibliográficas

CAPAMA. (19 de Abril de 2019). Comisión de Agua Potable y Alcantarillado del Municipio de Acapulco. Obtenido de Acerca de CAPAMA Nuestra historia: http://www.capama.gob.mx/historia

Cervantes Tafur, H. (02 de Mayo de 2013). Detección de fugas en la tubería de la red principal del sistema de agua potable de la Junta Administradora de Agua Potable Sumak - Yaku - Araque - Otavalo. Ibarra, Ecuador.

Chaves Campos, A., Araya Rodríguez, F., Chaves Jimenénez, A., & Yepez García, V. (2005). Desarrollo de una red de monitoreo por sensores remotos de la calidad de agua. Tecnología en Marcha, 4-8.

Chiaretta, S. (2018). Front-end Development with ASP.NET Core, Angular, and Bootstrap. Indiana: Wrox A Wiley Brand.

Chodorow, K. (2013). MongoDB: The Definitive Guide. United States of America: O'Reilly Media, Inc.

Comisión Nacional del Agua. (2016). Manual de agua potable, alcantarillado y saneamiento - Obras de captación superficiales. Coyoacán, México, D.F.: Conagua.

De Sousa, M. (2015). Internet of Things with Intel Galileo Employ the Intel Galileo board to design a world of smarter technology for your home. Birmingham, UK: Packt Publishing Ltd.

Deitel, D. &. (2005). C# Como programar . USA: Pearson Pretincel Hall.

Elec Freaks. (01 de 07 de 2013). Elec Freaks. Obtenido de https://www.electroschematics.com/wp-content/uploads/2013/07/HCSR04-datasheet-version-1.pdf

Emmit , A. J. (2016). SPA Design and Architecture - Understanding single-page web applications. New York, EU: Manning Publications Co.

Ferguson, J., Patterson, B., Beres, J., Boutquin, P., & Gupta, M. (2003). La biblia de C#. Madrid, España: ANAYA Multimedia.

Ferrando, J. (2015). KnockoutJS Essentials Implement a successful JavaScript-rich application with KnockoutJS, jQuery, and Bootstrap. Birmingham, UK.: Packt Publishing Ltd.

Forbes. (13 de Mayo de 2019). Forbes México. Obtenido de Forbes Tecnlogía: https://www.forbes.com.mx/el-internet-de-las-cosas-en-mexico-podria-detonar-un-mercado-de-4000-mdd-a-2022/

Gama Moreno, L., Sánchez Rodríguez, M., & Ochoa Franco, C. d. (2010). Diseño de una interfaz para la detección de fugas de agua. Revista Digital Universitaria, 1-13.

Garofalo, R. (2011). Building Enterprise Applications with Windows Presentation Foundation and the MVVM Model View ViewModel Pattern. California, United States of America: O'Reilly Media, Inc.

Gastón, H. C. (2017). MQTT Essentials - A Lightweight IoT Protocol - The preferred IoT publish-subscribe lightweight messaging protocol. Birmingham, UK: Packt Publishing Ltd.

Ghenassia, F. (2005). Transaction-Level Modeling with Systemc TLM Concepts and Applications for Embedded Systems. Netherlands: Springer.

Gómez Rivera, A., Velásquez Clavijo, F., & Jiménez López, A. (2016). Monitoreo de sensores en aplicaciones web emebebidas. Colombia: Researchgate.

Haugen, O., Moller-Pedersen, B., & Weigert, T. (2006). Introduction to UML and the Modeling of Embedded Systems. En O. Haugen, B. Moller-Pedersen, & T. Weigert, Embedded Systems Handbook (págs. 1-33). United States of America: Taylor & Francis Group.

Lock, A. (2018). ASP.NET Core in Action. United States of America: Manning Publications Co.

Make. (23 de Abril de 2019). Make. Obtenido de What is a maker? You are: https://makezine.com/2016/04/01/what-is-a-maker-you-are/

Marcillo, P., Bernal, I., & Macías, C. (2015). Sistema para el monitoreo y cuantificación de flujos de lodo basado en una red inalámbrica de sensores. Quito, Ecuador: Researchgate. Obtenido de https://www.researchgate.net/publication/301341627_Sistema_para_el_monitoreo_y_cuantificacion_de_flujos_de_lodo_basado_en_una_red_inalambrica_de_sensores

McLean Hall, G. (2014). Adaptive Code via C# Agile coding with design patterns ans SOLID principles. Washington, United States of America: Microsoft Press.

Microsoft. (20 de Abril de 2019). Microsoft .NET. Obtenido de Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application: https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application

Munro, J. (2015). Knockout.js Building dynamic clien-side web applications. United States of America: O'Reilly Media, Inc.

Nath, S., Stackowiak, R., & Romano, C. (2017). Architecting the Industrial Internet. Birmingham, UK: Packt Publishing Ltd.

NCYT Amazing. (07 de Mayo de 2018). Noticias de la Ciencia y la Tecnología, Divulgando la Ciencia por Internet desde 1997. Obtenido de Un software localiza en tiempo real fugas en ductos de agua, petróleo o gas: https://noticiasdelaciencia.com/art/13709/un-software-localiza-en-tiempo-real-fugas-en-ductos-de-agua-petroleo-o-gas

Ramon, M. C. (2014). Intel Galileo and Intel Galileo Gen 2 API Features and Arduino Projects for Linux Programmers. New York, United Stated of America: Apress Media, LLC.

Richardson, M. (2014). Make: Getting Started with Intel Galileo. United States of America: Maker Media, Inc.

Rodríguez Penin, A. (2007). Sistemas SCADA - Notas de diseño, Normática, Seguridad y comunicaciones industriales, primeros pasos con InTouch. Barcelona, España: MARCOMBO, S.A.

Rodríguez Penin, A. (2008). Comunicaciones Industriales - Guía práctica. Barcelona: MARCOMBO S.A.

Sarasa, A. (2016). Introducción a las bases de datos NoSQL usando MongoDB. Barcelona, España: UOC (Oberta UOC Publishing, SL.

Serna Ruiz, A., Ros García, F. A., & Rico Noguera, J. C. (2010). Guía práctica de sensores. España: Creaciones copyright, S. L.

Tedeschi, N. (01 de 01 de 2010). Microsoft Developer Network. Obtenido de https://msdn.microsoft.com/es-es/library/bb972240.aspx

Thomas, M. S., & McDonald, J. D. (2015). Power System SCADA and Smart Grids. Boca Raton, NW, Florida, E.U.: CRC Press Taylor & Francis Group.

Verma, S. (26 de Abril de 2019). DZone. Obtenido de IoT Zone Industrial Internet of Things [Infographic]: https://dzone.com/articles/industrial-internet-of-things-1?utm_medium=feed&utm_source=feedpress.me&utm_campaign=Feed:%20dzone